CEUR Workshop Proceedings
http://ceur-ws.org
ISSN 1613-0073

Vol-799
urn:nbn:de:0074-799-C

RuleML2011@BRF Challenge

5th International Rule Challenge

Proceedings of the RuleML2011@BRF Challenge, at the 5th International Web Rule Symposium

Fort Lauderdale, Florida (USA)

November 3-5, 2011

Edited by

Stefano Bragaglia University of Bologna, Bologna, Italy

Carlos Viegas Damásio Universidade Nova de Lisboa, Lisboa, Portugal

Marco Montali Free University of Bozen-Bolzano, Bozen-Bolzano, Italy

Charles Petrie Stanford University, Palo Alto (California), USA

Alun Preece Cardiff University, UK

Mark Proctor Red Hat Europe, UK

Umberto Straccia ISTI-CNR, Italy

Preface

This volume collects the selected contributions of the RuleML2011@IJCAI Doctoral Consortium, the RuleML@IJCAI poster session papers, and the demo papers accepted for presentation at the RuleML2011@BRF Challenge.

The RuleML doctoral consortium is a new initiative of the International Symposium on Rules, RuleML, to attract and promote Ph.D. research in the area of Rules and Markup Languages. The doctoral symposium offers to students a close contact with leading experts on the field, as well as the opportunity to present and discuss their ideas in a dynamic and friendly setting. The first edition of the RuleML Doctoral Consortium took place during the first part of the 5th International Symposium on Rules (RuleML 2011@IJCAI) held on July 19th, 2011 in Barcelona. We include here the four selected papers of the doctoral consortium that resulted in lively presentations and discussion, which unfortunately cannot be reflected in print. Elisa Marengo's work, supervised by Matteo Baldoni and Cristina Baroglio, provides a way of specifying patterns of interactions by extending commitment protocols to account for temporal regulations. Woznowski's work, supervised by Alun Preece, describes a system architecture integrating rules with sensor middleware, with a pilot application to tracking of visitor locations in the healthcare domain. Antonius Weinzierl addresses the problem of inconsistency management in heterogeneous knowledge bases, described as multi-context systems. Jak's work proposes a rule-based query answering method for relational data, using hybrid reasoning and forward chaining, exploiting the Jess based implementation for querying a knowledge base of economic crimes.

The RuleML@IJCAI poster authored by Selner, Schwarz, and Zinser presented a poster paper describing IT service management combining business rules and business processes described in SVBR.

The Rule Challenge has reached its 5th year anniversary. It has taken place on November 4th, 2011 in Fort Lauderdale, Florida (USA), in the context of the second part of the 5th International Symposium on Rules (RuleML2011@BRF). The Rule Challenge is devoted to disseminate the most advanced practical experiences with rule-based applications, where state-of-the-art solutions and recent research proposals meet the concrete needs of the market. This year, four main topics have emerged:

1. Combination of rules, objects and ontologies, towards the development of integrated systems able to deal with knowledge-intensive domains and hybrid reasoning.

2. eHealth and clinical applications, dealing with rules and very large information sources that combine structured and non-structured information.

3. Rule editors that exploit parsing mechanism, ontologies and semi-automated composition techniques in order to facilitate the modelling task.

4. Improvement of tools related to RuleML as a standardisation effort.

In particular, Bak, Falkowski and Jedrzejek present a system using rules and ontologies to query relational databases. O'Connor, Richards, Martins, Bingen, Tu and Das introduce a semantic web-enabled system to query and visualise temporal data. Erdem, Erdogan and Oztok describe a software system that provides biomedical query answering capabilities by exploiting Answer Set Programming. In their work, Sottara, Fry, Aliverti, Salatino, Harby, Killen, Nguyen and Wright define a unified architecture for a knowledge intensive patient healthcare management and implement it to warn the patients of an high probability of developing some diseases in the future. Salatino, Aliverti and Calcaprina show how processes and rules can be suitably combined to deal with complex scenarios such as service provision in the case of emergencies. Teymourian, Rohde and Paschke analyse the possibility to use background knowledge about events and their relation with other concepts to improve the quality of complex event processing, discussing the application of their approach in the stock market domain. Gordon presents a software that supports the construction, evaluation and visualisation of arguments by exploiting defeasible and semantic knowledge, and shows how it can be exploited to check compatibility issues among open-source software licenses. Marinos, Gazzard and Krause provide an implementation of a web-based SBVR editor capable of in-line automatic highlighting and auto-completion suggestions. Athan develops a web-based service for permitting the validation of XML instances using particular modules of the large family of RuleML1.0 language in Relax NG schemas. Finally Zaho, Paschke, Ali, and Boley present a web-based collaborative system that provides support to the organising committee of a symposium by efficiently handling complex queries on the domain.

We would like to warmly thank all students, supervisors, referees, co-chairs, members of the program committee and the organising team that made the doctoral consortium and the RuleML2011@IJCAI Doctoral Consortium and the RuleML2011@BRF Challenge a great success.

November 2011

Stefano Bragaglia
Carlos Viegas Damásio
Marco Montali
Charles Petrie
Alun Preece
Mark Proctor
Umberto Straccia

Contents

Part I

RuleML2011@IJCAI Doctoral Consortium

Doctoral Consortium Papers

Extend Commitment Protocols with Temporal Regulations: Why and How

Elisa Marengo, Matteo Baldoni, and Cristina Baroglio*

Dipartimento di Informatica, Università degli Studi di Torino
{emarengo, baldoni, baroglio}@di.unito.it

Abstract. The proposal of Elisa Marengo's thesis is to extend commitment protocols to explicitly account for *temporal regulations*. This extension will satisfy two needs: (1) it will allow representing, in a flexible and modular way, temporal regulations with a normative force, posed on the interaction, so as to represent conventions, laws and suchlike; (2) it will allow committing to complex conditions, which describe not only what will be achieved but to some extent also *how*. These two aspects will be deeply investigated in the proposal of a unified framework, which is part of the ongoing work and will be included in the thesis.

1 Commitment-based Interaction Protocols

The issues of communication and cooperation are crucial in the area of Multiagent systems (MAS). The common solution is to rely on agent interaction protocols. Among different proposals, *commitment protocols* [24] have been widely adopted. All agents, involved in an interaction ruled by a commitment protocol, share the semantics of a set of actions which affect the social state. This semantics is based on the notion of *social commitment*. The idea is that if an agent takes a commitment towards another agent to bring about a condition, then, it will behave in such a way to fulfill the engagement sooner or later. In this respect, commitment protocols have a *deontic nature*, because a commitment introduces a social expectation on the responsibility of some agent towards some other agent to perform something or to achieve some result. Commitment protocols suit well open MAS because they are *respectful* of the agents' autonomy, since no introspection to the agents' mental states is required [26]; they are *dynamic* because commitments can be created, released, deleted and suchlike, and *flexible* because agents are free to take advantage of opportunities or to follow shortcuts [29].

Commitment protocols have fundamentally changed the process of protocol specification from a procedural approach (i.e., prescribing *how* an interaction is to be executed) to a declarative one (i.e., describing *what* interaction is to take place) [28]. Agents decide which action to perform depending on the commitments they have taken and this is because they want to comply with the protocol and fulfill the engagements they have taken [28]. However, in many practical situations this is not sufficient. *Why*? Because, in many cases it is necessary to express some hints on how the interaction should evolve [7,18]. For example, it is necessary to express that some ways to fulfill the commitments

* M. Baldoni and C. Baroglio are the advisor and co-advisor.

are preferred over others, or that only some of them are legal. This does not mean going back to procedural approaches, but it means reconsidering the *how*. In particular, it could be necessary to express *commitments to temporal regulations* and to represent *legal patterns of interaction*. The former are needed to express the engagement of someone to achieve something and in a specific order. For instance, an insurance company commits to paying an in-network surgeon for a procedure only after a covered patient has undergone the procedure. Patterns of interaction, instead, can capture conventions, laws, preferences, habits, or, in general, rules that hold a given reality. For example, in a democratic assembly, a participant cannot speak if she has not obtained the floor.

The thesis, therefore, focuses on the *specification of interactions*, which require a degree of expressiveness that commitments alone do not have. We propose an extension of commitment protocols in order to (i) supply a way for expressing patterns of interaction, capturing laws, conventions and whatever constrains the interaction and (ii) extend the regulative nature of commitments with the possibility of explicitly committing to temporal regulations. A further challenge is how to provide a specification of the interaction in which agents can recognize the normative force of the temporal regulations and explicitly accept them. Indeed, since an agent is free to violate or to behave in accordance with a norm, for a regulation to *influence* the agents' behaviour it must be ascribed of a normative force and, then, it must be accepted as a norm by the agent [13]. The final step will investigate this aspect and propose a unified framework in which both patterns of interactions and commitments to temporal regulations can find place.

2 Temporal Regulations and Commitment Protocols

We discuss how temporal regulations and commitment protocols can be combined: Section 2.1 describes our proposal for including patterns of interaction in protocol specification; Section 2.2 describes commitments to temporal regulations.

2.1 *Why (and How)* expressing legal patterns of interaction

Commitment-based protocols represent a valid solution for interaction protocols specification in open and heterogeneous MAS, mostly because they take into account and respect the agents' and MAS characteristics. For instance, they are respectful of the agents' autonomy, and they do not require a particular implementation or architecture to the agents that are part of the system (heterogeneity). However, to be considered a complete tool for interaction specification, commitment protocols cannot disregard the possibility of specifying *patterns of interaction* as temporal regulations. This requirement is supported by many proposals in the literature (see Section 3) and is due to the need of discriminating those possible executions that are legal from those that are not. These patterns can specify rules of different nature, like habits, conventions, laws, protocol compositions or simply preferences. Basically, they capture a partial ordering between certain actions (or states of affair to be achieved).

Our proposal, described deeply in [4,7], relies on Searle's definition of a social reality [23]. In particular, he identifies a *constitutive* and a *regulative* specification. The former defines a set of actions as foundational of a certain context. In commitment protocols this corresponds to the *social meaning* of the actions, i.e. to actions' semantics, usually given in terms of effects on the social state. The latter, instead, captures how things should be carried on. In current proposals there is not a clear distinction between the constitutive and the regulative part of the specification. Some approaches only care of the regulative nature of commitments (once a commitment is taken, it must be be fulfilled), but completely disregard a specification of *how* things should be done. The thesis proposes an explicit and declarative definition of the regulative specification, given by means of a set of constraints expressed in Linear-time Temporal Logic (LTL). Such constraints [4] define a relative order among different conditions (facts and commitments) that become true in the social state. For example, it allows to express that something can become true only after something else holds, or that if something becomes true, something else must hold sooner or later. The choice of a declarative representation of constraints allows for the specification of what is mandatory and what is forbidden in a protocol, without the need to enumerate the allowed executions (on the contrary to procedural approaches). Indeed, such enumeration is often a huge task, when considering open and dynamics MAS, it limits reusability and the agents' autonomy.

One of the main differences w.r.t. other works from the literature, e.g. [25,17,20], is that temporal constraints are defined in terms commitments and facts, which are, broadly speaking, the effects of the social actions, and not directly on actions (events). This improves flexibility and easiness of reuse in different contexts. Suppose, for instance, to have a specification with the action "pay-by-cash" with semantics *paid*, and to have the constraint "$paid\ before\ sent$". Then, suppose that a change in the context requires that also payment by credit cards can be performed. In this case it is necessary to add a new action"pay-by-credit-card" with the same semantics of "pay-by-cash", i.e. *paid*. Since the constraint is not defined directly on actions, it is not necessary to change it or add a new constraint. Indeed, $paid\ before\ sent$ already constrains the execution of both actions. These aspects are studied in [6], where the adaptation of the Contract Net Protocol to different contexts is discussed. However, where needed 2CL constraints can be used to rule directly actions. The way this is done is by adding a specific effect for each action and then by using these effects in the definition of the constraints.

Orders among actions could be obtained also by adding ad-hoc preconditions to the executability of actions, as done in [28,11,16,12]. However, this solution is not flexible, since regulations are hidden in the actions' definitions and thus difficult to be recognized, updated or modified. Moreover, agents should be always free to decide whether sticking to regulations. If regulations are realized by means of preconditions, agents cannot but choose which action to perform among those that are executable. Thus, they are forced to respect the rules and this is against the normative nature of regulations [13]. In our approach, which is orthogonal to preconditions definition, agents are free to evaluate different alternative paths and to take advantage from opportunities by choosing, among these, the most convenient for them. The role of the constraints is to restrict the set of legal executions, but an agent is free to decide to stick at the rules or to violate them. In the second case the agent knows it could be punished (sanctioned).

A real case study in which this approach has been tested is for the representation of *MiFID: Markets in Financial Instruments Directive* [8]. This directive by the European

Union regulates the interaction of banks, clients, and financial intermediaries in order to guarantee the investor from the intermediaries. The complete example can be found at `http://www.di.unito.it/~alice/2CL/`.

2.2 *Why (and How)* Committing to regulations

In many practical situations, commitments involve rich temporal structures rather than simple conditions to achieve. Let us consider a few examples: (a) an insurance company commits to reimbursing a covered patient for a health procedure provided the patient obtains approval from the company prior to the health procedure; (b) a pharmacy commits to provide medicine only if the patient obtains a prescription for that medicine; (c) an insurance company commits to paying an innetwork surgeon for a procedure only after a covered patient has undergone the procedure. Presumably, the surgeon would bill the insurance company after performing the procedure.

Commitments alone do not have the degree of expressiveness required by these conditions. Indeed, conditions in conditional commitments do not impose a temporal ordering: the consequent condition can be achieved even if the antecedent condition does not hold. The contribution of the thesis for capturing these aspects is described in [18] and consists in a new formalization of commitments, where temporal regulations are incorporated as content of commitments themselves. In this way regulations assume a normative force which is due to the regulative nature of commitments. For example, $C(x, y, \top, a\ before\ b)$ expresses the engagement of x towards y not only to make a and b happen but also to make them happen in the given order. Participants to the interaction will be able to guide their actions locally, in order to not violate any commitment they have taken, and to judge the compliance of their counter-parties. Indeed, since regulations are placed inside commitments, the debtor will be considered responsible and thus liable for violations. Consider, for example, a regulation saying that a physician's referral should precede a surgeon's procedure; in this situation, in case of violation, it is not clear whether the physician is responsible for moving first or the surgeon is responsible for moving second. By placing the regulations in commitments, we make it explicit that it is the debtor of the commitment who needs to ensure its satisfaction.

For this reason it becomes fundamental for an agent to establish, before taking a commitment, if it has a sufficient support by the other agents. The elements the agent has to consider are both the set of actions it can perform and the cooperation it can get from the others, via the set of commitments of which it is the creditor. To this aim, we formalized the notions of *control* and *safety*. The former captures the capability, for an agent, to bring about a regulation. It depends on the actions a certain agent can perform and on commitments directed towards it. The latter is strictly related to the notion of control: a commitment is safe if its debtor has established sufficient control to guarantee being able to discharge it.

To the best of our knowledge, no approach for protocol specification based on commitments allows to express commitments to temporal regulations. Placing temporal regulation inside commitments, however, allows for the representation of a debtor and thus allows to precisely identify who is responsible for each regulation and potentially liable for a violation. This is an advantage w.r.t. approaches based on expectations [1] which are not scoped by a debtor and a creditor. Moreover, it helps make the regulations explicit within the system of interacting agents and thereby facilitates their coordination. Accordingly to [13] it allows also to explicitly represent the recognition and the acceptance of a regulation by the agents.

3 Related Works

The need of expressing temporal regulations is supported by many attempts in the literature to rule actions' execution along the interaction. However, in our opinion, all these attempts can be improved in order to better take into account regulative aspects without compromising the flexibility and all the good properties of commitment protocols.

Fornara and Colombetti [14] propose a model based on *interaction diagrams*, a kind of specification which is similar to UML sequence diagrams. The choice of relying on interaction diagrams is very strong because it forces the ordering of action execution defining a strict set of allowed sequences. It basically can be classified as a procedural approach, thus presenting the same shortcomings [20]: it weakens agents' autonomy to decide which action to perform and their capability to take advantage of opportunities; it is too rigid, where instead the openness and dynamicity of MASs require higher flexibility of the specification.

The use of a declarative approach is proposed by Singh [25] and by Mallya and Singh [17]. In these works they define a *before relation* applied to events. The idea is that when a before relation among two activities is specified, the only thing that matters is the order among the two, no matter what happens in-between. Even if the choice of adopting a declarative specification overcomes many limits of the proposals described before, the main limitation is that temporal regulations are defined over actions (events). As described in the previous section, a greater degree of decoupling between actions and temporal specifications can, in our opinion, support better the openness of MAS. Moreover, this kind of regulation is conceived as a solution for service composition external from protocol specification. In our proposal, instead, temporal regulations actively contribute to the definition of the protocol.

The same shortcoming can be found in other proposals. It is hard to be exhaustive but let us consider a proposal inspired from the neighboring area of business processes. Pesic and van der Aalst [22] propose ConDec, a declarative language for business process representation. ConDec is a graphical language grounded in Linear-time Temporal Logic, which is used to rule the activities that compose a process. Montali and colleagues [9,20] integrate ConDec with SCIFF thus giving a semantics to actions that is based on expectations. The authors use this approach to specify interaction protocols and service choreographies. As the previous one, also this proposal is based on actions, thus suffering the same shortcomings.

Dialogue games are another solution for communication specification. Different kinds of dialogues basically define different kind of schema according which the agents can interact. The differences among them are given by the aim of the communication (e.g. persuade, inform, negotiate). Our approach is more general, since it provide the basic components for interaction specification and since it is not limited to communication (message exchange), as in [15,19], but to interaction in general. By means of this tool the desired interaction can be declaratively drawn according to the needs and to the aim of the system, without having to choose one among predefined schema. This is along the line of the claim by Singh in [10]. The idea is that a standard is difficult to be used "as-is" for modelling a desired system. To model a desired interaction he proposes standards for standard definition. Similar considerations holds for works that propose to use commitments for ACL semantics: they define predefined schema (type) for actions specification, bringing to an undesired rigidity. Our approach, and in general approaches based on the meaning of messages [10], are more flexible.

4 Ongoing and Future Work

Our proposal, for patterns specification and commitments to temporal regulations, allows facing two different lacks of commitment protocols related to temporal regulations. The thesis will finally investigate a unified framework in which both aspects can be reconciled under a common normative force. In other words, agents will be provided of the necessary means to explicitly recognize and accept temporal regulations, thus accepting their behaviour to be influenced by them [13]. Of course, agents will be free to decide to violate them in every moment. Thus, this framework will allow agents to commit to complex conditions, it will allow for the specification of patterns of interaction representing norms, conventions and rules, and it will provide the tools necessary to the agents to verify their ability to fulfill the engagements (along the line of control and of safety).

Reconciling these two aspects, which are *strictly connected to one another* [5], opens the way to interesting considerations. In particular, a set of constraints restricts the set of commitments that can be taken by the agents to those that can be considered legal. For example, before getting on a train a person has to punch the ticket. Only after, he/she is allowed to travel on the train. However, think to a person that commits to travel to his/her destination first and, once he/she reached it, to punch the ticket. In this situation, it is impossible for the person to fulfill his/her commitment without violating the norm. More generally, in order to propose a unified framework some important questions are to be answered. For example, given a set of norms expressed in terms of patterns of interaction, how can one establish which commitments are compliant and which are incompatible? If norms change, how do these changes affect the set of commitments? Moreover, how can one monitor the interaction of the agents and discover violations? In this respect, a solutions could be to lean on e-institutions. In [3] an initial proposal is described, where the idea is to define specific artifacts [21] able to detect violations. The kinds of reasoning that can be performed in this way are many. For example, it is possible not only to detect a violation, but also to classify different violations according to how relevant they are or how costly would be to repair from the damage caused by the violation.

The modularity of our proposal suits well the needs of the dynamic specification of protocols, along the line of [2]. Artikis' proposal is to define a set of *meta-actions* that can be performed by the agents at run-time, and that can change the set of rules that define the protocol. During this phase, the interaction is suspended. It will be resumed once the definition of the new rules is finished. In our proposal, it is possible to define a set of meta-actions whose effects are to change the set of constraints representing the norms that must be respected in the MAS. By performing those actions, agents would be able to change at run-time and dynamically, i.e. without suspending the interaction, the set of rules. Also this extension opens the way to some important question. For instance, who and how is allowed to change the rules? As part of the future work we will investigate also these aspects.

Finally, our proposal can be applied also to business process representations, and in particular to those situations in which a sequential representation is not adequate, due to the many alternative executions. In these contexts the high number of possible sequences suggests that a declarative representation, based on rules or constraints, is preferable with respect to procedural approaches. We plan to investigate more deeply these aspects along the line of [27], where a business process is described in terms of the commitments of the actors, that are involved in the process. One advantage of adopting declarative specifications and a modular representation of the constitutive (actions) and the regulative (constraints) part, is a gain of time and money in the operation of update of the business process due, for example, to norms changes.

5 Acknowledgments

We would like to thank Viviana Patti, Munindar P. Singh, and Amit K. Chopra for the helpful discussions and the work done together. We would like to thank also the reviewers for the helpful suggestions.

References

1. Alberti, M., Chesani, F., Daolio, D., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and Verification of Agent Interaction Protocols in a Logic-based System. Scalable Computing: Practice and Experience. Spec. Iss. on Foundational Underpinnings for Pragmatic Agent-based Systems 8(1), 1–13 (2007)
2. Artikis, A.: Formalising Dynamic Protocols for Open Agent Systems. In: Proc. of the 12th International Conference on AIL. pp. 68–77 (2009)
3. Baldoni, M., Baroglio, C., Bergenti, F., Boccalatte, A., Marengo, E., Martelli, M., Mascardi, V., Padovani, L., Patti, V., Ricci, A., Rossi, G., Santi, A.: MERCURIO: An Iteraction-oriented Framework for Designing, Verifying and Programming Multi-Agent Systems. In: Proc. of COIN@MALLOW. pp. 66–83 (2010)
4. Baldoni, M., Baroglio, C., Marengo, E.: Behavior-Oriented Commitment-Based Protocols. In: Proc. of the European Conf. on Art. Int. (ECAI). pp. 137–142 (2010)
5. Baldoni, M., Baroglio, C., Marengo, E.: Commitment-based Protocols with Behavioral Rules and Correctness Properties of MAS. In: Proc. of Workshop DALT. pp. 66–83 (2010)
6. Baldoni, M., Baroglio, C., Marengo, E.: Constraints among Commitments: Regulative Specification of Interaction Protocols. In: Proc. of Int. Workshop on Agent Communication. pp. 2–18 (2010)
7. Baldoni, M., Baroglio, C., Marengo, E., Patti, V.: Constitutive and Regulative Specifications of Commitment Protocols: a Decoupled Approach. ACM Transactions on Intelligent Systems and Technology, Special Issue on Agent Communication (2011). To appear.
8. Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F.: Learn the rules so you know how to break them properly. In: Proc. of WOA 2011 (2011)

9. Chesani, F., Mello, P., Montali, M., Torroni, P.: Verifying A-Priori the Composition of Declarative Specified Services. In: Proc. of MALLOW. CEUR Workshop Proceedings, vol. 494. CEUR-WS.org, Turin, Italy (September 2009)
10. Chopra, A.K., Artikis, A., Bentahar, J., Colombetti, M., Dignum, F., Fornara, N., Jones, A.J.I., Singh, M.P., Yolum, P.: Research Directions in Agent Communication. ACM Trans. on Int. Sys. and Tech., Spec. Iss. on Agent Communication (2011). To appear.
11. Chopra, A.K., Singh, M.P.: Contextualizing Commitment Protocol. In: Proc. of AAMAS 2006. pp. 1345–1352. ACM (2006)
12. Chopra, A., Singh, M.P.: Constitutive interoperability. In: Proc. of AAMAS'08. pp. 797–804 (2008)
13. Conte, R., Castelfranchi, C., Dignum, F.: Autonomous Norm Acceptance. In: ATAL. Lecture Notes in Computer Science, vol. 1555, pp. 99–112. Springer (1998)
14. Fornara, N., Colombetti, M.: A Commitment-Based Approach To Agent Communication. Applied Artificial Intelligence 18(9-10), 853–866 (2004)
15. Gaudou, B., Herzig, A., Longin, D.: A Logical Framework for Grounding-based Dialogue Analysis. Electr. Notes Theor. Comput. Sci. 157(4), 117–137 (2006)
16. Giordano, L., Martelli, A., Schwind, C.: Specifying and verifying interaction protocols in a temporal action logic. J. Applied Logic 5(2), 214–234 (2007)
17. Mallya, A.U., Singh, M.P.: Introducing Preferences into Commitment Protocols. In: Agent Communication II, International Workshops on Agent Communication (AC 2005 and AC 2006). LNCS, vol. 3859, pp. 136–149. Springer, Utrecht, Netherlands (2006)
18. Marengo, E., Baldoni, M., Baroglio, C., Chopra, A.K., Patti, V., Singh, M.P.: Commitments with Regulations: Reasoning about Safety and Control in REGULA. In: Proc. of AAMAS. pp. 843–850. IFAAMAS (2011)
19. McBurney, P., Parsons, S.: Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents. Journal of Logic, Language and Information 11(3), 315–334 (2002)
20. Montali, M.: Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach, LNBIP, vol. 56. Springer, Heidelberg (2010)
21. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems 17(3), 432–456 (2008)
22. Pesic, M., van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In: Proc. of BPM 2006 International Workshops. LNCS, vol. 4103, pp. 169–180. Springer (2006)
23. Searle, J.: The construction of social reality. Free Press, New York (1995)
24. Singh, M.P.: An Ontology for Commitments in Multiagent Systems. Artif. Intell. Law 7(1), 97–113 (1999)
25. Singh, M.P.: Distributed Enactment of Multiagent Workflows: Temporal Logic for Web Service Composition. In: Proc. of AAMAS. pp. 907–914. ACM (2003)
26. Singh, M.P.: Agent Communication Languages: Rethinking the Principles. In: Communication in Multiagent Systems. LNCS, vol. 2650. Springer (2003)
27. Telang, P.R., Singh, M.P.: Business Modeling via Commitments. In: SOCASE. LNCS, vol. 5907, pp. 111–125. Springer (2009)
28. Winikoff, M., Liu, W., Harland, J.: Enhancing commitment machines. In: DALT 2004. LNCS, vol. 3476, pp. 198–220. Springer (2005)
29. Yolum, P., Singh, M.P.: Commitment Machines. In: Intelligent Agents VIII, ATAL 2001. LNCS, vol. 2333, pp. 235–247. Springer (2002)

Rule-Based Semantic Sensing

Przemyslaw Woznowski and Alun Preece

Cardiff University, School of Computer Science, 5 The Parade, Cardiff, UK
{p.r.woznowski,a.d.preece}@cs.cf.ac.uk

Abstract. Rule-Based Systems have been in use for decades to solve a variety of problems but not in the sensor informatics domain. Rules aid the aggregation of low-level sensor readings to form a more complete picture of the real world and help to address 10 identified challenges for sensor network middleware. This paper presents the reader with an overview of a system architecture and a pilot application to demonstrate the usefulness of a system integrating rules with sensor middleware.

Keywords: RBS, SNM, RFID, rules, tracking, sensors, JESS, GSN.

1 Introduction & Motivation

A single sensor provides only partial information on the actual physical condition measured, e.g. an acoustic sensor only records audio signals. For an application to reason over sensor data, raw sensor readings have to be captured and often aggregated to form a more complete picture of the real-world condition measured. Sensor Network Middleware (SNM) aids this process. As defined in [1], "The main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications". However, existing SNMs don't give the user – who can be an expert in some area that is not computer science – an opportunity to easily specify data aggregation logic themselves.

It is hypothesised that rules help to address this problem and can greatly improve the SNM. Moreover, such an approach to sensor networks addresses many of the 10 challenges for SNM, listed in [2], in the following way:

Data Fusion - Rules fuse simple facts to infer higher-level facts about the real world.
Application Knowledge - Expert's knowledge encoded into an automated system.
Adaptability - Applicable to any domain, non-programmers can write rules.
Abstraction Support - Each fact is an interpretation of data. How the data is interpreted is determined by an expert via rules.
QoS Support - Multiple combinations of rules and facts can often answer the same query. Solution can be explained by retracing the reasoning.

The remaining challenges: Network Heterogeneity, Dynamic Topology, Resource Constraints, Security and Scalability – need to be met by SNM. Additional benefits come from well-known advantages of using RBS systems: reproducibility, permanence, consistency, timeliness, efficiency, breadth, completeness, documentation, etc. – as identified in [3]. Finally, representing sensor data in the form of facts adds semantics. We propose the Rule-Based Semantic Sensor System (RBS3) which employs a Rule-Based System (RBS) on top of existing off-the-shelf SNM. The pilot application described in Section 3 was implemented to test the hypothesis that rules help to address the 10 Challenges and ease the development, maintenance, execution and extensibility of sensing-based applications.

2 Proposed System Architecture

The proposed system architecture in Figure 1 consists of four layers. The SNM layer serves as a bridge between physical sensors and the layer above it. It abstracts away the network heterogeneity by modelling hardware entities, and the output they produce, in software. The Interface layer is responsible for injecting sensor data, coming from the layer below it, into the Reasoning Engine layer in the form of facts. Its main function is to translate the SNMs output into facts, defined in terms of a semantic data model (for which we currently use RDF Schema for simplicity, although details of this are not included in this paper due to lack of space). The Reasoning Engine layer is the heart of the system. It continuously reasons over incoming facts and those already in the Working Memory (WM) to produce new, more complex facts. The more complex the facts, the higher the semantic enrichment and therefore more detailed picture of the real world. The Application layer bridges the user's interface with the Reasoning Engine. It exposes facts and queries, which persist in the KB, to the application. Moreover, it takes user's queries, pushes them to the layer below and returns the results in the format easily consumable by the application.

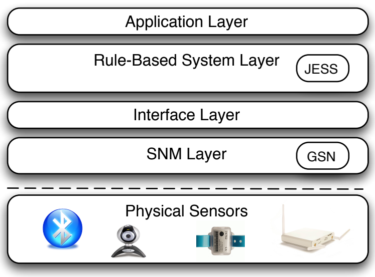


Fig. 1. System Architecture

The system architecture in Figure 1 is implemented in our RBS3 system as follows. The SNM layer currently consists of Global Sensor Networks (GSN) middleware, which serves XML data in response to queries. GSN (GNU GPL license) is a SNM, which deals with sensor network heterogeneity via use of Virtual Sensor (VS) abstraction. Any type of sensors, whether hardware or software, is represented by a single VS XML file, which specifies its inputs and output structure [4]. However, other SNMs such as: Pachube, ITA Sensor Fabric or SWE compliant middleware could replace GSN. The Interface Layer parses the XML data to JavaBeans, which are then injected into the Reasoning Engine (Jess) in the form of facts. Alternatively SweetRules, which is much more compact and offers extra features, could replace Jess, as both rule engines accept rules in CLIPS format. Queries, their arguments, and return parameters are available through the Application Layer, which serves data in JSON(JavaScript Object Notation) format, because it is a lightweight data-interchange format, which is easy for humans to read/write and easy for machines to parse.

3 Pilot Application

The aim of this application is to provide information on people's indoor locations, their history of visited locations, and information on walking speed between the locations - later referred in this paper as "corridor tests". The basic assumption for the system to work is that the tracked person wears either the RFID tag or any Bluetooth(BT) enabled device pre-registered with the mobileDevStore VS. Also corridor entities need to be defined in the corridorStore VS in order for the system to log corridor tests. This is part of a larger project looking at people's recovery from physical injury.

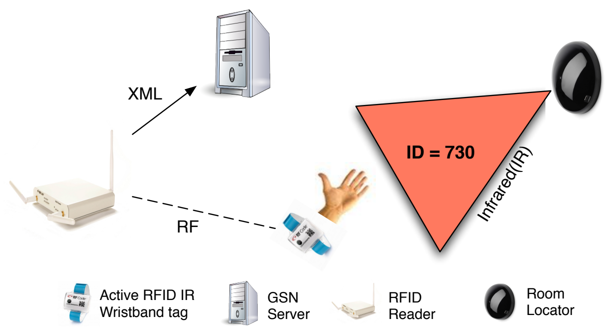


Fig. 2. Localisation with the Room Locator

The room-level localisation of active RFID tags is possible via use of the Room Locators, which broadcast a pre-set location code via IR (Figure 2). The active RFID wristband tags are IR enabled, therefore report the IR location code to the RFID reader. For this a direct line-of-sight between tag and the Room Locator is required. In the experiment, the network consisted of 1 laptop, 1 RFID reader, 2 Room Locators and 2 standard desktop PCs with Bluetooth, placed in two rooms, both running an instance of the GSN server. To clarify, the software/hardware components used in the experiment have the following functions:

RFID Active Tag: Every 2 seconds broadcasts it's unique ID, IR location code, motion status, etc. to the RFID Reader.

Bluetooth: Alternative source of information on user's location.
GSN: Connects to sensors, logs their readings and exposes them via a web server. Also serves as a source of information for static data.
RFID Reader: Receives active RFID tags' signals.
Room Locator: Transmits an IR pulse pattern containing a unique 3-digit location code to enable room-level accuracy localisation.
mobileDevStore VS: Lookup service. Stores name to RFID/BT address mappings.
corridorStore VS: Lookup service. Stores corridor entities (endA, endB, length).
btReader VS: Logs device's discovery time, BT address and reader's location.
rfidReader VS: Logs tag's discovery time, ID and reader's location.

3.1 Facts

Shadow fact, as described in [5], is "an unordered fact whose slots correspond to the properties of a JavaBean". Three shadow fact templates are defined in the Knowledge Base (KB): `MobileTrace, Person` and `Corridor`. They allow for quick insertion of JavaBean objects into the Working Memory (WM) and they directly represent GSN Virtual Sensor's outputs.

```
(deftemplate MobileTrace (declare (from-class javaBeans.MobileTrace)))
;Java class members/slots: location, address, time.
(deftemplate Person (declare (from-class javaBeans.Person)))
;Java class members/slots: name, deviceAddress.
(deftemplate Corridor (declare (from-class javaBeans.Corridor)))
;Java class members/slots: enda, endb, length.
```

Apart from shadow facts described above, the following set of unordered facts exists in the KB. All these facts originate from rules defined in the KB. To summarise, in this implementation, shadow facts (capitalised) represent sensor readings and unordered facts are used internally in Jess to represent fused sensor data. These fact templates are the semantic interface and we do have the RDF Schema for them, however, this is not included due to lack of space.

```
(deftemplate is-seen-at (slot name)(slot location)(slot time))
(deftemplate is-currently-at (slot name)(slot location)(slot tStart)
 (slot tFinish))
(deftemplate was-at(slot name)(slot location)(slot tStart)(slot tFinish))
(deftemplate was-tracked (slot name) (slot endA) (slot endB)
 (slot tStart)(slot tFinish)(slot distance)(slot tTaken)(slot velocity))
```

3.2 Rules

The set of rules defined in the KB, allows the system to infer four types of observations from sensor and static data: `is-seen-at, is-currently-at, was-at` and `was-tracked`. First rule, `seen_at`, simply aggregates `Person` and `MobileTrace` facts to assert `is-seen-at` to the WM. It also retracts all the `MobileTraces` that are successfully fused with `Person` facts.

```
(defrule seen_at
 (Person (deviceAddress ?address)(name ?name))
 ?mob <- (MobileTrace (location ?loc)(time ?time)(address ?address))
 =>  (retract ?mob)
     (assert (is-seen-at(name ?name)(location ?loc)(time ?time))))
```

The next rule, `was_at`, asserts two facts to the WM: `was-at` and `is-currently-at`. The latter contains information about a person's current location; therefore whenever the same person is seen at different location, the `is-currently-at` fact becomes `was-at` and a new `is-currently-at` fact is added. This time both facts, which are used to infer new information (`is-seen-at` and `is-currently-at`) are retracted from the WM, as at any point in time there should only exist one of each of these facts, simply because some person can only be seen at one location at any time. However, `was_at` will never fire unless the initial `is-currently-at` fact is inserted as `is-currently-at` facts are only produced by this rule. Therefore, a dummy fact is defined for each person tracked by the system, e.g. for Pete we have `(assert(is-currently-at(name ‘‘Pete’’)(location ‘‘dummyLoc’’)(tStart 0)(tFinish 0)))`.

```
(defrule was_at
 ?c <- (is-currently-at(name ?n)(location ?l1)(tStart ?tS)(tFinish ?tF))
 ?seen <- (is-seen-at (name ?n)(location ?l2)(time ?t))
 =>(retract ?c ?seen)
   (assert(was-at(name ?n)(location ?l1)(tStart ?tS)(tFinish ?tF)))
   (assert(is-currently-at(name ?n)(location ?l2)(tStart ?t)(tFinish ?t))))
```

As opposed to `was_at`, the `update_current_loc` rule deals with the situation when the location reported by `is-seen-at` is the same: it simply updates the `tFinish` of the `is-currently-at` fact.

```
(defrule update_current_loc
 ?c <- (is-currently-at (name ?n)(location ?loc)(tStart ?tS)(tFinish ?tF))
 ?seen <- (is-seen-at (name ?n)(location ?loc)(time ?time))
 (test(< ?tF ?time))
 => (retract ?seen)(modify ?c (tFinish ?time)))
```

The three rules discussed above can already provide information on a subject's current location and history of visited locations. If a human expert was to analyse this data, they could easily answer questions on where the person currently is/was at any point in time. Additionally, it wouldn't be a problem to tell how much time it took somebody to transfer from one location to another, as this can be worked out from `was-at` facts. `Find_corridor_events` does exactly this, but in a slightly different way. Instead of analysing consecutive `was-at` facts it works with `is-currently-at` and `was-at` facts, whose locations are defined as ends of some `Corridor` in the KB. However, `was-tracked` fact is asserted if the subject travels from A to B and not B to A. This logic is there in purpose, as one may be interested in journeys in only one direction.

```
(defrule find_corridor_events
 (was-at (name ?name)(location ?loc1)(tStart ?t1S)(tFinish ?t1F))
 (is-currently-at (name ?name)(location ?loc2)(tStart ?t2S)(tFinish ?2tF))
 (Corridor (enda ?loc1)(endb ?loc2)(length ?length))
 => (bind ?tTaken (- ?t2S ?t1F))
    (assert (was-tracked (name ?name)(endA ?loc1)(endB ?loc2)(tStart ?t1F)
    (tFinish ?t2S)(distance ?length)(tTaken ?tTaken)
    (velocity (/ ?length (/ ?tTaken 1000)))))
```

It seemed to be enough to define only four rules in the KB. However, test results have revealed the missing logic. Assuming a scenario where somebody visits locations in the following order: 730, 000, 740, 000, 730, 000, 740 and the corridor is defined as `(Corridor (enda 730)(endb 740)(length 20))` any person would know that there are two journeys of interest: 2 x “730 through 000 to 740”. However, the system inferred one additional fact: “730 through 000, 740, 000, 730, 000 to 740”. Since it does not make sense to consider cyclic journeys, we also have rules to retract these from the WM. Obviously the `find_corridor_events` rule could be replaced with a query, which looks for `was-at` and `Corridor` facts, however the general idea is to infer new, more complex facts from existing lower-level facts, rather that to come up with a clever query which can provide information on one’s journeys. By inserting new and often more complex facts, the KB is populated with more data what allows for defining new rules that can simply look at existing facts and infer even more complex ones. `Find_corridor_events` is an example of a rule that does not modify facts that are already in the WM but instead populates new facts, which can then be used by other rules.

3.3 Results

Three queries, that take name as the parameter, are defined in the KB: `find_journeys`, `where_is` and `location_history`. They simply look for `was-tracked`, `is-currently-at` and `was-at` facts respectively for some person. Querying the WM becomes very simple, as neither new data needs to be inferred, nor any calculations done - simply query parameter needs defining. Hence query of the form `"find_journeys Pete"` lists all the `was-tracked` facts (corridor test results) for Pete.

To validate this application some tests were carried out. The table below contains results of the corridor tests recorded by the system, contrasted with times recorded by the subject of these tests via use of an ordinary watch synchronised with system’s time. For simplicity, times represented in the table are of form HH:MM:SS and do not include milliseconds. From Table 1 it is easy to see that the system never underestimates the

Table 1. Experiment Results

Recorded by the system			Recorded by hand		
tStart	tFinish	tTaken	tStart	tFinish	tTaken
13:30:44	13:31:26	42	13:30:43	13:31:21	38
13:36:21	13:37:00	39	13:36:18	13:36:56	38
13:59:25	14:00:08	43	13:59:22	14:00:03	41
14:13:41	14:14:16	35	14:13:38	14:14:08	30

tTaken but is rather an overestimate of it. This behaviour was predictable due to the following two factors. Firstly, RFID tags broadcast their signal every 2 seconds (when in motion) and therefore introduce a maximum delay of 2 seconds on both ends of the corridor. Therefore, if somebody arrives at some location, this information may not be injected into the system until the next round of broadcasting, which in worst case is 2 seconds later. Secondly, in order for the tag to report it’s location it has to receive the IR signature of some location. If there is no direct line-of-sight between the tag and Room Locator, the tag reports location code 000 instead of the broadcasted location code. To account for both these factors the system could subtract the average delay time from the results returned.

4 Related Work

Many of the popular Sensor Network Middlewares, such as GSN, ITA Sensor Fabric, Pachube or SWE-compatible products are rather low-level [4,6,7,8]. They simply provide sensor data (using different models and abstractions) and do not make it very easy for the programmer to program with them. We consider these as candidates for the SNM layer rather than complete solutions that meet all the 10 challenges to a satisfactory level. Application knowledge, adaptability and abstraction support are not very well addressed by these SNMs.

To the best of our knowledge there are no systems that implement a Rule-Based System on the top of SNM. The most similar is Semantic Streams (SS) – "... a framework ... that allows users to pose declarative queries over semantic interpretations of sensor data" [9]. SS and RBS3 are both very high-level in terms of ability to query for real world facts. RBS3 adapts ideas from SS in a sense that rules have analogous function to the semantic services – both take some inputs and produce outputs as a result of data aggregation. Moreover, streams of data (in case of RBS3 - facts) are reused in both systems. However, SS uses a modified version of Prolog and connects to sensors using MSR Sense toolkit, hence lacks openness at the lower layer, and can only use sensors compatible with this toolkit – according to Microsoft [10] "MSR Sense has only been tested on TinyOS-based sensor motes, although in theory, it should work with any 802.15.4 compatible wireless sensors". Therefore SS is hard to extend with new sensors or other sensor middleware. In addition rules are coded implicitly using SS markup language another specification to learn in order to use the system. RBS3, on the other hand, defines rules explicitly in a well-known "standard" form of rule (CLIPS) and allows adding new rules at the runtime. Semantic Streams use a variant of backward chaining to find semantic services that can satisfy the query. In contrast, RBS3 implements forward-chaining mechanism and only allows the user to query the system using queries defined in the KB.

5 Conclusion & Future Work

In this paper, we have proposed a system architecture which combines rules and sensor middleware to better address 10 identified challenges for sensing systems. The proposed system architecture provides several benefits amongst which are: flexibility and extensibility. This approach also aids application development, maintenance, deployment, and execution. Other benefits come from using a Rule-Based System and they help to address half of the 10 challenges for SNM: Data Fusion, Application Knowledge, Adaptability, QoS and Abstraction Support.

The current implementation of the system only has GSN in the SNM layer. In the next version of RBS3, wrappers to interface with other popular sensor middleware, such as Pachube, ITA Sensor Fabric or SWE, will be present. The proposed system architecture makes the system extensible – if the user is constrained to use a specific type of SNM they can implement their own wrapper for it; and flexible – if the user does not want to be limited to use one SNM but wants to use sensor data from various sources. Another improvement to the system would be to modify the Interface Layer, so that RDF data serialised in JSON is parsed and injected into the Reasoning Engine, instead of XML parsed to JavaBeans – "since XML just describes grammars there is no way of recognising a semantic unit from a particular domain of interest" [11]. The Reasoning Engine would then be processing semantically rich data.

Because the system works in a forward-chaining way, only when a rule that produces certain type of facts is specified, these facts become available for queries. The next version of the system may use both: backward- and forward-chaining mechanisms to allow the user to query for data for which production rules are specified just before the query – therefore, historical data stored in DBs can participate to the query. Another, desirable enhancement to the system is at the Application layer – the system is easier to interface with if the user has a choice whether to receive data in JSON or RDF format.

Scalability is something that the entire system, once fully implemented, has to be extensively tested for in order to provide good response times and good level of reliability – the more sensors used, the more data to parse and store. However, the system as it is, is proven to work correctly and starts to reveal it's potential. As facts are injected into the system, they are not only aggregated together but also they are re-used across multiple rules. The more complex the facts are the better they re-create the real world conditions measured by sensors and can answer more complex queries.

Acknowledgements

We would like to thank Prof. Robert van Deursen, Arshi Iqbal (School of Healthcare Studies, Cardiff University) and Dr Allison Cooper (Swansea University) for providing RFID equipment and continuous help with system development and testing. Many thanks to Chris Gwilliams for proofreading this paper.

References

1. K.Römer, O.Kasten, and F.Mattern, "Middleware challenges for wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol.6, pp.59–61, Oct. 2002.
2. M.Molla and S.Ahamed, "A survey of middleware for sensor network and challenges," in *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on Parallel Processing Workshops*, p.6, 2006.
3. C.E. Brown and D.E. O'Leary, "Introduction to Artificial Intelligence and Expert Systems." available at `http://www.carfield.com.hk/document/ai/Expert+Systems.html`.
4. K.Aberer, M.Hauswirth, and A.Salehi, "A Middleware For Fast And Flexible Sensor Network Deployment," in *Proceedings of the 32nd international conference on Very large data bases ACM*, 2006.
5. E.Friedman-Hill, *Jess in Action: Java Rule-based Systems.* Greenwich, CT: Manning, 2003.
6. U.Hague, "Pachube" available at `http://www.pachube.com`.
7. J.Wright, C.Gibson, F.Bergamaschi, K.Marcus, R.Pressley, G.Verma, and G.Whipps, "A dynamic infrastructure for interconnecting disparate isr/istar assets (the ita sensor fabric)," in *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pp. 1393 –1400, 2009.
8. M.Botts, G.Percivall, C.Reed, and J.Davidson, "OGC Sensor Web Enablement: Overview and high level architecture," in *GeoSensor Networks*, vol.4540 of *Lecture Notes in Computer Science*, pp.175–190, Springer Berlin / Heidelberg, 2008.
9. K.Whitehouse, F.Zhao and J.Liu, "Semantic Streams: a Framework for Declarative Queries and Automatic Data Interpretation," *Technical Report*, 2005.
10. Microsoft Corporation, "MSR Sense" available at `http://research.microsoft.com/en-us/projects/msrsense/default.aspx`.
11. S.Decker, F.V. Harmelen, and J.Broekstra, "The Semantic Web - on the respective Roles of XML and RDF," *IEEE Internet Computing*, 2000.

Advancing Multi-Context Systems by Inconsistency Management*

Antonius Weinzierl

Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
weinzierl@kr.tuwien.ac.at

Abstract. Multi-Context Systems are an expressive formalism to model (possibly) non-monotonic information exchange between heterogeneous knowledge bases. Such information exchange, however, often comes with unforseen side-effects leading to violation of constraints, making the system inconsistent, and thus unusable. Although there are many approaches to assess and repair a single inconsistent knowledge base, the heterogeneous nature of Multi-Context Systems poses problems which have not yet been addressed in a satisfying way: How to identify and explain a inconsistency that spreads over multiple knowledge bases with different logical formalisms (e.g., logic programs and ontologies)? What are the causes of inconsistency if inference/information exchange is non-monotonic (e.g., absent information as cause)? How to deal with inconsistency if access to knowledge bases is restricted (e.g., companies exchange information, but do not allow arbitrary modifications to their knowledge bases)? Many traditional approaches solely aim for a consistent system, but automatic removal of inconsistency is not always desireable. Therefore a human operator has to be supported in finding the erroneous parts contributing to the inconsistency. In my thesis those issues will be adressed mainly from a foundational perspective, while our research project also provides algorithms and prototype implementations.

1 Introduction

Multi-Context Systems (MCSs) are an expressive formalism for (possibly) non-monotonic knowledge exchange between heterogeneous knowledge sources. These sources are called contexts and formalized as abstract 'logics'. Information flow between contexts is specified using bridge rules which look and behave similar to rules in non-monotonic logic programming (cf. [15]):

$$(k : s) \leftarrow (c_1 : p_1), \ldots, (c_j : p_j), not(c_{j+1} : p_{j+1}), \ldots, not(c_m : p_m). \qquad (1)$$

Such a rule states that information s is added to context k if for $1 \leq i \leq j$ knowledge p_i is present in context c_i and for $j + 1 \leq i \leq m$ knowledge p_i is

* Supported by the Vienna Science and Technology Fund (WWTF), grant ICT08-020.

absent in c_i. Following common terminology $p_1, \ldots, p_m$ are called beliefs (each of their respective context) and s is the head formula of the bridge rule.

Consider a hospital where a database with patient records, a medical ontology, and an expert system shall be working together giving decision support on patient medications. The MCS framework is a good choice to realize this. Assume for patient Sue, the database knows that a) her X-Ray result indicates pneumonia, b) a certain blood marker is present, and c) she has no known allergies. The ontology imports information on X-Ray and blood tests using bridge rules

$$(C_{onto} : \mathit{xray}(Sue)) \leftarrow (C_{patients} : \mathit{labresult}(Sue, \mathit{xray})).$$
$$(C_{onto} : \mathit{marker}(Sue)) \leftarrow (C_{patients} : \mathit{labresult}(Sue, \mathit{marker})).$$

As the ontology contains the axiom $\mathit{xray} \sqcap \mathit{marker} \sqsubseteq \mathit{atyp\_pneu}$ it concludes that Sue has a atypical pneumonaia, severe kind of pneumonia. Finally, the expert system, a logic program containing rules $\mathit{give\_weak} \lor \mathit{give\_strong} : -\mathit{pneumonia}.$ and $\mathit{give\_strong} : -\mathit{atyp\_pneumonia}.$ suggests one out of two kinds of antibiotics if a patient has pneumonia. But it also respects potential allergies by the constraint $: -\mathit{give\_strong}, \mathit{not\ allowed\_strong}.$ As Sue has atypical pneumonia, only the strong antibiotic will help, so the logic program suggests this.

Now assume that Sue is allergic to strong antibiotics, a case that actually happens in the real world. Then the expert system can give no valid suggestion as strong antibiotics have to be given, but at the same time they are forbidden to be applied. This results in the whole system having no 'model' satisfying deductions of all knowledge bases and bridge rules. We call such an MCS inconsistent. [1]

By this example, we identify the following *open problems*:

– the inconsistency above is present due to tuples in the database, terminological assertions in the ontology, logic programming rules in the expert system and, a set of bridge rules establishing the information exchange. In what terms should the inconsistency be described and is there a uniform description irrespective of the specific formalisms used in contexts? Non-monotonicity of bridge rules and contexts is an additional challenge to such a description.
– Given such a description it is very likely that multiple ways exist to restore consistency. Removing some bridge rules would make the above example consistent, but also removal of tuples describing lab results. Similarly, addition of new bridge rules could resolve the inconsistency. If multiple options exist, which is the most preferred to restore consistency? Is it possible to do this in a heterogeneous way, i.e., can the designer of an MCS use a formalism of his own choice to specify his preference? Can such preference be given only for specific parts of an MCS and preference for other parts differently expressed?
– In the above example, the inconsistency can be dealt with locally, e.g., the expert system could switch to use paracoherent semantics and the MCS

[1] A complete formalisation of this example is available in [12].

becomes consistent. For MCSs with cyclic information flow, however, this might be impossible as cyclic information flow can be such that each context returns valid belief sets ("models"), but still for the overall system it does not fit together. How far does local inconsistency management help to resolve inconsistency, e.g., for MCSs with acyclic information flow?
– Besides inconsistency, is the MCS framework so versatile as to use other kinds of rules to connect contexts, e.g., SPARQL queries for information exchange?

As research on these topics has been started two years ago, the rest of this paper will briefly present results adressing above questions. Regarding research methodologies, we built analogies from existing techniques, e.g., Reiter's diagnosis. For algorithms we resorted to reductions to computational logic and meta-reasoning transformations, e.g., preference is handled in this way. Whenever possible, our invented methods are open so that legacy systems may be integrated to achieve certain tasks, e.g., local inconsistency management.

Contributions summary:

– we developed a uniform representation of inconsistency in terms of bridge rules. This representation leads a) to the notion of inconsistency explanation which separates different sources of inconsistency and points out those bridge rules creating inconsistency and b) to the notion of diagnosis which induce all possible repairs of an inconsistent MCS. Notably, both notions coincide on the overall set of bridge rules which are marked 'faulty.
– on top of those notions, we developed a transformation-based technique to allow meta-reasoning on diagnoses of an inconsistent MCS. This allows system designers to express preferences over diagnoses in a formalism of their own choice. The same techinque also allows to filter out undesired diagnoses.
– for local inconsistency management, a generalization of the MCS formalism was developed allowing to use existing methods of inconsistency management locally for a context. The introduced notion of a context manager allows to employ arbitrary knowledge management techniques locally at a context. It is important that the employed manager can change a knowledge base in a broad range and therefore it can also do other operations like view updates, belief revision, logic program updates, etc.
– for above notions the computational complexity also was analysed.
– to show the versatility of the ideas behind MCS, we also introduced a modified notion of MCS where knowledge exchange is specified using SPARQL queries.

Finally, we also implemented prototypes for evaluating MCSs and computing diagnoses and explanations of inconsistent MCSs.

The remainder of this paper is organized as follows: In Section 2 related work is discussed while Section 3 recapitulates the formal semantics of MCS and our basic notion for inconsistency diagnosis/explanation, it is followed by a short presentation of major achievements in the last two years in Section 4. Finally, Section 5 is an outlook on future work.

2 Related Work

With the seminal work of [19] and [16] the notion of context has been introduced to artificial intelligence and logic. In these works, a context is a regarded as a certain point of view in which formal reasoning takes place. The Trento school (cf. [17,22]) formalized and improved this understanding of context. It is notable, however, that those first frameworks consider homogeneous, monotonic logics for representing a context. With [9,21] non-monotonicity was introduced to Multi-Context Systems. Although default negation is added to bridge rules, contexts still are homogeneous or monotonic. Only with [7] the framework has been generalized for non-monotonic bridge rules and heterogeneous contexts. This finally allows to use arbitrary knowledge sources that are connected by (possibly) non-monotonic bridge rules. Our research is based on this notion of MCSs.

To deal with inconsistency, in [5] defeasible rules are introduced as a way of establishing information exchange in MCS. Defeasible rules are similar to bridge rules, but their semantics differs as a defeasible rule does not fire if it would cause an inconsistency by doing so. Several algorithms based on preference orders (or argumentation frameworks [4]) have been proposed. Inconsistency is resolved inherently, but no deeper inconsistency analysis is possible. For our hospital example this would mean that some information simply would not be passed along, e.g., forgetting the illness of Sue. Most of the proposed algorithms are based on provenance, which means that context internals have to be exhibited to other contexts. A company making profit by allowing third parties to use its knowledge base, however, will not risk its business by providing such information.

Aside from MCS, other areas deal with knowledge integration and its issues. Peer-to-Peer (p2p) systems [24,10] are similar as knowledge sources interchange pieces of information. Although the notion of a peer is very similar to a context in MCS, the essential feature of p2p systems is that peers may leave and join the system arbitrarily. Therefore research seeks to cope with inconsistency by isolating faulty contexts and simply ignore their information instead of analysing the inconsistency and aiming for a consistent system.

Information integration on the other hand deals extensively with issues like constraint violations that stem from the integration of several databases into a single one (cf. [6] for a survey on data fusion). Its main differences to MCS are that the result of data fusion is one single database which usually uses relational algebra for knowledge representation. MCSs, however, require inconsistency management for multiple, heterogeneous knowledge bases which are not restricted to a relational setting.

For many formalisms, methods of inconsistency handling have been invented, e.g., belief revision or possibilistic reasoning (e.g. [3]) for classical logic, paracoherent semantics for logic programs, etc. These methods can resolve inconsistency locally at a context (cf. Section 4), but they can not guarantee a consistent system. Also, most of those methods are only applicable to a specific formalism instead of a heterogeneous non-monotonic system.

3 MCS Preliminaries

Each context of an MCS is seen as a knowledge base built on an underlying logic. To capture different kinds of logics, this notion is general and not defined in the bottom-up style of inductive definitions for syntax and semantics. Instead, its approach is top-down, directly working with sets of well-formed formulas (wffs) and models (called belief sets). The semantics of a logic then only maps each set of wffs to a set of belief sets, i.e., the models of the wffs.

Formally, a logic $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$ consists, of the following components: 1) $\mathbf{KB}_L$ is the set of well-formed knowledge bases of L where each element of $\mathbf{KB}_L$ is a set (of formulas). 2) $\mathbf{BS}_L$ is the set of possible belief sets where we assume that each element of $\mathbf{BS}_L$ is a set (i.e.,a model containing all formulas that are considered true). 3) $\mathbf{ACC}_L : \mathbf{KB}_L \to 2^{\mathbf{BS}_L}$ is a function describing the semantics of L by assigning each knowledge base a set of acceptable belief sets. This concept of a logic captures many monotonic and non-monotonic logics, e.g., classical logic, description logics, modal, default, and autoepistemic logics, circumscription, and logic programs under the answer set semantics.

A *Multi-Context System* $M = (C_1, \ldots, C_n)$ is a collection of contexts $C_i = (L_i, kb_i, br_i)$, $1 \leq i \leq n$, where $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ is a logic, $kb_i \in \mathbf{KB}_i$ a knowledge base, and br_i is a set of bridge rules of form (1) over logics $(L_1, \ldots, L_n)$. Furthermore, for each bridge rule $r \in br_i$ its head formula s is compatible with C_i, i.e., for each $H \subseteq \{s \mid r \in br \text{ and } (i : s) \text{ is the head of } r\}$ holds $kb \cup H \in \mathbf{KB}_{L_i}$.

A belief state $S = (S_1, \ldots, S_n)$ of an MCS $M = (C_1, \ldots, C_n)$ is a belief set for every context, i.e., $S_i \in \mathbf{BS}_i$ for all $1 \leq i \leq n$. The semantics of MCS is defined in terms of equilibria, i.e., belief states that reproduce themselves under the application of bridge rules. Formally, let M be an MCS, C_i a context of M and $S = (S_1, \ldots, S_n)$ a belief state of M, then an bridge rule r of form (1) is applicable wrt. S, denoted by $S \models \mathit{body}(r)$, iff $p_\ell \in S_{c_\ell}$ for $1 \leq \ell \leq j$ and $p_\ell \notin S_{c_\ell}$ for $j < \ell \leq m$. Let $\mathit{app}_i(S) = \{\mathit{hd}(r) \mid r \in br_i \wedge S \models \mathit{body}(r)\}$ denote the heads of all applicable bridge rules of context C_i under S, then $S = (S_1, \ldots, S_n)$ is an *equilibrium* of M if and only if $S_i \in \mathbf{ACC}_i(\mathit{app}_i(S))$ for $1 \leq i \leq n$.

Basic Notions for Inconsistency Analysis (cf. [12]): We call an MCS M *inconsistent* iff no belief state of M is an equilibrium. To analyse and explain the inconsistency in an MCS, two notions have been developed: consistency-based diagnosis and entailment-based inconsistency explanation. Both notions use bridge rules to characterize 'faulty' information exchange. Intuitively, a diagnosis states how an inconsistent MCS can be changed to get a consistent system and an explanation shows what parts of the system create the inconsistency.

For an MCS M, br_M denotes the set of all bridge rules occuring in M, $M[R]$ denotes a modified MCS where all bridge rules of M are replaced by those of R, and $M \models \perp$ denotes that M is inconsistent. Given an MCS M, a *diagnosis* of M is a pair (D_1, D_2),$D_1, D_2 \subseteq br_M$, s.t. $M[br_M \setminus D_1 \cup \mathit{heads}(D_2)] \not\models \perp$. An *explanation* of M is a pair (E_1, E_2) of sets $E_1, E_2 \subseteq br_M$ of bridge rules s.t. for all (R_1, R_2) where $E_1 \subseteq R_1 \subseteq br_M$ and $R_2 \subseteq br_M \setminus E_2$, it holds that $M[R_1 \cup heads(R_2)] \models \bot$.

For a concise characterization, one usually focuses on subset-minimal diagnoses and explanations. The basic ideas behind both notions appear also in Reiter's seminal work on diagnosis [20]. Our diagnosis is similar to his notion and our explanation is similar to (minimal) inconsistent sets. For differences, we assume the source of inconsistency to be some faulty information exchange, so we only consider bridge rules, and because of the non-monotonic nature of MCSs, a bridge rule can be faulty by firing when it should not and also by not firing when it should. In classical diagnosis, only the former is relevant as monotonic logics only become inconsistent by that. The set of minimal diagnoses can also be seen as describing all minimal repairs, while the set of minimal explanations show hows inconsistency is caused in the system. The set E_2 in an explanation also shares some ideas with consistency restoring rules (cf. [2]) of logic programs.

4 Contributions: Methods of Inconsistency Management

This section presents contributions and answers the motivational questions raised in the introduction. These are the major published results of my graduate research. Note that authors are listed alphabetically for the respective publications.

Inconsistency Assessment: Having jointly developed and investigated, the basic notions for inconsistency analysis, the next step was developing methods to assess inconsistency qualitatively, i.e., filter diagnoses with undesired properties and select most preferred ones. In the spirit of MCS, we do not apply a specific formalism for preference or filters on diagnoses, but rather show how a transformation of the MCS and slight adaption of the notion of diagnosis is sufficient to achieve the desired effects in [13].

As one of the strengths of MCS is the ability to allow arbitrary formalisms for knowledge representation inside contexts, we do not want to restrict the users to a specific kind of representation of filters (or preferences). We therefore devised a meta-reasoning transformation which allows certain contexts to observe which diagnosis is applied to the MCS. The desired filter then is realized inside such an observer context (in a formalisms which is best suited for this task). So an MCS M is transformed into an MCS M_f where an additional observer context ob is added together with some additional bridge rules (details cf. [13]). As M_f contains all contexts and bridge rules of M, every diagnosis of M can also be applied to M_f. If ob detects an undesired diagnosis D', then ob simply becomes inconsistent, i.e., having no acceptable belief set. Therefore D' is no diagnosis of M_f, but all other diagnoses of M are diagnoses of M_f. This allows to compute all filtered diagnoses with the same algorithm as for computing subset-minimal diagnoses and it also allows to specify the filter in any desired formalism.

The meta-reasoning transformation also can be applied for multiple observation contexts where each observer only sees some bridge rules instead of all, thus preserving information hiding. As a similar meta-reasoning transformation can be used for comparison of diagnoses, it is possible to realize any given preference order on diagnoses and select the most preferred one. In general, however, this requires exponentially many more bridge rules in the transformed system, but for restricted classes of preference orders it is feasible.

Inconsistency management at the level of contexts: For many specific logics and knowledge formalisms, solutions to deal with inconsistency have been developed in the past, e.g., belief revision and paraconsistency for logics, paracoherent logic programming for logic programs, etc. For contexts using the underlying formalism it is desireable that MCSs also offer the same methods of inconsistency handling. Those methods, however, require to modify a knowledge base in more ways, than just the addition of formulas as bridge rules can do.

We therefore propose *managed Multi-Context Systems* (mMCS) in [8] where each context is equiped with a manager that can apply arbitrary changes to the context's knowledge base. Bridge rules in an mMCS are like those of MCS, but their head contains a unary command *op*, e.g., *revise*(s), *delete*(s), *add*(s), to apply the resp. operation on the formula s and the knowledge base of the context.

Managed MCS are a significant generalization of MCS as management functions can be used to realize a multitude of tasks: belief revision, view updates, updates of logic programs. To us, the most interesting is to ensure that contexts have a 'model' for any input. Such contexts are called *totally coherent*. Most notably even mMCS with totally coherent contexts cannot guarantee that the overall system has an equilibrium, but they ensure that inconsistency is only caused by odd-cyclic information flow. It directly follows that any acyclic mMCS with totally coherent contexts is consistent, thus proving local inconsistency management sufficient for acyclic MCS.

Beyond bridge rules: In [23] we introduce MCS where knowledge exchange is realised using SPARQL construct-queries. This is surprisingly simple and again shows the versatility of MCS. The resulting SPARQL-MCS framework is related to the MWeb approach [1], but our treatment of variables is different.

5 Future Work

As shown above, we were able to answer several foundational questions, give a uniform representation of inconsistency in heterogeneous MCSs, an open integration of preference-based inconsistency assessment, investigating the impact of local inconsistency handling, and making the MCS formalism capable of dealing with arbitrary changes to the knowledge bases of an MCS.

To evaluate the feasibility of the developed methods, we also aim for a reference application which is currently in the making: querying of a DNA database posing questions in (almost) natural language using an ontology and answer-set programs. Intital steps towards exchanging large amounts of information (cf. [14]) also showed that more specialised algorithms are needed.

Investigations whether approximation operators of [11] for logic programs can be translated to MCSs and transferring optimisations for abductive diagnosis (e.g.,[18]) to MCSs are also open tasks.

6 Acknowledgements

I am very grateful to my advisor Thomas Eiter, the principal investigator of our research project Michael Fink, and my colleague Peter Schüller who provided guidance, and helped with many fruitful discussions. Thank you.

References

1. Analyti, A., Antoniou, G., Damasio, C.V.: MWeb: A principled framework for modular web rule bases and its semantics. ACM Trans. Comput. Logic 12(2) (2011)
2. Balduccini, M., Gelfond, M.: Logic programs with consistency-restoring rules. In: International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series. pp. 9–18 (2003)
3. Benferhat, S., Lagrue, S., Yahi, S.: Bridging possibilistic conditional knowledge bases and partially ordered bases. In: JELIA. pp. 38–50 (2010)
4. Bikakis, A., Antoniou, G.: Contextual argumentation in ambient intelligence. In: LPNMR. pp. 30–43 (2009)
5. Bikakis, A., Antoniou, G., Hassapis, P.: Alternative strategies for conflict resolution in multi-context systems. In: AIAI. pp. 31–40 (2009)
6. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
7. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI. pp. 385–390 (2007)
8. Brewka, G., Eiter, T., Fink, M., Weinzierl, A.: Managed multi-context systems. In: IJCAI (2011), to appear.
9. Brewka, G., Roelofsen, F., Serafini, L.: Contextual default reasoning. In: IJCAI. pp. 268–273 (2007)
10. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency tolerance in p2p data integration: An epistemic logic approach. Inf. Syst. 33(4-5), 360–384 (2008)
11. Denecker, M., Marek, V.W., Truszczynski, M.: Ultimate approximation and its application in nonmonotonic knowledge representation systems. Inf. Comput. 192(1), 84–121 (2004)
12. Eiter, T., Fink, M., Schüller, P., Weinzierl, A.: Finding explanations of inconsistency in multi-context systems. In: KR (2010)
13. Eiter, T., Fink, M., Weinzierl, A.: Preference-based inconsistency assessment in multi-context systems. In: JELIA. pp. 143–155 (2010)
14. Fink, M., Ghionna, L., Weinzierl, A.: Relational information exchange and aggregation in multi-context systems. In: LPNMR. pp. 120–133 (2011)
15. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)
16. Giunchiglia, F.: Abstract contextual reasoning (1993)
17. Giunchiglia, F., Serafini, L.: Multilanguage hierarchical logics or: How we can do without modal logics. Artif. Intell. 65(1), 29–70 (1994)
18. de Kleer, J.: Focusing on probable diagnoses. In: AAAI. pp. 842–848 (1991)

Rule-based query answering method for a knowledge base of economic crimes

Jaroslaw Bak
Institute of Control and Information Engineering,
Poznan University of Technology,
M. Sklodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland
Jaroslaw.Bak@put.poznan.pl

Abstract. We present a description of the PhD thesis which aims to propose a rule-based query answering method for relational data. In this approach we use an additional knowledge which is represented as a set of rules and describes the source data at concept (ontological) level. Queries are posed in the terms of abstract level. We present two methods. The first one uses hybrid reasoning and the second one exploits only forward chaining. These two methods are demonstrated by the prototypical implementation of the system coupled with the Jess engine. Tests are performed on the knowledge base of the selected economic crimes: fraudulent disbursement and money laundering.

Keywords. Rule-based query answering, relational database access, Jess engine, economic crimes, SDL library

1 Introduction

Data stored in relational databases are described only by their schema (syntactic structure of data). Therefore, it is often difficult to pose a query at a higher level of abstraction than in a language of database relations and attributes. There is also a mismatching problem with table and column names without strictly defined semantics. A lack of a conceptual knowledge can be overcome by introducing ontologies which for evaluation purposes can be transformed into a set of rules. This kind of additional rule-based knowledge allows reasoning and query answering at an appropriate abstract level and relieves a user of using structural constructions from SQL. This kind of query evaluation is called a rule-based query answering method.

In our rule-based system we apply rules that are Horn clauses [8]. If there is conjunction of several predicates in the head, the rule can be easily transformed into Horn clauses with the Lloyd-Topor transformation [8].

We assume that only unary or binary predicates exist in our system, according to the terms that appear in OWL language [2] (since we decided to use this standard as a way to express conceptual knowledge).

Every rule consists of two parts: the left-hand-side, which is called the body, and the right-hand-side, which is called the head. Generally both parts are the sets of atoms that are interpreted conjunctively. In the body of the rule we use premises (patterns, conditions), which have to be satisfied by the appropriate atoms (facts) to allow the rule to be fired and produce conclusions from the rule's head. Next section describes the problem statement of the proposed PhD thesis. Section 3 presents current knowledge of the problem domain and existing solutions. Section 4 contains results achieved so far, the current state of the work and author's contributions. In Section 5 the concluding remarks are given.

2 Problem statement

The presented PhD thesis is trying to cope with the following research question:

How to efficiently query a relational database
at the conceptual level defined in a rule-based system?

This question is strongly connected with the following three main problems:

i. Rule-based query answering,
ii. The combination of a rule-based system and a relational database,
iii. The construction of the knowledge base (i.e. knowledge of economic crimes).

In a rule-based query answering method we assume that there exists a knowledge base which contains two parts: intensional and extensional. The intensional knowledge is represented as a set of rules and describes the source data at a conceptual (ontological) level. The extensional knowledge consists of facts that are stored in the relational database as well as facts that were derived in the reasoning process. Queries can be posed in the terms of the conceptual level. Thus, one gets an easier way to create a query than using structural constructions from SQL (Structured Query Language). The rule-based query answering method uses the reasoning process to obtain an answer for a given query. During this process facts from database are gathered and used to derive new facts according to a given set of rules. Next, the answer is constructed and presented.

In the first two problems (i, ii), we need to deal with the following questions:

1. What kind of rule-based system do we want to use?
2. How to express and represent the conceptual knowledge in the form of rules?
3. What is the language of the queries that can be evaluated by the system?
4. What kind of reasoning is involved in the rule-based query answering?
5. How to ensure the decidability of the query answering method?
6. How to combine a relational database with a rule-based system?
7. Which reasoning engine should be used for the prototypical implementation?
8. What are other potential applications of the proposed method and system?

We also assume that the rule-based query answering method will be used with the knowledge base of the selected economic crimes: fraudulent disbursement and money laundering. Particularly, we assume our system to be aimed at determining legal sanctions for crime perpetrators and to discover crime activities and roles (of particular types of owners, managers, directors and chairmen) using concepts, appropriate relations and rules.

The answers to the majority of the given questions and current achievements are presented in Section 4.

3 Overview of existing solutions

The presented problem, the rule-based query answering task [14], has many times been subject to research. Generally, there are two kinds of reasoning method applied in the rule-based query answering task. The first one is a backward chaining method, where reasoning is goal-driven. In this case our goal is the query posed to the system. This scheme of reasoning is implemented, for instance, in Prolog engine, and takes the form of the Selective Linear Definite clause resolution (SLD). In the backward reasoning technique facts are obtained only when they are needed in derivations.

On the contrary a forward chaining approach, which is data-driven, needs reasoning about all facts. In the working memory some of the inferred facts are useless and many rules are fired unnecessarily. It has a negative impact on the efficiency of the answering process. Moreover, because all facts should exist in the working memory, the scalability of reasoning task is poor due to the limited RAM memory. This drawback occurs also in the backward chaining.

The rule-based query answering task in rule-based systems, which exploits forward chaining is generally an inefficient method. The results of the OpenRuleBench initiative [1] show that efficiency of tabling Prolog and deductive database technologies surpasses the ones obtained from the corresponding pure rule-based forward chaining engines.

The most comprehensive approaches concerning optimizations of bottom-up query evaluation (in forward chaining) were given in [14, 15]. The general method relies on the transformation of a program *P* (set of rules) and a query *Q* into a new program, *magic(P ∪ Q)*, as shown in [15]. This *magic transformation* modifies each original rule by additional predicates to ensure that the rule will fire only when the values for these predicates are available. There were also other improvements and modifications of magic approach [14]. According to the work presented in [12] we also believe that the bottom-up approach has still room for improvements in order to increase the performance of the rule-based query answering task.

There exist also some works about the combination of rules (or logic programming) with relational databases. Notable are approaches presented in [18], [10] and [19] where ontology-based data access is performed with Prolog rules or Disjunctive Datalog.

The problem of applying rules in economic crimes is quite new. Most of the research work in the legal area relies on using ontologies in the field of information management and exchange [23, 24], not reasoning [16]. Other solutions, developed for instance in FFPoirot project [25, 26], concern descriptions of financial frauds, mainly the Nigerian letter fraud and fraudulent Internet investment pages. The ontologies developed in this project are not publicly available.

In our approach rules and queries are used to reflect data concerning documents and their attributes, formal hierarchy in a company, parameters of transactions, engaged people actions and their legal qualifications. To the best of our knowledge it is the first such approach in the field of economic crimes, besides the work presented in [17], which concerns cybercrimes.

4 Achievements and the current work

4.1 General assumptions

As mentioned in Section 2, the first two tasks include eight questions. For most of them, in the current state of our work, the answers are already known:

1. We wanted to use a production rule system because we need to apply our solutions in the real world applications.
2. We decided to express the conceptual knowledge with the Horn-SHIQ ontology combined with SWRL (Horn-like) [3] rules. We are aware of the restrictions that Horn-SHIQ imposes on ontology creation [21], but this fragment of OWL is sufficient for our needs.
3. Currently we assume conjunctive queries only, which are built of the terms from ontology (concepts and relations).
4. We developed two ways of applying reasoning process in the rule-based query answering task. In the first one [6] *hybrid* reasoning (forward and backward chaining) is used and in the second one only *forward* chaining and *extended* rules are executed. The second approach is still in progress.
5. We use the *Datalog Safety* [11] restriction in the rule-based query answering and *DL-safe* rules in ontology creation [22].
6. We developed the special mapping method which is presented later in this section and also in [6].
7. We decided to use the *Jess* (Java Expert System Shell) engine [4, 5], since it is one of the fastest commercial engines (with the free academic use) and it can be easily integrated with the Java language (which is the implementation language of our tool). The Jess engine also supports both forward and backward chaining.
8. We are convinced that our knowledge base of economic crimes [27, 30] would not be the only application of the defined system. Our methods are general and can be used in every application, which requires additional knowledge for query evaluation or need to offer an easier way of query creation than with the traditional SQL.

Our current results were presented in Polish and English papers [6, 27, 28, 29, 30].

4.2 Query answering with the hybrid reasoning

The approach described in [6] concerns the hybrid reasoning in the rule-based query answering task. In this work we described also the method of mapping between an ontology and a relational database. We presented our prototypical implementation of a library tool, the Semantic Data Library (SDL), which integrates the Jess engine, rules and ontology to effectively query a relational database.

In our hybrid reasoning process the backward chaining engine is responsible only for gathering data from a relational database. Data is added (asserted in Jess terminology) as triples into the working memory. The forward chaining engine can answer a query with all constraints put on variables in a given query (=, !=, <, > etc.).

The queries are constructed in Jess language in terms of ontology concepts. The mapping between the ontology classes and properties and the relational database schema is defined to fit syntactic structures and to preserve the semantics of the data.

Extensional data itself is stored in a relational database. The ontology and the mapping rules transformed into Jess language format provide the additional semantic layer to the relational database. Such an approach allows for answering queries to a relational database with a reasoning process performed in the Jess system over rules and ontology. The hybrid reasoning and query execution is supported by the SDL library. More details are given in [6].

4.3 Mapping between ontology terms and relational database

A mapping between ontology terms and relational data [6] is defined as a set of rules where each rule is of the following form:

SQL query => essential predicate

where "essential" means that the instance of the term cannot be derived from the rules. We assume that every "essential" ontology term has its appropriate SQL query and can be obtained only in a direct way, as a result of the SQL query. For example, in the following OWL hierarchy of classes *Mother is-a Woman is-a Person*, where the class *Mother* is a subclass of the class *Woman* etc., every instance of the class *Mother* is an "essential" term.

We assume that every SQL query has the following form:

SELECT [R] FROM [T] <WHERE> <C, AND, OR>

where:

- *R* denotes the result attributes (columns) – one or two according to the unary or binary terms (OWL Class, OWL DataProperty or OWL ObjectProperty),
- *T* stands for the tables, which are queried,
- *WHERE* is an optional clause to specify the constraints,
- *C* abbreviates the constraints in the following form: <attribute, comparator, value>, for example: *Age > 21*,
- *AND, OR* are the optional SQL commands.

As an example, let us assume that we have a table *Person* with the following attributes: *Id, Name, Age* and *Gender*. To obtain all adult men, we would define the following SQL query: *SELECT Id FROM Person WHERE Age>21 AND Gender='Male'*.

The mapping process requires defining SQL queries for all "essential" classes and properties. Other terms can be mapped too, but this is not necessary, since instances of them can be derived in the reasoning process.

4.4 Knowledge base of economic crimes

The approaches presented in [27, 28, 29] concern construction of the knowledge base of the selected economic crimes: fraudulent disbursement and money laundering. We analysed current related works and proposed the formal model of these economic crimes. We developed the ontology, which is the result of an analysis of about 10 crime cases. This means that the ontology is crafted to a task rather than attempting to describe the whole conceivable space of concepts and relations (top ontologies). The intensional part of the knowledge base contains also SWRL rules, which are very important when we want to determine legal sanctions for crime perpetrators and to discover crime activities and roles (not only to describe a crime scheme).

The methodology consists of several steps:

1. Design of a hierarchical data representation with 'minimal' ontology, which is used to uncover a crime scheme. This means using only necessary concepts that follow in the logical order of uncovering a crime. In the first stage goods/services transfer data is analyzed with relation to 3 basic flows: money, invoices, and documents (i.e., confirming that the service or goods have been delivered). In addition, responsible or relevant people within companies are associated with particular illegal activities.
2. Provision of a framework in which the graph building process and queries are executed.
3. Relating answers to queries with crime qualifications.

This approach is limited, but provides an essential model for evidence-building of a very important class of financial crimes: among them acting to do damage to a company and money laundering. Both crimes occurred in the example *Hydra Case* which was tested with the hybrid approach and artificially generated data. The work and results are presented in [30].

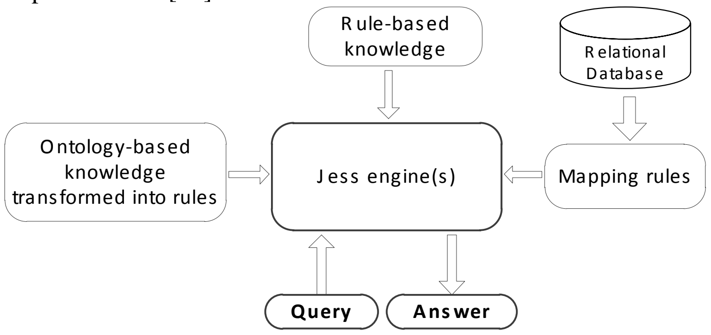


Figure 1. The architecture of the rule-based query answering system

4.5 Current work

In the current state of our work we are focused on the new rule-based query answering method which uses *extended* rules. *Extended* means that these rules are generated automatically from the basic ones for the evaluation purposes, and the modification is strongly connected with the magic transformation method. The extended rules are generated in the goal- and dependency-directed transformation. In this method we are interested in dependencies between variables appearing in predicates inside each rule.

The rule-based query answering method in this approach needs the different assumptions from the hybrid one because we use only one Jess engine to obtain relational data and answer a query. Obviously, we have to modify our query answering algorithm prepared for the hybrid system. This work is still in progress and results will be presented as soon as possible.

Figure 1 presents the architecture of our system which covers both solutions (hybrid reasoning and reasoning with extended rules).

5 Conclusions

In this paper we have outlined the content of the PhD thesis titled *Rule-based query answering method for a knowledge base of economic crimes*. Up to date we have obtained some achievements in the research, particularly related to the special crime-oriented ontological knowledge, its representation in rules of the Jess system, the connection with extensional data in a database and query answering by reasoning over the different data representations. We continue our research aiming to elaborate a new method of rules transformation, which will allow for more efficient application of rules in query answering task. We have to manage with problems presented in Section 2 and to provide a precise, clear and formal description of our solutions. We have already obtained positive results of tests performed on the prototype system but we also plan to execute queries prepared by the OpenRuleBench initiative. The comparison of our results and those obtained in a pure Jess system seems to be an adequate and objective assessment of usefulness of our work.

References

1. Liang S., Fodor P., Wan H., Kifer M., OpenRuleBench: An Analysis of the Performance of Rule Engines, Proceedings of the 18th international conference on World wide web, ACM, 2009, o. 601-610.
2. McGuinness D., van Harmelen, F.: Owl web ontology language overview. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features/
3. Horrocks I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member Submission (May 21 2004), http://www.w3.org/Submission/SWRL/
4. Jess (Java Expert System Shell), http://jessrules.com/
5. Friedman-Hill, E.: Jess in Action, Manning Publications Co. (2003)
6. Bak, J., Jedrzejek, C., Falkowski, M.: Usage of the Jess engine, rules and ontology to query a relational database. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 216–230. Springer, Heidelberg (2009)
7. Forgy C., Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19, pp. 17-37, 1982.
8. Lloyd J.W., Foundations of logic programming (second, extended edition). Springer series in symbolic computation. Springer-Verlag, New York, 1987.
9. Horridge M., Bechhofer S. The OWL API: A Java API for Working with OWL 2 Ontologies. OWLED 2009, 6th OWL Experienced and Directions Workshop, Chantilly, Virginia, October 2009.
10. Poggi A., Lembo D., Calvanese D., De Giacomo G., Lenzerini M., Rosati R., Linking Data to Ontologies, Journal on Data Semantics, Volume 10, 2008, p. 133-173.

11. Gallaire H., Minker J. (Eds.): Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, 1977. Advances in Data Base Theory, Plenum Press, New York, 1978
12. Brass, S., Implementation Alternatives for Bottom-Up Evaluation, 26th International Conference on Logic Programming, ICLP (Technical Communications), Edinburgh 2010, 44-54.
13. Liang S., Fodor P., Wan H., Kifer M., OpenRuleBench: Detailed Report, May 29, 2009, http://projects.semwebcentral.org/docman/view.php/158/69/report.pdf.
14. F. Bry, N. Eisinger, T. Eiter, T. Furche, G. Gottlob, C. Ley, B. Linse, R. Pichler, and F. Wei. Foundations of Rule-Based Query Answering. In Proceedings of Summer School Reasoning Web 2007, Dresden, Germany (3rd–7th September 2007), volume 4634 of LNCS, pages 1–153. REWERSE, 2007.
15. Beeri C., Ramakrishnan R., On the power of magic, Journal of Logic Programming, v.10 n.3-4, p.255-299, April/May 1991.
16. Breuker, J. (2009). Dreams, awakenings and paradoxes of ontologies, invited talk presentation, 3rd Workshop on Legal Ontologies and Artificial Intelligence Techniques, http://ontobra.comp.ime.eb.br/apresentacoes/keynoteontobra-2009.ppt.
17. Bezzazi, H. (2007). Building an ontology that helps identify articles that apply to a cybercrime case, Proceedings of the Second International Conference on Software and Data Technologies, ICSOFT 2007, Barcelona, Spain, pp. 179–185.
18. Lukácsy G., Szeredi P., Scalable Web Reasoning Using Logic Programming Techniques. In A. Polleres and T. Swift, editors, Proceedings of the Third International Conference on Web Reasoning and Rule Systems, RR 2009, Chantilly, VA, USA, October 25-26, 2009, volume 5837 of Lecture Notes in Computer Science, pages 102{117. Springer, 2009.
19. U. Hustadt, B. Motik, U. Sattler. Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. Proc. of the 9th International Conference on Knowledge Representation and Reasoning (KR2004), June 2004, Whistler, Canada, pp. 152-162.
20. Hustand U., Motik B., Sattler U., Data Complexity in Very Expressive Description Logics, In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005), 2005.
21. Horn-SHIQ restrictions: http://www.w3.org/2007/OWL/wiki/Tractable_Fragments#5
22. Motik B., Sattler U., and Studer R., Query answering for owl-dl with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 3(1):41–60, 2005.
23. Biasiotti, M., Francesconi, E., Palmirani, M., Sartor, G. and Vitali, F. (2008). Legal Informatics and Management of Legislative Documents. Global Centre for ICT in Parliament, Working Paper No. 2, United Nations, Department of Economics and Social Affairs.
24. Casellas, N. (2008). Modelling Legal Knowledge through Ontologies. OPJK: the Ontology of Professional Judicial Knowledge, PhD thesis, Universitat Autónoma de Barcelona, Barcelona.
25. Kerremans, K. and Zhao, G. (2005). Topical ontology of VAT, Technical Report of the FFPOIROT IP project (IST-2001- 38248). Deliverable D2.3 (WP 2), STARLab VUB.
26. Zhao, G. and Leary, R. (2005). AKEM: an ontology engineering methodology in FFPoirot, Technical Report of the FFPOIROT IP project (IST-2001-38248). Deliverable D6.8 (WP 6), STARLab VUB.
27. Bak, J., Jedrzejek, C., Falkowski, M.: Application of an Ontology-Based and Rule-Based Model to Selected Economic Crimes: Fraudulent Disbursement and Money Laundering, Lecture Notes in Computer Science, 2010, Volume 6403/2010, 210-224
28. Bak J., Jedrzejek C., Application of an Ontology-based Model to a Selected Fraudulent Disbursement Economic Crime. In P. Casanovas, U. Pagallo, G. Ajani, and G. Sartor, editors, AI approaches to the complexity of legal systems, LNAI, Vol 6237, 2010
29. Jedrzejek C., Cybulka J., Bak J., Towards Ontology of Fraudulent Disbursement, Agent and Multi-Agent Systems: Technology and Applications, Proceedings of 5th International Conference, LNCS 2011, to be published.
30. Bak J., Jedrzejek C., Falkowski M., Application of the SDL Library to Reveal Legal Sanctions for Crime Perpetrators in Selected Economic Crimes: Fraudulent Disbursement and Money Laundering, In Proceedings of the 4th International RuleML-2010 Challenge, Washington, DC, USA, October, 21-23, 2010. Edited by: Palmirani M., Omair Shafiq M., Francesconi E., Vitali F., Volume 649.

Poster Session Papers

Semantic-ontological combination of Business Rules and Business Processes in IT Service Management

Alexander Sellner[1], Christopher Schwarz[1], Erwin Zinser[1]

[1]FH JOANNEUM University of Applied Sciences, Department of Information Management
Alte Poststrasse 147, 8020 Graz, Austria
{Alexander.Sellner|Christopher.Schwarz|Erwin.Zinser}
@fh-joanneum.at

Abstract. IT Service Management deals with managing a broad range of items related to complex system environments. As there is both, a close connection to business interests and IT infrastructure, the application of semantic expressions which are seamlessly integrated within applications for managing ITSM environments, can help to improve transparency and profitability. This paper focuses on the challenges regarding the integration of semantics and ontologies within ITSM environments. It will describe the paradigm of relationships and inheritance within complex service trees and will present an approach of ontologically expressing them. Furthermore, the application of SBVR-based rules as executable SQL triggers will be discussed. Finally, the broad range of topics for further research, derived from the findings, will be presented.

Keywords: Semantic IT Service Management, SBVR-based SQL statements, Ontological ITSM service trees

1 Introduction

IT Service Management (ITSM) can be seen as a large and complex environment for business processes and rules with a certain potential for automation. On the one hand, ITSM puts a strong focus on providing tools for managing business topics such as outsourcing costs, licensing fees and negotiated prices within Service Level Agreements (SLAs), on the other hand the goal is to provide clearly defined processes for managing IT resources, which are, for instance defined in the ITIL framework [1].

Having a look at ITSM from large IT service providers' perspectives, there is an tremendous amount of so-called configuration items which are combined together to services, which are ultimately sold to clients. Due to changes in system environment or technological development, these services are in a constant state of change, turning the task of outsourced service provision to a rather stable price to a quite difficult challenge.

Most ITSM activities affecting existing relationships between IT service providers and their clients are triggered by so called "requests for changes" (RFCs). RFCs usually add, remove, or change existing services e.g. upsize the RAM of a server or add an SLA to a database application. Almost every ITSM item will have a certain dependency to another ITSM item and, depending on the complexity of the services, these relationships will lead to the formation of a so-called service tree. The service tree can be seen as one or more graphs, as there is no single parent node and items, such as SLAs can exist multiple times.

A well-suited approach for making such graphs human-readable is to make use of ontologies using semantic expressions [2]. By applying ontologies, it becomes possible to create graphical representations of the complex service trees, making it possible to discover all dependencies and to keep them well-managed. A further advantage becomes obvious, having a look at natural-language based semantic expressions used by ontologies. Besides adding further advances to the comprehensibility and integrity of service trees, this also creates the possibility of performing commonly understandable modifications of service relationships and therefore can help to create a mutual understanding between ITSM service providers and their customers [3].

Because of the strong business connection and great involvement of rules within SLAs, the idea of establishing a rule repository for service level definitions which are based on natural language, seems obvious. Over the last years, several standards for such definitions have been created such as RuleSpeak[1], R2ML[2] and Semantics of Business Vocabulary and Business Rules (SBVR)[3]. Most recently, it seems that there is a strong support for the SBVR standard both within the academic community as well as the industry.

This paper focuses on the challenge of displaying complex graphs of service relations to human-readable ontologies, based on semantic models. Furthermore the paper will discuss on the special topic of enriching these complex service trees with SLAs assembled by SBVR-based business rules. Another topic being discussed will be the establishment of SBVR-based business rules using DBMS triggers for execution [4].

The paper is structured as follows: Section 2 will have a closer look at service trees, discuss the paradigm of service inheritance and present ways of ontologically displaying existing service structures. Section 3 will explain the main categories of SLA rules within service trees and present samples for converting SBVR definitions to database-executable statements. Section 4 will discuss topics for further research within the presented areas.

[1] www.rulespeak.com/
[2] http://rewerse.net/
[3] http://www.omg.org/spec/SBVR/1.0/

2 Establishing ontological views of service trees based on existing semantic definitions

As already briefly discussed in the previous section, the structure of service trees in real-world business environments can be quite complex and is therefore hard to handle and maintain by people carrying out various tasks within ITSM processes. The following example will provide closer insights into the process of establishing RFCs as well as the related legacy data and the necessary procedures to involve ontological views which allow establishing a well-structured perspective on the configuration items' relationships and corresponding inherited attributes.

Within the given case, which is based on a project carried out for a large ITSM provider, the following types of configuration items (CIs) can be identified:

- Service: A service is a collection of ITSM assets with a business-relevant impact. This means that they are the elementary constructs within contracts between service providers and customers. Having a look at the function of services within the service tree, they can also be used as logical containers for putting together various "low-level" services to more sophisticated "high-level" services. Therefore, a low level service can for instance be a domain name resolution service and a high level service could be a billing application. As a matter of fact, services can exist multiple times within the service tree.
- Host: Hosts are the fundamental basis for establishing services. As a result, a service can be put together by one or more hosts. Because of a strong advance in virtualization technology over the recent years, hosts don't necessarily need to be actual hardware but multiple hosts could also depend on a host-service relationship providing the actual hardware for virtualization as well as on an additional host-service relationship for data storage. The same host can be used by multiple services and therefore can appear multiple times within the service tree.
- Service Level Agreement: SLAs include legal implications regarding the quality of services. For instance, the loss of an email service for a certain time could have a significant business impact for organizations. As a result, SLAs also handle penalty fees which need to be paid under certain circumstances. This can, for instance, be as soon as a service becomes unavailable (first failure fines and concurrent failure fines) or if a certain value for the availability of a service over a certain period is not met (availability percentage per day, month or year). SLAs follow the principle of inheritance and are therefore valid for all services or hosts which are hierarchically subordinate within the service tree. In addition, SLAs can appear multiple times within the service tree and they need to be prioritized in order to prevent discrepancies by allowing only one valid SLA per host or service.
- Maintenance Contract (MTC): MTCs contain contractually defined regulations regarding repair and renewal tasks on hardware. This can for instance be the replacement of a defect part or the regular upgrade to newer hardware parts. MTCs can be closed for services or hosts have the same characteristics as SLAs regarding inheritance. In contrast to SLAs, there can be multiple MTCs valid at the same time for one service or host which leads to accumulated maintenance liabilities.

Besides these CIs, there can be a wide range of additional CIs involved in real-world ITSM processes [5]. The above-mentioned items will serve to establish a simplified demonstration of a sample ITSM service tree which already contains a great variety of characteristics worth to undergo closer investigation. The sample service construct is demonstrated in Figure 1.

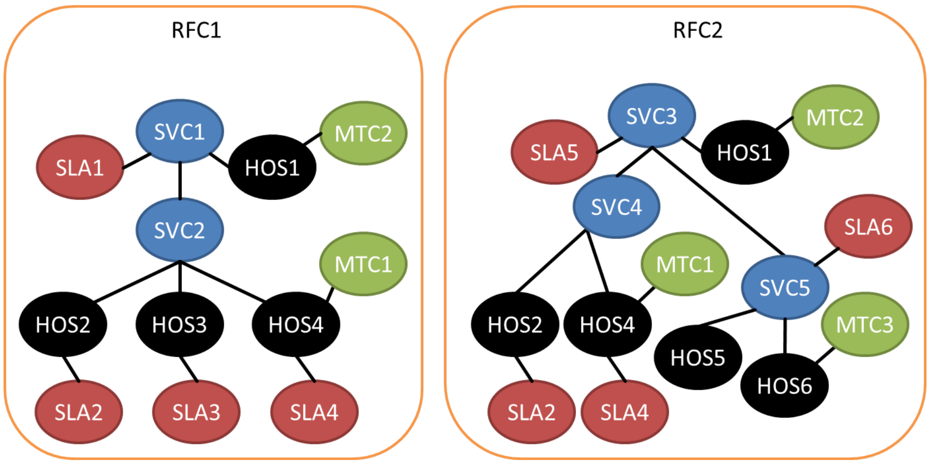


Figure 1. Sample ITSM service construct

The sample demonstrates two RFCs with their respective hierarchically subordinate services, hosts, SLAs, and MTCs. RFC1 might, for instance, be a database service hosted on a virtual machine (HOS1) with HOS2, HOS3 and HOS4 being redundant database servers, each having different SLAs. RFC2 could be a Web-based application consisting of SVC4 as database backend and SVC5 as Web servers, whereas SVC4 consists of two machines (HOS2 and HOS4) already being used by RFC1.

Considering the characteristic of inheritance for SLAs and MTCs within service trees, the example clearly illustrates why more complex service trees are almost impossible to manage and handle from a business perspective.

Having a look at the SLAs in RFC1, it clearly needs to be determined whether SLA1 or one of the respective SLAs for HOS2, HOS3 and HOS4 is valid. This is a prime example where priorities need to be assigned to SLAs in order to ensure legal correctness and optimized business benefit. Such a prioritization even needs to be put in place across boundaries of RFCs. For instance within RFC2 it also needs to be determined whether SLA4 or SLA2 and SLA5 respectively are valid for HOS2 and HOS4. Changing priority of SLA2 or SLA4 would again have an impact on the overall situation of RFC1

When focusing closer on the question of how prioritization between two SLAs needs to be done, it becomes clear that sometimes it is quite difficult to say which SLA is preferable to be applied . Assuming that within RFC1, the fines for first failure are higher in SLA1 than in SLA2 and the availability fines are higher in SLA2 than in

SLA1, it is impossible to instantly determine the most cost effective solution. It is rather imaginable that the availability of service or hosts is monitored over a certain period leading to statistical forecast regarding its behavior. Only then it would be possible to pick the optimal SLA, taking all types of fines into account.

In contrast, the challenge regarding MTCs is rather on deciding whether individual MTCs should be discharged because of redundancies or existing MTCs should be extended. As a result, the accumulated MTC benefits should be considered for every individual host or service.

3 Invoking semantics within ITSM service trees

To overcome the challenge of managing complex relations within ITSM service trees, it seems quite obvious to introduce ontologies and semantic expressions. This makes it possible to display dependencies between configuration items in a human-readable way and supports extending or rearranging the service tree [6]. An example of such an ontological view taken from the prototype of an ITSM application of a large IT service provider is given in Figure 2. The figure displays an RFC which is linked to an SLA, an MTC, a hardware asset and a high level infrastructure service which is then linked to a low level infrastructure service. The figure also reveals the concept of service instances used to express configuration items which appear multiple times within a service tree.

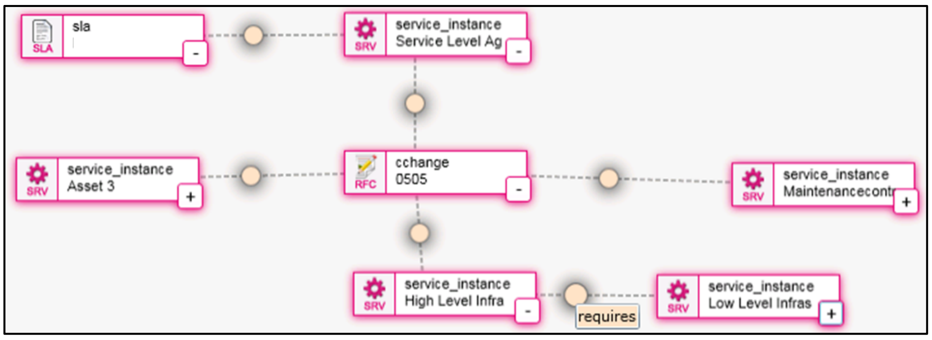


Figure 2. Ontological view of ITSM service tree

Integrating semantic expressions, adds a further level of complexity regarding the underlying data structure [7]. This can lead to quite a heavy integration effort regarding database design and operations performed to read, write or update data.

On the other hand, the advantages for integrating semantic expressions are obvious. It becomes possible to create human-understandable illustrations of service trees based on natural language. Additionally, rules can be applied on the service structure based on the existing semantic expressions. Consequently, these rules can contribute to handle the situation regarding SLA and MTC inheritance within the service tree.

4 Integrating SBVR-based rules using DBMS triggers for execution

In general, three categories of rules can be identified within ITSM service trees:

1. Rules to prevent adding new SLAs or MTCs due to inconsistencies or disadvantageous business impacts
2. Rules to analyze and improve existing SLA and MTC structure
3. Rules within SLAs themselves

There are various ways following the goal of achieving a loosely coupled execution of these SLA rules within a repository on the specified service tree. A quite feasible approach is to aim for rule execution within Database Management Systems (DBMS) using SQL triggers. This approach leverages characteristics of DBMS to establish as rule engines allowing the execution of Event-Condition-Action (ECA) rules [8].

Recent findings dealing with the conversion of SBVR definitions to SQL statements were taken as basis for developing a prototype making it possible to express the first category of rules for SLAs in structured English and carry out the respective database operations to put them in action [9,10].

Listing 1 gives an example for the conversion of the first category of rules from structured English to an ECA rule which is placed on the DBMS.

Structured English statement:

```
T:SLA
T:SVC
T:total fines
F: SLA has total fines
F:SLA is linked to SVC
NR: For an SLA that is linked to an SVC it is obligatory that
the total fines of the new SLA are less than the total fines
of the old SLA.
```

SQL expression:

```
CREATE TRIGGER "NR1" BEFORE UPDATE OF “SLA_id”
  ON "SLA-is_linked_to-SVC"
WHEN NOT
(SELECT "total fines" from "SLA" where id=new.SLA_id)<
(SELECT "total fines" from "SLA" where id=old.SLA_id)
  BEGIN
    SELECT RAISE(ABORT, ‘Requirement of NR1 not met’);
  END;
```

Listing 1. Conversion from structured English to DBMS trigger statement

The terms “SLA” and “SVC” and “total fines” are expressed as lines starting with “T”. The fact types “SLA has total fines” and “SLA is linked to SVC” are marked by a preceding “T”. Finally, the normative rule is denoted by “NR”.

Regarding the definition of fact types, the identification of table attributes takes place using conjugations of the predicate “have” and a table relationship is established by the predicate “being linked to”.

By choosing “NR” as markup, the creation of the SQL trigger is either based on a fact type involving a relationship or on a table related to a single term. The identification is based on the phrase being placed in front of the rule. If a fact type is identified as input, the first term used to describe the identity column for the SQL trigger.

Based on the fact type definition for the attribute “total fines”, the reverse order “total fines of the new SLA” forms that basis for the SELECT statement (SELECT "total fines" from "SLA" where id=new.SLA_id), whereas the definitions “new” and “old”, are explicitly required because of the targeted syntax of the SQL trigger.

Because of the fact that the trigger should not abort when the defined rule is met, the WHEN clause needs to be followed by NOT in order to display an error message in any other cases.

5 Conclusions and Future Work

The approach presented, shows the advantages and challenges regarding the application of ontologies and semantic expressions. Findings derived, led to conclusions regarding prerequisites which need to be fulfilled, especially when aiming for a close connection to underlying data structure.

Adding semantic expressions to ITSM service trees can increase presentability and manageability. On the other hand, such an integration also implies a challenge towards the underlying data, because of the paradigm of inherited SLAs and MTCs within the service tree.

The application of SBVR based business rules adds further needs which must be met by database models. In fact, the database design needs to be carried out in accordance with naming conventions established within SQL statements converted from SBVR.

Having a look at the originating statements defined in structured English, a strongly controlled natural language needs to be applied in order to enable a conversion to SQL definitions. The approach presented, focused on SBVR-based rules for preventing negative business impacts in connection with the addition of new SLAs. It needs to be investigated how these statements need to be verbalized for MTCs. It will also be necessary to look at the conversion to stored procedures, which would allow apply rules not only on a data manipulation level.

Further research will also be necessary to mathematically investigate paradigm of inheritance within ITSM service trees. Most likely, this will be closely related to graph theory and provide further insights how service trees can be optimized.

Another area worth performing research on will be investigating the application of upper level ontologies within the specific scenario. This might lead to an enterprise ontology for combining business processes and rules or to an ontology being closely focused on the area of ITSM.

References

1. Addy, R.: Effective IT Service Management: To ITIL and Beyond! Springer, Berlin (2007)
2. Guarino, N., Giaretta, P.: Ontologies and Knowledge Bases - Towards a Terminological Clarification. Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (1995)
3. Baioco, G., Costa, A., Calvi, C., Garcia, A.: IT service management and governance modeling an ITSM Configuration process: A foundational ontology approach. Integrated Network Management-Workshops (2009)
4. Vasilecas, O., Bugaite, D.: Ontology-Based Information Systems Development: The Problem of Automation of Information Processing Rules. ADVIS 2006, Proceedings (2006)
5. Ayachitula, N., Buco, M., Diao, Y., Surendra, M., Pavuluri, R., Shwartz, L., Ward, C.: IT service management automation - A hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows. Proceedings of IEEE SCC'2007 (2007)
6. Betz, C.: Architecture and Patterns for IT Service Management, Resource Planning, and Governance: Making Shoes for the Cobbler's Children. Morgan Kaufmann (2006)
7. Storey, V.: Understanding and Representing Relationship Semantics in Database Design. Lecture Notes in Computer Science (2001)
8. Paschke, A.: ECA-RuleML: An Approach combining ECA Rules with temporal interval-based KR Event/Action Logics and Transactional Update Logics. Internet-based Information Systems (2005)
9. Moschoyiannis, , Marinos, A., Krause, P.: Generating SQL queries from SBVR rules. RuleML'10 Proceedings of the 2010 international conference on Semantic web rules (2010)
10. Marinos, A., Moschoyiannis, S., Krause, P.: SBVR to SQL Compiler. Proceedings of the RuleML-2010 Challenge (2010)

Part II
RuleML2011@BRF Challenge

The SDL Library: Querying a Relational Database with an Ontology, Rules and the Jess Engine

Jaroslaw Bak, Maciej Falkowski and Czeslaw Jedrzejek
Institute of Control and Information Engineering,
Poznan University of Technology,
M. Sklodowskiej-Curie Sqr. 5, 60-965 Poznan, Poland
{firstname.lastname}@put.poznan.pl

Abstract. In this demo we present the Semantic Data Library (SDL) which is used to query a relational database at a concept (ontological) level. The SDL integrates a rule engine, a relational database and a set of rules obtained from the transformation of an OWL ontology. This combination allows querying and inferring with data stored in a relational database using concepts, roles and rules. We propose an implementation of the method of querying relational database with extended rules and the transformation of OWL ontologies into sets of rules. Our demonstration is based on the previously presented financial crime 'minimal model' ontology and artificially generated data sets. Prospects of the future development of the SDL tool are presented.

Keywords. SDL library, Jess engine, rules, OWL ontology, query answering

1 Introduction

The most of data processed in modern applications come from relational databases. Such data is described only by their schema (a structure of data). Without strictly defined semantics there is often a mismatching problem with table and column names in databases. Moreover, it is rather difficult to query data at a more abstract level than only in a language of database relations and attributes. A lack of conceptual knowledge can be overcome by introducing ontologies. For the evaluation purposes, an ontology (and other knowledge) can be transformed into a set of rules (however, several of the OWL axioms cannot be transformed [1]). The additional rule-based knowledge allows reasoning and query answering at an appropriate abstract layer. Moreover, it simplifies posing a question than using structural constructions from SQL. This kind of query evaluation is called the rule-based query answering method.

As a result a user gets an easy way to query a relational database and both a query and an answer are based on the semantics defined in an OWL [2] ontology. The ontology describes data at the concept (ontological) level and introduces a formal definition of concepts and roles which do not exist directly in the database. For example, let us assume that we have a table *persons(id, fatherID, motherID, gender)*. In the corresponding OWL ontology we can define the following concepts:

Grandfather, *Grandmother*, *Cousin* etc. and roles: *hasBrother*, *hasSister*, *hasCousin* etc. These concepts/roles are not defined directly in the database. But with the use of the OWL ontology and SWRL [3] rules we can obtain instances of the above-mentioned terms. Moreover, we can use these terms in queries which are in the form of directed graphs.

In this paper we present a prototypical implementation of the Semantic Data Library (SDL) tool which integrates an OWL ontology, SWRL rules, the Jess [4] reasoning engine and a relational database. Our integration allows to pose a query to a relational database at concept (ontological) level. We assume that OWL ontology which is handled by the SDL can contain both OWL axioms and SWRL rules.

During the development and research process, we have proposed and implemented two methods of querying relational database: hybrid reasoning [5] and forward reasoning with extended rules [6]. In this work, we are focused on the implementation and the evaluation of the latter method. The paper makes the following contributions:

- We present the SDL library in details: characterizing the functionalities and the OWL to Jess transformation methods,
- We evaluate our 'minimal model' ontology with all our approaches achieved so far,
- We show that our approaches increase the scalability of the Jess engine and outperforms its rule-based query answering method.

The paper is organized as follows. Section 2 presents the SDL architecture and functionalities. Section 3 describes an example evaluation and application of the SDL tool to the previous constructed 'minimal model' ontology. Section 4 contains concluding remarks and future work plans.

2 SDL Architecture and Features

2.1 SDL Overview and Architecture

SDL integrates ontologies, relational data and rules which represent domain knowledge. We need such tool when we have to pose complicated queries to the standard relational database. Due to the formally defined semantics (OWL) we can pose a semantic query and get a corresponding semantic answer. The SDL generates rules automatically which is very important for knowledge bases that often change.

The architecture of this system is presented in Figure 1. The central part, which gathers input from other system elements and processes rules, are one [6] or two [5] Jess engines used for forward and backward chaining. The hybrid approach [5] exploits both forward and backward reasoning. The backward method is responsible for gathering data from a relational database and the forward chaining is used to answer a given query. One instance of the Jess engine is created for each reasoning method. It means that we use two instances of the Jess engine in the hybrid approach.

In the extended rules [6] approach we use one instance of the Jess engine, because only the forward reasoning method is used. *Extended* means that these rules are generated automatically from the basic ones for the evaluation purposes, and the modification is strongly connected with the magic transformation [7] method. The set of basic rules consists of rules which constitute the knowledge base. The rule-based knowledge base comes from an OWL to Jess transformation. The set of extended rules is semantically equivalent to the set of basic rules. The extended rules are generated in the goal- and dependency-directed transformation. In this method we are interested in dependencies between variables appearing in predicates inside each rule. Together with the mapping rules, the extended ones are used in the rule-based query answering algorithm.

The rule-based query answering method in this approach needs the different assumptions from the hybrid one because we use only one Jess engine to obtain relational data and answer a query. Obviously, we modified our query answering algorithm prepared for the hybrid system. More theoretical information can be found in [6].

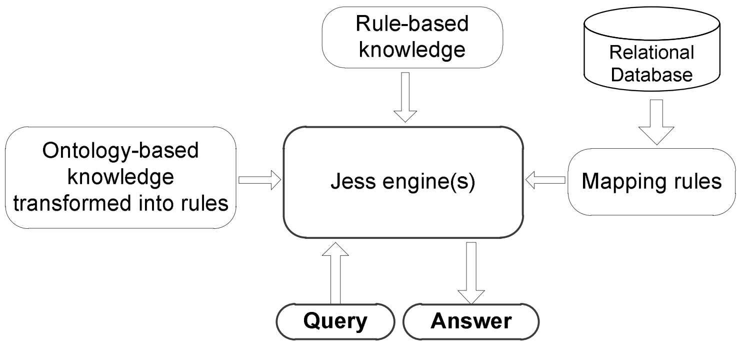


Figure 1. The architecture of the rule-based query answering system

Figure 2 presents the integration scheme of an OWL ontology with SWRL rules, the Jess engine and a relational database. We assume that the ontology is in the Horn-SHIQ language and contains SWRL rules (Horn-like clauses). Such OWL+SWRL ontology is transformed into a set of rules in the Jess language. The set of rules is stored as a Jess script file (*.clp). The script is then transformed into a set of extended rules (*ExRScript.clp*). A user can load: *ExRScript.clp* and a mapping rules Jess script; then establish a database connection and pose queries with SDL and Jess. It is worth noting that such a transformation needs to be done only once (besides changes of the OWL ontology, SWRL rules or the database schema).

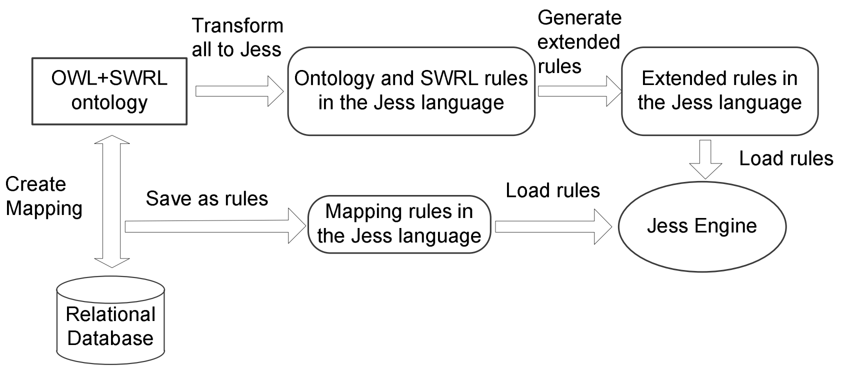


Figure 2. The integration scheme executed in the SDL library

2.2 SDL Features

The SDL tool is implemented in Java language. It is split into two modules:

- SDL-API (Application Programming Interface), which provides all functions,
- SDL-GUI (Graphical User Interface), which exploits SDL-API functions for defining the mapping between ontology terms and relational data; and provides automatic transformation of ontology into rules and the generation of Jess scripts.

The SDL-API module provides the following functionalities:

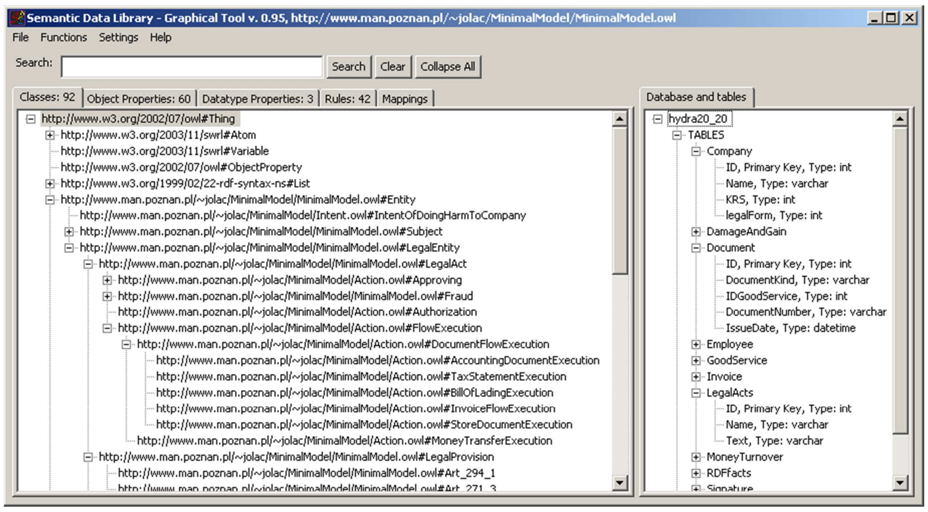
- reading a relational database schema,
- executing SQL query or procedure (results are added into Jess engine as facts),
- reading OWL ontology and Jess scripts,
- Jess scripts generation (forward and backward chaining, extended rules, Horn-SHIQ transformation) from OWL ontology,
- mapping between ontology concepts/roles and relational data,
- executing a Jess query which consists of the concepts and roles from OWL ontology or templates defined in Jess language,
- rule-based query answering methods: hybrid and extended rules,
- Jess engine reasoning management (in forward and backward chaining).

Due to SDL-GUI module the library enables executing the following functions:

- reading ontology and viewing of concepts/roles hierarchies; the view contains classes hierarchy, object properties hierarchy and datatype properties hierarchy. These hierarchies are calculated by the Pellet engine [8],
- viewing a relational database schema which contains tables, views, columns and data types,
- mapping between ontology concepts/roles and relational data,
- populating an ontology with data from a relational database according to the specified mapping,
- creating Jess facts from a relational database according to the specified mapping,
- transforming OWL ontologies to Jess scripts,
- transforming Jess scripts into Jess scripts with extended rules (only *triple* template of facts is currently supported).

SDL supports interaction with the Pellet engine (for TBox reasoning with ontology and its classification), exploits OWL API [9] (for handling OWL files) and uses JDBC library for MS SQL 2008 Server access. The taxonomies of ontology classes and properties are classified by SDL-GUI with Pellet 2.3.0 and prepared for a user, who can define SQL mapping queries on these calculated taxonomies.

Figure 3 presents our minimal model ontology loaded into SDL-GUI and established connection to the corresponding relational database. A user gets a presentation of tables and views which exist in a database.



```
⊟ http://www.w3.org/2002/07/owl#Thing
   ⊞ http://www.w3.org/2003/11/swrl#Atom
     http://www.w3.org/2003/11/swrl#Variable
     http://www.w3.org/2002/07/owl#ObjectProperty
   ⊞ http://www.w3.org/1999/02/22-rdf-syntax-ns#List
   ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#Entity
       http://www.man.poznan.pl/~jolac/MinimalModel/Intent.owl#IntentOfDoingHarmToCompany
     ⊞ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#Subject
     ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#LegalEntity
       ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#LegalAct
         ⊞ http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#Approving
         ⊞ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#Fraud
           http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#Authorization
         ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#FlowExecution
           ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#DocumentFlowExecution
               http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#AccountingDocumentExecution
               http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#TaxStatementExecution
               http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#BillOfLadingExecution
               http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#InvoiceFlowExecution
               http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#StoreDocumentExecution
             http://www.man.poznan.pl/~jolac/MinimalModel/Action.owl#MoneyTransferExecution
       ⊟ http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#LegalProvision
           http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#Art_294_1
           http://www.man.poznan.pl/~jolac/MinimalModel/MinimalModel.owl#Art_271_3
```

```
⊟ hydra20_20
   ⊟ TABLES
     ⊟ Company
         ID, Primary Key, Type: int
         Name, Type: varchar
         KRS, Type: int
         legalForm, Type: int
     ⊞ DamageAndGain
     ⊟ Document
         ID, Primary Key, Type: int
         DocumentKind, Type: varchar
         IDGoodService, Type: int
         DocumentNumber, Type: varchar
         IssueDate, Type: datetime
     ⊞ Employee
     ⊞ GoodService
     ⊞ Invoice
     ⊟ LegalActs
         ID, Primary Key, Type: int
         Name, Type: varchar
         Text, Type: varchar
     ⊞ MoneyTurnover
     ⊞ RDFfacts
     ⊞ Signature
```

Figure 3. The SDL tool with minimal model ontology and database connection

SDL is available as a binary distribution and is free of charge for non-commercial academic usage (for universities only) and can be downloaded from the Web site [10].

2.3 OWL to Jess Transformation Methods

The SDL library supports two main methods of transforming OWL ontologies into rules expressed in Jess language: *simple* and *Horn-SHIQ*. The simple method transforms taxonomies of concepts and roles into Jess rules. These taxonomies are calculated by the Pellet engine first. SWRL rules and SWRLB [11] predicates are also transformed into rules and Jess expressions. The simple transformation can be done in the following modes:

1. Jess script assigned to forward chaining.
2. Jess script assigned to backward chaining.
3. Jess script assigned to forward chaining with extended rules.

The Horn-SHIQ transformation is an extension of the simple one. In this case, additional rules are generated according to (not all) OWL axioms. Rather than transforming the semantics of the OWL language into rules we create rules according to this semantics and a given ontology (in contrast to work presented in [12] and [13]). For example, when we have an ObjectProperty which is a SymmetricObjectProperty we create a rule which reflects that when an instance of this property occurs, a symmetric instance will also occur:

```
(defrule MAIN::HST-SymmetricProperty-inComplicityWith
  (triple (predicate "inComplicityWith") (subject ?x) (object ?y))
  =>
  (assert
     (triple (predicate "inComplicityWith") (subject ?y) (object ?x))))
```

Currently, the implementation is prototypical and does not support all Horn-SHIQ axioms from the W3C specification [14]. The SDL allows for use of simple atomic concepts (A, C), and roles (R). We assume that a concept C is simple if it is of the form: $A, \exists R.A, \forall R.A,$ or $\leq 1R.A$. Complex constructions are not supported. The universal and the existential quantifiers are used only as restrictions in the same way as presented in [15].

Currently supported OWL axioms are taken from the official Horn-SHIQ specification [14] and cover the following list:

a) class axioms:
   - equivalentClasses: URI | ObjectIntersectionOf | ObjectSomeValuesFrom
   - subClass: URI | ObjectUnionOf | ObjectIntersectionOf | ObjectSomeValuesFrom
   - superClass: URI | ObjectIntersectionOf

b) property axioms: URI | equivalentObjectProperties | subObjectPropertyOf | objectPropertyDomain | objectPropertyRange | functionalObjectProperty | inverseFunctionalObjectProperty | symmetricObjectProperty

The Horn-SHIQ transformation can be executed only in two modes: 1 and 3. The SDL also provides the Horn-SHIQ transformation without hierarchy rules. This feature can be helpful to use scripts in different Jess engines.

3 Example Evaluation

For a practical demonstration of the SDL library we used the ‘minimal model’ ontology (the one that fully models a fraudulent disbursement economic crime, but not other economic crimes) with artificially generated data sets. These data sets contain information about: companies, employees, documents, invoices, money turnovers, legal sanctions for this class of crimes, etc. We prepared three databases which differ in the size of the generated documents, values of money, turnovers, etc. The number of companies and employees are the same in every database (20 companies and 240 people). Generated databases contain the following numbers of documents (and money turnovers): 20, 100, 200. An example crime scheme and more information about ‘minimal model’ ontology are presented on the demo description site [10].

We executed five test queries for which description and graphical representation are presented on the demo site. Queries where executed on a computer with the following parameters: Intel Core2Duo 2GHz, 2GB Ram; Java Heap Space was set at 1024MB.

We compared the extended rules approach (marked 2011) with the results presented on the last RuleML Challenge [16] (marked 2010). The comparison is made using the same 5 queries as in 2010. Our current approach outperforms the hybrid one. Since we did not apply all possible optimizations, we are convinced that the efficiency of our method can be improved.

We compared our results with pure forward and backward reasoning in Jess system. Results of this comparison can be found in [10] in the Section called ‘Evaluation’. In these tests while loading data from the third database, the size of the Java heap space was reached (in both engines), so the queries could not be executed.

It seems that for small databases, it is better to store data (facts) in the engines' working memory. But for the larger databases, the problem with scalability occurs. In such cases our extended rules approach seems promising.

Table 1. Results of queries execution in comparison to the RuleML Challenge 2010 results [6]

Query and info		**Database 20**		**Database 100**		**Database 200**	
		2010	**2011**	**2010**	**2011**	**2010**	**2011**
Query 1	[ms]	781	219	1 328	891	1 922	969
Results	[number]	54	54	474	474	1 036	1 036
Rules Fired	[number]	74	251	441	1 630	796	3 001
Query 2	[ms]	2 734	437	37 141	4 125	163 968	19 391
Results	[number]	1	1	1	1	1	1
Rules Fired	[number]	1076	1 506	36 260	13 179	225 381	29 593
Query 3	[ms]	2 875	359	36 344	14 938	183 047	116 593
Results	[number]	18	18	322	322	1004	1 004
Rules Fired	[number]	1 367	2 005	38 457	41 755	232 583	359 681
Query 4	[ms]	5 437	1 859	128 719	35 656	Time	347 110
Results	[number]	1	1	1	1	exceeded	1
Rules Fired	[number]	2 040	5 467	57 091	58 520	10 minutes	597 711
Query 5	[ms]	9 312	1 234	Time	34 500	Time	343 469
Results	[number]	1	1	exceeded	1	exceeded	1
Rules Fired	[number]	2 540	5 828	10 minutes	61 199	10 minutes	608 925

We also executed test queries with extended rules method and Horn-SHIQ transformation rules and compared them to the results achieved with the simple transformation rules. The results and the comparison are shown in [10], in the 'Evaluation' section. An addition of Horn-SHIQ rules makes query answering process more complicated and computationally demanding. It results from fact that Horn-SHIQ transformation contains more OWL axioms than the simple transformation.

Presented results confirm that our approach significantly improves a scalability of a rule-based system in the rule-based query answering. It is a very important, because in the forward chaining rule-based systems, facts have to be stored in the working memory which is, in general, limited by the RAM memory (we call it the traditional approach). If we store facts outside of the memory and load them only when they are needed, we achieve better scalability.

The SDL Demo with above test queries and presented query answering method are available on the demo site [10]. The 'minimal model' ontology is added to the demo material. On the demo site a user has an option to pose her/his own query constructed from concepts and roles from the minimal model ontology. Two databases are available: Database 20 and 100.

4 Conclusions and future work

In this paper we described the SDL library and demonstrated its application to the previously developed the 'minimal model' ontology. We presented a generalization (that is containing more OWL axioms) of the previously introduced hybrid method [5] to the case of transformation of an OWL ontology into Horn-SHIQ rules in the Jess language. The implementation was executed in the dedicated SDL framework. We also confirmed that our approaches significantly improve a scalability of a rule-based system compared with the pure Jess approach.

The SDL library is useful for queries creation because a user of our system gets an easier way to pose queries (due to ontology origin of rules) than using structural constructions from SQL. The creation of queries, presented in the performance evaluation, is extremely difficult when we want to use pure SQL constructions. The strictly defined semantics (in the form of an ontology) is another advantage of our tool.

In future, we are going to use other ontologies to test our tool. We will also extend our approach to handle predicates with an arbitrary number of arguments. We will improve the rule-based query answering algorithm by using optimizations that concern extended rules and magic transformation.

Acknowledgement. This work was supported by PUT BW 45-087/11 BW 2010 and PUT DS 45-085/11 PB grants.

References

1. Hitzler P., Parsia B., Ontologies and Rules, in: Steffen Staab and Rudi Studer (eds.), Handbook on Ontologies. Springer, 2nd Edition, 2009, pp. 111-132
2. McGuinness D., van Harmelen, F.:. Owl web ontology language overview. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-features/
3. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member Submission (May 21 2004), http://www.w3.org/Submission/SWRL/
4. Jess (Java Expert System Shell), http://jessrules.com/
5. Bak, J., Jedrzejek, C., Falkowski, M.: Usage of the Jess engine, rules and ontology to query a relational database. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009. LNCS, vol. 5858, pp. 216–230. Springer, Heidelberg (2009)
6. Bak J., Brzykcy G., Jedrzejek C., Extended Rules in Knowledge-based Data Access, F. Olten, M. Palmirani, D. Sottara (Eds.): RuleML - America 2011, LNCS 7018, pp. 112--127. Springer, Heidelberg (2011)
7. Beeri, C., Ramakrishnan, R.: On the Power of Magic. J. Log. Program.(1991) 255--299, (1991)
8. Pellet Reasoner, http://clarkparsia.com/pellet/
9. Matthew Horridge, Sean Bechhofer. The OWL API: A Java API for Working with OWL 2 Ontologies. OWLED 2009, 6th OWL Experienced and Directions Workshop, Chantilly, Virginia, October 2009, http://owlapi.sourceforge.net/
10. SDL demo and download page, http://draco.kari.put.poznan.pl/
11. SWRL Built-ins, http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
12. Mei, J., Paslaru Bontas, E., Reasoning Paradigms for OWL Ontologies, FU Berlin, Fachbereich Informatik, Technical Reports B 04-12), 2004.
13. G. Meditskos, N. Bassiliades, "A Rule-based Object-Oriented OWL Reasoner", IEEE Transactions on Knowledge and Data Engineering (TKDE), 08 Oct 2007. IEEE Computer Society Digital Library. IEEE Computer Society, 9 October 2007
14. http://www.w3.org/2007/OWL/wiki/Tractable_Fragments#Horn-SHIQ
15. Grosof B., Volz R., Horrocks I. and Decker S. Description Logic Programs: Combining Logic Programs with Description Logics. In Proc. of the 12th International World Wide Web Conference (WWW 2003), 2003.
16. Bak J., Jedrzejek C., Falkowski M., Application of the SDL Library to Reveal Legal Sanctions for Crime Perpetrators in Selected Economic Crimes: Fraudulent Disbursement and Money Laundering, In Proceedings of the 4th International RuleML-2010 Challenge, Washington, DC, USA, October, 21-23, 2010. Edited by: Palmirani M., Omair Shafiq M., Francesconi E., Vitali F., Volume 649, Washington (2010)

SWEETInfo: a Web-Based System for Visualizing and Querying Temporal Data

Martin J. O'Connor, Amanda Richards, Susana Martins, Mike Bingen, Samson W. Tu, Amar Das

Stanford Center for Biomedical Informatics Research
Stanford, CA 94305, U.S.A. martin.oconnor@stanford.edu

Abstract. We describe SWEETInfo (Semantic Web-Enabled Exploration of Temporal Information), a highly-interactive web-based system for querying and visualizing time-oriented data. SWEETInfo was developed as an open-source Web-based infrastructure and combines Semantic Web technologies, such as OWL and SWRL, with standard Web development software, such as the Google Web Toolkit. Investigators can use SWEETInfo to collaboratively manipulate, explore, and visualize temporal data. It allows users to interactively generate complex temporal analyses and to share these analyses with other SWEETInfo users. It also supports importation of data from spreadsheets and relational databases and the final export of analysis results. SWEETInfo was publicly released in September 2011 and is available at `www.sweetinfo.org`.

1 Introduction

The temporal dimension of data is central in many domains. Answering non-trivial temporal questions in these domains typically requires generating complex temporal criteria. For example, a question in a medical domain might combine temporal durations at multiple granularities ("Did a patient have elevated blood sugar levels for more than two days in the past week?"), queries with aggregates ("What was the average post-breakfast blood sugar level of a patient over the last week?"), and the use of periodic patterns ("Did a patient have more than one three-week interval of drug treatment followed by a suppressed viral load lasting at least a week?"). Encoding these temporal queries is difficult using currently available tools. Although commonly-used tools such as SAS and R support complex analysis strategies, they offer very poor support for expressing temporal criteria. They generally support only simple instant-based timestamps and a small set of basic temporal functions. A key shortcoming is that they lack a temporal model. Hence, there is no principled means of associating a piece of information with its temporal dimension. The end result is that temporal criteria have to be expressed at a very low level and customized to the details of the source data layout and structure. Expressing non-trivial criteria, and, in particular, developing complex multi-layered analyses, requires the time-consuming development of customized analysis routines, a process that often requires specialist knowledge. Sharing these analyses with others is also problematic.

There is a clear need for tools to support development and sharing of complex temporal analyses by non specialist users. To address this need, we developed a system that allows researchers to interactively develop temporal analyses that can be published and shared with other researchers. The application, Semantic Web-Enabled Exploration of Temporal Information (SWEETInfo) [1], is a web-based tool that lets investigators collaboratively manipulate, explore and visualize temporal data.

2 SWEETInfo System Architecture

SWEETinfo combines semantic web technologies such as the Web Ontology Language (OWL) [2] and the Semantic Web Rule Language (SWRL) [3] with a variety of standard web technologies. It was developed as a web-based application using standard open source technologies. Its architecture consists of a fairly standard n-tier web layering (Figure 1). The three primary layers are: (1) a presentation layer; (2) a business logic layer; and (3) a data access layer. The presentation layer uses a combination of CSS and the Google Web Toolkit (GWT) [4]. GWT provides a convenient Java wrapper around Javascript, allowing an application to be developed completely in Java. The business logic layer was developed using semantic web technologies. In particular, it uses an OWL-based temporal information model to provide a standardized means of representing all information in a system. SWRL-based temporal reasoning methods are used in SWEETInfo to perform operations on data in this information model. The data access layer uses Protégé-OWL APIs (`protégé.stanford.edu`) to manipulate OWL ontologies. Its SWRL APIs (`protege.cim3.net/cgi-bin/wiki.pl?SWRLTab`) provide the ability to execute SWRL rules. These APIs use the Jess rule engine (`www.jessrules.com`) to execute the SWRL rules.

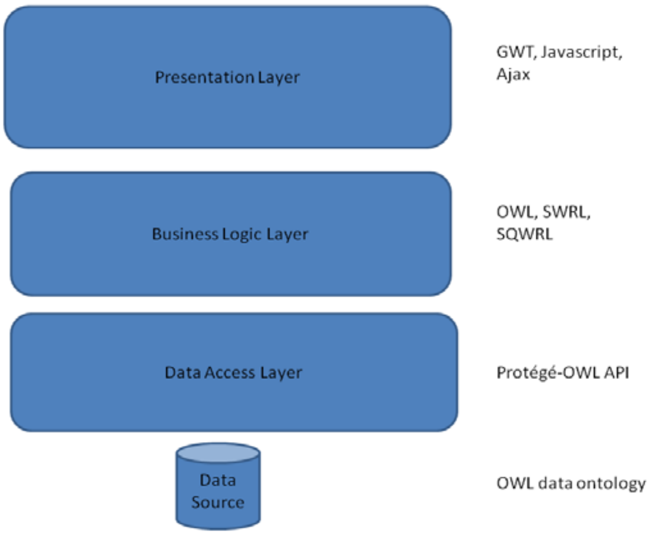


Figure 1. Layers in SWEETInfo System Architecture.

3 SWEETInto Information Architecture

A principled temporal model is key when performing temporal analysis because it enforces a consistent representation of temporal information in a system. One important result of research in the area of temporal data representation is convergence on the valid-time temporal model [5]. Although numerous models have been used to represent temporal information in relational databases and other information systems, we selected this one because it couples simplicity with considerable expressivity. We adopted an OWL-based representation of this model in SWEETInfo [6]. This model is used to encode the temporal dimension of all user data, thus allowing standardized approaches to temporal reasoning and querying.

Using this temporal model as a base, we developed an information model in OWL to provide a consistent representation of all user data in our system. Our model provides a consistent representation of the data's temporal dimension. All data are represented using *variables*, which are atomic pieces of data. Each variable has a value and a temporal dimension. Variables can be numeric, string-based, or include ontology term references. Example variables include viral load values and blood pressure values. All variables are associated with an object of interest (e.g., a patient). All operations and displays in SWEETInfo are defined in terms of objects of interest and the variables associated with them.

Once all temporal information is represented consistently in an information model, it can be manipulated using reusable methods. While OWL has very limited temporal operators for manipulating time values, its associated rule language SWRL provides a small set. However, these operators are very basic, and provide simple instant-based comparisons only. SWRL provides built-ins, which are a mechanism for creating user-defined libraries of custom methods and using them in rules. We have used them to define a library of methods that implement Allen's interval-based temporal operators [7]. The library also has a native understanding of our temporal information model and supports an array of temporal operations on entities defined using it. It can thus be used to directly reason about data defined using our information model.

A SWRL-based query language called SQWRL (Semantic Query-Enhanced Web Rule Language) [8] provides querying support in the system. SQWRL defines a set of SQL-like query operators that can be used to construct retrieval specifications for information in an OWL ontology. SQWRL uses our temporal library to provide complex temporal selection of results. For example, it can make queries such as List the first three doses of the drug DDI or Return the most recent dose of the drug DDI. SQWRL's ability to work directly with our temporal model allows expressive, yet relatively concise temporal queries for expressing criteria. All criteria in SWEETInfo can be expressed directly in SQWRL. Some operations are implemented by combining rules and queries to generate intermediate results incrementally at successively higher levels of abstraction.

A set of predefined temporal operations provide the system's basic set of data transformations. Each operation was designed to be simple enough to be defined using straightforward forms-based dialogs. They were also designed to be easily combined with other operations in a sequence. In this way, users can build complex functionality from relatively simple operations. We identified an initial set of four generalized operations necessary to express temporal criteria. They are:

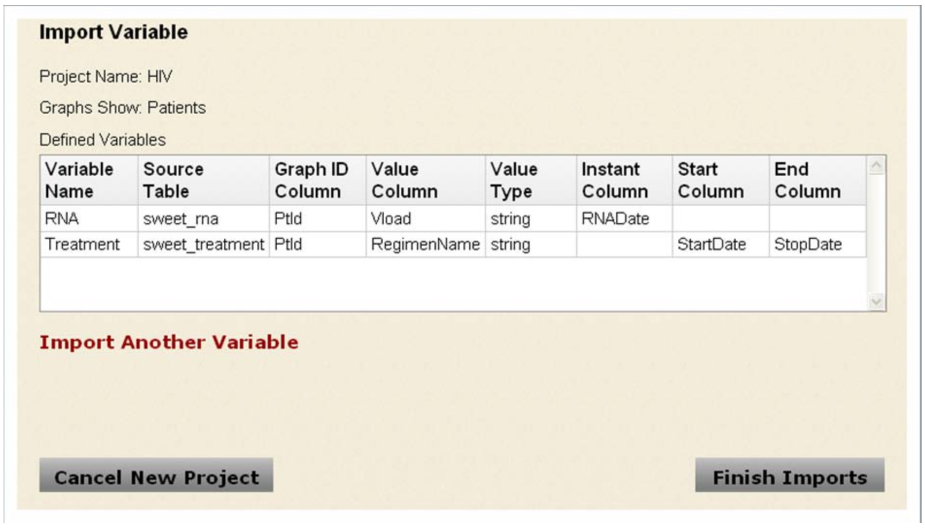
Filter Operations. These select a defined subset of data out of a larger dataset. We currently provide four filtering criteria for operations. They are *value criteria*, which can express basic equality or inequality criteria on numeric or string data, *aggregate criteria*, which can filter on the count, average or maximum or minimum of data, and two forms of temporal constraints. They are *duration criteria*, which can select data based on the length of time between two time points, and *timing criteria*, which support a range of temporal criteria using standard Allen operators. Instances of these four criteria can be composed in a single filter operation (conjunctively or disjunctively) to build complex selections of data. The four criteria can also be used when defining the other three operations.
Grouping Operations. Dividing data into groups is common when performing analysis. For example, patients can be grouped into related categories (e.g., short, medium, tall), which are then analyzed in different ways. A grouping operation allows users to define a set of named groups meeting different sets of criteria. Each group is specified using a criteria set, and data for each group is generated from the subset of input to a grouping operation that meet its associated criteria.
New Variable Operations. SWEETInfo uses variables to define data elements (e.g., blood pressure, viral load; see Section 3). A new variable operation defines a new variable by creating a restriction on an existing variable. For example, a High RNA variable is defined by selecting RNA values that are greater than some number, using a value criterion. Alternatively, a temporal duration criterion could be used to find patients who had been treated with a particular drug for longer than one month.
Temporal Context Operations. Defining temporal patterns is a central requirement when working with temporal data. Temporal context operations are basic building blocks in this regard because they allow users to specify periods that meet a certain pattern. For example, if a user is interested in post-surgical patient outcomes, he could create a new temporal context to represent the period from the beginning of a hospitalization to 30 days after release. This operation allows complex temporal criteria to be built iteratively from a smaller set of simple criteria.

4 SWEETinfo System Features

SWEETInfo uses this information architecture to provide five core functionalities: (1) data importation; (2) temporal data analysis; (3) data visualization; (4) publication and sharing; and (5) analysis results exporting.

4.1 Data Importation

SWEETInfo allows users to import data into its information model from relational databases or spreadsheets. It provides an interactive wizard that allows users to first select a data source and then to select subsets of that data that are to be represented as SWEETinfo variables (Figure 2).


Import Variable

Project Name: HIV

Graphs Show: Patients

Defined Variables

Variable Name	Source Table	Graph ID Column	Value Column	Value Type	Instant Column	Start Column	End Column
RNA	sweet_rna	PtId	Vload	string	RNADate		
Treatment	sweet_treatment	PtId	RegimenName	string		StartDate	StopDate

Figure 2. Screen shot of a preview stage of SWEETInfo import wizard showing user-specified mapping of relational tables to SWEETInfo variables.

4.2 Temporal Data Analysis

As mentioned, operations in SWEETInfo ultimately specify the action taken when certain conditions are met. The conditions are specified using a criteria set. Each operation has an associated creation dialog, which has four subtabs, one for each criteria type. Users can specify criteria interactively, and, on creation, they are displayed in summary form.

To support a typical analysis workflow, we used a pipeline-based representation of analyses (Figure 3). A pipeline is composed of chained operation-view pairs. This approach allows users to see intermediate results of the step-by-step operations that generate the overall analysis. Multiple parallel paths can be defined with filter and grouping operations. A visualization node is automatically produced by each operation and can be used to explore data. It can also be used to summarize the number of patients meeting criteria at each stage of the pipeline. Each view node can also be opened to view detailed displays of the data.

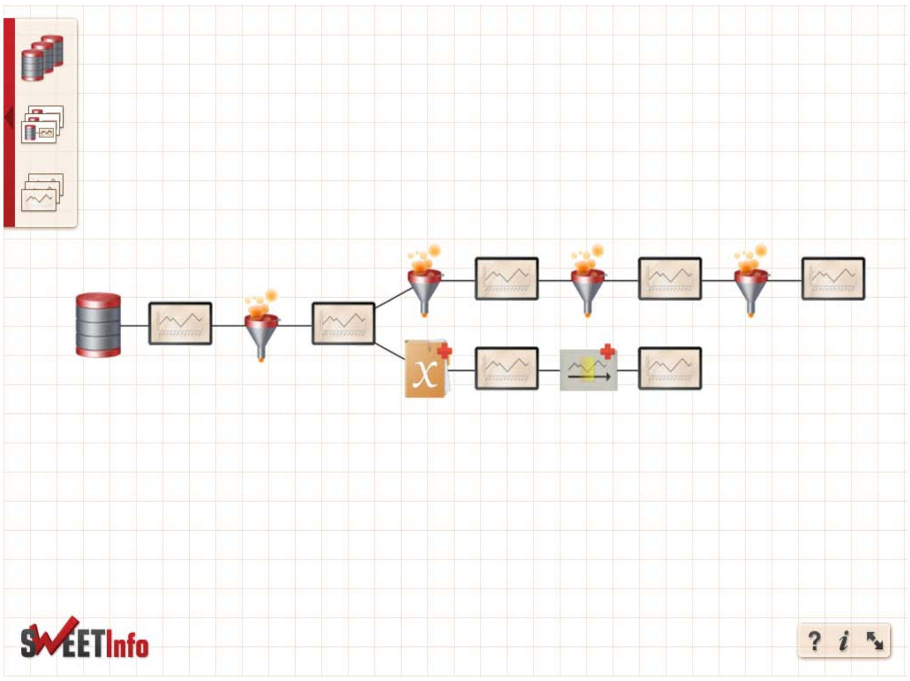


Figure 3. Screen shot of SWEETInfo showing a pipeline definition.

4.3 Data Visualization

Within a pipeline, users can immediately execute an operation that they have defined and examine the results in a set of graphical displays, which show population-level data. They can also drill down to examine individual data elements (Figure 4). The population-level view provides a simultaneous display of data for multiple data elements. The view node can also provide summary statistics for an operation. SWEETInfo provides customized displays for different types of temporal data, and allows users to customize display options. After viewing the data, a user can modify operations in a pipeline immediately or define additional operations. Users can also define branching points using filter and group operations to define parallel analysis paths. The immediate visual feedback and the ability to quickly modify or extend pipeline operations facilitate rapid development of analyses.

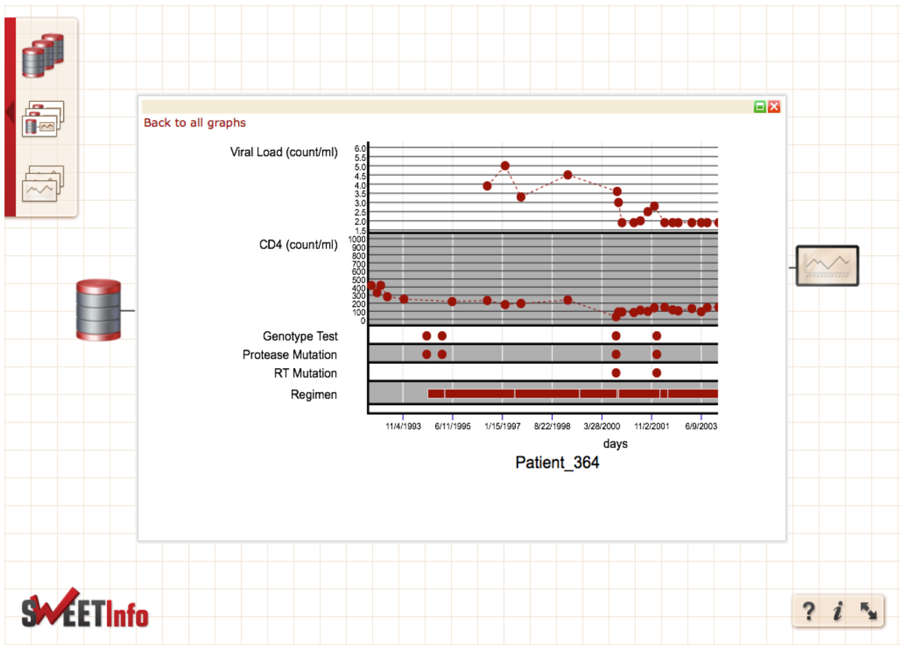


Figure 4. Screen shot of example SWEETInfo visualization.

4.4 Publication and Sharing

SWEETInfo includes a project-based mechanism that allows users to save their analyses and pipeline definitions. It supports multiple projects per user, with each project holding a data set and a pipeline. This mechanism allows users to store, reload, and execute the analyses defined in a pipeline. SWEETInfo also allows users to share pipelines with other users. Additional fine-grained sharing is also provided. Each operation node, which can define a set of complex constraints, can be shared. A library of operations can be maintained for each user and selectively shared. Shared operations can be easily modified to deal with similar data sets.

4.5 Results Exporting

After running an analysis users may wish to export their data for further analysis by other tools. SWEETInfo allows users to export data from any view node in a pipeline. These data are exported as a set of ZIPped CSV files, with one CSV file per variable.

5 Conclusion

A key feature of SWEETInfo is its use of semantic web technologies, particularly OWL, SWRL, and SQWRL. OWL is used to represent an information model for storing all user data. It is also used to represent all system information, such as user accounts, data transformations, and pipeline configurations. Additionally, all data transformations are executed using SWRL and SQWRL. These operations make use of an expressive temporal model and library developed for these languages [6]. The system demonstrates that these technologies can support the demands of complex highly interactive web-based applications, and that they can be successfully combined with traditional web technologies. The information model ontology and data analysis modules that result from this combination are reusable and can provide temporal representation and analysis functionality in other semantic web applications. SWEETInfo was released in September 2011 and is available at `www.sweetinfo.org`.

We are currently adding more expressivity through additional pipeline operations, such as, for example, statistical analysis operations. These operations will invoke external packages like R or SAS. We also intend to allow users to assemble individual operations into higher level components and then to share them. Our ultimate goal is to produce a library of reusable analysis modules, much like those provided by standard statistical packages.

Performance is also a key goal. While the current system performs satisfactorily, it does not scale to a large number of users. In particular, performing simultaneous analyses on a single server with SWRL and SQWRL is computationally expensive. We are investigating the use of a variety of strategies to tackle this problem.

Acknowledgements. This research was supported by grant 1R01LM009607 from the National Library of Medicine.

References

1. O'Connor, M.J., Bingen, M., Tu, S.W., Richards, A, Das, A.K. Web-Based Exploration of Temporal Data in Biomedicine. 7th International Conference on Web Information Systems and Technologies (WEBIST), Noordwijkerhout, Netherlands, 352-359 (2011)
2. OWL Overview: http://www.w3.org/TR/owl-features/
3. SWRL Submission: http://www.w3.org/Submission/SWRL/
4. Google Web Toolkit: http://code.google.com/webtoolkit/
5. Snodgrass, R.T. The TSQL2 Temporal Query Language, Boston, MA: Kluwer. (1995)
6. O'Connor, M.J. and Das, A.K. A Method for Representing and Querying Temporal Information in OWL. Springer-Verlag, CCIS 127, 97-110 (2011)
7. Allen, J.F. Maintaining Knowledge about Temporal Intervals. Communications of the ACM, 26(11) (1983)
8. O'Connor, M.J. and Das, A.K. SQWRL: a Query Language for OWL. OWL: Experiences and Directions, 6th International Workshop, Chantilly, VA (2009)

~~BIOQUERY-ASP: Querying Biomedical Ontologies using~~ Answer Set Programming

Esra Erdem, Halit Erdogan, and Umut Oztok

Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul, Turkey

Abstract. We describe a software system, called BIOQUERY-ASP, that finds answers and generates explanations to complex biomedical queries over the available knowledge resources, such as, PHARMGKB, DRUGBANK, CTD, SIDER, BIOGRID, using the computational methods/tools of Answer Set Programming.

1 Introduction

Recent advances in health and life sciences have led to generation of a large amount of biomedical data. To facilitate access to its desired parts, such a big mass of data has been represented in structured forms, like biomedical ontologies and databases. On the other hand, representing these ontologies and databases in different forms, constructing them independently from each other, and storing them at different locations have brought about many challenges for answering queries about the knowledge represented in these ontologies.

One of the challenges for the users is to be able to represent a complex query in a natural language, and get its answers in an understandable form. Another challenge is to be able to answer complex queries that require appropriate integration of relevant knowledge from different knowledge resources and/or that require auxiliary definitions, such as, chains of drug-drug interactions, cliques of genes based on gene-gene relations, or similarity/diversity of genes/drugs. Furthermore, once an answer is found for a complex query, the experts may need further explanations about the answer.

Consider, for instance, the following queries:

Q1 What are the genes that are targeted by the drug Epinephrine and that interact with the gene DLG4?
Q2 What are the genes that are targeted by all the drugs that belong to the category Hmg-coa reductase inhibitors?
Q3 What are the genes related to the gene ADRB1 via a gene-gene relation chain of length at most 3?
Q4 What are the 3 most similar genes that are targeted by the drug Epinephrine?

Most of the existing biomedical querying systems (e.g, web services built over the available knowledge resources) support keyword search but not complex queries like Q1–Q4. Some of these complex queries, such as Q1 and Q2, can be represented in a formal query language (e.g., SQL/SPARQL) and then answered using Semantic Web technologies. However, queries, like Q3, that require auxiliary recursive definitions (such as transitive closure) cannot be directly represented in these languages; and thus such queries cannot be answered directly using Semantic Web technologies. The experts usually compute auxiliary relations externally, for instance, by enumerating all drug-drug interaction chains or gene cliques, and then use these auxiliary relations to represent and answer a query like Q3. Similarity/diversity queries, like Q4, cannot be represented directly in these languages either, and require a sophisticated reasoning algorithm. Also, none of the existing systems can provide informative explanations about the answers, but point to related web pages of the knowledge resources available online.

We have built a software system, called BIOQUERY-ASP,[1] that handles all these challenges using Answer Set Programming (ASP) [9]. To address the first challenge, we have developed a controlled natural language for biomedical queries about drug discovery; this language is called BIOQUERY-CNL [5]. For instance, the queries Q1–Q4 are in BIOQUERY-CNL. Then we have built an intelligent user interface that allows users to enter biomedical queries in BIOQUERY-CNL and that presents the answers (possibly with explanations or related links, if requested) in BIOQUERY-CNL [6]. To address the second challenge, we have developed a rule layer over biomedical ontologies and databases, that not only integrates the concepts in these knowledge resources but also provides definitions of auxiliary concepts [1]. We have introduced an algorithm to identify the relevant parts of the rule layer and the knowledge resources with respect to the given query, and used automated reasoners of ASP to answer queries considering these relevant parts [4]. To address the third challenge, we have developed an algorithm to generate an explanation for a given answer, with respect to the query and the relevant parts of the rule layer and the knowledge resources [10,4]. The overall system architecture for BIOQUERY-ASP is presented in Figure 1.

We have shown the applicability of BIOQUERY-ASP to answer queries over large biomedical knowledge resources about genes, drugs and diseases, such as PHARMGKB,[2] DRUGBANK,[3] BIOGRID,[4] CTD,[5] and SIDER,[6] using efficient solvers of ASP [4]. For queries that are not concerned about similarity/diversity of genes/drugs, we have used the ASP solver CLASP [7]. For similarity/diversity queries, we have utilized the online methods of [2] for finding similar/diverse solutions, and thus used the ASP solver CLASP-NK, a variant of CLASP.

2 Demonstration of BIOQUERY-ASP by Examples

Let us demonstrate the use of BIOQUERY-ASP with four examples: the queries Q1, Q2, Q3 and Q4 presented in the introduction.

2.1 Representing a Query in BIOQUERY-CNL

BIOQUERY-ASP allows users to construct queries in the grammar of BIOQUERY-CNL (an extended version of the grammar introduced in our earlier work [5]), by providing

[1] http://krr.sabanciuniv.edu/projects/BioQuery-ASP/
[2] http://www.pharmgkb.org/
[3] http://www.drugbank.ca/
[4] http://thebiogrid.org/
[5] http://ctd.mdibl.org/
[6] http://sideeffects.embl.de/

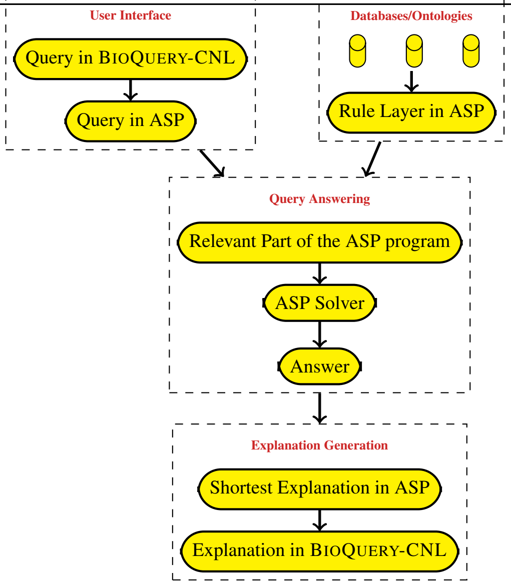


Fig. 1. System overview of BIOQUERY-ASP.

them two options: by showing them some templates so that they can choose one from among them (as shown in Figure 2 for the query Q1); or by guiding them to construct the query by showing the possibilities with an auto-completion feature (as shown in Figure 3 for the query Q1).

2.2 Transforming the Query in BIOQUERY-CNL to ASP

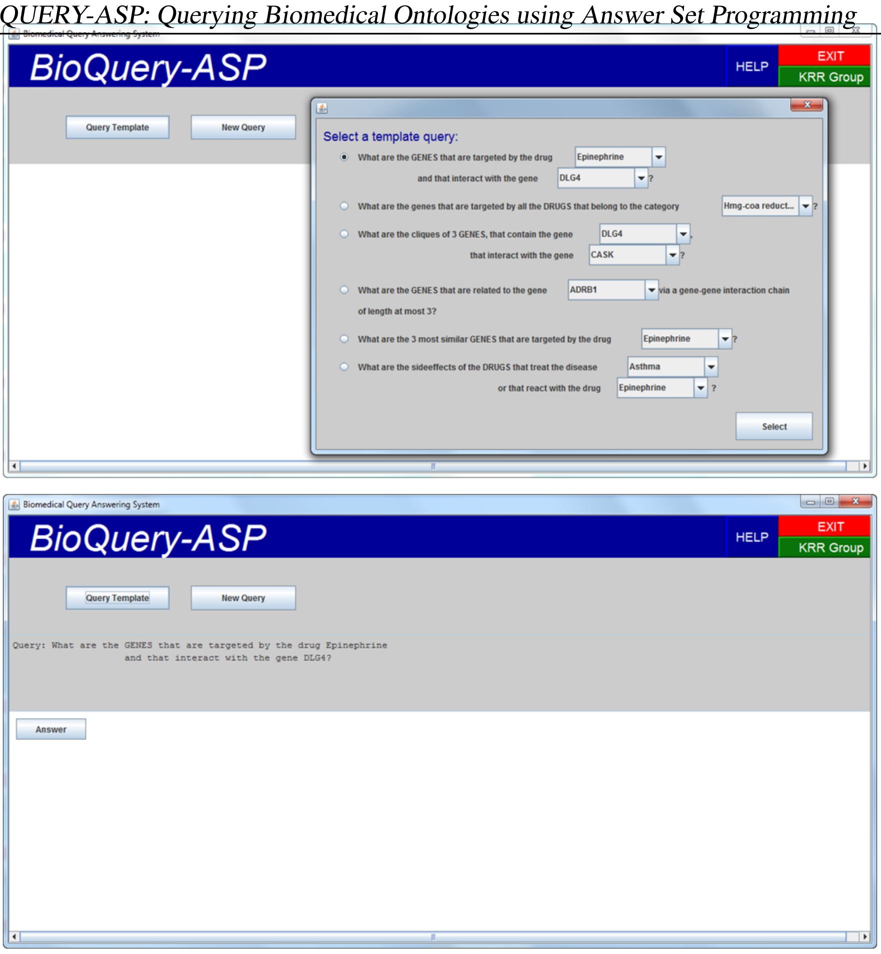
Once a query is constructed in BIOQUERY-CNL, we transform the query into ASP by an extension of the algorithm introduced in our earlier work [5]. For instance, the query Q1 is translated into the following ASP program:

```
what_be_genes(GN1) :- condition1(GN1), condition2(GN1).
condition1(GN)  :- drug_gene("Epinephrine",GN).
condition2(GN)  :- gene_gene(GN,"DLG4").
answer_exists  :- what_be_genes(GN1).
```

where `condition1` and `condition2` are invented relations, and `gene_gene` and `drug_gene` are defined in the rule layer as follows:

```
drug_gene(D,G)  :- drug_gene_pharmgkb(D,G).
drug_gene(D,G)  :- drug_gene_ctd(D,G).

gene_gene(GN1,GN2)  :- gene_gene_biogrid(GN1,GN2).
gene_gene(GN2,GN1)  :- gene_gene(GN1,GN2).
```

Similarly, the query Q2 is translated into the following ASP program:



Select a template query:

[✓ ●] What are the GENES that are targeted by the drug Epinephrine and that interact with the gene DLG4?

○ What are the genes that are targeted by all the DRUGS that belong to the category Hmg-coa reduct... ?

○ What are the cliques of 3 GENES, that contain the gene DLG4, that interact with the gene CASK?

○ What are the GENES that are related to the gene ADRB1 via a gene-gene interaction chain of length at most 3?

○ What are the 3 most similar GENES that are targeted by the drug Epinephrine?

○ What are the sideeffects of the DRUGS that treat the disease Asthma or that react with the drug Epinephrine ?

```
Query: What are the GENES that are targeted by the drug Epinephrine
                   and that interact with the gene DLG4?
```

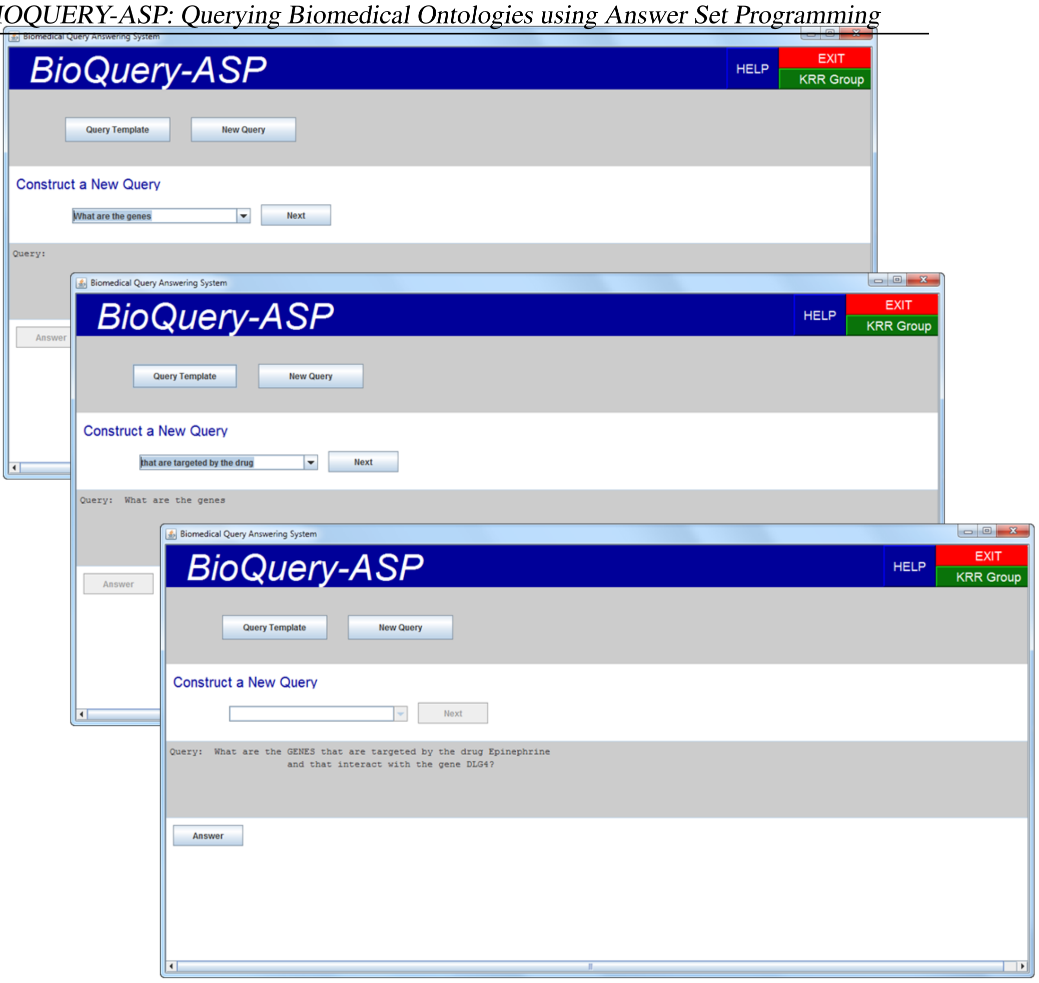
Fig. 2. In BIOQUERY-ASP a query can be constructed by making use of a template.

```
what_be_genes(GN1) :- not notcommon(GN1),
   gene_name(GN1), notcommon_exists.
notcommon(GN1) :- not drug_gene(D2,GN1),
   condition1(D2), gene_name(GN1).
notcommon_exists :- notcommon(X).
condition1(D) :- drug_category(D,"Hmg-coa reductase inhibitors").
answer_exists :- what_be_genes(GN).
```

The query Q3 is translated into the following ASP program:

```
what_be_genes(GN) :- condition1(GN).
condition1(GN) :- gene_reachable_from(GN,L).
start_gene("ADRB1").
max_chain_length(3).
answer_exists :- what_be_genes(GN).
```

where `gene_reachable_from` is defined in the rule layer as follows:



```
Query:
```

```
Query:  What are the genes
```

```
Query:  What are the GENES that are targeted by the drug Epinephrine
               and that interact with the gene DLG4?
```

Fig. 3. In BIOQUERY-ASP a query can be constructed by making use of the auto-completion feature.

```
gene_reachable_from(X,1) :- gene_gene(X,Y), start_gene(Y).
gene_reachable_from(X,N+1) :- gene_gene(X,Z),
   gene_reachable_from(Z,N), 0 < N, N < L,
   max_chain_length(L).
```

Note that unlike the rules for `gene_gene` and `drug_gene` that integrate knowledge resources, the rules for `gene_reachable_from` define an auxiliary concept to be used for deep reasoning.

The query Q4 is translated into the following ASP program:

```
1{what_be_genes(GN) : condition1(GN)}1.
condition1(GN) :- drug_gene("Epinephrine", GN).
answer_exists :- what_be_genes(GN).
```

2.3 Extracting Information from the Knowledge Resources using ASP

Some of the ASP solvers, such as DLVHEX [3], provide constructs to import external theories that may be in different formats (e.g., ontologies in RDF(S)/OWL). For instance, consider as an external theory a Drug Ontology described in RDF. All triples from this theory can be exported using the external predicate `&rdf`:

```
triple_drug(X,Y,Z) :- &rdf["URI for Drug Ontology"](X,Y,Z).
```

Then the names of drugs can be extracted by DLVHEX using the rule:

```
drug_name(A) :- triple_drug(_,"drugproperties:name",A).
```

Similarly, gene-gene interactions could be extracted from a Gene Ontology by DLVHEX using the rules

```
gene_gene(G1,G2) :- triple_gene(X,"geneproperties:name",G1),
   triple_gene(X,"geneproperties:related_genes",B),
   triple_gene(B,Z,Y), Z!="rdf:type",
   triple_gene(Y,"geneproperties:name",G2).
```

Some knowledge resources are provided as relational databases, or more often as a set of triples (probably extracted from ontologies in RDF). In such cases, we introduce special algorithms to transform the relations into ASP.

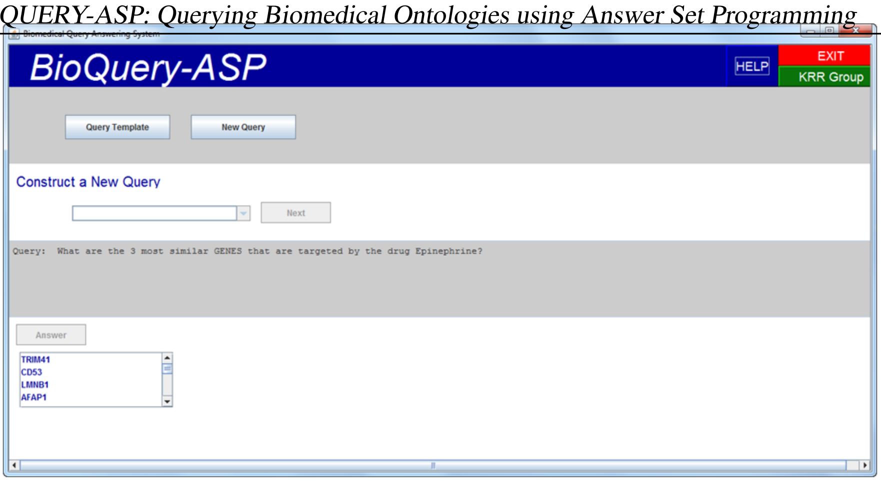
2.4 Answering the Queries using ASP

At this stage, the given biomedical query Q, the rule layer, and the information extracted from the knowledge resources are all in ASP. Let us denote by Π the union of the rule layer and the information extracted from the knowledge resources. To be able to answer the given query Q efficiently, BIOQUERY-ASP extracts the relevant part of Π with respect to Q [4], and then computes a model (called "an answer set" [8]) for the relevant part (if exists) using a state-of-the-art ASP system, such as CLASP. After that, BIOQUERY-ASP extracts the answers to Q from the answer set, and presents them as a list. For instance, the answers to the query Q3 is shown in Figure 4.

For queries about similarity/diversity of genes, BIOQUERY-ASP uses the answer set solver CLASP-NK [2], an extension of CLASP that can compute similar/diverse answer sets for an ASP program with respect to a given distance measure. The idea behind CLASP-NK is to define the distance measure (as a C++ program) and modify the search algorithm of CLASP accordingly in the style of a branch-and-bound algorithm. BIOQUERY-ASP considers the semantic and functional similarity of genes defined over the gene ontology [11]; this measure can be computed by the software GOSEMSIM.

2.5 Generating Explanations for the Answers

Once an answer to a query (that does not involve aggregates as in Q3 and Q4) is computed, BIOQUERY-ASP can generate an explanation for it [4,10]. For instance, an explanation for the answer "ADRB1" to the query Q1 is shown in Figure 5. If an explanation cannot be found then related links to the knowledge resources are provided.



```
Query:  What are the 3 most similar GENES that are targeted by the drug Epinephrine?
```

```
TRIM41
CD53
LMNB1
AFAP1
```

Fig. 4. BIOQUERY-ASP presents answers to a query as a list.

3 Conclusion

We have described the software system BIOQUERY-ASP that finds answers and generates explanations to complex biomedical queries over the available knowledge resources, such as, PHARMGKB, DRUGBANK, CTD, SIDER, BIOGRID, using the computational methods/tools of Answer Set Programming. These complex biomedical queries require appropriate integration of relevant knowledge from different knowledge resources; auxiliary definitions, such as, chains of drug-drug interactions, cliques of genes based on gene-gene relations, or similarity/diversity of genes/drugs; and further deep reasoning, like finding similar/diverse genes/drugs. No existing biomedical query answering systems (e.g., web services built over the available knowledge resources, that answer queries by means of keyword search) can directly answer such queries, or can generate explanations for answers. In that sense, BIOQUERY-ASP is a novel biomedical query answering system that can be useful for experts.

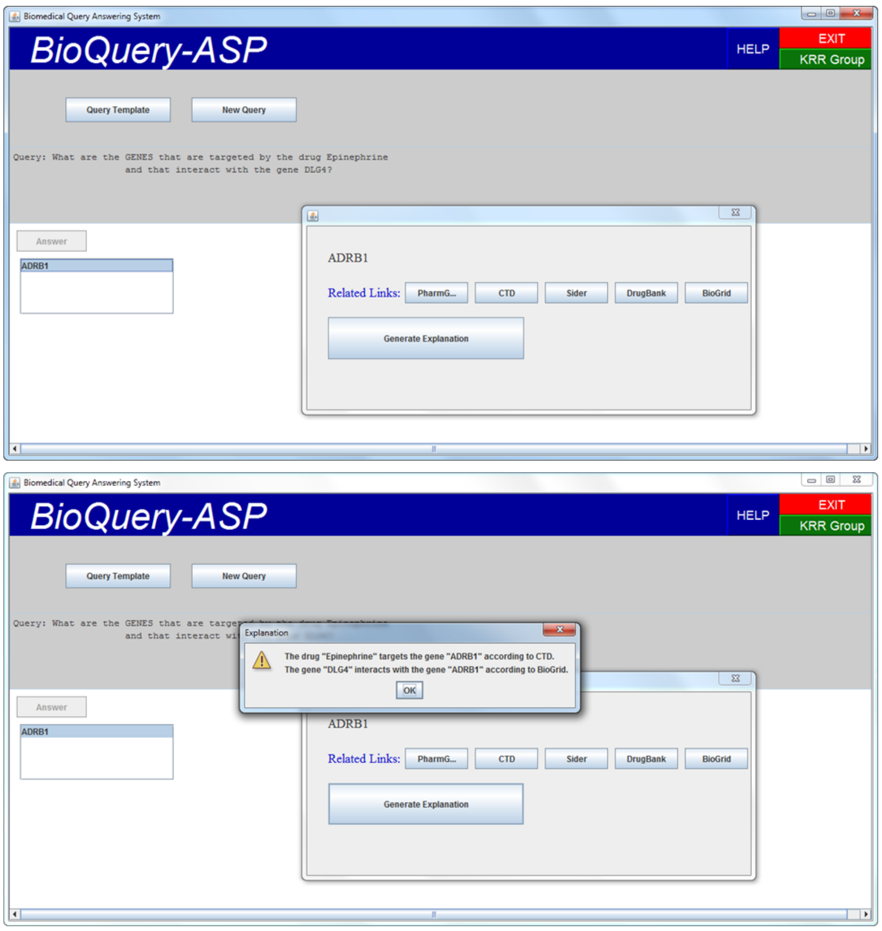
Acknowledgments.

We are grateful to Yelda Erdem (Sanovel Pharmaceuticals) for her suggestions about the biomedical queries related to drug discovery. This work has been supported by TUBITAK Grant 108E229.

References

1. Bodenreider, O., Coban, Z.H., Doganay, M.C., Erdem, E.: A preliminary report on answering complex queries related to drug discovery using answer set programming. In: Proc. of ALPSWS (2008)
2. Eiter, T., Erdem, E., Erdogan, H., Fink, M.: Finding similar or diverse solutions in answer set programming. In: Proc. of ICLP. pp. 342–356 (2009)
3. Eiter, T., G.Ianni, R.Schindlauer, H.Tompits: Effective integration of declarative rules with external evaluations for Semantic-Web reasoning. In: Proc. of ESWC (2006)

4. Erdem, E., Erdem, Y., Erdogan, H., Oztok, U.: Finding answers and generating explanations for complex biomedical queries. In: Proc. of AAAI (2011)
5. Erdem, E., Yeniterzi, R.: Transforming controlled natural language biomedical queries into answer set programs. In: Proc. of the Workshop on BioNLP. pp. 117–124 (2009)
6. Erdogan, H., Oztok, U., Erdem, Y., Erdem, E.: Querying biomedical ontologies in natural language using answer set programming. In: Proc. of SWAT4LS (2010)
7. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A Conflict-Driven Answer Set Solver. In: Proc. of LPNMR. pp. 260–265 (2007)
8. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
9. Lifschitz, V.: What is answer set programming? In: Proc. of AAAI (2008)
10. Oztok, U., Erdem, E.: Generating explanations for complex biomedical queries. In: Proc. of AAAI (2011)
11. Wang, J.Z., Du, Z., Payattakool, R., Yu, S.P., Chen, C.F.: A new method to measure the semantic similarity of go terms. Bioinformatics 23, 1274–1281 (2007)



```
Query: What are the GENES that are targeted by the drug Epinephrine
              and that interact with the gene DLG4?
```

The drug "Epinephrine" targets the gene "ADRB1" according to CTD.
The gene "DLG4" interacts with the gene "ADRB1" according to BioGrid.

Fig. 5. BIOQUERY-ASP can generate an explanation for an answer to the given query.

Managing High Disease Risk Factors : a use case in the KMR-II Healthcare Infrastructure

Davide Sottara[1], Emory Fry[2], Esteban Aliverti[3], Mauricio Salatino[3], and John Harby[4], Stefan Killen[4], Tuyet Nguyen[4], Mike Wright[4]

[1] Università di Bologna (davide.sottara2@unibo.it)
[2] Naval Health Research Center (eafry@gmx.com)
[3] PlugTree
[4] Soadex Inc.

Abstract. The goal of the KMR-II project is the creation of a unified architecture for a knowledge intensive patient healthcare management. Using a combination of knowledge integration techniques, it provides a framework for the interaction between patients, providers and managers. At the same time, is provides intelligent clinical decision support and automates management procedures. In this paper, we introduce some of the architectural aspects and show how it allows to manage a simple use case, where a predictive model is used to notify both a patient and their provider that there is a high probability that the patient will develop some disease in the future.

1 Introduction

Supply and demand forecasting in the healthcare industry is predicated on the dual requirement of an accurate analysis of care resources within a community and equally accurate demand forecasting. Unfortunately, current predictive models for projecting population-based demand and the resource required to provide quality care are simply inadequate. It is not unusual for organizations to estimate demand using simplistic models for generalized populations regardless of the clinical characteristics of the patients being served, or to adequately analyze the capacity of availability of resources.

The Distributed Decision Support and Knowledge Management Repository (KMR-II) system provides healthcare planners with the analytic tools they require to develop and deploy sensitive predictive models tailored to the specific characteristics of the target population. It allows organizations to (i) develop disease specific models for accurate forecasting of demand (anticipated number of patients with a specific disease), and (ii) plan for the optimal utilization of existing resources to care for them. It is focused on the integration of commodity rule and workflow management systems for we believe this approach represents the best opportunity to deliver “knowledge services” that can be layered on both military and civilian health information networks.

The second mission of the KMR infrastructure is to change behavior - its inference engines decide what to do, its workflow capabilities automate those tasks, and its notification capabilities communicate with intended recipients. Ultimately, all these capabilities need to be exposed to the end user if behavior is to change. KMR provides managed Presentation Services to assist developers in deploying tightly integrated graphical user interfaces that not only ensure the results of Clinical Decision Support (CDS) and Predictive Analytics can alert and inform the intended user, but can also materially effect behavior by writing orders, booking appointments, changing device settings, etc . . .

To this end, we are developing a knowledge intensive infrastructure, capable of leveraging different forms of knowledge: from a semantic description of the application domain, to predictive, diagnostic and planning models for decision support, to complex event processing for real-time operation management, to business rules and workflows for policy enforcement. We have adopted, but not limited ourselves to, a patient-centric architecture, in which a context ("virtual medical record" or VMR) is built for each patient, where all their historical data are stored. Each context is actually the working memory of a reasoning engine, capable of applying various inference strategies to the the clinical data contained therein. The engine, then, is provided at runtime with the appropriate knowledge, according to the specific evaluations which must be performed on the patient.

To give an overview of the architecture and its design principles, we will show a simple use case built on top of our general-purpose infrastructure. In this scenario, a healthcare organization recommends the usage of a set of predictive models, in order to evaluate the risk level a person may have, to develop one or more diseases. A provider is given the capability to apply one or more of those models to their patients. The models would leverage the clinical, historical data present in a patient's medical record - or allow the provider to fill in any missing piece of information - to estimate the probability (with an associated confidence interval) that the patient might suffer from a certain disease in the future. The result of this evaluation, usually based on a comparison of the patient's characteristics with the features of a reference population of individuals, can then be used to apply diagnostic and preventive actions[5]. In particular, should a model detect that a risk threshold has been exceeded for a disease, the system would make sure that both the patient and their provider are notified of this event, delivering a content-rich, informative alert message, using one or more predefined channels (e-mail, SMS, voice call, . . .) and ensuring that an acknowledgment is returned.

The knowledge-based part of the architecture has been implemented using the open source Knowledge Integration Platform Drools. In addition to its friendly licensing model, it is a suite of tools which allows to model, combine and execute different forms of knowledge in the same environment, facilitating the development of complex applications such as the one we are describing. Drools comes in the form of a core rule engine, supporting a rich, semi-declarative language, and is extended with a set of additional modules which greatly enhance its inferential capabilities. In section 2, then, we will describe how the tool has been used to

[5] also recommendable by the system, although the topic is not covered in this paper

develop the core of our application, while section 3 will provide more details on the use case we are presenting.

2 A Knowledge Intensive Architecture

Nowadays, large-scale applications are implemented either using Service-Oriented Architectures (SOA), Event-Driven Architectures, or a combination thereof. The two have been proven to be complementary ([4]), since the distinction mostly depends on how the components interact (synchronously vs asynchronously) rather than on structural differences. In our setting, we need both event-driven and request-driven services. A patient record would be continuously updated with information coming from various and heterogeneous sources, possibly triggering the parallel execution of different policies and processes. At the same time, various individuals in the healthcare management organization could request access or even perform operations on the patient's VMR to manage their status.

Given the nature of the data in the records, the knowledge-intensive nature the operations to be performed, the necessity to adapt the policies as the patient's conditions change in time and ultimately the need to provide a highly dynamic set of functionalities, we chose not to implement the core functionalities strictly as services. While still using more traditional (web) service interfaces for peripheral components such as the persistence, security and GUI sub-systems, we have decided to found the core of the architecture on the concept of rich, intelligent agents rather than services. Even from a modelling perspective, an "agent" better represents the idea of a dedicated patient manager: it is the agent, then, who is in charge of processing the incoming events and providing the required services.

Drools Agents. A Drools Agent is itself a knowledge-based application: its internal logic is programmed using rules rather than traditional imperative code. Its external interface is based on the concept of *communication performatives* and has been designed according to the principles of the FIPA[6] standard. This standard regulates the interactions between intelligent agents, defining formats for messages and their content, in addition to formalizing protocols for agent-to-agent communications. To integrate agents in a non agent-based SOA, the agents expose a single endpoint as a (web) service, named `tell`. This service allows to deliver messages to the agent, either from another agent or a more traditional service. When a message is received, it is immediately inserted into a knowledge session, where a set of rules will interpret and process it. The agent currently understands most of the FIPA performatives and is designed to rely on the concept of *expectation* [1] to make sure that the appropriate protocols are respected when exchanging messages (e.g. either refusals or responses are received in a timely fashion for a given request).

[6] http://www.fipa.org/

Internally, the agent leverages the functionalities of `drools-grid` to deploy and maintain separate knowledge sub-sessions (i.e. runtime instances of the reasoning engine) on different nodes of a network. Each session is usually dedicated to a single patient. The rules in the main agent session, then, preprocess the messages and implement a dynamic content-based routing service [3], to ensure that messages regarding a specific patient are ultimately processed in the appropriate sub-session. The master session, in fact, converts each message into a set of working memory commands (assert, retract, query, . . .). These commands are executed in the appropriate sub-session: when the results are available, they are delivered back to the main session where the appropriate response message is generated. In addition to the obvious load balancing benefits, this architecture allows to separate the messaging and protocol management rules (applied in the main session) from the clinical knowledge, since the sub-sessions are not aware of the agent-oriented nature of the environment sustaining them. An overview of the agent structure is summarized in figure 1.

While our architecture will support most FIPA performatives, the most relevant are `Inform`, `Request` and `Query`.

- `Inform` is used mainly for event notifications: it allows the asynchronous delivery of both data and events to any patient session. The events will be able to trigger consequences, but the source of the event will generally not be notified.
- `Query` is used to access the patient records and extract any information present therein in the form of facts
- `Request` is used to request the execution of specific actions on a patient session. A request is implemented using a fact insertion (to trigger the execution) followed by a query (to get the results).

The use of the `Request` performative, in combination with a rule-based management of the request message content allows to provide dynamic, declarative services. A `Request` contains an `Action` with a name and a list of arguments. The management rules are based on patterns matching the `Action`s and, as their consequences, trigger the actual executions aimed at satisfying the request, either directly or through chaining. These rules, then, are equivalent to the implementation of a service. With respect to more traditional services, interface contracts and registration and discovery functionalities are not (yet) supported, but the declarative nature of the implementation has many advantages in terms of deployment, maintenance and lifecycle management in general. Rules, in fact, are generally managed using a (Business) Rule Management System (`drools-guvnor`, in our case) which takes care of issues such as authoring, publication and versioning.

Rich Knowledge Sessions. To actually implement the patient specific services, the individual sessions can leverage the full power of the reasoning engine. The extension of production rules with event processing (`drools-fusion`) and workflow management (`drools-jbpm`) capabilities are well known and will not be

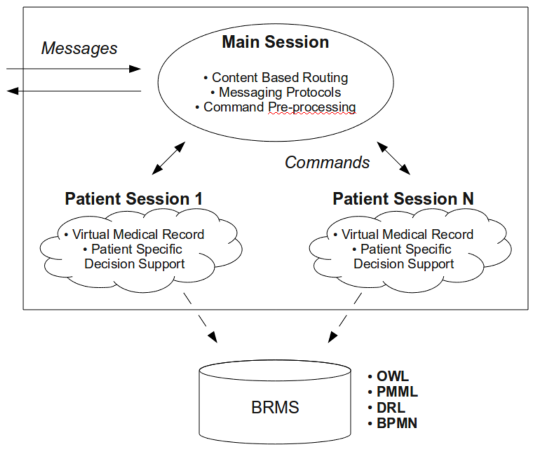


Fig. 1. Clinical Decision Support Agent Architecture

further discussed. To manage our complex use cases, however, additional extensions are being implemented as a part of the KMR-II project. The enhanced modules[7] include:

– `drools-semantics` adds semantic capabilities to the engine. In addition to providing a basic form of semantic reasoning [2], it currently implements some techniques to extract a data model from an ontology [5]. This data model can then be used to ground rules in the concepts defined in the ontology itself. The model is based on java interfaces: in order to bind to the interfaces either object instances or - in alternative - triple-described individuals, a technique known as *traiting* is used, similarly to what has been recently proposed in [8]. A high-level KMR-II ontology is currently being defined, as a part of the project, to describe a medical domain. It will be integrated with other medical "ontologies" to include standard vocabularies.
– `drools-pmml` allows to deploy PMML-encoded predictive models into a knowledge session, where they can be evaluated reactively. PMML, a standard for predictive model exchange, has been designed to declaratively describe the models' structure and parameters, and allows to exchange models between different engines. Models, then, can be trained using tools such as Knime or Weka and imported into Drools for runtime execution. Interestingly, a Drools-based compiler is used to translate a model into a serie of rules which emulate the behaviour of the predictive models themselves, allowing a seamless, homogeneous integration of the models into the rule session [7].
– `drools-chance` extends the traditional inference mechanisms to support uncertain and vague reasoning in a native way [6]. It will be used by the semantic and predictive reasoners, since their outputs are not boolean in general.

[7] https://github.com/droolsjbpm/drools-chance

– `drools-informer` is another rule-based module, derived from a refactoring of the open source tool previously known as "Tohu"[8]. It allows to automatically create and manage *Questions*, special objects which can be bound, at the same time, to a widget in a GUI interface and to a target object's field. Sets of `Questions`, called `Questionnaires`, generate dynamic forms for the dynamic update of facts inside the working memory, effectively making a knowledge session interactive.

3 Use Case - Risk Factor Prediction and Management

To demonstrate the flexibility of the infrastructure, we propose the following simple scenario. We assume that a soldier, just returned from deployment in a war zone, visits their provider for a check-up. Given the patient history, the provider has chosen to evaluate a risk assessment model to estimate the probability that the soldier will suffer from post-traumatic stress disorder (PTSD). The system loads the appropriate predictive model and notifies the provider that some relevant model inputs are missing. Those inputs, however, can be easily acquired by interviewing the patient and provided to the system filling a questionnaire. Once the questionnaire is complete, the model will be evaluated: depending on the actual answers, the estimated risk value might be above the threshold set by the provider. If this is the case, the system will generate two alert messages: one for the provider and one for the patient. The messages are accessible through a "universal inbox", a web-based frontend which displays messages in a fashion inspired by traditional email clients. The body of the messages, however, contains dynamic content, including an interactive form that is used by the recipient to acknowledge the message after it has been read. If the message is not acknowledged within a given deadline, a second message - in the form of an SMS - is delivered as a reminder.

Use Case Implementation. This use case leverages most of the components in the architecture. The presentation services, interacting with the GUI, use three of the decision support agent "services": `getRiskModelDetails`, `getForm` and `setForm`. The first is used to request the deployment (if not already available) of a specific PMML model in a patient's session and it subsequent evaluation, returning the estimated percentage and any correlated information (e.g. the confidence interval)[9]. The second is used to get any interaction questionnaire metadata (generating it if necessary), which allows the GUI to render a web form to collect the answers from the user. In combination with `getRiskModelDetails`, it returns the questionnaire associated to a predictive model instance. The third, instead, delivers the answers to specific questions and returns the result of any validation check, if present.

When the risk threshold is exceeded, a message generation and management process is started internally. The agent has been configured to generate the ap-

[8] http://www.jboss.org/tohu
[9] In the application demo, we use a few mock models

propriate alert messages and deliver them to the appropriate recipients using the appropriate channels. It collects data from the current context (patient, provider, disease, risk level, . . .) and uses them to instantiate one or more message templates. (The actual delivery of the messages is not done directly, but delegated to another dedicated agent, which is out of scope for the purpose of this paper). Each message will also contain the reference to an interaction form, prepared to collect the acknowledgments from the recipient. When the inbox client renders the message content, in fact, the GUI will again invoke the `getForm` and `setForm` services to let the user and the agent interact.

4 Conclusions

This paper shows some aspects of the current state in the development of the KMR-II clinical decision support system infrastructure. The system allows to integrate different types of knowledge (semantic, predictive, operational, . . .) regarding a patient and the best way to provide healthcare for them. The system then applies this knowledge to the data definining a patient's clinical history, automating some management actions and acting as a (clinical) decision support system for the patient's provider and their organization.

The knowledge-based core of the application is deployed in a broader service and event oriented architecture, integrating various data sources and services. The application, moreover, provides a unified web-based interface for both patients and providers, allowing them to interact with the system and between each other in a seamless way.

Acknowledgments

The KMR project is supported by Award Number: W81XWH-11-2-0062, administered through the U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick, MD 21702-5014.

The opinions of the authors do not necessarily state or reflect those of their respective employers, including the Department of Navy, the Department of Defense, and the United States Government, and shall not be used for advertising or product endorsement purposes.

The whole platform will be released under a BSD-like open source license in early 2012.

References

1. Bragaglia, S., Chesani, F., Fry, E., , Mello, P., Montali, M., Sottara, D.: Event condition expectation (ece-) rules for monitoring observable systems (2011)
2. Bragaglia, S., Chesani, F., Mello, P., Sottara, D.: A Rule-Based Implementation of Fuzzy Tableau Reasoning. In: RuleML. pp. 35–49 (2010)

3. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
4. Luckham, D.: SOA, EDA, BPM and CEP are all Complementary
5. Meditskos, G., Bassiliades, N.: Clips-owl: A framework for providing object-oriented extensional ontology queries in a production rule engine. Data Knowl. Eng. 70(7), 661–681 (2011)
6. Sottara, D., Mello, P., Proctor, M.: Adding uncertainty to a rete-OO inference engine. Rule Representation, Interchange and Reasoning on the Web pp. 104–118 (2008)
7. Sottara, D., Mello, P., Sartori, C., Fry, E.: Enhancing a production rule engine with predictive models using pmml. In: Proceedings of the 2011 workshop on Predictive markup language modeling. pp. 39–47. PMML '11 (2011)
8. Stevenson, G., Dobson, S.: Sapphire: Generating java runtime artefacts from owl ontologies. In: CAiSE Workshops. pp. 425–436 (2011)

Emergency Services: Process, Rules and Events

Mauricio Salatino, Esteban Aliverti, and Demian Calcaprina

Plugtree
salaboy@gmail.com

Abstract. The Emergency Service Application was built as a blue print for architect applications using the Drools and jBPM5 platform. The main goal of this application is to show how we can model and implement a complete and complex business scenario using declarative approaches like: Business Rules, Business Process and Complex Event Processing. From the technical point of view, this application shows how components like a Rule Engine and Business Process Engine can be integrated with technologies that are widely used in most system architectures. Components - like Distributed Caches, Transactional Frameworks, Messaging Systems, Web Services Stacks and Web Frameworks - can be combined to work around the declarative power of rules, processes and events.

1 Application Scope, Domain and Requirements

The application was created to represent complex scenarios that are being executed by an Emergency Services company that deals with concurrent emergencies within a city. The company needs to solve different situations where different entities need to be coordinated to deal with an emergency situation.

The Emergency Services Application shows how we can provide a tool that helps the company to improve their services by giving them full visibility of their actions, traceability of their resources, suggestions and advice based on the context without sacrificing any degree of flexibility that they need to solve real life situations. One of the most important requirements of the application is related to the the fact that most of the emergencies can be classified under different categories based on their characteristics. We can model guidelines to solve each of these categories but in real life, each emergency will be treated differently, depending on the context, the state of the company and their resources. For each particular situation, different actions will be evaluated, suggested and executed.

Two generic situations will be described by this paper: Heart Attack Emergencies and Fire Related Emergencies. But the application is in no way limited to these two scenarios.

To design the application we gather, understand and formalize the knowledge that the company experts use to drive their activities. The formalization of this knowledge will be introduced in the following sections where the Business Processes, Business Rules and Events modeled for this application are described.

2 Declarative Knowledge Representation

The application uses the concept of procedure to define the set of business processes, business rules and domain specific services that will be used to deal with an emergency. We have defined a set of default procedures that describe the activities that can be executed during specific emergencies. The "Default Heart Attack Procedure" can be used to drive a standard set of activities that needs to be executed each time a person suffers a heart attack. The "Default Fire Emergency Procedure" describes the same but for emergencies that include fire situations where we need to coordinate the firefighters department in order to mitigate the dangerous situation as soon as possible.

Both procedures set up the basic activities that the company needs to execute in each specific situation, but in no way do they limit the company to add, remove or execute more activities in parallel.

Before executing these procedures, the company needs to identify the context of the emergency by executing a set of activities that were designed to pick up the initial information and find out what is happening. These activities are contained in a Generic Emergency Procedure that is executed each time the phone rings in the central offices.

3 Generic Emergency Procedure

This procedure will initiate the information about an emergency and is based on the initial information that is being gathered by the phone operators of the company. Using this information, a suggestion mechanism based on Business Rules will be in charge of suggesting the most appropriate, specific procedures out of all the available procedures.

This generic procedure is driven by the following business process:

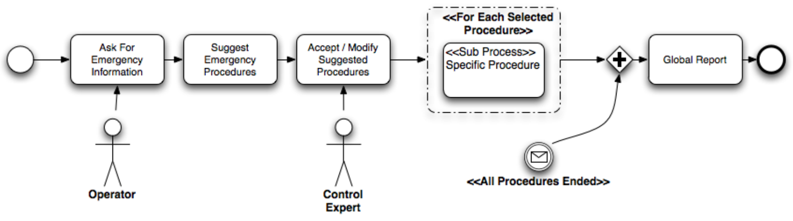

Ask For Emergency Information
Suggest Emergency Procedures
Accept / Modify Suggested Procedures
<<For Each Selected Procedure>>
<<Sub Process>> Specific Procedure
Global Report
Operator
Control Expert
<<All Procedures Ended>>

Fig. 1. Heart Attack Emergency Procedure

The Suggest Emergency Procedures activity inside this business process uses a set of rules to analyze the context of the emergency gathered to suggest a set of procedures that fits with that specific situation. These rules will evaluate the status of the company in order to suggest viable procedures.

Once the Suggestion rules are executed, the control expert in charge reviews the suggestions and modifies or approves the selected procedures. The business process automatically will start each selected procedure in parallel and it will wait until it receives a notification that all the specific procedures have ended.

The main purpose of this generic procedure is to speed up the activities that need to be executed for every emergency that the central offices handle. The suggestion rules help the experts by giving them a clear set of procedures that can be executed based on the current status of the company and also on the contextual and semantic information gathered for each specific situation. Once the procedures are started, a separate group of resources will be used to monitor all the activities executed for each emergency.

4 Default Heart Attack Procedure

As soon as we identify a heart attack situation, the system will automatically suggest to the expert which procedures best fit based on contextual information. If the Default Heart Attack Procedure is selected, the activities described by the following business process will be executed:

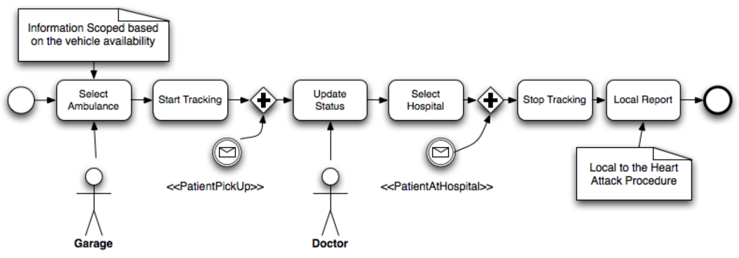


Fig. 2. Fire Emergency Procedure

This sequence of activities and events defines exactly how the company must deal with a Heart Attack situation. Briefly, an ambulance will be selected according to the company status and the patient information and it will be dispatched to the emergency location. Once that ambulance arrives, the Doctor will send an updated report to the central about the situation. This information will be correlated with the emergency location, the location of different hospitals based on distance, and the availability of the health-care services in each hospital to select the most appropriate facility. Once the patient is at the hospital, a report about this procedure will be created.

This business process is enriched by different sets of rules that are being executed in different activities to take automated actions to speed up the service and the human involvement times. For this particular use case, a set of rules is being defined to automatically select the best hospital based on the available information. Taking advantage of the Complex Event Process features provided by the platform, another group of rules are defined to take care of more dynamic and reactive aspects that need to be covered.

The following rule is one of the set of rules created for the hospital selection mechanism:

Listing 1.1. Rule 2

```
rule "Select Closest Hospital"
  ruleflow-group "hospital-selection"
  when
    $pI: WorkflowProcessInstance( $pid : id )
    $emergency: Emergency( $type: type.name )
    $selectedHospital: Hospital() from accumulate (
       $hospital: Hospital() from externalEntities.getAllHospitals(),
         hospitalDistanceCalculator(
           new HospitalDistanceCalculationData(
             $hospital, $emergency )
           )
         )
  then
    String callId = ( (Call) $pI.getVariable("call") ).getId();
    //Send Hospital Selected Message
    MessageFactory.sendMessage(
      new HospitalSelectedMessage( callId, $selectedHospital ) );
end
```

This basic rule calculates the closest hospital to the emergency location iterating the location of all the available hospitals. The available hospitals are being dynamically calculated based on periodical updates reports received in the central.

The following business rule uses the temporal operators to analyze and react based on the patients vital signs. Once the patient is inside the ambulance, all his/her vital signs are sent to the central offices and monitored by a set of rules that are designed to analyze anomalous situations and generate warnings. These warnings are automatic reactions executed by the system when a specific pattern is found in multiple sources of real time events. These warnings can be used to influence the hospital selection, the route to reach the selected hospital, or even trigger new on-demand procedures.

Listing 1.2. Rule 3

```
rule "Patient heart attack pattern"
  when
    ArrayList( $num : size > 7 ) from collect (
      PulseEvent( processed == false, $pulse: value )
        over window:time(1s)
        from entry-point "patientHeartbeats" )
  then
    MessageFactory.sendMessage(
      new PatientMonitorAlertMessage(
        callId, vehicleId,
        "Warning, patient suffering a heart attack ",
        new java.util.Date() ) );
end
```

This simple rule evaluates in real time the events that are coming from a stream called “patientHeartBeats”. If we find more than 7 events per second filtering the values of those particular events, we can say that it is very likely that the patient is having a heart attack. The application provides a configurable module that allows us to set up different devices to be used as input for these events. We have designed a set of bindings for the Wii Remote Control Accelerometer, the IPhone Accelerometer and Android Devices Accelerometers that can be plugged as event sources. Based on the values that are being sent by these devices, the rules will react if a pattern is found.

5 Default Fire Emergency Procedure

This procedure will be executed each time that the company needs to deal with a fire situation. Once again, a set of business processes and business rules will compose this procedure. For this procedure, we will analyze a business process that describes a more dynamic set of activities that needs to be executed.

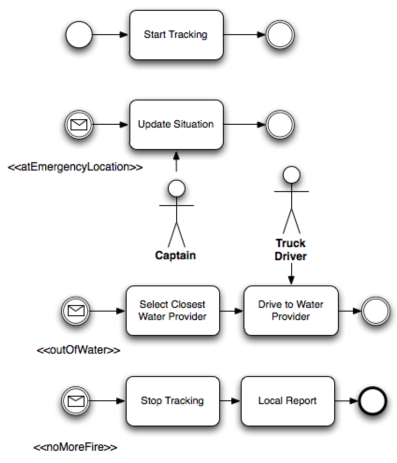


Fig. 3. High Level Architecture

This more unstructured process allows us to represent a situation where we send one or more fire trucks to a fire emergency. Each truck will have a limited amount of water that can be recharged in the fire departments. This process is being driven by the events that are being received in the central offices, which allow us to coordinate if we need more trucks; or if the situation is under control, we can reduce the number of trucks that we are using.

For this procedure a set of rules is defined to control the water tanks and select the closest water provider; a planning algorithm is also used to calculate the initial amount of trucks required to deal with the fire situation.

6 Inferences, Correlation, Aggregation and Dynamic Knowledge Composition

In order to provide the flexibility required to handling these complex situations we need to provide generic mechanisms to gather the knowledge required to deal with each emergency on demand. This application was designed to allow company experts to pick different business knowledge assets to solve specific situations. Composing different pieces knowledge into a single runtime will allow us to build very flexible and reactive services that can solve high and low level situations. These knowledge runtimes will be populated with the information that belongs to the specific situation. Once the runtime is created and populated with information, the rules, processes and events will be analyzed and the correspondent actions will be triggered.

Each knowledge runtime will be an isolated entity that can be distributed in different nodes of a computer grid allowing the application to scale. Each of these knowledge runtimes will provide a context smart enough to solve the specific situation that causes its creation.

7 Architectural Overview

This application was created to take advantage of different technologies to solve very specific problems. All these technologies are being used to demonstrate how we can solve all the technical problems that arise when we try to provide a solution that needs to drive a company. The following technical components are being used to solve infrastructural problems such as scalability and robustness, delegating the business logic and business definitions to the Business Rule Engine and Business Process Engine.

Current technical components that are being integrated to the application:

– Distributed Cache (Infinispan)
– NOSQL Graph Based Database (Neo4J)
– Query and Graph Transversal frameworks (Gremlin and Chyper)
– Reliable Messaging System (HornetQ)
– Web Frameworks for Presentation Layer (FreeMarker, Spring MVC)
– Interaction Component for dynamic form builder (Smart Tasks)

The architecture of the application was created with the concept of distribution in mind. Usually the problems that we want to solve using this approach are extremely complex and can involve huge amounts of data, therefore each knowledge runtime can be instantiated in different physical or virtual machines. Using different techniques, we can coordinate and monitor these knowledge runtimes so decoupling them in order to improve performance and scalability. Most of the interactions are being handled by messaging queues which allow us to configure the reliability of the channels completely decoupled from the problems that we are trying to solve.

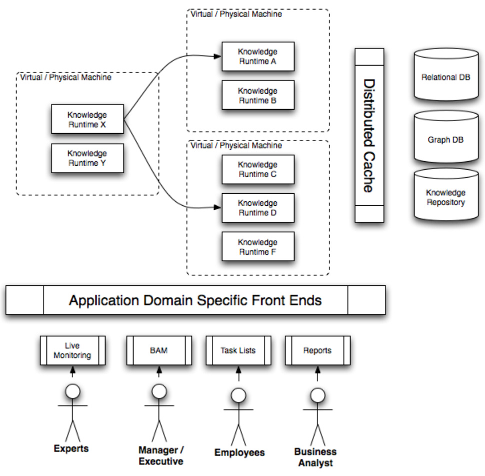


Fig. 4. Caption here

8 Conclusions

The application right now has as the main goal to show how can we mix the declarative power of rules, business processes and complex event processing to create application that can be understood by business people. The future of the application will be driven by the Drools and jBPM5 projects that are continuously evolving. Ontologies, smart and distributed agents, more powerful domain specific languages, predictive models and planning algorithms will be included as part of the design of the application architecture to test and demonstrate how all these features can be complemented to provide a more flexible platform to build applications.

Older versions of this application were presented in six international events during 2010 and 2011. For more information about the application, you can browse the main developers blogs [3, 1, 2] . All the application features are open to the community and we encourage people to participate from the project to learn about these technologies. The application source code is fully available to download [4] and is licensed under the Apache Software License 2.0.

References

1. Aliverti, E.: Blog. `http://ilesteban.wordpress.com/` (2011)
2. Calcaprina, D.: Blog. `http://dcalca.wordpress.com/` (2011)
3. Salatino, M.: Blog. `http://salaboy.wordpress.com/about/` (2011)
4. Salatino, M.: Emergency Services Application. `https://github.com/Salaboy/emergency-service-drools-app` (2011)

~~Processing of Complex Stock Market Events Using~~ Background Knowledge

Kia Teymourian, Malte Rohde, and Adrian Paschke

Corporate Semantic Web Research Group
Institut for Computer Science, Freie Universität Berlin
`http://www.inf.fu-berlin.de/groups/ag-csw/`
{kia, malte.rohde, paschke}@inf.fu-berlin.de

Abstract. Usage of background knowledge about events and their relations to other concepts in the application domain, can improve the quality of event processing. In this paper, we describe a system for knowledge-based event detection of complex stock market events based on available background knowledge about stock market companies. Our system profits from data fusion of live event streams and background knowledge about companies which are stored in a knowledge base. Users of our system can express their queries in a rule language which provides functionalities to specify semantic queries about companies in the RDF SPARQL language for querying the external knowledge base and combine it with event data streams. Background makes it possible to detect stock market events based on companies attributes and not only based on syntactic processing of stock price and volume.[1]

Keywords: Complex Event Processing, Knowledge-Based Complex Event Processing

1 Motivation

The reality in many business organizations is that some of the important complex events can not be used in process management because they are not detected from the workflow data and the decision makers can not be informed about them. Detection of events is one of the critical factors for the event-driven systems and business process management. Semantic models of events can improve event processing quality by using event metadata in combination with ontologies and rules (knowledge bases). The successes of the knowledge representation research community in building standards and tools for technologies such as formalized and declarative rules are opening novel research and application areas. One of these promising application areas is *semantic event processing*.

Several complex event processing systems are already proposed and developed [4]. Existing methods for event processing can be categorized into two main categories, logic-based approaches and non-logic-based approaches [9]. One of the logic-based

[1] This work has been partially supported by the “InnoProfile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions.

approaches is introduced in [8] which proposes a homogeneous reaction rule language for complex event processing. It is a combinatorial approach of event and action processing, formalization of reaction rules in combination with other rule types such as derivation rules, integrity constraints, and transactional knowledge.

Examples of commercial CEP products are: TIBCO BusinessEvents[2], Oracle CEP [3], Sybase CEP[4] Some of these existing CEP systems can integrate and access external static or reference data sources. However, these systems do not provide any inferencing on external knowledge bases and do not consider reasoning on relationships of events to other non-event concepts. Previously, we proposed in [11, 10] a new approach for the Semantic enabled Complex Event Processing (SCEP). We claim that semantic models of events can improve the quality of event processing by using event stream data in combination with background knowledge about events and other related concepts in the target application domain. We described how to semantically query and filter events and how to formalize complex event patterns based on a logical knowledge representation interval-based event/action algebra, namely the interval-based Event Calculus [5–7]. Other related approaches like [2, 3] are also relevant for our approach, but these approaches are not combining the complex event detection based on event correlations and detection semantics with the relationships between events and other non-event concepts/individuals in the background knowledge base.

In this paper, we describe a demonstration system for knowledge-based complex event processing to extract complex stock market events using live stock market events and background knowledge about companies and other related concepts. Fusion of event data streams and background knowledge can build up a more complete knowledge about events and their relationships to other concepts. The rest of this paper is organized as follows. In Section 2, we focus on use case scenario and show which kind of complex events can be detected using a background knowledge base. The use case is described by providing a concrete example. Section 3 describes our method for knowledge-based event processing which includes methods for data fusion with the background knowledge base. In Section 4 we describe our demonstration system in details and provide an other example.

2 Use Case Scenario

Stock market brokers gather information from different parties and monitor different stock market graphs to be able to make the best possible stock market handling strategy. They have to be able to make the right decision at the right time. They get all of the "chunks" of information and are face with the difficult tasks of mentally/intuitively combining them together, enriching/aggregating and inferencing on them. The vision of our system is to have real-time information processing support system for stock market brokers.

Consider that Mr. Smith is a stock broker and has access to stock exchange event stream like listed in Listing 1.1. He is interested in special kinds of stocks and would

[2] `http://www.tibco.com/`
[3] `http://www.oracle.com`
[4] `http://www.sybase.de`

like to be informed if there are some interesting stocks available for purchasing. His special interest or his special stock handling strategy can be described in a high level language which describes his expressed interest using background knowledge about companies.

Mr Smith would like to start the following query on the event stream: Buy Stocks of Companies, who have *production facilities in Europe* and produce products from *iron* and have more than *10,000 employees* and are at the moment in *reconstruction phase* and their price/volume *increased stable* in the *past 5 minutes*.

Listing 1.1: Stock Exchange Event Stream

```
{ {(Name, ''GM'')(Price , 20.24)(Volume, 8,835)}, {(Name, ''SAP'')(Price , 48.71)(
    Volume, 8,703)}, {(Name, ''MSFT'')(Price , 24.88)(Volume, 46,829)} , ... }
```

As we can see the above query cannot be processed without having background knowledge which can define the used concepts in this query. Mr. Smith needs an intelligent system which can use background knowledge about companies like listed in Listing 1.2. This background knowledge should be integrated and processed together with event data stream in real-time manner so that interesting complex events can be timely detected.

We can also consider that Mr. Smith works for a company and may need to share this knowledge base with other brokers. Each of the brokers may be able to gather new information about companies and update this knowledge base, e.g., the Opel company is not in reconstruction phase, or the Apple company has a new chief executive officer.

Listing 1.2: An Excerpt of a Knowledge Base about Companies

```
(OPEL, belongsTO, GM ), (OPEL, isA, automobilCompany),
(automobilCompany, build, Cars), (Cars, areFrom, Iron),
(OPEL, hasProductionFacilitiesIn, Germany), (Germany, isIn, Europe),
(OPEL, isA, MajorCorporation), (MajorCorporation, have, over10,000employees),
(OPEL, isIn, reconstructionPhase), ...
```

3 Semantic Enabled Event Processing

The fusion of background knowledge with the data from an event stream can help the event processing engine to know more about incoming events and their relationships to other related concepts. We propose to use an external knowledge base which can provide background conceptual and assertional information about the events as it is shown in Figure 1. This means that events can be detected based on reasoning on their type hierarchy relationships, or temporal/spatial relationships. It can also be based on their connections to other relevant concepts from the domain, e.g., relationship of a stock price to the products or services of a company.

The realization of SCEP is a challenging task, because it should provide real-time processing and high scalability. The naïve approach for SCEP might be a storage-based approach. This means to store all of the background knowledge in knowledge bases and start pulling the knowledge base, every time when a new event comes into the system, and then process the result from the external knowledge base with event data. This approach may have several problems when the throughput of the event stream is high,

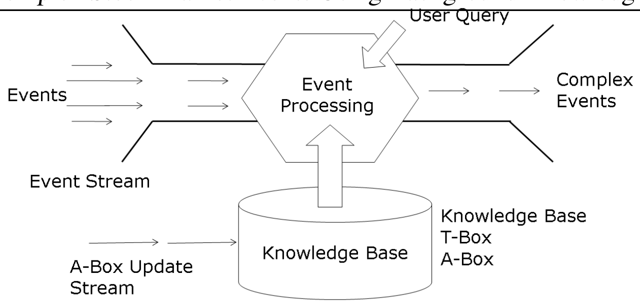


Fig. 1: High Level Architecture of Semantic-Enabled Complex Event Processing

the size of background knowledge is high, or even when expressive reasoning should be done on the knowledge base.

Event Query Pre-Processing:

We propose to do an Event Query Pre-Processing (EQPP) before the event processing is down on the event stream. In this approach, the original complex event query can be pre-processed by use of a knowledge base and rewritten into a single *new query*. This *new query* is a query which can be syntactically processed only with the knowledge from the event stream and without an external knowledge base.

In this paper, we are addressing a simple pre-processing of event queries and illustrates the potential of such a pre-processing approach for SCEP. In our method the user query is pre-processed and rewritten into a single new query which has the same semantic meaning as the original one. The advantage of this method is that the user can define event queries in a high level abstraction view and does not need to care about some details, e.g., the user can specify queries like *"companies who produce products from iron"* and does not need to know all of the products of companies which might not be simple for humans to remember. One other advantage is that the SCEP system is able to provide real-time event processing as events arrive into the system because the external reasoning on knowledge base is done in advance. On the other side, one disadvantage of this approach is that the query needs to be updated each time when the knowledge base is changed (or when a part of the KB is changed). We assume that in most of the use cases the rate of background knowledge updates is not very high as the rate of the main event stream.

4 System Architecture and Demonstration

In this section we describe the architecture of our system and describe how our demonstrator works. The architecture of our implementation is shown in Figure 2, it shows different components of our system, event procudcer, APIs and user interfaces, main event processing engine and a knowledge component base which can be used to store background knowledge and doing reasoning on background knowledge base.

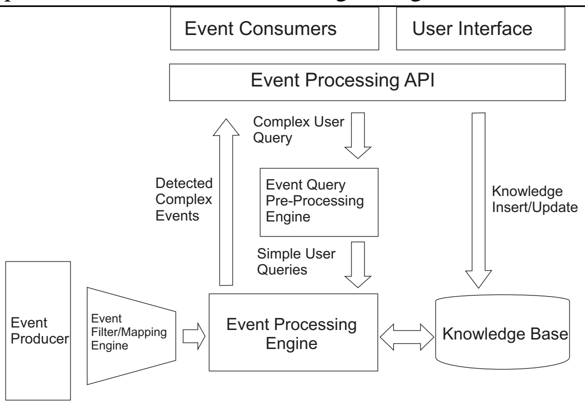


Fig. 2: Architecture of our SCEP Implementation

As main event processing engine, we use Prova[5] as a rule-based execution which can be used as event processing engine and as reaction rule language formalization. Prova uses reactive messaging[6], reaction groups and guards[7] for complex event processing. Multiple messages can be received using revMult(XID, Protocol, Destination, Performative, Payload) ; XID, a conversation id of the message; Protocol, message passing protocol; Destination, an endpoint; Performative, message type; Payload, the content of message. Prova implements a new inference extension called literal guards. During the unification only if a guard condition evaluates to true, the target rule will proceed with further evaluation.

We implemented a Prova *sparql_select* built-in[8] to run SPARQL queries from Prova which can start a SPARQL query from inside Prova on an RDF file or a SPARQL endpoint. This Prova buit-in can use results which come from the SPARQL query and use them inside Prova.

During the processing of a SPARQL query inside Prova rules, the rule engine sends the embedded SPARQL query to the triple store and gets the results back. Afterwards it waits for incoming events to process. It processes the sequence of events using the provided results from the knowledge base. The SPARQL_select has the following syntax: sparql_select(QueryString, [SetOfOutputVariables], [SetOfInputVariables], ServiceEndpoint). The set of output variables are the results which come from the SPARQL query, the set of input variables provide the possibility to replace variables

[5] Prova, ISO Prolog syntax with extensions `http://prova.ws` , July 2011

[6] Prova Reactive Messaging `http://www.prova.ws/confluence/display/RM/Reactive+messaging` , July 2011

[7] Event Processing Using Reaction Groups `http://www.prova.ws/confluence/display/EP/Event+processing+using+reaction+groups`, July 2011

[8] Source codes for Semantic Web extensions in Prova 3 can be found in `https://github.com/prova/prova/tree/prova3-sw` , October 2011

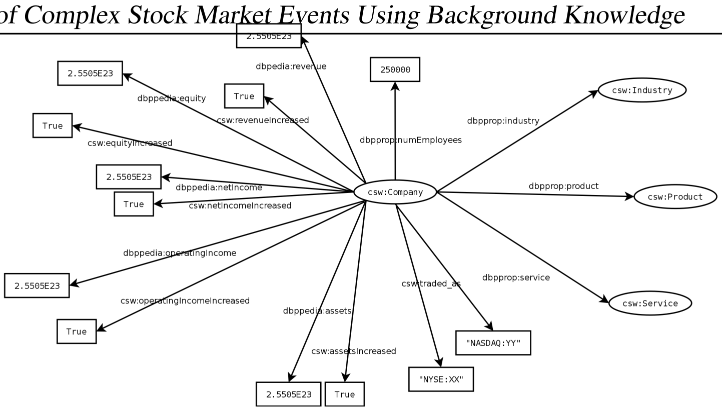


Fig. 3: A Simple Company Ontology

in SPARQL string which are starting with $ with variables in Prova, and the service endpoint is the SPARQL endpoint.

We created a light-weight ontology for companies as shown in Figure 3. We analyzed the available data from DBpedia, about 500 Companies from the S&P 500 Index[9], we managed to extract about 302 different properties referring to companies. We included only the most important properties in our ontology. Our system is able to query DBpedia and other linked data endpoints directly. The stocks of company can be detected based on a set of properties for the concept "Company" and available knowledge on the background knowledge base. The knowledge base might be updated by the users but with a very lower frequency than the stock market events.

Listing 1.3: Prova Example for Semantic Event Detection.

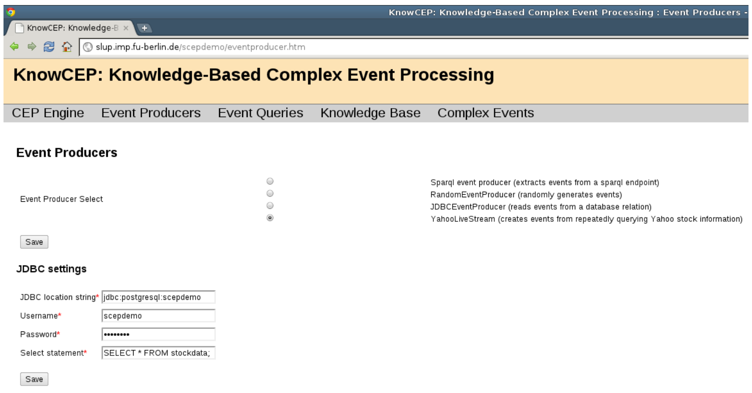
```
:- eval(server()).

server() :-
sparqlrule(CompanySymbol, CompanyEmployees),
rcvMult(XID, Protocol, Sender, event,
{time->Time, symbol->Symbol, name->Name, lastprice->Lastprice, volume->Volume,
     high->High, low->Low})
[Symbol = CompanySymbol, CompanyEmployees > 5000],
sendMsg(XID, Protocol, Sender, testrule, {name->Name}).

sparqlrule(CompanySymbol, CompanyEmployees) :-
  Query = ' PREFIX DBPPROP: <http://dbpedia.org/property/>
           PREFIX DBPEDIA: <http://dbpedia.org/resource/>
           PREFIX CSW: <http://corporate-semantic-web.de/scep/>
     SELECT ?symbol ?employees WHERE {
           ?company DBPPROP:industry DBPEDIA:Computer_software .
           ?company CSW:traded_as ?symbol .
           ?company DBPPROP:numEmployees ?employees .  }',
sparql_select(Query, [symbol(CompanySymbol), employees(CompanyEmployees)], [], '
     http://localhost:8890/sparql').
```

[9] http://www.standardandpoors.com/

The Listing 1.3 provides an excerpt of the Prova code example which illustrate our implementation. In this query, a broker is interested in software companies which have more than 5000 employees.

The complete pre-processing step should be updated on the knowledge base, whenever there is a change in the knowledge base, e.g., if new products are added to the product lists of a company. In many use cases like ours, the frequency of such updates can be considered to not be very high. Here, one useful approach is to implement the updates also in an event-based manner, if any relevant changes are done on the knowledge base a notification informs the event processing engine to update the event query.

Prova follows a workflow paradigm in event processing. It is possible to use Prova for the realization of Plan-based complex event detection as proposed in [1]. However, in our experiments we assume that all of the event streams come to a central processing point. Our system shows clearly that the EQPP can achieve a better performance than the naïve storage-based (or pulling) approach. It also demonstrates that the EQPP approach is an applicable approach for the above described use case. It shows also that the scalability of SCEP systems has five different dimensions; 1. Discharge rate of events,2. Number of rules in main memory, 3. Number of triples in the knowledge base (amount of knowledge), 4. Rate of knowledge updates,5. Expressive level of reasoning on background knowledge. Our demonstration system can be found on the Web at `http://slup.imp.fu-berlin.de/scepdemo/`.


KnowCEP: Knowledge-Based Complex Event Processing : Event Producers -
KnowCEP: Knowledge-B
slup.imp.fu-berlin.de/scepdemo/eventproducer.htm
KnowCEP: Knowledge-Based Complex Event Processing
CEP Engine
Event Producers
Event Queries
Knowledge Base
Complex Events
Save
Save

Event Producers

Event Producer Select
- ○ Sparql event producer (extracts events from a sparql endpoint)
- ○ RandomEventProducer (randomly generates events)
- ○ JDBCEventProducer (reads events from a database relation)
- [✓ YahooLiveStream (creates events from repeatedly querying Yahoo stock information)]

JDBC settings

JDBC location string*	jdbc:postgresql:scepdemo
Username*	scepdemo
Password*	•••••••••
Select statement*	SELECT * FROM stockdata;

Fig. 4: Screenshot of KnowCEP: Knowledge-Based Complex Event Processing System

The user interface of our demonstration system consist of four parts, each of them have different functionalities; The first part is *CEP Engine Status*, a user can see the current status of the system, can start or stop the CEP engine. The second part is called *Event Query*, a user can give an event query in form of a Prova rule. The third part is *Knowledge Base* of the system, in this part a user can specify a SPARQL endpoint as an external knowledge base, can send SPARQL queries to end point and get the results back to the system, the user can also post and add RDF data to the external RDF store.

The next component is the configuration of the event sources, as shown in Figure 4, users can select between different event producers, live stock market data stream, stored stream from a database, or randomly generated stock market events for demonstration purposes. After the configuration of the CEP system, it can be started from the main system menu.

5 Conclusion and Outlook

We described our initial work on semantic event processing and semantic pre-processing of event queries, and illustrated the potential of this approach by means of a demonstration. Our future steps are to work on the semantics of event processing languages and define which semantics can be adequate for Complex Event Processing. Furthermore, we are working on an algorithm for rewriting of complex event queries to several simple queries which can be distributed on an event processing network to achieve high performance and scalability.

References

1. Mert Akdere, Uğur Çetintemel, and Nesime Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1:66–77, August 2008.
2. Liming Chen and Chris D. Nugent. Ontology-based activity recognition in intelligent pervasive environments. *IJWIS*, 5(4):410–430, 2009.
3. Claudia d'Amato, Nicola Fanizzi, Bettina Fazzinga, Georg Gottlob, and Thomas Lukasiewicz. Combining semantic web search with the power of inductive reasoning. In Amol Deshpande and Anthony Hunter, editors, *SUM*, volume 6379 of *Lecture Notes in Computer Science*, pages 137–150. Springer, 2010.
4. Alessandro Margara and Gianpaolo Cugola. Processing flows of information: from data stream to complex event processing. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS '11, pages 359–360, New York, NY, USA, 2011. ACM.
5. A. Paschke. Eca-lp / eca-ruleml: A homogeneous event-condition-action logic programming language. In *RuleML-2006*, Athens, Georgia, USA, 2006.
6. Adrian Paschke. Eca-ruleml: An approach combining eca rules with temporal interval-based kr event/action logics and transactional update logics. *CoRR*, abs/cs/0610167, 2006.
7. Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated sla management. *Decis. Support Syst.*, 46(1):187–205, 2008.
8. Adrian Paschke, Alexander Kozlenkov, and Harold Boley. A homogeneous reaction rule language for complex event processing. *CoRR*, abs/1008.0823, 2010.
9. Kay-Uwe Schmidt, Darko Anicic, and Roland Stühmer. Event-driven reactivity: A survey and requirements analysis. In *SBPM2008: 3rd international Workshop on Semantic Business Process Management in conjunction with the 5th European Semantic Web Conference (ESWC'08).* CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073), June 2008.
10. Kia Teymourian and Adrian Paschke. Semantic rule-based complex event processing. In *RuleML 2009: Proceedings of the International RuleML Symposium on Rule Interchange and Applications*, 2009.
11. Kia Teymourian and Adrian Paschke. Towards semantic event processing. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1–2, New York, NY, USA, 2009. ACM.

Combining Rules and Ontologies with Carneades

Thomas F. Gordon[1]

Fraunhofer FOKUS, Berlin
thomas.gordon@fokus.fraunhofer.de

Abstract. The Carneades software system provides support for constructing, evaluating and visualizing arguments, using formal representations of facts, concepts, defeasible rules and argumentation schemes. This paper illustrates how rules and ontologies can be combined in Carneades with a prototype legal application for analyzing open source software license compatibility issues in particular cases.

Introduction

This paper illustrates how rules and ontologies can be used in combination in the Carneades argumentation system [14] with examples from a prototype legal application for analyzing open source license compatibility issues [15]. As Bing [3], Fiedler [7], McCarty [17] and many others have noted, legal argumentation is not primarily deductive, but rather a modeling process of shaping an understanding of the facts, based on evidence, and an interpretation of the legal sources, to construct a theory for some legal conclusion [3]. The parties in a legal dispute construct competing theories and argue about their relative merits. Carneades is designed to support all the steps in this process of theory construction, argumentation and evaluation.[1]

The Carneades software system is based on a well-founded formal model of structured argumentation with support for proof burdens and standards [9,10], now called Carneades Argument Evaluation Structures (CAES). It has been formally proven that the Carneades model of argument is a specialization of both Prakken's ASPIC+ model of structured argumentation [8] and Brewka's Abstract Dialectical Frameworks [5] and thus an instantiation of Dung's Abstract Argumentation Framework [4,8]. Carneades has also been shown by Governatori to be closely related to Defeasible Logic [16]. A formal model of abduction in Carneades argument evaluation structures has been developed [2], which is useful for identifying relevant issues and computing minimal sets of statements, called positions, which, if proven, would make some goal statement acceptable (in) or not acceptable (out) in a stage of a dialogue.

Building on this formal foundation, the Carneades software provides a number of tools for interactively constructing, evaluating and visualizing arguments, as well as computing positions. Arguments are constructed using formalizations of facts, concepts, defeasible rules and argumentation schemes [11,12]. Facts and concepts are represented using the Web Ontology Language (OWL), an XML schema for description logic [1], a subset of first-order logic and thus with a monotonic (strict) entailment relation. Legal rules and argumentation schemes [18] are both modeled as defeasible inference rules, represented in the Legal Knowledge Interchange Format (LKIF) [6]. The rules of alternative, competing theories of the law can be included in a single model.

A combination of forwards and backwards reasoning is used to construct arguments: a description logic reasoner constructs the deductive closure of the concepts and facts in a forwards manner; the Carneades rules engine uses backwards reasoning to apply the defeasible inference rules in a goal-directed and stratified way to the deductive closure of the description logic theory of facts and concepts. The LKIF rule language has been extended to provide a way to declare the domain of variables using predicates defined in OWL, similar to the way variables are typed in programming languages. These domain declarations provide important control information that enables the rule engine to iterate over instances of the domains to more efficiently instantiate the rules.

[1] http://carneades.github.com

In addition to the arguments constructed automatically from a knowledge-base of facts, concepts and rules, arguments can manually entered into the system by the user. These arguments can be completely ad hoc or instantiations of argumentation schemes. The Carneades system currently includes a library of about 20 of Walton's most important argumentation schemes along with a software assistant which steps the user through the process of selecting and instantiating schemes.

As the arguments are constructed and edited, they are visualized in an argument map [13]. The graphical user interface, called the Carneades Editor, supports argument evaluation by providing tools to accept and reject statements, assign proof standards and weigh arguments. After every modification, the underlying computational model of argument is used to update and visualize the acceptability status of statements in the map. The differential legal effects of competing theories can be analyzed by assuming their rules to be valid and then checking how this effects the acceptability of issues of interest in the argument map. Moreover Carneades provides a *find positions* assistant which can be used to abduce theories with desired legal effects.

The rest of this paper show how ontologies and rules can be used in combination in Carneades with examples from the prototype legal application for analyzing open source license compatibility issues. We start with examples from a simple OWL ontology for describing software licenses and use and derivation relationships between works of software. Next we show how to use the ontology to model the facts of a case. We then show how to model some rules of copyright law in LKIF, focusing on the issue of whether linking to a software library produces a derivative work.

Concepts and Facts

Carneades uses the Web Ontology Language (OWL), a World Wide Web standard XML schema for representing and interchanging description logic knowledge bases. These knowledge bases have two parts, for concepts (TBox) and facts (ABox). The top-level concepts, called *classes* in OWL, for our application are CopyrightLicense, CopyrightLicenseTemplate, LegalEntity, LicenseTerm and Work. The Work class is for all works protectable by copyright. There is a SoftwareEntity subclass of Work, intended to cover all kinds of software artifacts.

The main property of software entities of interest for license compatibility issues is the isDerivedFrom property, expressing that one entity has been derived from another. The ontology includes properties for representing various ways that software can use other software, such compiledBy and linksTo.

The software ontology was used to model an example software project, roughly based on the current version of the Carneades system.

Rules

Description logic (DL) is semantically a decidable subset of first-order logic. This means that the inferences of description logic reasoners are *strict*: if the axioms of a DL knowledge base are true in some domain, then all of the inferences made by a (correctly implemented) DL reasoner are necessarily also true, without exception. While DL is very powerful and useful, monotonic logics are not sufficient for modeling legal rules, such as the rules of copyright law, in a maintainable and verifiable way, isomorphic with the structure of legislation and regulations. Legislation is typically organized as general rules subject to exceptions. Arguments made by applying legal rules are *defeasible*. Their conclusions can be defeated with better counterarguments. Various legal rules may conflict with each other. Theses conflicts are resolved using legal principals about priority relationships between rules, such as the principal of *lex superior*, which gives rules from a higher authority, such as federal law, priority over rules from a lower authority, such as state law. These properties of legal rules are well known in AI and Law and have been studied extensively. References are omitted for lack of space.

Thus we model legal rules using a defeasible rule language which has been developed especially for this purpose, as part of the Legal Knowledge Interchange Format (LKIF), and use description logic (OWL more specifically) for more limited purposes: 1) to declare the language of unary and binary predicate symbols and 2) to make assertions about these predicates, using DL axioms, which are judged to be universally true and beyond dispute in the domain.

Here we illustrate the LKIF rule language by modeling two interpretations of the concept of a derivative work in copyright law. We begin with the general rule that the copyright owner of software may license the software using any license template he chooses.

```
<rule id="DefaultLicenseRule">
    <head>
        <s pred="&oss;mayUseLicenseTemplate">
            <v>SE</v> may be licensed using
            the <v>T</v> template
        </s>
    </head>
</rule>
```

Since LKIF is an XML schema, rules are represented in XML. This particular rule has a head (conclusion) but no body (conditions). Even though the rule has no conditions, inferences made using this rule are not necessarily or universally true, but remain defeasible. We will make use of this feature to express exceptions to this general rule below.

The rule has been assigned an identifier, DefaultLicenseRule, which may be used to formulate statements about the rule. That is, rules are reified and may be reasoned about just like other objects.

The predicate symbol of the statement (proposition) in the head of the rule is specified using the *pred* attribute. Its value can be the name of a class or property in a OWL ontology, as in this example. The ? entity reference refers to the ontology, using its URI.

Declaring predicate symbols in ontologies makes it possible to divide up the model of a complex domain theory into modules, with a separate LKIF file for each module. OWL provides a way to import the classes and properties of other OWL files, recursively. Similarly, LKIF provides a way to import both LKIF and OWL files. OWL makes it easy to manage predicate symbols across the boundaries of modules and to make sure that symbols in different modules refer to the same class or property when this is desired.

The XML syntax for rules in LKIF is rather verbose and not especially readable. Fortunately, it is easy to write programs for converting XML into more readable formats. Moreover, XML structure editors exist which use style sheets to enable authors to edit XML documents directly in a more readable form. Using this feature, the above rule can be displayed in the editor as follows:

```
rule DefaultLicenseRule
    head SE may be licensed using the T template
```

We will use this more readable format for displaying LKIF rules in the remainder of this article. Next let us formulate an exception to the general rule that any license template may be used for reciprocal licenses:

```
rule ReciprocityRule
    head
        not: SE1 may be licensed using the T1 template
    domains
        SE1 uses SE2
        SE2 has license L
    body
        L is reciprocal
        SE1 is derived from SE2
        unless exists T2 : L is an instance of template T2
            such that T1 is compatible with T2
```

This reciprocity rule states that a software entity, SE1, may not be licensed using a license template, T1, if the software is derived from another software entity, SE2, licensed using a reciprocal license, L, unless L is an instance of a template license, T2, which is compatible with T1. The use of domains in this rule provides control information to make use of forward chaining in the description logic reasoner, as discussed in the introduction. Notice that the conclusion of the rule is negated and that the last condition of the rule expresses a further exception, using an *unless* operator.

These two rules illustrate two kinds of exceptions. In argumentation terms, arguments constructed using the ReciprocityRule *rebut* arguments constructed using the DefaultRule and arguments which make use of the explicit exception of the ReciprocityRule, by showing that the licenses are compatible, *undercut* the reciprocity argument.

Let us end this brief overview with rules modeling two conflicting views about whether or not linking creates a derivative work. According to the lawyers of the Free Software Foundation, linking does create a derivate work. Lawrence Rosen, a legal expert on open source licensing issues, takes the opposing point of view and argues that linking per se is not sufficient to create derivate works.

```
rule FSFTheoryOfLinking
    head
        SE1 is derived from SE2
    body
        SE2 is a software library
        SE1 is linked to SE2
        The FSF theory of linking is valid

rule RosenTheoryOfLinking
    head
        not: SE1 is derived from SE2
    body
        SE2 is a software library
        SE1 is linked to SE2
        The Rosen theory of linking is valid
```

The last condition of each of these rules requires that the interpretation of copyright law represented by the rule is legally valid. Making this condition explicit enables us to argue about which theory of linking is correct, to compare the effects of these two theories on particular cases, and to use abduction to derive positions about which theory to prefer.

Conclusion

We have illustrated how ontologies and rules can be used together in the Carneades argumentation system with a prototype legal application for analyzing software licensing issues. To our knowledge, no other argumentation or rule-based system currently provides the combination of tools required for this application: 1) automatic argument construction from a knowledge base of strict and defeasible rules; 2) argument mapping; 3) argument evaluation; 4) interactive construction of arguments using argumentation schemes; 5) exploration of effects of alternative legal theories; and 6) computation of positions, using abduction. The source code of the application is freely available, as open source software.

The current user interface is a desktop application, written in Java using the Swing user interface library. Work is in progress on a web version of Carneades. The user interface of this web version is a Rich Internet Application (RIA) implemented using only World-Wide-Web Concortiums standards, in particular XML and Javascript (AJAX). Argument graphs in the web version are rendered using Scalable Vector Graphics (SVG), another W3C standard.

Acknowledgements

This work was partially funded by the European projects Qualipso (IST-FP6–034763) and IMPACT (IST-FP7–247228). Earlier version of this paper were presented at the Jurix 2010 workshop on Modeling Legal Cases and Legal Rules and the Thirteenth International Conference on Artificial Intelligence and Law (ICAIL 2011).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., eds. *The Description Logic Handbook — Theory, Implementation and Applications.* Cambridge University Press, 2003.
2. Ballnat, S. and Gordon, T.F. Goal Selection in Argumentation Processes — A Formal Model of Abduction in Argument Evaluation Structures. *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, IOS Press (2010), 51–62.
3. Bing, J. Uncertainty, Decisions and Information Systems. In C. Ciampi, ed., *Artificial Intelligence and Legal Information Systems.* North-Holland, 1982.
4. Brewka, G., Dunne, P.E., and Woltran, S. Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI–2011)*, (2011), in press.
5. Brewka, G. and Gordon, T.F. Carneades and Abstract Dialectical Frameworks: A Reconstruction. *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, IOS Press (2010), 3–12.
6. ESTRELLA Project. The Legal Knowledge Interchange Format (LKIF). 2008.
7. Fiedler, H. Expert Systems as a Tool for Drafting Legal Decisions. In A.A. Martino and F.S. Natali, eds., *Logica, Informatica, Diritto.* Consiglio Nazionale delle Richere, Florence, 1985, 265–274.
8. Gijzel, B.V. and Prakken, H. Relating Carneades with abstract argumentation. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI–2011), (2011), in press.
9. Gordon, T.F., Prakken, H., and Walton, D. The Carneades Model of Argument and Burden of Proof. *Artificial Intelligence 171*, 10–11 (2007), 875–896.
10. Gordon, T.F. and Walton, D. Proof Burdens and Standards. In I. Rahwan and G. Simari, eds., *Argumentation in Artificial Intelligence.* Springer-Verlag, Berlin, Germany, 2009, 239–260.
11. Gordon, T.F. and Walton, D. Legal Reasoning with Argumentation Schemes. *12th International Conference on Artificial Intelligence and Law (ICAIL 2009)*, ACM Press (2009), 137–146.
12. Gordon, T.F. Constructing Arguments with a Computational Model of an Argumentation Scheme for Legal Rules – Interpreting Legal Rules as Reasoning Policies. *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, (2007), 117–121.

13. Gordon, T.F. Visualizing Carneades Argument Graphs. *Law, Probability and Risk 6*, 2007, 109–117.
14. Gordon, T.F. An Overview of the Carneades Argumentation Support System. In C.W. Tindale and C. Reed, eds., *Dialectics, Dialogue and Argumentation. An Examination of Douglas Walton[2BC?]s Theories of Reasoning.* College Publications, 2010, 145–156.
15. Gordon, T.F. Report on a Prototype Decision Support System for OSS License Compatibility Issues. 2010.
16. Governatori, G. On the Relationship between Carneades and Defeasible Logic. *Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL–2011)*, (2011).
17. McCarty, L.T. Some Arguments About Legal Arguments. *International Conference on Artificial Intelligence and Law*, (1997), 215–224.
18. Walton, D., Reed, C., and Macagno, F. *Argumentation Schemes.* Cambridge University Press, 2008.

An SBVR Editor with Highlighting and Auto-completion

Alexandros Marinos[1], Pagan Gazzard[1], Paul Krause[2]

Rulemotion Ltd., Surrey Technology Centre,
30 Occam Rd., Surrey Research Park,
GU2 7YG, Guildford, Surrey, United Kingdom
{pagan, alexandros}@surrey.ac.uk

Department of Computing, FEPS, University of Surrey,
GU2 7XH, Guildford, Surrey, United Kingdom
p.krause@surrey.ac.uk

Abstract. This paper presents the implementation of an SBVR editor. Our editor supports automatic highlighting and offers auto-completion suggestions as the model is being typed. These capabilities have been designed to reduce the overhead in the writing of SBVR models as much as possible. The editor has been built with web technologies, and can run in any browser.

Keywords: SBVR, editor, OMeta, highlighting

1 Introduction

Semantics of Business Vocabulary and Rules (SBVR) [1] is a modeling language standardised by Object Management Group and is the result of many years of research by the Business Rules Group. SBVR holds a lot of promise due to its being completely declarative, having a solid logical foundation, and the possibility of representing its rules in a subset of English, readable by an untrained user. The standard has been in development for several years; however the tooling support seems to be lagging.

Anyone wanting to write rules as they are seen in the standard is expected to type them and highlight them by hand, possibly using Microsoft Word templates or something of that kind. This means that highlighting is left up to the human mind to determine, which becomes less and less reliable a method the more complex a vocabulary becomes. Even in the standard itself [1], highlighting inconsistencies can readily be noticed, for instance by searching for the string 'the set of'. The authors of this paper have been researching potential use cases for SBVR [2][3], but these use cases cannot be fully exploited without the proper environment for writing SBVR. For this reason we have invested effort in developing an editor that can infer the correct highlighting and offer auto-completion suggestions for models that are written in SBVR Structured English. These models can then feed into our parser which generates SBVR Logical Formulation. The result is SBVR in its native representation and a whole range of possible use cases opens up, unhindered by the difficulties of attaining properly formatted SBVR-LF.

1.1 Related Work

Saying that tooling is lagging is not intended to mean that there are no tools whatsoever. But the range and capabilities they offer are limited. The earliest project that appeared in this space was an Eclipse IDE add-on called SBeaVeR [4]. Unfortunately SBeaVeR has not been updated since 2006 and the only release was marked as an alpha prototype. SBeaVeR did do some highlighting, but that was predicated upon adding an inelegant requirement for those writing SBVR models. Any term or verb that consisted of multiple nouns had to be joined by a dash. So a fact type that pertained to a student's registration for a study programme would have to be written as follows:

student *is-registered-for* **study-programme**

This adds unnecessary cognitive overhead for the modeler and the reader of the model, for the benefit of making the parsing significantly easier. We found it preferable to invest additional time in the one-off task of writing a parser rather than roll-over the difficulty to the modeler, which the tool was supposed to help. Additionally SBeaVeR did not offer a path to extract SBVR Logical Formulation from the rules. One can imagine that this was on the developers' roadmap, but the project has never been continued.

Another tool for writing SBVR models is RuleXpress by RuleArts [5]. RuleXpress offers impressive options in terms of vocabulary management, but only highlights terms, not verbs or keywords. Besides the reduced functionality, this simple string-matching approach may lead to errors if a word that can be used both as a verb and a noun is declared as a term (e.g. conduct, digest, escort, insult, produce, record, set).

2 Features

2.1 SBVR coverage

Our parser does not implement the full breadth of the SBVR specification yet, but rather a large and usable subset with a focus on expressing complex rules. The parser can be extended to include less common features of SBVR and indeed this is part of the future work planned. The features currently implemented are: declaration of terms and fact types, all modalities for rules, all quantifiers, and the keyword 'that' as a means of introducing atomic formulations that constrain variables.

With this subset of SBVR, even complex rules such as the following can be highlighted appropriately and parsed into their logical formulation:

It is necessary that each **student** that *is registered for* a **module** *is enrolled in* a **study programme** that the **module** *is available for*

We also support attributes for terms and fact types, although only definitions are highlighted at the moment. Finally, the editor can recognise fact types of any arity.

There exist in the literature mentions of ambiguities in SBVR-SE[10], but the paper mentions that using a lexicon should solve the problem for the example given. Our parser uses the vocabulary to inform the parsing of rules and therefore is not vulnerable to that kind of ambiguity presented. Using this setup, we have not come across any other ambiguous formulations, although we are open to the fact that they may appear. We take an engineering approach instead of a formal approach to this problem and have not attempted to prove that SBVR-SE as we parse it is completely impervious to contradictions.

2.2 Pluralisation

One of the more interesting aspects of our system is the automatic recognition of plurals. With a term such as **student** declared, any rule that uses the plural form **students** will be highlighted correctly. This also follows in the auto-complete suggestions. This recognition is accomplished with the help of an inflection component from the library Active Support for JavaScript [6]. This tool uses a number of well-known patterns of English for determining the plural of a given singular noun, and also includes a list of exceptions.

This of course does not mean all possible exceptions can be included. Even if it included every single irregular pluralisation in the largest available corpus of English nouns, new terms are coined continuously, and in the case of businesses, product names are often terms borrowed from other languages or coined de novo, which may have plurals that don't conform to obvious patterns. In these cases it would be useful for the modeler to have a way of declaring the plural of the relevant term, with this declaration overriding the judgment of the inflector.

The SBVR specification is the most authoritative document on SBVR Structured English, even though it does not claim to be a normative specification for it. While we have not reached a point where the guidance of the standard does not suffice for implementation, if we were to try plural parsing an exception as mentioned above, the best way would be to have available a 'Plural' attribute that can be defined for any term. This may not make sense for the original conception of SBVR which didn't necessarily anticipate support for tooling, however the ease of use that such tools offer may be worth accommodating in the standard. It is important to note that such an attribute would have no effect on the logical formulation. Its influence would be limited in assisting the automated parsing of models into logical formulation (which applies to highlighting and auto-completion as well) and stop there.

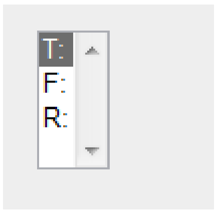
This extension of the SBVR attributes would not be something that would only be used in English. In fact, the grammar of English has in a way obscured this problem which would be much more obvious if another language was used as the basis for the specification. While nouns in English can only be in singular and plural forms (and perhaps their possessive), nouns in other languages have many more cases, each of which dictates a different form for the noun, and is subject to much the same difficulties with regard to exception handling.

2.3 Highlighting

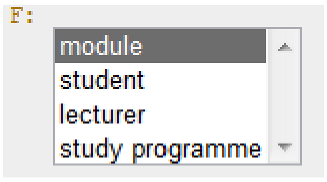
During the development of this editor, we considered the automatic highlighting of SBVR Structured English to be of great importance, as the writing of SBVR can be quite an ordeal otherwise, which can turn potential users away from SBVR. The editor highlights the SBVR features it implements as one would expect, recognising keywords, terms, and verbs according to the specification. One novel feature is that because we use the complete SBVR parser for the highlighting functionality, any input that cannot be highlighted, is input that cannot be parsed. This gives instant feedback to the modeler, which indicates that there is either some error in the rule, or the feature being used is not supported.

2.4 Auto-completion

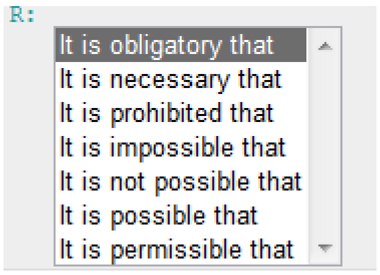
At any point during the process of writing a fact type or rule, a user can press Ctrl+Space to get options for the next tokens.



Requesting auto-completion at the start of a blank line gives the only 3 options which are to choose between a term, a fact and a rule.

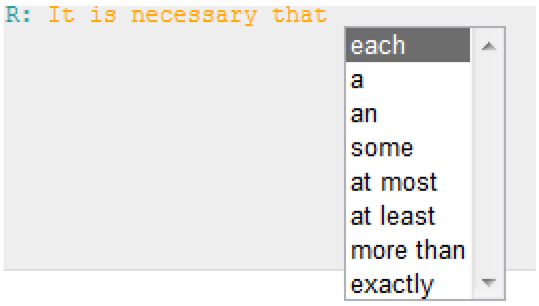


At the start of a fact, the only allowed options are terms so only terms are displayed, however these terms are all in their singular form as showing both singular and plural versions could make the number of options unwieldy.



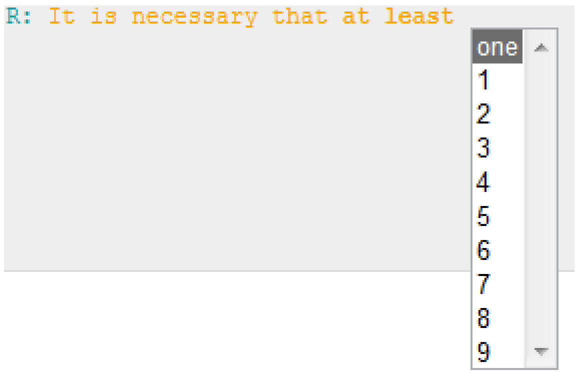
```
R:
```

At the start of a rule, the only options available are the predefined modalities.



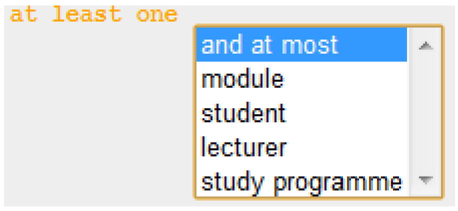
```
R: It is necessary that
```

After a modifier there has to be a quantifier so a list of available quantifiers are given.

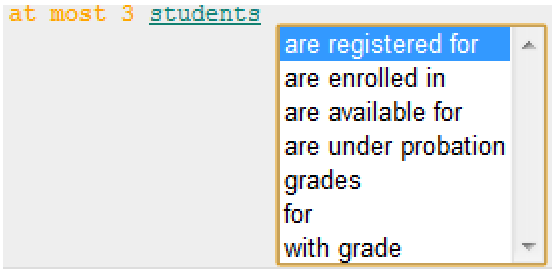


```
R: It is necessary that at least
```

We are now offered a simple list of quantities as the ‘at least n’ quantifier requires. Although any number is allowed only the numbers from 1-9 are listed to keep it manageable and to give the user the idea that a number is necessary here. Any other number can simply be typed in and will be highlighted and accepted.



If a quantifier that can join into another is chosen, we are given a list of the available joining quantifiers as well as the terms.



After the term we are offered available verbs which are in their plural form due to following a plural term. Unfortunately all verbs are given even though only a subset of verbs applies to the chosen term.

3 Implementation

To accomplish editing SBVR in the browser, we needed to build on an editing component, intended for writing editors for programming languages. This came at the benefit of reusing mature code for complex functionality, even if the intended purposes were slightly different (modeling vs. programming), which led to a number of issues during the development process.

To choose the appropriate editing component, we reviewed a number of available ones, such as Ace [7], CodeTextArea [8], EditArea [9]. We ended up using CodeMirror2 [11], as it represents an optimal mix of features, simplicity, and project activity for our purposes.

3.1 Implementing in CodeMirror

The system needs to be able to highlight SBVR text and to provide auto-completion. To implement a syntax in highlighter in CodeMirror 2 you must provide a JavaScript closure which contains a member function called “token”, with 3 optional functions, "`startState`", "`indent`" and "`copyState`" as well as one optional variable "`electricChars`".

The token function takes two arguments, the first being a `StringStream` as defined in codemirror.js and the second being a state object, which starts as either true or whatever is returned by `startState` if it is implemented. The state object will stay consistent throughout the document and reflect changes made during previous token operations. The function returns the style that should be used for all text that has been read from the stream object during the execution of this function.

The `startState` function is available to return an initialised state variable if required, we will use this to return an object containing empty arrays for storing terms and verbs we have encountered during tokenising of the document.

The indent function takes two arguments, the state and the text of the line and returns the appropriate indentation level. As SBVR does not use indentation we can override this to always return a level of 0.

The `copyState` function takes one argument, the state, and returns a copy of the state, if this function is not implemented then the state object is just copied as-is, since we do not need any specific copying functionality we can safely ignore this function.

The `electricChars` string contains characters which when found in the string will trigger indentation to be performed; as we have no need for indentation we can safely ignore this.

3.2 Patching OMeta

Initially to enable highlighting we modified OMeta, the language the grammar is written in, to store a rule token which included the rule name, starting index, and length for each OMeta rule that was successfully matched. Within the highlighting wrapper we then picked out the rule names we were interested in highlighting and were able to generate a list of highlighting tokens using the starting indices and lengths.

Whilst this solution worked, it meant storing an absolute minimum of one rule token per character of the string (and generally a lot more, e.g. char, exactly, seq, token, etc.), most of which we were never interested in. So to get around this we modified OMeta to accept a list of rule names we are interested in tokens for and for OMeta to only store tokens for rules that match this list, this reduced the number of rule tokens stored dramatically and also meant that the highlighting wrapper did not need to check through for only the rules it was interested in and so could have its complexity reduced.

Due to the nature of the highlighting being a one to one mapping with the rules that we store it becomes necessary for the parser to use separate rules for parsing each token that needs to be highlighted differently, so some modification may be necessary. However the result of these modifications being required seems to be one of enforcing a good code style rather than one of creating an annoyance, similar to the use of significant white-space for block indentation like in python.

For auto-completion we modified OMeta to store the rule name and starting index for every attempt to match a rule, this way we can find all possible branches that

OMeta attempted to take at a given point. This is only guaranteed to be a complete list of branches at the point the match fails, so for our purposes we take the point in the line at which the user requested auto-completion hints and tell OMeta to parse up until that point, as such we know that the parsing will fail so we will get all possible branches that can be taken. The auto-completer then looks at the map of rule names to possibilities provided by the OMeta based parser and offers those possibilities.

4 Conclusion & Future Work

We have found using the SBVR editor useful and intend to release it as a commercial application soon. However, there still remain a number of potential improvements that can be made, and we will keep improving the codebase.

The obvious direction for improvements is in extending the amount of SBVR that our grammar can handle, improving both the highlighter and the parser at the same time. Also, adding support for multiple vocabularies and inclusion of vocabularies in others will make the environment more suitable for larger projects.

Another intriguing possibility, which may help in making SBVR more popular, is to add the possibility for publishing models on the Web, with an easily shareable URL. This will hopefully address the dearth of SBVR examples online currently, another barrier for newcomers.

We also expect to receive a lot of feedback as we make the tool available for use to wider audiences, and have reserved significant resources in our roadmap so we can be responsive to users' suggestions.

References

1. Object Management Group, "Semantics of Business Vocabulary and Rules Formal Specification v1.0", 2008, URL: http://www.omg.org/spec/SBVR/1.0/, Accessed: 16/9/2011.
2. Marinos, A., Krause, P., "An SBVR Framework for RESTful Web Applications" In Semantic Web Rules - International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings. LNCS 6403, Springer-Verlag Berlin, Heidelberg, 2010, pp. 144-158.
3. Marinos, P. Krause. "What, not How: A generative approach to service composition", IEEE Conference on Digital Ecosystems Technologies 2009 (DEST 2009)
4. SBeaVeR, URL: http://sbeaver.sourceforge.net, Accessed: 16/9/2011.
5. RuleXpress, URL: http://www.rulearts.com/RuleXpress, Accessed: 16/9/2011.
6. Inflection-js, URL: http://code.google.com/p/inflection-js/, Accessed: 16/9/2011.
7. Ace, URL: http://ace.ajax.org, Accessed: 16/9/2011.
8. CodeTextArea, URL: http://code.google.com/p/codetextarea/, Accessed: 16/9/2011.
9. EditArea, URL: http://www.cdolivet.com/editarea/, Accessed: 16/9/2011.
10. Kleiner M., Albert, P., Bézivin, J., "Parsing SBVR-Based Controlled Languages", In Model Driven Engineering Languages and Systems, LNCS, 2009, Volume 5795/2009, pp. 122-136
11. CodeMirror2, URL: http://codemirror.net/, Accessed: 16/9/2011.

MYNG: Validation with RuleML 1.0 Parameterized Relax NG Schemas

Tara Athan

Athan Services, Ukiah, CA, USA
`taraathan AT gmail.com`

Abstract. The knowledge representation language RuleML Version 1.0 has recently been re-engineered using the Relax NG schema language, introducing several new features, including on-the-fly schemas with fine-grained, freely-combinable modules. The web application Modular sYNtax confiGurator (MYNG) provides GUI access to a PHP-based parameterized schema. To ensure monotonicity when combined, the modules follow a schema design pattern that is enforced by a meta-schema. The schema design pattern also facilitates user-extension of the language. The usage of these new features of RuleML are demonstrated at the website `http://wiki.ruleml.org/index.php/MYNG#Demo` using H. Sivonen's on-line, open source validator, `http://Validator.nu`.

1 Introduction

The knowledge representation language RuleML Version 1.0 has recently been re-engineered using the Relax NG schema language [ABss]. The Relax NG schema language is known to be more expressive than XSD or DTD [Gen07]. It has also been shown to be more efficient for XML validation [SK07], where benchmark studies comparing the performance of the Jing validator (against Relax NG schemas) to the Xerces and MSV validators (against XSD schemas) showed reductions by 25% to 50% in the time required for validation, (scaling linearly with file length).

Although validating against Relax NG schemas has advantages over the XML Schema Definition Language (XSD) schemas in many respects, Relax NG is not as widely used as XSD. While some commercial XML editors, notably oXygen[1] support Relax NG schemas, some other popular editors do not. Therefore, a large segment of the population of current and prospective RuleML users is, in all likelihood, unfamiliar with validation using Relax NG.

Further, the re-engineered RuleML Relax NG schemas introduce several new features, including

- a PHP-driven parameterized schema driver that delivers a customized schema on-the-fly based on query string parameters in the schema URL;
- a schema design pattern that ensures monotonicity when schema modules are freely combined.

[1] oXygen: `http://oxygenxml.com`

Advanced users who wish to fully exploit the customizability and extensibility of the RuleML Relax NG schemas require a thorough understanding of these features.

Therefore we present a demonstration of validation using RuleML 1.0 schemas, beginning with the simplest case based on validation of a pure RuleML instance using a redirected link, progressing to validation of mixed-namespace instances using customized and user-extended schemas, and finishing with validation of Relax NG schemas against a meta-schema defining a schema design pattern.

2 Overview of Validator.nu

For this demo, we will make use of Henri Sivonen's Validator.nu[2], a free validation webservice [Siv07] that can validate an XML instance against schemas, including Relax NG schemas and Namespace-Based Validation Dispatching Language[3]. The validation engine used by Validator.nu is Jing[4], an opensource application with command-line interface developed by James Clark, one of the authors of Relax NG [ISO08] itself.

Validator.nu has a simple user interface, allowing the user to specify a single instance document and zero to many schemas. Options include namespace-based filtering, allowing a particular namespace to be ignored by the validator.

3 Examples

The following Validator.nu cases may be accessed via links from the RuleML wiki[5].

3.1 Redirected Links

Example 1 demonstrates an attempt to validate a test RuleML instance in the `bindatagroundlog` sublanguage with the `bindatagroundfact` relaxed schema[6], accessed via a redirected link to the parameterized schema. As expected, the validator finds errors.

The actual URL that is accessed in this example is `http://www.ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x1&default=x7&termseq=x2&lng=x1&propo=xf&implies=x6&terms=xf0f&quant=x1&expr=x0&serial=xf`. Redirections to the parameterized schema have been implemented for the original fifteen named

[2] Validator.nu: `http://validator.nu`
[3] NVDL: `http://nvdl.org`
[4] Jing: `http://code.google.com/p/jing-trang/`
[5] Validator.nu Links:`http://wiki.ruleml.org/index.php/MYNG#Validator.nu_Examples`
[6] See `http://ruleml.org/1.0/relaxng/bindatagroundfact_relaxed.rnc` for the most inclusive schema.

RuleML sublanguages in the Deliberation family (except for the SWSL languages), from `bindatagroundfact` to `naffologeq`, in both serializations (normal and relaxed form). A complete listing of these redirects is available at the website `http://ruleml.org/1.0/relaxng/`.

3.2 Direct Links

In example 2, we use a direct link to the PHP-driven parameterized schema for validating a RuleML instance with a foreign namespace element. Any elements or attributes whose names belong to the foreign namespace are ignored by the validator. This mode of validation accepts RuleML that is emebedded in other documents.

3.3 NVDL

Example 3 shows the validation of a RuleML instance against an NVDL script that refers to the parameterized schema and also allows arbitrary elements from foreign namespaces. The NVDL script is:

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <namespace ns="http://www.ruleml.org/0.91/xsd">
    <validate
      schema="http://ruleml.org/0.91/relaxng/schema_rnc.php?backbone=x3f&default=x7&
termseq=x7&lng=x1&propo=x3f&implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf"
      schemaType="application/relax-ng-compact-syntax"/>
  </namespace>
  <anyNamespace>
    <allow/>
  </anyNamespace>
</rules>
```

3.4 Static Schema

There are several reasons why a user may want to validate against a static copy of the RuleML schema, including performance and offline operation. Example 4 demonstrates validation of a RuleML instance against a static copy of the default output of the parameterized schema. The schema may be obtained by two methods:

- use the MYNG GUI[7] to display a direct link to the PHP script, click on the link, and save the output to a file;
- scrape the schema driver displayed on the RuleML MYNG GUI;
- download one of the zip archives, built on-demand for either normal and relaxed form, (see `http://ruleml.org/1.0/relaxng/` and extract the schema driver file for any of the named sublanguages;

In all cases, the directory containing the schema driver must also contain the module directory, which is included in both of the zip archives.

[7] MYNG GUI: `http://ruleml.org/1.0/myng`

3.5 Including Extensions

Users may wish to extend the RuleML syntax, and the Relax NG modular schema was designed to make such extensions as convenient as possible. As a template for such extensions, we show in example 5 the expansion of the RuleML parameterized schema by a Boolean operator for exclusive disjunction.

Four modules were written to implement this extension:

- the definition module[8] contains the definition of new elements;
- the stripe-skipping module[9] contains the code that allows redundant edge elements to be skipped;
- the dishornlog expressivity module[10] contains the code that makes the extended schema compatible with languages having
- the folog expressivity module[11] contains the code that makes the extended schema compatible with languages having at least the expressive power of first-order logic.

Other modules might be required for some extensions, including:

- specification of different content models for normal and relaxed serializations;
- specification of different content models for attributes with default values;
- additional expressivity modules if content models change with other levels of expressivity;

The schema driver contains the following include statements:

```
namespace rulemlx = "http://www.ruleml.org/0.91/ext"
include "http://ruleml.org/0.91/relaxng/schema_rnc.php"
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_stripe_skipping_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_dis_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_fo_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
```

The first statement includes the RuleML language `naffologeq` with the relaxed serialziation. The second statement includes the required expansion module for the Xor element. The other three statements include optional expansion modules that

- allow the `formula` edge to be skipped;
- allow the Xor element to appear in the conclusions of implications;
- allow the Xor element to appear in rulebase assertions and retractions.

[8] `http://ruleml.org/1.0/relaxng/modules-ext/xor_expansion_module.rnc`
[9] `http://ruleml.org/0.91/relaxng/modules-ext/xor_stripe_skipping_expansion_module.rnc`
[10] `http://ruleml.org/1.0/relaxng/modules-ext/xor_dis_expansion_module.rnc`
[11] `http://ruleml.org/1.0/relaxng/modules-ext/xor_fo_expansion_module.rnc`

3.6 In xhtml

In keeping with the original purpose of Validator.nu, which is (X)HTML5 validation, we demonstrate the validation of RuleML that is embedded in the `header` section of an xhtml document in Example 6. NVDL is used to validate against three schemas, the xhtml Relax NG schema, the xhtml Schematron restrictions and a RuleML schema.

3.7 RNG Schema Validation

In [ABss], a schema design pattern was introduced that ensures monotonicity of the language when modules are freely mixed. Validator.nu can be used to validate a schema in the XML-based Relax NG syntax (RNG) against the meta-schema, also in the RNG syntax, that defines the schema design pattern. The meta-schema includes and redefines the standard RuleML schema[12], restricting the vocabulary of named patterns to three categories based on their suffixes:

– Choice combine elements: with suffixes

  `( choice | main | content | value | datatype | sequence | defs )`

– Interleave combine elements: with suffixes

  `( attlist | header | notallowed )`

– No combine elements: with suffix

  `( def )`

Monotonicity is achieved by restricting patterns in the interleave combine category to be optional [ABss].

3.8 Performance

The greatest part of the execution time of these examples is spent on remote access of the schemas and instance. Thus any XML validation with serious concerns about performance must utilize local copies of the validator and schemas, or use caching, a feature not available in Validator.nu.

4 Conclusion

We have demonstrated a number of examples of instance validation using the RuleML Relax NG schemas, ranging from simple cases to multi-namespace instances, and customized, user-extended schemas. We have also shown how a schema in the XML-based Relax NG syntax may be validated against a custom schema that enforces a schema design pattern.

Because of the limitations of the Validation.nu webservice, there are several uses of these schemas that we are not able to demonstrate here, including

[12] Standard Relax NG schema in RNG: `http://relaxng.org/relaxng.rng`

– validation of a schema in the compact Relax NG syntax
– conversion between schema languages (Relax NG compact, XML-based, XSD)
– conversion of a modular Relax NG schema into a simplified monolithic schema
– generation of an XML parser from a Relax NG schema

All of these tasks can be accomplished on the desktop using opensource software[13] and all but the last are available in the commercial oXygen framework. Future work includes the development of Java Webstart services that provide these additional capabilities.

References

[ABss] Tara Athan and H. Boley. Design and implementation of highly modular schemas for xml: Customization of ruleml in relax ng. In F. Olken, M. Palmirani, and D. Sottara, editors, *Rule-Based Modeling and Computing on the Semantic Web*. RuleML 2011 - America, LNCS 7018, in press.

[Gen07] Pierre Geneves. Logics for xml. `http://hal.inria.fr/docs/00/13/35/91/PDF/geneves-phd.pdf`, 2007.

[ISO08] ISO. ISO/IEC 19757-2: Document Schema Definition Language (DSDL) Part 2: Regular-grammar-based validation - RELAX NG. `http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008(E).zip`, 2008.

[Siv07] Henri Sivonen. About validator.nu. `http://about.validator.nu`, 2007.

[SK07] Mikko Saesmaa and Pekka Kilpelinen. On-the-fly validation of xml markup languages using off-the-shelf tools. `http://conferences.idealliance.org/extreme/html/2007/Saesmaa01/EML2007Saesmaa01.html`, 2007.

[13] See `http://wiki.ruleml.org/index.php/MYNG#Tools`

SymposiumPlanner-2011: Querying Two Virtual Organization Committees

Zhili Zhao[1], Adrian Paschke[1], Chaudhry Usman Ali[2], and Harold Boley[2,3]

[1] Computer Science Department, Freie Universität Berlin, Germany
{zhili.zhao,paschke}[AT]inf.fu-berlin.de,
[2] Faculty of Computer Science, University of New Brunswick, Canada
maniali[AT]gmail.com,
[3] Institute for Information Technology, National Research Council Canada
harold.boley[AT]nrc.gc.ca

Abstract. SymposiumPlanner-2011 is the newest in a series of Rule Responder instantiations for the Questions&Answers (Q&A) sections of the official websites of the RuleML Symposia. It supports committee members via personal agents based on member-encoded rule knowledge. The personal agents are invoked by an organizational agent which selects them using a responsibility assignment matrix. This paper describes SymposiumPlanner-2011, which goes beyond the previous instantiations by providing a more powerful user interface and reducing redundancy in the internal conference data repositories.

1 Introduction

SymposiumPlanner[4] is a series of Rule Responder[1, 5] instantiations for the Questions&Answers (Q&A) sections of the official websites of the RuleML Symposia since 2007. SymposiumPlanner utilizes an Organizational Agent (OA) to handle the filtering and delegation of incoming queries. Each committee chair has a Personal Agent (PA) that acts in a rule-based manner on behalf of the chair. This PA manages the chair's personal information, such as a FOAF(Friend of a Friend[5])-like profile containing a layer of facts about the committee member as well as FOAF-extending rules. These facts and rules allow PAs to automatically respond to requests concerning the RuleML Symposia. Query responsibility for the organization is managed through a responsibility assignment matrix, which defines the classes of queries PAs of chairs are responsible for. External Agents (EAs) constitute the public interface to the OA of a Symposium's virtual organization, through which enquiry users can send queries and receive answers.

While the instantiations from 2007 to 2009 employed the same Rule Responder infrastructure, the 2010 implementation of SymposiumPlanner additionally incorporated the EMERALD[6] framework, which was built on top of the JADE[7]

[4] http://ruleml.org/SymposiumPlanner/
[5] http://www.foaf-project.org/
[6] http://lpis.csd.auth.gr/systems/emerald/
[7] http://jade.tilab.com/

multi-agent system. This project successfully bridged EMERALD and Rule Responder, and added new functionalities such as the mapping of paper topics to Symposium tracks.

For the two installments of the 5th International RuleML Symposium[8], "RuleML 2011 - Europe" and "RuleML 2011 - America", our SymposiumPlanner-2011 employs two Sub-Organizational Agents (Sub-OAs) and an overarching Super-Organizational Agent (Super-OA). Along with the personal agents, these agents handle other issues associated with Super-OA-to-Sub-OA delegation, intelligent decision-tree-like Q&A, e.g. about where to submit theory papers, etc. We upgraded the communication middleware and rule agents of SymposiumPlanner to their latest versions and complement the user client to issue queries via a controlled natural language.

The rest of the paper is structured as follows: Section 2 introduces the SymposiumPlanner-2011 system and highlights its novel features. Section 3 describes the implementation of SymposiumPlanner-2011 in detail. Based on our experience and analysis, we close with some suggestions about SymposiumPlanner's future development.

2 SymposiumPlanner-2011 Use Case Description

The SymposiumPlanner-2011 agents support the organizing committee members in their organization tasks, such as: helping the program chair to monitor and possibly move important dates, finding contact information about selected chairs of the symposium, answering questions of participants about the conference, etc. The organization of the symposium committee is modeled by a coordinator, who is responsible for communicating with the executors which complete tasks on behalf of the symposium chairs. In SymposiumPlanner, the coordinator is implemented as an organizational agent (OA), which knows the responsibilities and the roles of each executor (Symposium chair). The OA manages the executors, which are implemented as personal agents (PAs).

The 5th International Symposium on Rules 2011 has two installments "RuleML 2011 - Europe" (IJCAI 2011) and "RuleML 2011 - America" (BRF 2011). To better manage disparate symposiums, SymposiumPlanner-2011 provides three Organizational Agents (OAs) to model the overall RuleML-2011 organization and its two sub-committees of the two Symposium installments:

- The Super-OA delivers and filters queries and requested tasks to the Sub-OAs, single point of entry for that specific instalment of RuleML 2011.
- The Sub-OAs manage the organization committee members of the two RuleML-2011 installments. They filter, decide and delegate incoming queries to responsible personal agents in the appropriate sub-organization.

An ontological RAM represents the roles and responsibilities of the Sub-OAs and the personal agents in the virtual organization of RuleML Symposium 2011.

[8] http://2011.ruleml.org

Table 1 gives a RAM fragment of the committee chairs of RuleML Symposium 2011. Negotiation and distributed coordination protocols are applied to manage and communicate with the organizations' agents and external agents.

Table 1. Responsibility Assignment Matrix: Each committee chair has different roles, such as: responsible, supportive, consulted, informed, for a particular task.

	General Chair	Program Chair	Publicity Chair	...
Symposium	responsible	consulted	supportive	...
Website	accountable	responsible	...	
Sponsoring	informed, signs	verifies	responsible	...
Submission	informed	responsible	...	
...	...	...	...	

The personal agents are self-autonomous agents and have their own rule-based decision and behavioral logic on top of their personal information sources, Web services, vocabularies/ontologies and knowledge structures. The rules are executed by different platform-specific rule engines which communicate via an Enterprise Service Bus (ESB) using standardized Reaction RuleML messages (event messages).

Besides consulting knowledge from the internal data repositories, which was very common in previous instantiations, the SymposiumPlanner-2011 rule agents access external data sources to reduce redundancy in the conference data via query languages such as SQL, SPARQL, etc. External data sources such as calendars, vocabulary definitions, databases, web pages, metadata sources, personal data are dynamically queried at runtime and used as facts in the internal knowledge base of an agent. For instance, in SymposiumPlanner-2011, the rule agents access data coming from the Semantic Web Dog Food RDF database, which contains information about the RuleML-2011 proceedings such as authors, papers, reviewers.

As a Web-based Q&A application, the previous instantiations of SymposiumPlanner allowed users to issue queries via selection of the queries in adaptable Web form menus. The potential queries needed to be defined before delivering the system to users. It becomes a real burden when there are many kinds of queries available. For the purpose of resolving this problem, SymposiumPlanner 2011 provides a translator service, which can automatically translate public interface descriptions of function rules to Reaction RuleML messages. Meanwhile, a controlled English has a restricted syntax and a restricted semantics described by a small set of construction and interpretation rules. It is not difficult for users to learn it. SymposiumPlanner 2011 also strives for allowing users to issue the queries in controlled English and then translate them into the standardized Reaction RuleML messages.

3 SymposiumPlanner-2011

SymposiumPlanner-2011 provides three Organizational Agents (OAs) to model the overall RuleML-2011 organization and its two sub-committees of the two Symposium installments. The general architecture of SymposiumPlanner 2011 is shown in Figure 1. In SymposiumPlanner-2011, we use the latest Prova 3.1.3 rule engine for the OAs, which is now an OSGi bundle and can run in an OSGi container with just a few dependencies. It packs a lot of features, some of them are inspired by the latest developments in modern computational logic, functional programming, distributed systems, and event driven architectures, some of them completely new and original. Some of very unique features include: reaction groups for event processing, guards and guarded cut, dynamic branches in workflows, and etc. Meanwhile, we utilize latest Mule ESB 3.1, which includes major improvements to Cloud Connect, including custom schemas for each connector, much simpler invocation of connectors, a new polling mechanism, message enrichment capabilities, and a simple yet powerful logging facility. In what follows, we will detail the implementation of the SymposiumPlanner-2011 system.

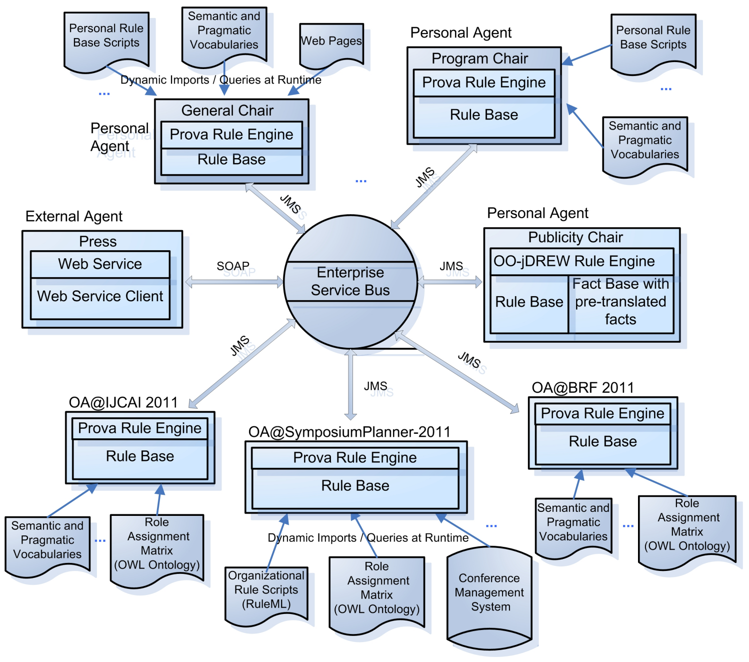

Personal Rule Base Scripts
Semantic and Pragmatic Vocabularies
Web Pages
Dynamic Imports / Queries at Runtime
Personal Agent
General Chair
Prova Rule Engine
Rule Base
Personal Agent
Program Chair
Prova Rule Engine
Rule Base
Personal Rule Base Scripts
Semantic and Pragmatic Vocabularies
JMS
JMS
External Agent
Press
Web Service
Web Service Client
SOAP
Enterprise Service Bus
JMS
Personal Agent
Publicity Chair
OO-jDREW Rule Engine
Rule Base
Fact Base with pre-translated facts
JMS
JMS
JMS
OA@IJCAI 2011
Prova Rule Engine
Rule Base
OA@SymposiumPlanner-2011
Prova Rule Engine
Rule Base
OA@BRF 2011
Prova Rule Engine
Rule Base
Semantic and Pragmatic Vocabularies
Role Assignment Matrix (OWL Ontology)
Dynamic Imports / Queries at Runtime
Organizational Rule Scripts (RuleML)
Role Assignment Matrix (OWL Ontology)
Conference Management System
Semantic and Pragmatic Vocabularies
Role Assignment Matrix (OWL Ontology)

Fig. 1. SymposiumPlanner 2011 Architecture: Each rule engine is implemented as a Web-based service consisting of a set of internal or external knowledge sources. Reaction RuleML (interchange language) messages are transported via the ESB to the appropriate agent with different transport protocols.

3.1 Enterprise Service Bus

Mule open-source ESB[9] allows deploying the rule-based agents on the Mule object broker and supports the communication in this rule-based agent processing network via a multitude of transport protocols. Mule provides a distributable object broker which follows the Staged Event Driven Architecture (SEDA)[10] pattern to manage all sorts of service components. This design decouples event and thread scheduling from application logic and avoids the high overhead associated with thread-based concurrency models[1].

The distributed agent services which at their core run the rule engines are installed as Mule components which listen at configured endpoints, e.g., JMS message endpoints, HTTP ports, SOAP server/client addresses or JDBC database interfaces. Reaction RuleML[11] is used as a common platform independent rule interchange format between the agents (and possibly other rule execution / inference services). The translator services are used to translate inbound and outbound messages from platform-independent Reaction RuleML and/or controlled natural language into the platform-specific rule engines execution syntaxes and vice versa.

3.2 Prova Rule Engine

Generally speaking, each agent service might run one or more arbitrary rule engines. Here we only describe the rule engine Prova[12] which is used for implementing the complex logic of the organizational agents. Prova follows the spirit and design of the recent W3C Semantic Web initiative and combines declarative rules, ontologies and inference with dynamic object-oriented Java API calls and access to external data sources via query languages such as SQL, SPARQL and XQuery [6].

```
File Input / Output
..., fopen(File,Reader), ...
XML (DOM)
document(DomTree,DocumentReader) :- XML(DocumenReader),...
SQL
... ,sql_select(DB,cla,[pdb_id,"1alx"],[px,Domain]).
RDF
...,rdf(http://...,"rdfs",Subject,"rdf_type","gene1_Gene"),...
XQuery
..., XQuery = 'for $name in StatisticsURL//Author[0]/@name/text()
return $name', xquery_select(XQuery,name(ExpertName)),...
SPARQL
...,sparql_select(SparqlQuery,...
```

Prova has its separation of logic, data access, and computation and its tight integration of Java and Semantic Web technologies. Due to the natural integration of Prova with Java, it offers an economic and compact way of specifying

[9] http://www.mulesoft.org
[10] http://www.eecs.harvard.edu/∼mdw/proj/seda/
[11] http://reaction.ruleml.org
[12] http://prova.ws

agents' behavior while allowing for efficient Java-based extensions to improve performance of critical operations. The main language constructs of messaging reaction rules in Prova are: sendMsg, rcvMsg and rcvMult. For instance, the following query retrieves the tracks of the symposium RuleML 2011@IJCAI.

```
getTracks(XID,Track):-
   % look-up responsible agent (Program Chair) from RAM
   assigned(XID,Agent,ruleml2011ATijcai_ProgramChair,ruleml2011ATijcai_responsible),
   % send the query to personal agent
   sendMsg(XID,esb,Agent, "query", getTrack(Track)),
   % receive answers multiple times
   rcvMult(XID,esb,Agent, "answer", substitutions(Track)).
```

3.3 Reaction RuleML Rule Interchange Format

Reaction RuleML acts as an interchange language between distributed rule agents in SymposiumPlanner. It is a general, practical, compact and user-friendly XML-serialized sub-language of RuleML for the family of reaction rules and incorporates various kinds of production, action, reaction, and knowledge representation temporal/event/action logic rules as well as (complex) event/action messages into the native RuleML syntax using a system of step-wise extensions [4]. For the communication between distributed rule-based (agent) systems Reaction RuleML provides a general message syntax as follows:

```
<Message directive="<!-- pragmatic context -->">
   <oid> <!-- conversation ID--> </oid>
   <protocol> <!-- transport protocol --> </protocol>
   <sender> <!-- sender agent/service --> </sender>
   <receiver> <!-- receiver agent/service --> </receiver>
   <content> <!-- message payload --> </content>
</Message>
```

Distributed rule agents can be programmed by the proprietary languages and engines. Reaction RuleML provides a translator service framework which translates the rule messages from and to Reaction RuleML, controlled natural language and platform-specific rule languages, such as Prova, Drools, POSL, EMERALD, etc. For example, the query in Attempto Controlled English (ACE) "What is the contact-information of the general-chair-of-RuleML-2011-IJCAI?" can be translated into a Reaction RuleML query message, in order to get the contact information of the RuleML 2011@IJCAI general chair.

```
<RuleML>
 <Message mode="outbound" directive="query-sync">
   <oid><Ind>SymposiumPlannerSystem</Ind></oid>
   <protocol><Ind>esb</Ind></protocol>
   <sender><Ind>User</Ind></sender>
   <content>
      <Atom>
         <Rel>getContact</Rel>
         <Ind>ruleml2011ATijcai_GeneralChair</Ind>
         <Var>Contact</Var>
      </Atom>
   </content>
 </Message>
</RuleML>
```

This Reaction RuleML message is then translated into a Prova message query which is executed in the engine:

```
[SymposiumPlannerSystem,esb,User,query-sync,[getContact,ruleml2011ATijcai_GeneralChair,Contact]].
```

3.4 SymposiumPlanner User Client

The SymposiumPlanner user client supports two solutions to construct queries. The straightforward solution uses an XML based file, which describes publicly accessibly rule functions together with their mode and type declarations, to create HTML forms. After user initializes the parameters, the translator service combines the structure of function rules and the parameters values to create the standardized Reaction RuleML message.

SymposiumPlanner user client also allows issuing queries in a controlled natural language (Attempto Controlled English [2]) via a web browser. The ACE2RML translator forwards the text to the Attempto Parsing Engine (APE), which translates the text into a discourse representation structure (DRS) and/or advices to correct malformed input. The DRS gives a logical/structural representation of the text which will be fed to an XML parser and to be translated into a domain specific Reaction RuleML representation of the query. For example, the query follows ACE format: "Who are the authors of 'Rule-based Distributed and Agent Systems' "? is translated by the APE into the following DRS:

```
<DRS domain="">
  <Question>
    <DRS domain="A B C">
      <query obj="A" question="who" sentid="1" tokid="1"/>
      <relation obj1="C"  rel="of" obj2="string('Rule-based Distributed and Agent Systems')"
        sentid="1"  tokid="5"/>
      <object ref="C" noun="author" struct="countable" unit="na" numrel="geq" num="2"
        sentid="1" tokid="4"/>
      <predicate ref="B" verb="be" subj="A" obj="C" sentid="1" tokid="2"/>
    </DRS>
  </Question>
</DRS>
```

By parsing the DRS and applying the domain-specific rules which map named objects to constants and predicate relations to atomic predicates. The core element of the result is shown as follows:

```
<Atom>
  <Rel>getAuthorsOfPaper</Rel>
  <Ind>'Rule-based Distributed and Agent Systems'</Ind>
  <Var>author</Var>
</Atom>
```

4 Conclusion and Future Work

The communication middleware Mule uses SEDA, which decomposes the processes of Q&A in SymposiumPlanner with event-driven stages connected by explicit queues. Theses queues enable Mule to decouple the receiver of a message from the other steps in processing the message. That means the operations of SymposiumPlanner lie mostly on the SEDA processing mechanism and its performance. As a well developed framework, the performance of SEDA has been evaluated by many efforts. In [7, 3], the authors shown that SEDA maximizes throughput and exhibits higher performance and more robust behavior under load than traditional service designs. Accordingly, these efforts can reflect the performance of SymposiumPlanner from the other side, that is, SymposiumPlanner can process users' queries reasonably and prevent resources from being overcommitted when demand exceeds agent capacity.

In future, while there is always a potential need to add more human support to the system in order to enhance flexibility, we will also aim at achieving more efficiency in terms of responses from agents as they automate redundant tasks that human users can avoid and be able to respond to increasingly complex queries. This may also lead to the need for peer to peer communication between PAs in order to help each other in answering queries posed by external agents. In such a scenario we can see PAs using FOAF-like profiles to advertise their capability of solving any query and cooperating with other PAs and the OA to achieve a collaborative environment of query response. In the case of a complex query, the query can be decomposed into parts, the OA can then delegate parts of the decomposed query to relevant PAs to solve individually. The OA can then eventually assemble the responses from PAs into a complete solution and send it back to the External Agent. In view of the aforementioned and proposed developments in our effort to build a framework to assist human organizations, we can see the SymposiumPlanner truly provides the basis for our gradual transformation of workplaces into an efficient and productive environment.

References

1. Boley, H., Paschke, A.: Rule responder agents framework and instantiations. In: Eli, A., Kon, M., Orgun, M. (eds.) Semantic Agent Systems, Studies in Computational Intelligence, vol. 344, pp. 3–23. Springer Berlin / Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-18308-9_1
2. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto controlled english meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Sutcliffe, G., Goebel, R. (eds.) FLAIRS Conference. pp. 664–669. AAAI Press (2006), http://dblp.uni-trier.de/db/conf/flairs/flairs2006.html#FuchsKS06
3. Lucian, M.: Building a scalable enterprise applications using asynchronous io and seda model. Tech. rep., TheServerSide.com (2008), http://www.theserverside.com/news/1363672/Building-a-Scalable-Enterprise-Applications-Using-Asynchronous-IO-and-SEDA-Model
4. Paschke, A., Kozlenkov, A., Boley, H.: A homogenous reaction rules language for complex event processing. In: International Workshop on Event Drive Architecture for Complex Event Process (2007), http://ibis.in.tum.de/staff/paschke/
5. Paschke, A., Boley, H., Kozlenkov, A., Craig, B.: Rule responder: Ruleml-based agents for distributed collaboration on the pragmatic web. In: Proceedings of the 2nd international conference on Pragmatic web. pp. 17–28. ICPW '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1324237.1324240
6. Paschke, A., Kozlenkov, A.: A rule-based middleware for business process execution. In: Multikonferenz Wirtschaftsinformatik (2008)
7. Welsh, M., Culler, D., Brewer, E.: Seda: an architecture for well-conditioned, scalable internet services. SIGOPS Oper. Syst. Rev. 35, 230–243 (October 2001), http://doi.acm.org/10.1145/502059.502057