SDRule-L: Managing Semantically Rich Business Decision Processes

Yan Tang Demey and Christophe Debruyne

Semantics Technology & Application Research Lab,
Department of Computer Science, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussels, Belgium
`yan.tang@vub.ac.be, christophe.debruyne@vub.ac.be`

Abstract. Semantic Decision Rule Language (SDRule-L) is an extension to Object-Role Modelling language (ORM), which is one of the most popular fact based, graphical modelling languages for designing information systems. In this paper, we want to discuss how SDRule-L models can be formalized, analysed and applied in a business context. An SDRule-L model may contain *static* (e.g., data constraints) and *dynamic* rules (e.g., sequence of events). A reasoning engine is created for detecting inconsistency. When an SDRule-L model is used to manage linked data, a feasible way is to align SDRule-L with Semantic Web languages, e.g. OWL. In order to achieve this, we propose to map dynamic rules into a combination of static rules and queries for detecting anomalies. In this paper, we will illustrate a model reification algorithm for automatically transforming SDRule-L models that contain dynamic rules into the ones containing static rules, which can be formalized in Description Logic.

Keywords: business process modelling, fact based modelling, Description Logic, semantic decision support

1 Introduction

Ontologies can be applied in many fields, such as system engineering, requirement analysis, bioinformatics, information categorization and Semantic Web (SW). One interesting and appealing domain is semantic decision support (SDS) for business, which can be further considered as a means to enhance decision support using business domain knowledge. We call a system for SDS as SDSS, with which we can assist communications between decision makers by enhancing the shareability and improve interoperabilities among business decision tools and services.

A fundamental requirement of SDSS is that its business semantics that is important to make a decision must be properly captured. In order to fulfil this need, we use Semantic Decision Rule Language (SDRule-L, [1]), which is a dialect in the family of fact based modeling (FBM) languages [2] and an extension to Object-Role Modelling language (ORM [3]), to capture decisional semantics and graphically present it.

In this paper, we will discuss the SDRule-L constraints that do not exist in most FBM dialects, or, have different semantics. We also propose using SDRule-L for

checking the consistency of linked business data. An SDRule-L model may contain *static* rules (e.g., data constraints), *dynamic* rules (e.g., sequence of events), and *second-order* attributes (e.g., clusters). Unfortunately, current solutions of managing linked data are based on Description Logic (DL) family, which does not directly deal with dynamic rules. And, DL by default is first-order logic instead of second-order logic. In this paper, we will illustrate how SDRule-L models can be mapped into OWL (Web Ontology Language)-compatible SDRule-L model and DL. In order to check consistency of business data, we have implemented an SDRule-L engine.

It is organized as follows. Sec. 2 is the paper background. How to map dynamic rules into a combination of static rules and queries for detecting anomalies will be discussed in Sec. 3. We will show the implementation issues and present the discussions in Sec. 4. In Sec. 5, we will conclude.

2   Background and Related Work

For over three decades, FBM dialects, such as ORM [3], have been intensively studied for modeling business information. When comparing FBM dialects to the languages in the related work, FBM has many outstanding advantages as a modeling tool for ontologies. For example, Entity-Relationship diagrams (ER, [4]) and Unified Modeling Language (UML, [5]) cannot express relevant constraints on or between attributes. Business Process Models and Notations (BPMN, [6]) and its extensions (e.g., rBPMN that focuses on expression of constraints in BPMN, [7]) mainly focuses on processes and researchers pay less attention to other models, such as data models. Compared to Conceptual Graph (CG, [8]), FBM languages contain more semantically rich graphical notations and have *verbalization* mechanisms, which enable modelers to easily learn and communicate with domain experts. Hence, FBM is more suitable for conceptual analysis, especially when *non-technical* domain experts are involved. In the domain of business, this is an extremely important reason.

Since 1999, the FBM methodological principles have been adopted for modeling ontologies and supporting *verbalization* of ontology models in the paradigm of Developing Ontology-based Methodologies and Applications [9] [10]. Later on, ORM/ORM2 is extended for modeling ontologies. One extension is called Semantic Decision Rule Language (SDRule-L, [1]) and is used for modeling semantically rich decision support rules within the context of business. Its markup language – SDRule-ML – has been designed to store and exchange ontology-based decision rules.

3   Model Transformation

SDRule-L extends ORM by introducing contains, operators and corresponding graphical notations such as instance, sequence, cluster, negation, exception and modality. In this section, we will illustrate those graphical notations and explain their semantics. In the meanwhile, we will show how SDRule-L models can be transformed into OWL-compatible models and the SPARQL queries used for checking the consistency of business data.

Formalization of Objectification: Before going into the details of SDRule-L constraints and operators, it is necessary to explain objectification and the formalization.

Objectification is a way of treating a role pair as an object [3]. Graphically, it is represented as shown in **Fig. 1** (O1), which is a minimum constrained fact type with an objectification. A and B are two object types, r1 and r2 are the roles that A and B can play with, and C is an objectified role pair r1/r2. The bar on r1/r2 is a uniqueness constraint, meaning that the populations of A and B with regard to r1/r2 are unique.
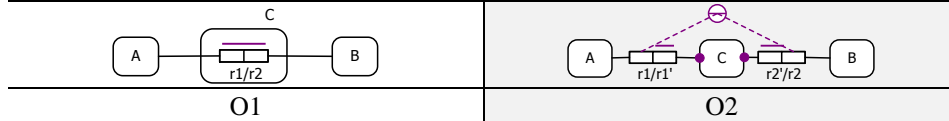

Fig. 1. Example of objectification and its equivalent owl-compatible model

The objectification from **Fig. 1** (O1) can be mapped to **Fig. 1** (O2) without losing any semantics. In **Fig. 1** (O2), the objectified role pair r1/r2 is treated as a new object type C. Two new roles r1' and r2' are introduced for the issues of formalization and implementation. Two mandatory constraints (graphically represented as dots) are applied between C and A, and between C and B. The constraints on roles r1' and r2' ensures 1:1 population between C and A, and between C and B. The circled bar in **Fig. 1** (O2) is an external uniqueness, which is a direct adaptation from the uniqueness constraint on r1/r2 from **Fig. 1** (O1). We use $\mathcal{SOIQ}(D)$ – a DL dialect – to formalize **Fig. 1** (O2) as follows:

$$\exists r1.\top \sqsubseteq A \qquad r1^- \equiv r1' \qquad \exists r2.\top \sqsubseteq B \qquad r2^- \equiv r2'$$

$$\exists r1'.\top \sqsubseteq C \qquad \exists r2'.\top \sqsubseteq C \qquad C \equiv\, \leq 1r1'.\top \sqcap \exists r1'.\top \sqcup \leq 1r2'.\top \sqcap \exists r2'.\top$$

In what follows, we will use objectification to objectify roles.

Sequence is a common constraint for an event. In SDRule-L, two events can have the relations as indicated in **Table 1**.

Table 1. SDRule-L Sequence (E_1: event on the right of the connector; E_2: event on the left)

ID	Name	Graphical Notation	Verbalization
1	Succession	——≫—▶	E_1 is before E_2
2	Continuation	——_ _▶	E_1 is exactly before E_2
3	Overlap	◀-_-—▶	E_1 and E_2 overlap
4	Trigger	≻—≫——▶	E_1 triggers E_2
5	Terminator	——≫—○	E_1 is terminated by E_2
6	Coincidence	◀—|=|—▶	E_1 and E_2 are in parallel

Allow us to use E for denoting an event. An event contains two basic time indicators: *begin time stamp* (which we indicate as T_1) and *end time stamp* (indicated as T_2). E is a valid event iff $T_2 \geq T_1$. We use a dot "." to indicate the holder of an element. For example, for an event E_i, its begin time stamp is denoted by $E_i.T_1$. Given two

events $-E_1$ and E_2 – a succession (E_1 is before E_2) is valid iff $E_1.T_1 > E_2.T_1$. A continuation (E_1 is exactly before E_2) is valid iff $E_1.T_2 = E_2.T_1 + \alpha$ where α is a given time interval. An overlap is valid iff

- $E_1.T_1 \leq E_2.T_1$ and $E_1.T_2 > E_2.T_1$. Or,
- $E_2.T_1 \leq E_1.T_1$ and $E_2.T_2 > E_1.T_1$

A trigger is similar to (but stricter than) a succession. The fact that E_1 triggers E_2 is valid iff $E_1.T_1 > E_2.T_1$ and when E_1 happens, E_2 must happen. For a succession like "E_1 is before E_2", when E_1 happens, E_2 will possibly (but not necessarily) happen. A terminator $-E_1$ is terminated by E_2 – is valid iff $E_1.T_1 \geq E_2.T_2$ and $E_1.T_2 = E_2.T_2$. A coincidence is valid iff $E_1.T_1 = E_2.T_1$ and $E_1.T_2 = E_2.T_2$.

Fig. 2 shows an example containing all the six sequence constraints. Each role pair (see the rectangles) is constrained with a uniqueness constraint (graphically represented as a bar above role pairs). Without it, a role pair cannot be populated.
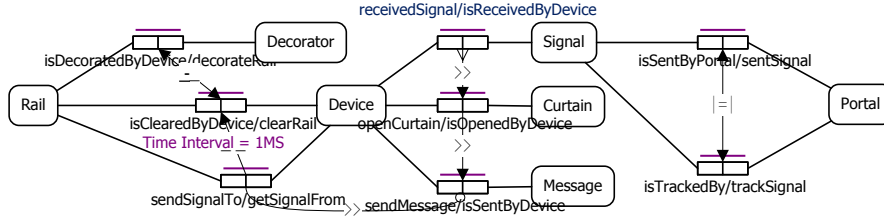

Fig. 2. An example of sequence

An example of the **verbalization**[1] of **Fig. 2** is Device open(s) Curtain <u>before</u> Device send(s) Message.

We can transform the succession constraint modelled in **Fig. 2** into an OWL-compatible model as illustrated in **Fig. 3**. Role pairs are objectified and new concepts concerning event and time stamps are added with mandatory constraints (graphically represented as dots). The part in **Fig. 3** that contains extra concepts can be verbalized as "Open Curtain is a subtype of Event; Send Message is a subtype of Event; Event has Time Stamp T1; Event has Time Stamp T2".

1. $OpenCurtain \sqsubseteq \exists openCurtain'.Device \sqcap \exists isOpenedByDevice'.Curtain \sqcap \leq 1 openCurtain'.Device \sqcap \leq 1 isOpenedByDevice.Curtain.$
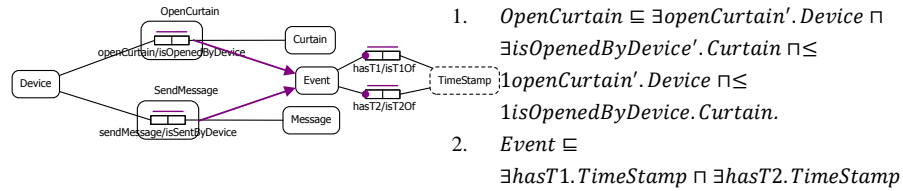2. $Event \sqsubseteq \exists hasT1.TimeStamp \sqcap \exists hasT2.TimeStamp$

Fig. 3. An OWL-compatible model transformed partly from **Fig. 2** and the DL axioms

Note that one part of the semantics of sequence from the original design in **Fig. 2** and the discussed DL axioms is missing in **Fig. 3**. It does not specify that T1 of "Open

---

[1] Verbalization is a process of mapping a graphical model to (or from) a few sentences in a controlled language.

Curtain" must be smaller than T1 from "Send Message". It is normal because it can be modelled neither in an OWL-compatible model nor in any DL dialects that are supported by OWL.

Such semantics is captured using a query language. We can check data consistency by querying linked data. In this paper, we adopt an approach similar to the one in [11] for checking constraints, namely to translate constraints into SPARQL ASK queries to check whether counterexamples (i.e. constraint violations) exist. In our engine, the ASK query looks for counterexamples and upon a positive answer, will return that this particular constraint has been violated.

Cluster is a way of treating a set of fact types as an object. By having clusters, we can reify a model by looking into the details of an object, or, we can abstract a model by hiding the design details of its objects. The graphical notation of cluster is a round-cornered box indicated with a cluster name. A cluster can be composed of another cluster, fact types and objectified roles. A composition can be possible or necessary, the graphical notations of which are shown in**Table 2**.

Table 2. SDRule-L Cluster

ID	Name	Graphical Notation	Verbalization
1	Possible composition	ParentCluster / ChildCluster (M); ParentCluster (M)	… possibly con-tains …
2	Necessary composition	ParentCluster / ChildCluster (L); ParentCluster (L)	… must contain …

The modality operators are used to specify whether it is necessary (or possibly) for a cluster to include a component. In SDRule-L, there are two modality operators – necessity and possibility. The graphical notation of necessity is a square marked with "L". A possibility operator is a diamond square marked with "M". Note that we shall not mistake M for mandatory. Since we want to align our graphical notations with the logical symbols from Modal Logic that are commonly accepted, we choose L ($\square$) for necessity and M ($\lozenge$) for possibility.

Fig. 4 shows an example of cluster and the zoom-out view. The cluster "Opening Curtain" is composed of a necessary cluster "Listen and React" and a possible cluster "Sending Msg". The cluster "Listen and React" contains two fact types – Device received Signal and Device open(s) Curtain. The cluster "Sending Msg" contains one fact type – Device send(s) Message. The three clusters are subtypes of "Task".

If a role that is connected with a cluster is populated, then the necessary components of this cluster must be populated while it is not required to have its optional components populated. Each component of a cluster is by default optional.

With regard to design issues, when a necessary component of a cluster contains a possible component, then the necessary component is treated as if it were optional. **Fig. 5** shows two models of cluster. Cluster C2 in the figure on the left is a necessary component for cluster C1, while C2 on the right is an optional component for C1. Their equivalence can be proven with a truth table.
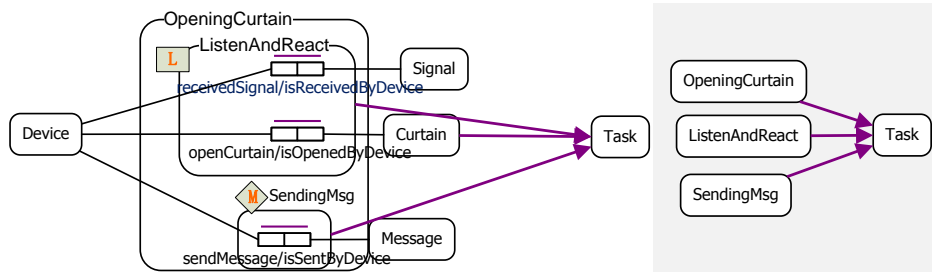
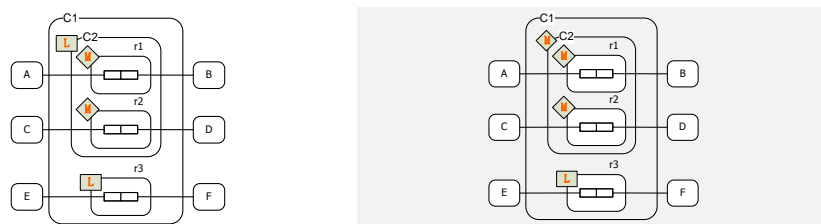**Fig. 4.** Left: An example of cluster in SDRule-L; Right: a zoom-out view

Fig. 5. Two equivalent models

Fig. 4 can be mapped into an OWL-compatible model as illustrated in **Fig. 6**. Mandatory constraints are assigned to the roles that come from a mandatory cluster. The semantics of composition from **Fig. 4** is missing in **Fig. 6**.
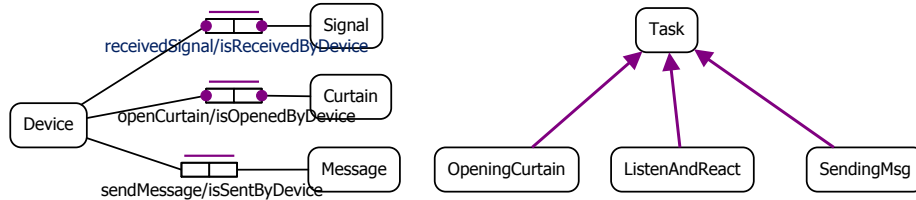

Fig. 6. OWL-compatible models partly transformed from **Fig. 4**

Other Constraints and Operators

In general, an implication is used to draw conclusions based on statements. In SDRule-L, we use it to control the population of a role based on alternatives. It is often used for modeling dynamic and non-monotonic decision rules.

Fig. 7 shows an example of implication and its verbalization. An arrow tipped bar indicated with ¬ is an operator of negation. When negation is applied on a role of the antecedent of an implication, it is a checksum of empty population. When it is applied on a role of the consequence of an implication, it is a denial of populating this role. For instance in **Fig. 7**, if *openCurtain/isOpenedByDevice* is populated, then *isSentByDevice/sendMsg* must be populated; otherwise, the latter must not be populated.
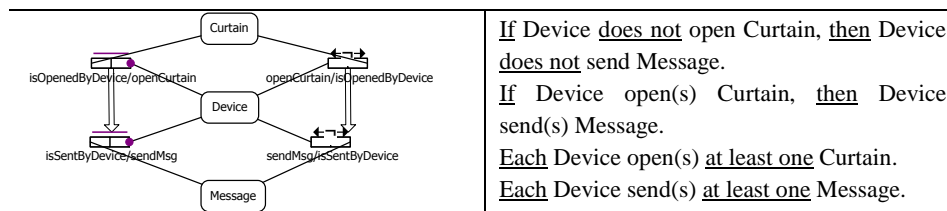
	If Device <u>does not</u> open Curtain, <u>then</u> Device <u>does not</u> send Message. If Device open(s) Curtain, <u>then</u> Device send(s) Message. <u>Each</u> Device open(s) <u>at least one</u> Curtain. <u>Each</u> Device send(s) <u>at least one</u> Message.

Fig. 7. An example of implication and its verbalization

Due to the limitation of DL, negation and conditional alternatives cannot be formalized. Implication could be partly modeled in DL as a subset. For instance, the non-negative part in **Fig. 7** can be formalized as: $Device1 \sqsubseteq \exists\, openCurtain.\top$ $Device2 \sqsubseteq \exists openCurtain.\top$; and, $Device1 \sqsubseteq Device2$. However, we shall avoid this complicated construction and opt for queries to detect counterexamples instead.

When negation is used in a conditional statement, it is a constraint. When it is used in a conclusion, it is an operator. Another important operator in SDRule-L is skipper.

A skipper allows us to give an exception to the studied constraints. It is quite useful especially for the domains like law, which in most cases contains inconsistency. **Fig. 8** shows the graphical notation of skipper.
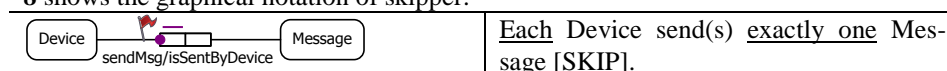
	<u>Each</u> Device send(s) <u>exactly one</u> Message [SKIP].

Fig. 8. An example of skipper (exception)

4 Implementation, Discussion and Future Work

The paper idea has been implemented in the SDRule-L engine, which can be downloaded from https://sourceforge.net/projects/sdrulel/.

An SDRule-L model is stored and shared in a mark-up language called SDRule-ML [1]. Our SDRule-L engine takes SDRule-ML files as inputs and generates analysis messages (e.g., whether all the constraints in a model are satisfied or not) as outputs. Including the method of model transformation that is discussed in Sec. 3, it is also required to specify any possible implicit constraints. Otherwise, it would be difficult to link the components in an XML file to the elements in a query.

In this paper, a sequence constraint (e.g., continuation) is applied on two fact types, which share at least one object entity. In general, we allow a sequence constraint to be applied on any two fact types that are indirectly connected. When we want to compare two facts from these two different fact types, we need to find the right connection between them; otherwise, we cannot compare them. **Fig. 9** shows an example of sequence that is applied on indirectly connected and two possible paths of building the connection. The two different paths might lead to different conclusions. Finding right connections for indirectly connected fact types is a challenge, which we will study in the future.
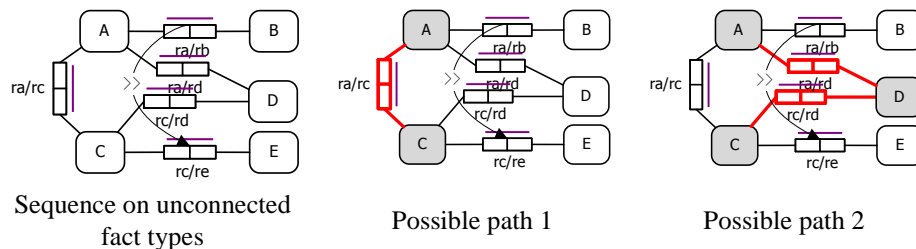
| Sequence on unconnected fact types | Possible path 1 | Possible path 2 |

Fig. 9. An example of a sequence applied on unconnected fact types and two possible paths of connections

5   Conclusion

In this paper, we have discussed the most recent results concerning SDRule-L, which is a semantic decision support language. In particular, we have presented constraints of *sequence*, *cluster* and *implication*, and operators of *negation* and *skipper*. We have shown a method of mapping dynamic rules into a combination of static rules and queries for detecting model anomalies. This method is further implemented in the SDRule-L reasoning engine.

References

1. Tang, Y., Meersman, R.: SDRule Markup Language: Towards Modeling and Interchanging Ontological Commitments for Semantic Decision Making. In Giurca, A., Gasevic, D., Taveter, K., eds. : Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches Sec. I. Chapter V. IGI Publishing, USA (2008)

2. FBM: What is Fact Based Modeling? In: Fact Based Modeling Official Website. Available at: http://www.factbasedmodeling.org/

3. Halpin, T., Morgan, T.: Information Modeling and Relational Databases 2nd edn. Morgan Kaufmann (2008)

4. Calì, A., Gottlob, G., Pieris, A.: The Return of the Entity-Relationship Model: Ontological Query Answering. In : Semantic Search over the Web. Springer Berlin Heidelberg, Germany (2012) 255-281

5. Wang, X., Chan, C.: Ontology Modeling Using UML. In Konstantas, D., Léonard, M., Pigneur, Y., Patel, S., eds. : 7th International Conference on Object Oriented Information Systems Conference (OOIS'2001), Geneva, vol. LNCS 2817, pp.59-68 (2001)

6. Prater, J., Mueller, R., Beauregard, B.: An ontological approach to oracle BPM. In

Pan, J. Z., Chen, H., Kim, H. G., Li, J., Wu, Z., Horrocks, I., Mizoguchi, R., Wu, Z., eds. : JIST'11 Proceedings of the 2011 joint international conference on The Semantic Web, Hangzhou, vol. LNCS7185, pp.402-410 (2011)

7. Milanovic, M., Gasevic, D., Rocha, L.: Modeling Flexible Business Processes with Business Rule. In : the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011, Helsinki, Finland, vol. 3.1, pp.65-74 (2011)

8. Comparot, C., Haemmerle, O., Hernandez, N.: Conceptual Graphs and Ontologies for Information Retrieval. In Priss, U., Polovina, S., Hill, R., eds. : the 15th international conference on Conceptual Structures: Knowledge Architectures for Smart Applications, Beijing, vol. LNAI 4604, pp.480-483 (2007)

9. Meersman, R.: Semantic Ontology Tools in IS Design. In Raś, Z., Skowron, A., eds. : ISMIS 99 Conference, London, vol. LNCS1609, pp.30-45 (1999)

10. Spyns, P., Tang, Y., Meersman, R.: An Ontology Engineering Methodology for DOGMA. Journal of Applied Ontology 3(1-2), 13-39 (2008)

11. Tao, J., Sirin, E., Bao, J., McGuinness, D. L.: Integrity Constraints in OWL. In Fox, M., Poole, D., eds. : AAAI, Atlanta, Georgia, USA (2010)