Metis: The SDRule-L Modelling Tool

Yan Tang Demey and Christophe Debruyne

Department of Computer Science,
Vrije Universiteit Brussel, 1050 Brussel, Belgium
{yan.tang,christophe.debruyne}@vub.ac.be

Abstract. Semantic Decision Rule Language (SDRule-L), which is an extension to Object-Role Modelling language (ORM), is designed for modelling semantic decision support rules. An SDRule-L model may contain *static* (e.g., data constraints) and *dynamic* rules (e.g., sequence of events). In this paper, we want to illustrate its supporting tool called Metis, with which we can graphically design SDRule-L models, verbalize and reason them. We can store and publish those models in its markup language called SDRule-ML, which can be partly mapped into OWL2. The embedded reasoning engine from Metis is used to check consistency.

Keywords: fact based modeling, object role modeling, conceptual modeling, semantic decision making, SDRule-L.

1 Introduction

Semantic decision support systems are a means to support group decision making using domain ontologies. Interoperability – the basic characteristic from any ontology-based systems – will enhance the mutual understanding between decision makers within a decision group (or community). Semantic Decision Rule Language (SDRule-L, [1]) is a bridge connecting ontologies and decision making processes. In other words, we can use SDRule-L to model decisions based on domain ontologies.

SDRule-L is an extension to ORM2 [2]. SDRule-L supports both *static* (e.g., data constraints) and *dynamic* rules (e.g., sequence of events). It also supports higher-order modelling. As most first-order static rules can be modelled in ORM2, in this paper, we will focus on the extended constraints and operators, which are recently implemented in a tool called Metis. Metis contains functions of designing SDRule-L models graphically and verbalizing the graphical models. The models are stored and published in the markup language called SDRule-ML, which is a hybrid language of ORM-ML [3] and FOL Rule-ML [4]. SDRule-ML can be partly mapped into Web Ontology Language (OWL2). In addition, it has an embedded SDRule-L reasoner, with which we can validate linked data.

The paper is organized as follows. Sec 2 is the related work. We will discuss specific SDRule-L constraints and operators in Sec. 3. Metis is illustrated in Sec. 3.5. We will discuss in Sec. 4. We will conclude and illustrate our future work in Sec.5.

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 492–502, 2013.

2 Related Work

Since 1999, the Fact Based Modeling (FBM[1]) methodological principles have been adopted for modeling ontologies. The authors in [5] [3] have illustrated how a particular FBM dialect – ORM [2] – can be used for modeling ontologies and ontology verbalization. Dogma Modeler[2] and Collibra Studio[3] were thus developed. Dogma Modeler supports simple verbalization in English, Dutch, German, French, Spanish, Arabic and Russian. Ontologies, in view of DOGMA Modeler, are made of reusable modules (or patterns). Collibra Studio is a commercialized tool, with which business people can model and share their data semantics.

Later on, ORM/ORM2 has been extended for modeling ontology-based application rules and enabling an application to easily commit to a domain ontology. One extension is called Semantic Decision Rule Language (SDRule-L, [1]), which is used to model semantically rich decision support rules. A markup language – SDRule-ML – has been designed to store and exchange SDRule-L models. The focus of this paper is the supporting tool called Metis[4]. Since the tool is to enhance decision making processes in order to get better and wiser decisions, it is named after Metis, a goddess of wisdom (or wise counsel/wise decision) in Greek mythology.

A related work of the sequence constraint discussed in this paper is UML[5] sequence diagram, which focuses on message interchanges between lifelines. Each lifeline corresponds to exactly one individual participant in the interaction. Another related work is BPMN[6] flow objects for describing sequence flows and message flows. Compared to their work, the sequence constraint in this paper focuses on different types of sequences and the relation between two events. One event may be played by more than one participant when a combination of sequence and cluster is applied.

Other FBM tools, which are more for modeling databases and information in general, are VisioModeler, NORM[7], CogNIAM Studio[8], Richmond[9] and Active Facts[10]. Unlike most of the mentioned FBM tools, which deal with local databases, Metis uses the Resource Description Framework (RDF) to deal with *Linked Data*, which can be considered as a collection of relevant databases or web data. Linked Data is referred to as a Semantic Web initiative to interlink web resources. It uses Uniform Resource Identifiers (URI) for accessing the resources and RDF for representing knowledge and annotating the resources. In addition, Metis contains extended graphical notations, such as cluster, sequence and decisional alternatives (implication), which are specific for modelling decision rules. In the following section, we will discuss the extensions.

---

[1] FBM official website: `www.factbasedmodeling.org`

[2] DOGMA Modeller can be downloaded from `www.jarrar.info/Dogmamodeler`

[3] Collibra Studio is now merged into Collibra Data Governance Software (`www.collibra.com`)

[4] Metis can be downloaded from `sourceforge.net/projects/sdrulel/files/Metis/`

[5] `http://www.omg.org/spec/UML/2.0/`

[6] `http://www.bpmn.org/`

[7] Both Visio Modeller and NORMA can be downloaded from `www.ormfoundation.org`

[8] A commercialized tool developed by PNA Group `www.pna-group.nl`

[9] A tool created by Victor Morgante.

[10] Active Facts can be retrieved from `dataconstellation.com`

3 SDRule-L Constraints and Operators

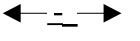
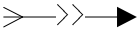
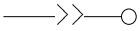
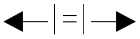
The extensions include constraints and operators like annotation (including sample instances), sequence, cluster, implication, necessity, possibility, cross-context subtyping and cross-context equivalence. In this paper, we want to focus on *sequence*, *cluster* and *implication*.

3.1 Sequence

We often use sequences to describe the relations between events. The graphical notation of a sequence is an arrow-tipped bar that connects two event types. Each event type is represented as an object type.

Currently, SDRule-L contains six types of sequence as illustrated 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

Given two events $E_1$and E_2, and their begin time stamps ($E_1.T_1$ and $E_2.T_1$) and end time stamps ($E_1.T_2$ and $E_2.T_2$), the semantics of the sequences illustrated in **Table 1** is illustrated as follows.

- **Succession:** $(E_1.T_1 < E_2.T_1) \wedge (\{\exists e | e \in E_2\} \rightarrow \{\exists e | e \in E_1\})$
- **Continuation:** $E_1.T_2 + \alpha = E_2.T_1$ where α is a given time interval
- **Overlap:** $(E_1.T_1 \leq E_2.T_1) \wedge (E_1.T_2 > E_2.T_1) \vee (E_2.T_1 \leq E_1.T_1) \wedge (E_2.T_2 > E_1.T_1)$
- **Trigger:** $(E_1.T_1 < E_2.T_1) \wedge (\{\exists e | e \in E_1\} \leftrightarrow \{\exists e | e \in E_2\})$
- **Terminator:** $(E_1.T_1 \leq E_2.T_2) \wedge (E_1.T_2 = E_2.T_2)$
- **Coincidence:** $(E_1.T_1 = E_2.T_1) \wedge (E_1.T_2 = E_2.T_2)$

Note that, given an event E, it is valid iff $E.T_1 \leq E.T_2$.

An SDRule-L model containing sequences (containing both *dynamic* and *static* rules) can be partly mapped into an OWL[11]-compatible model (containing *static* rules). Suppose we have a succession as illustrated in the left figure from **Fig. 1**. The OWL-compatible model that is *partly* mapped from the succession model is shown on the right in **Fig. 1**.

[11] Web Ontology Language: `http://www.w3.org/TR/owl2-overview/`

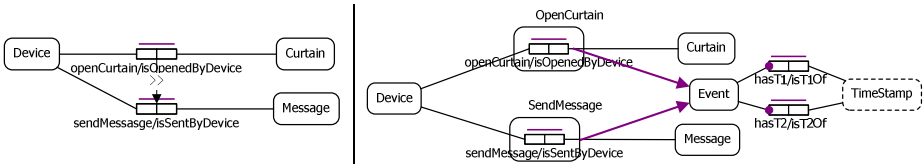


Fig. 1. Left: an example of succession; Right: an OWL-compatible model (partly mapped)

When a set of linked data is claimed to comply with the succession model in **Fig. 1**, we can check the data consistency by embedding the dynamic rule – $E_1.T_1 > E_2.T_1$ – in a query language (in our case, SPARQL[12] is chosen). This idea is adopted from [6]. In particular, sequences are translated into SPARQL ASK queries to check whether counterexamples exist.

For example for a succession – "E_1 is before E_2", it is valid iff $E_2.T_1 < E_1.T_1$. Counterexamples are sought by looking for two facts that violate this condition ($E_2.T_1 \geq E_1.T_1$). It is also invalid if E_2 happens but E_1 did not happen. We give the SPARQL query as illustrated below.

```
ASK {
  {
    ?a <http://…/ont#r1> ?rc1.
    ?a <http://…/ont#r2> ?rc2.
    ?rc1 <http://…/#T1> ?t1.
    ?rc2 <http://…/#T2> ?t2.
    FILTER(?t1 >= ?t2)
  } UNION {
    OPTIONAL{ ?a <http://…/ont#r1> ?rc1. }
    ?a <http://…/ont#lr2> ?rc2.
    FILTER(!BOUND(?rc1))
  }
}
```

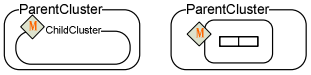
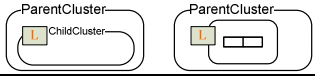
3.2 Cluster

A cluster is used to group a set of fact types. We treat such a set as an object, with which we can reify a model. For instance, the concept "helpdesk task", which points to a task, may contain other concepts/tasks, such as "people dial a number" and "people pick up a phone", which are used to explain "helpdesk task".

The graphical notation of a cluster is a round-cornered box indicated with a cluster name. A cluster (also called 'parent cluster') may contain another cluster (also called 'component cluster'), which is attached with a symbol of modality. **Table 2** shows the graphical notions of possible and necessary compositions.

[12] Query Language for Resource Description Framework (RDF): http://www.w3.org/TR/rdf-sparql-query/

Table 2. SDRule-L Cluster

ID	Name	Graphical Notation	Verbalization
1	Possible composition		… possibly contains …
2	Necessary composition		… must contain …

A component cluster is either possible or necessary. A *possible* component cluster is a component cluster that may be or may not belong to its parent cluster. The task that corresponds to a parent cluster can be executed with or without the tasks that correspond to its possible component cluster. A component cluster is by default possible. A *necessary* component cluster is a component cluster that must be included in its parent cluster; otherwise, the task that corresponds to the parent cluster cannot be executed without the tasks in the necessary component cluster.

Fig. 2 shows an example of cluster. 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".

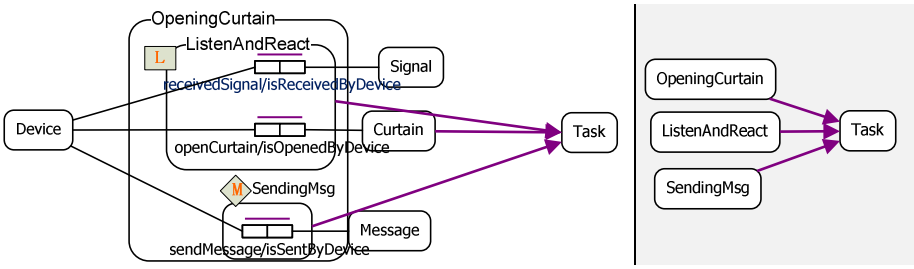


Fig. 2. Left: An example of cluster in SDRule-L; Right: a zoom-out view

When a cluster is populated, its necessary components must be populated while we do not require its optional components to be populated. Suppose that we have a data table containing a set of linked data as illustrated in **Table 3**.

Table 3. A table containing linked data that needs to be complied with **Fig. 2**

ID	*Device*	*Signal*	*Curtain*	*Message*
1	D1	S1	C1	M1
2	D2	S2	C2	NULL
3	D3	S3	NULL	NULL
4	D4	NULL	NULL	M4
5	D5	NULL	C5	M5
6	D6	NULL	NULL	NULL

When we validate it with the model as illustrated in **Fig. 2**, the data record with ID 4 does not satisfy the model because $SendingMsg$ is populated, which implies that $OpenCurtain$ must be populated. $OpenCurtain$ contains a necessary cluster called "$ListenAndReact$", which consists of the two following optional fact types.

$$l_1 = \langle Device, receivedSignal, isReceivedByDevice, Signal \rangle$$
$$l_2 = \langle Device, openCurtain, isOpenedByDevice, Curtain \rangle$$

Since $Opencurtaion$ must be populated, at least one from the above two fact types must be populated. However, in record with ID 4, none of these two fact types are populated. Therefore, it does not satisfy the model.

For the other data records, which are valid facts, we present the analysis as follows.

- 1: all fact types are populated
- 2: the optional component from cluster "$OpeningCurtain$" – $SendingMsg$ – is not populated. All the rest are populated.
- 3: the optional component (referred to l_1, see above) from the cluster "$ListenAndReact$", which is a necessary component from the cluster "$OpeningCurtain$", is populated. All the rest are not populated.
- 4: invalid (see the earlier discussion)
- 5: the optional component (referred to l_2, see above) from the cluster "$ListenAndReact$", which is a necessary component from the cluster "$OpeningCurtain$", is populated. Another optional component (referred to l_1, see above) from the cluster "$ListenAndReact$" is not populated.
- 6: the cluster "$ListenAndReact$" is not populated.

With cluster, a model can easily embrace higher ordered rules. The advantage is that we can simplify the process of model reification. The disadvantage is that it brings complexity to the reasoning engine. We will explore the issues on how to combine cluster with other constraints in the future.

Similar to the constraint of sequence, we first map clusters to OWL-compatible models as illustrated in **Fig. 3**.

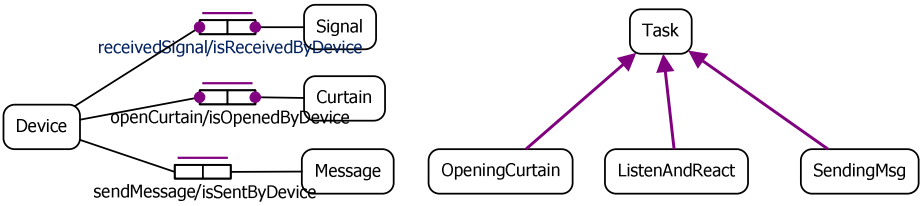


Fig. 3. OWL-compatible models partly transformed from **Fig. 2**

Then, we provide the following SPARQL queries[13] for finding counterexamples.

[13] "l2", "l3" and "l1" are the identifiers for respectively the first, second and third binary fact type on the left hand side of the example in Fig.2.

```
ASK {
  ?a a <http://…/ont#Device>.
  OPTIONAL {
    ?a <http://…/ont#l2f> ?x1.
    ?x1 a <http://…/ont#l2>.
  }
  OPTIONAL {
    ?a <http://…/ont#l3f> ?x2.
    ?x2 a <http://…/ont#l3>.
  }
  FILTER(!BOUND(?x1) || !BOUND(?x2))
}
```

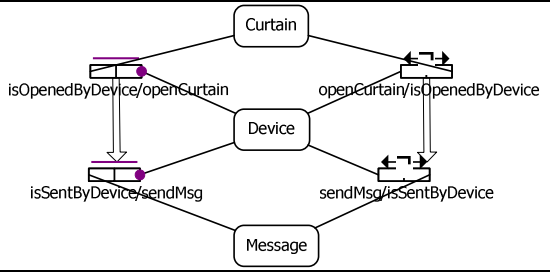
```
ASK {
  ?a <http://…/ont#l1f>
?b.
  ?b a
<http://…/ont#l1>.
  OPTIONAL {
    ?a ?z ?c. ?c a
<http://…/ont#l2>.
  }
  FILTER(!BOUND(?c))
}
```

3.3 Other Operators and Constraints

SDRule-L also includes other extensions, such as implication, negation, skipper (exception), cross-context equivalence and cross-context subtyping. Due to the limit of paper length, we refer to [1] for the details.

An important point we want to make in this paper is, all the operators and constraints that we have discussed earlier are supposed to be further used for modeling decisional alternatives (also called non-monotonic decision rules). *Implication* can be used for this purpose.

Fig. 4 shows an example of implication and its verbalization. An arrow tipped bar indicated with ¬ is a negation[14]. Implication can be mapped into a subset constraint when the decision rule is monotonic. When the rules are non-monotonic, we must use implication instead. Sometimes, one fact type may appear more than once within one SDRule-L model (e.g., as shown in **Fig. 4**). Note that in ORM, such duplications are not allowed.



If Device does not open Curtain, then Device does not send Message.
If Device open(s) Curtain, then Device send(s) Message.
Each Device open(s) at least one Curtain.
Each Device send(s) at least one Message.

Fig. 4. An example of implication and its verbalization

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. 4**, if

[14] When negation is used in a conditional statement, it is a constraint. When it is used in a conclusion, it is an operator.

openCurtain/isOpenedByDevice is populated, then *isSentByDevice/sendMsg* must be populated; otherwise, the latter must not be populated. Therefore, if we get an empty set of linked data, then it is valid. If we get a set of data as illustrated in **Table 4**, then the first record is valid and the second one is invalid.

Table 4. A table containing linked data that needs to be complied with **Fig. 4**

ID	*Curtain*	*Device*	*Message*
1	C1	D1	M1
2	C1	D1	NULL

The SPARQL queries are provided as follows.

```
ASK {
  ?a ont:OpenCurtain ?y.
  OPTIONAL{?a ont:sendMsg ?x.}
  FILTER(!BOUND(?x))
}
```

```
ASK {
  OPTIONAL{?a ont:sendMsg ?x.}
  ?a ont:OpenCurtain ?y.
  FILTER(!BOUND(?x))
}
```

Although implication can be used to model decisional alternatives, we want to use a neater and more concise means to model ontology-based decision rules. This particular approach is called semantic decision tables [7] [8]. How to use semantic decision tables for modeling decisional alternatives will be illustrated in the next subsection.

3.4 Using Semantic Decision Tables to Model Decisional Alternatives

A decision table is a table representing an exhaustive set of mutually exclusive conditions. It contains conditions, actions and decision rules. A condition consists of a condition stub represented as a label and a condition entry represented as a value or value range. Similarly, an action is composed of an action stub represented as a label, and an action entry, which is often a Boolean value[15]. A decision rule is a decision column or a decision row, depending on the layout of the decision table. A decision rule consists of a set of conditions and a set of actions.

A semantic decision table (SDT) is a decision table annotated with a domain ontology. With the annotation, we are able to specify the meta-rules and relations between table elements into axioms, which are processable by machines. **Fig. 5** shows an example of SDT that is equivalent to the SDRule-L illustrated in **Fig. 4**.

Condition	1	2
Device open Curtain	Yes	No
Action		
Device send Message	*	

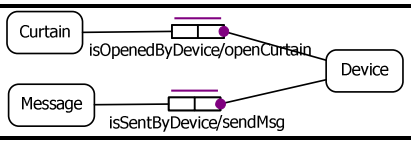


Fig. 5. An SDT example that is equivalent to **Fig. 4**

[15] If it is not a Boolean value, then we need to map it into a Boolean value.

In the decision table from **Fig. 5**, "*Device open Curtain*" is a condition stub. "*Yes*" and "*No*" are the two condition entries. "*Device send Message*" is an action stub. " * " and the absence of "*" are the two action entries. There are two conditions in the decision table: $\langle Device\ open\ Curtain, Yes \rangle$ and $\langle Device\ open\ Curtain, No \rangle$. There are also two actions: $\langle Device\ send\ Message, * \rangle$ and $\langle Device\ send\ Message, \rangle$. The two columns indicated with 1 and 2 are the two decision rules. We annotate this decision table with the fact types (from a domain ontology), which are constrained in the ontology as shown on the right in **Fig. 5**.

When we have complicated decision rules, SDTs are much better than the models containing implications for the following reasons.

- Within an SDT, decisional alternatives are easier to be compared.
- Decision rules in an SDT are an exhaustive set of mutually exclusive conditions. Therefore, the completeness of a rule set is ensured. When there are a lot of conditions, it is difficult to manually check the completeness of an SDRule-L model containing implications.
- It is easier to group rules and detecting irrelevant rule sets using the existing SDT rule engine. For SDRule-L implications, it is a big challenge to group similar rules.

3.5 Metis

As most tools from other fact-based modeling dialects as discussed in Sec. 2, Metis contains functions of graphically design and verbalization of SDRule-L models. The SDRule-L reasoner is also embedded to ensure the consistency of data.

Metis has been developed as a plugin framework. We have used Eclipse[16] Java software development kit (SDK) for the implementation. In particular, the Graphical Editing Framework (GEF) has been adopted for the rich graphical editor and views of Metis user interface. **Fig. 6** and **Fig. 7** show two screenshots of Metis.

Including the function of graphical modeling using drag and drop, Metis also supports importing/exporting SDRule-L models from/into SDRule-ML.

4 Discussion

An SDRule-L model can contain *dynamic* and *static* rules, and, *monotonic* and non-*monotonic* rules. Sequence can be used to create dynamic rules seeing that it contains dynamic aspects like time. Most SDRule-L constraints that are inherited from ORM2 are considered to be static. Implication (and the equivalent SDTs) is the only one constraint that can make a rule non-monotonic. All the rest, if they are not combined with implications, can only be used to model monotonic decision rules.

In this paper, we have illustrated how to use SDRule-L to validate linked data, which complies with an SDRule-L model. Our method of checking data consistency contains three steps: 1) transform into a static, OWL-compatible model; 2) create SPARQL queries that contain dynamic rules; 3) find counterexamples.

[16] www.eclipse.org

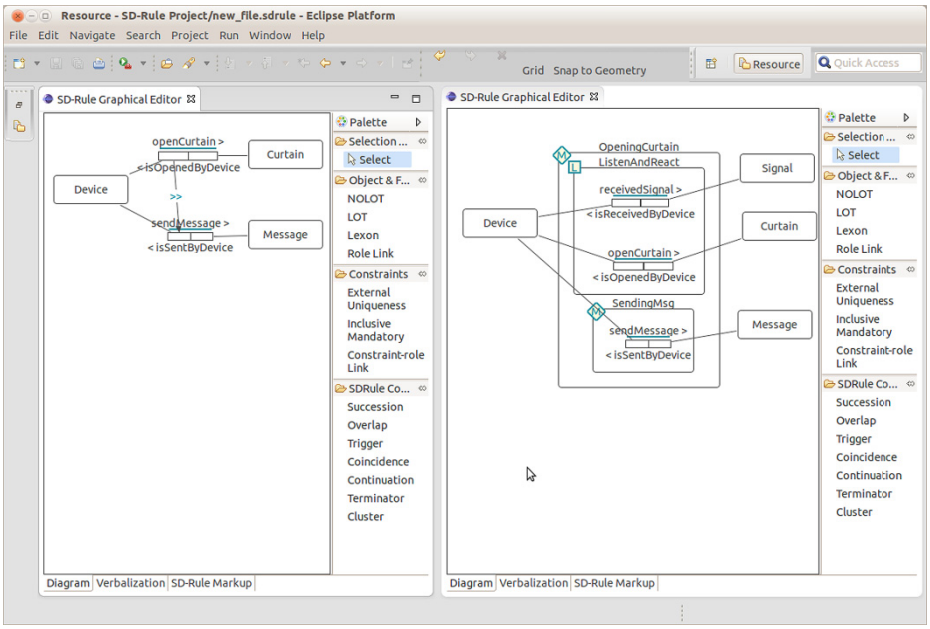


Fig. 6. Screenshots of Metis: graphical models w.r.t. **Fig. 1** and **Fig. 2**

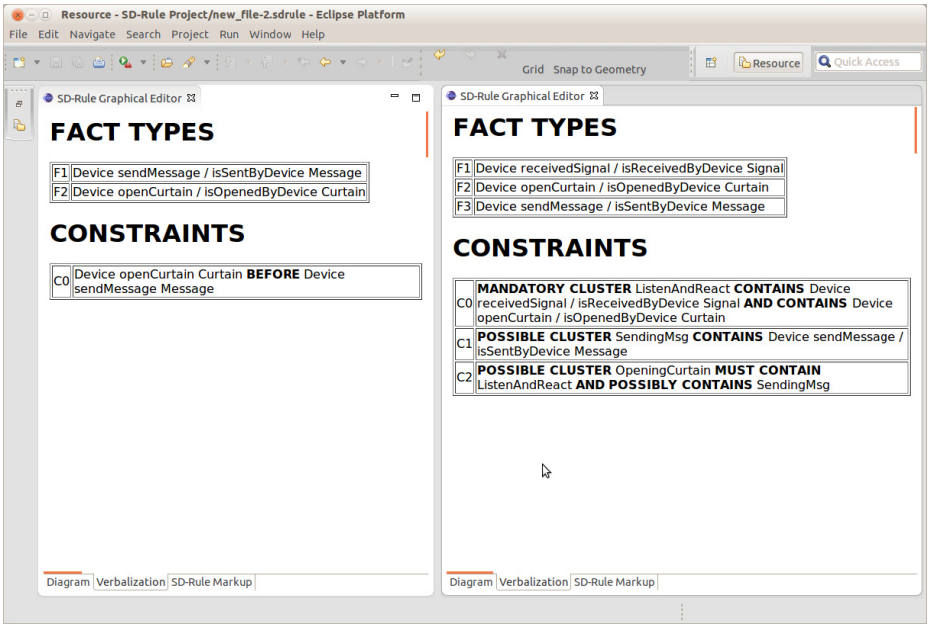


Fig. 7. Metis screenshot: Views of verbalization for the two graphical models in **Fig. 6**

5 Conclusion and Future Work

In this paper, we have discussed three constraints from SDRule-L – sequence, cluster and implication – and how we can use them to validate linked data. Implications can be further mapped into semantic decision tables, with which we can analyse the decision rules. We have also illustrated the first version of SDRule-L tool called Metis.

In the future, we want to add a function of model checksum to Metis. For example, we shall allow two continuation constraints with inverse directions to be used on two events.

Acknowledgements. This study was supported by the INNOViris Open Semantic Cloud for Brussels project, financed by the Brussels Capital Region. Our use case and experimental data in this paper are taken from this project.

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, ch. V. IGI Publishing, USA (2008)
2. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann (2008)
3. Demey, J., Jarrar, M., Meersman, R.: A Conceptual Markup Language that Supports Interoperability between Business Rule Modeling Systems. In: Meersman, R., Tari, Z. (eds.) CoopIS/DOA/ODBASE 2002. LNCS, vol. 2519, pp. 19–35. Springer, Heidelberg (2002)
4. Biletskiy, Y., Boley, H., Ranganathan, G.: RuleML-based learning object interoperability on the Semantic Web. Interact. Techn. Smart Edu. 5(1), 39–58 (2008)
5. Jarrar, M.: Towards Methodological Principles for Ontology Engineering. PhD Thesis, Vrije Universiteit Brussel, Brussel (2005)
6. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity Constraints in OWL. In: Fox, M., Poole, D. (eds.) AAAI, Atlanta, Georgia, USA (2010)
7. Tang, Y.: On Semantic Decision Tables. PhD Thesis, Department of Computer Science, Vrije Universiteit Brussel, Brussels (2009)
8. Tang, Y.: Semantic Decision Tables - A New, Promising and Practical Way of Organizing Your Business Semantics with Existing Decision Making Tools. LAP LAMBERT Academic Publishing AG & Co., Saarbrücken, Germany (2010)