

# **Semantic Search Interface for an Intelligent Network Management System—A Case Study**

Kasper Apajalahti

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 30.9.2015

**Thesis supervisor:**

Prof. Eero Hyvönen

**Thesis advisor:**

D.Sc. (Tech.) Vilho Räsänen

Author: Kasper Apajalahti		
Title: Semantic Search Interface for an Intelligent Network Management System—A Case Study		
Date: 30.9.2015	Language: English	Number of pages: 12+89
Department of Computer Science		
Professorship: Media Technology		
Supervisor: Prof. Eero Hyvönen		
Advisor: D.Sc. (Tech.) Vilho Räsänen		
<p>The thesis is part of a project which aims to enhance the automation in mobile network management with a statistical reasoner. The statistical reasoner analyses given network data and creates configuration proposals to the network.</p> <p>This thesis is a case study for building a semantic search interface which demonstrates how the statistical reasoner behaves in the mobile network. The demonstrator is implemented as a faceted search interface which uses Semantic Web technologies for data management and HTML 5 for the graphical user interface. The objective of the case study is to discover the information need of the user and to find methods to provide the needed information. The final implementation presents three separate faceted search views in order to answer the information need: one for describing the relation between the input and output of the reasoner, one for the relation between the output and its impact on the network, and one for the rule base of the reasoner.</p> <p>The evaluation of the implementation shows that the chosen techniques and methods support user in exploring the relevant network- and reasoner-related information. The evaluation also discovered some deficiencies which is valuable information for the case study. Finally, this case study produced new thoughts with respect to the initial objectives and to future directions for the project.</p>		
Keywords: LTE, SON, Semantic, Web, RDF, OWL, SPARQL, Network Management, User Interface, GUI, Exploratory, Faceted, Search, Browsing, Search Interface, Artificial Intelligent, AI, Reasoning		

Tekijä: Kasper Apajalahti		
Työn nimi: Tapaustutkimus: Semanttinen hakukäyttöliittymä älykkääseen verkkohallintajärjestelmään		
Päivämäärä: 30.9.2015	Kieli: Englanti	Sivumäärä: 12+89
Tietotekniikan laitos		
Professuuri: Mediatekniikka		
Työn valvoja: Prof. Eero Hyvönen		
Työn ohjaaja: TkT Vilho Räisänen		
<p>Tämä diplomityö on osa projektia, jossa tutkitaan, miten tilastollisella päättelijällä voidaan parantaa mobiiliverkon automaattista hallintaa. Tilastollinen päättelijä analysoi mobiiliverkosta saatavaa dataa ja tuottaa muutosehdotuksia mobiiliverkon parametreihin.</p> <p>Diplomityö toteutetaan tapaustutkimuksena, jossa suunnitellaan ja toteutetaan semanttinen hakukäyttöliittymä havainnollistamaan tilastollisen päättelijän toimintaa. Käyttöliittymä toteutetaan moninäkömähäulla, joka hyödyntää Semanttisen Webin tekniikoita tiedonhallinnassa ja HTML 5 -tekniikoita graafisessa käyttöliittymässä. Tapaustutkimuksen tavoitteena on löytää käyttäjän tiedontarve sekä menetelmät olennaisen tiedon tarjoamiselle. Toteutettu hakukäyttöliittymä esittelee kolme erillistä moninäkömähakua, joilla vastataan käyttäjän tiedontarpeeseen: ensimmäinen näkymä kuvaa päättelijän syötteen ja vasteen välistä suhdetta, toinen kuvaa vasteen ja sen vaikutusten suhdetta ja kolmas kuvaa päättelijän sääntökantaa. Sovelluksen arviointi osoittaa, että valitut tekniikat ja menetelmät tukevat hyvin käyttäjän tiedontarvetta ja tiedonhakuprosessia. Arvioinnissa käytiin läpi myös joitakin toteutuksen puutteita, jotka antoivat uutta arvokasta tietoa tapaustutkimukselle. Lopussa esitellään uusia ajatuksia liittyen tutkimuksen tavoitteisiin sekä mahdollisiin jatkotutkimusaiheisiin.</p>		
Avainsanat: Semanttinen, Web, LTE, SON, mobiiliverkko, tekoäly, käyttöliittymä, tiedonhaku, tapaustutkimus, RDF, OWL, SPARQL, tutkiva haku, moninäkömähaku, fasettihaku, fasettiselaus, fasetti		

## Preface

It has been a pleasure for me to do the thesis from such an interesting research topic. This project has been a great opportunity for me to learn technical aspects of mobile network management, knowledge engineering, and search interfaces.

I want to thank my employer Nokia Networks for giving me this opportunity. Also, I thank my instructor Vilho Räsänen for his feedback and guidance throughout the project. My thanks go also to my supervisor Eero Hyvönen for his guidance and support during the thesis work and throughout my studies.

I would also like to thank my family for all the valuable help and support during my studies. Finally, special thanks go to Mirva for all the love and support during these years.

Espoo, September 2015

Kasper Apajalahti

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Abstract (in Finnish)</b>	<b>iii</b>
<b>Preface</b>	<b>iv</b>
<b>Contents</b>	<b>v</b>
<b>I Background</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Overview . . . . .	2
1.2 Motivation . . . . .	2
1.3 Outline . . . . .	3
<b>2 Telecommunication Network Technologies</b>	<b>5</b>
2.1 Overview . . . . .	5
2.1.1 Mobile Network . . . . .	5
2.1.2 Long Term Evolution (LTE) . . . . .	6
2.1.3 Network Management . . . . .	6
2.2 Self-Organizing Network (SON) . . . . .	8
2.2.1 SON Overview . . . . .	8
2.2.2 Self-configuration . . . . .	8
2.2.3 Self-optimization . . . . .	9

2.2.4	Self-healing . . . . .	9
2.2.5	Coordination and Co-design of SON Functions . . . . .	10
2.3	SON Evolution . . . . .	11
2.3.1	Challenges in SON . . . . .	11
2.3.2	SON Future . . . . .	12
2.3.3	Vision of Cognitive Network . . . . .	12
<b>3</b>	<b>Knowledge Engineering</b>	<b>13</b>
3.1	Methodology . . . . .	13
3.1.1	Knowledge Representation . . . . .	13
3.1.2	Ontology . . . . .	14
3.2	Semantic Web . . . . .	14
3.2.1	Semantic Web Overview . . . . .	14
3.2.2	RDF and RDFS . . . . .	15
3.2.3	OWL . . . . .	15
3.2.4	SKOS . . . . .	16
3.2.5	SPARQL . . . . .	17
3.3	Markov Logic Network . . . . .	17
3.4	Knowledge Systems in Telecommunication Field . . . . .	18
3.4.1	RDF Dataset for Mobile Network . . . . .	18
3.4.2	OWL Reasoner for SON Conflict Analysis . . . . .	19
3.4.3	MLN Reasoner in SON Verification . . . . .	19
<b>4</b>	<b>Visualization and Interaction Methods</b>	<b>20</b>

4.1	Visualization Techniques . . . . .	20
4.2	Exploratory Search . . . . .	20
4.2.1	Overview . . . . .	20
4.2.2	Search Activities . . . . .	21
4.3	Faceted Search . . . . .	22
4.3.1	Overview . . . . .	22
4.3.2	Determination of Facets . . . . .	23
4.3.3	Semantic Faceted Search . . . . .	24
4.3.4	Fuzzy Faceted Search . . . . .	24
<b>II</b>	<b>Design and Preliminaries</b>	<b>27</b>
<b>5</b>	<b>System Design</b>	<b>28</b>
5.1	System Description . . . . .	28
5.2	MLN Reasoner . . . . .	28
5.3	Use Case Analysis . . . . .	30
5.4	Demonstrator Architecture . . . . .	31
5.4.1	Overview . . . . .	31
5.4.2	Architecture . . . . .	31
5.5	Demonstrator Views . . . . .	32
5.5.1	Cell Status View . . . . .	32
5.5.2	Cell Event View . . . . .	34
5.5.3	Rule View . . . . .	35

5.5.4	MLN Rule Updates . . . . .	36
5.6	Demonstrator Ontology . . . . .	36
5.6.1	Network Ontology . . . . .	36
5.6.2	MLN Ontology . . . . .	37
5.6.3	Facet Ontologies . . . . .	40
<b>6</b>	<b>Tools and Libraries for the Demonstrator</b>	<b>42</b>
6.1	Ontology Management . . . . .	42
6.1.1	SPARQL Engine . . . . .	42
6.1.2	DL Reasoners for OWL 2 Ontology . . . . .	42
6.2	Graphical User Interface . . . . .	43
6.2.1	Sortable Table . . . . .	43
6.2.2	Network Graph . . . . .	43
6.2.3	Timeline . . . . .	44
6.2.4	Faceted Browsing . . . . .	45
<b>III</b>	<b>Implementation</b>	<b>47</b>
<b>7</b>	<b>Graphical User Interface</b>	<b>48</b>
7.1	GUI Overview . . . . .	48
7.2	Cell States . . . . .	49
7.2.1	Description . . . . .	49
7.2.2	Use Case: Selecting Facets . . . . .	50
7.3	Cell Events . . . . .	53



7.3.1	Description . . . . .	53
7.3.2	Use Case: Selecting Facets . . . . .	54
7.4	Rules . . . . .	56
7.4.1	Description . . . . .	56
7.4.2	Use Case: Selecting Facets . . . . .	56
7.4.3	Use Case: Creating a Rule . . . . .	57
7.4.4	Use Case: Updating Weights . . . . .	58
7.4.5	Use Case: Removing Individual Rules . . . . .	59
7.4.6	Use Case: Removing a Set of Rules . . . . .	60
7.5	Plugins for the GUI . . . . .	61
<b>8</b>	<b>Data Management</b>	<b>63</b>
8.1	Data Sequence Diagram . . . . .	63
8.2	Input Data to RDF . . . . .	63
8.3	Consistency Checking of the Ontology . . . . .	67
8.4	SPARQL Server . . . . .	67
8.5	Creating Facets . . . . .	67
8.6	SPARQL Patterns for the GUI . . . . .	68
8.6.1	Result Set . . . . .	68
8.6.2	Facets . . . . .	69
8.6.3	Rule Updates . . . . .	69
<b>IV</b>	<b>Discussion</b>	<b>71</b>

<b>9</b>	<b>Evaluation</b>	<b>72</b>
9.1	Information Quality . . . . .	72
9.1.1	Implementation . . . . .	72
9.1.2	Deficiencies . . . . .	73
9.2	Facets . . . . .	73
9.2.1	Implementation . . . . .	73
9.2.2	Deficiencies . . . . .	74
9.3	Visualization Methods . . . . .	75
9.3.1	Implementation . . . . .	75
9.3.2	Deficiencies . . . . .	75
9.4	Other Functionality . . . . .	75
9.4.1	Implementation . . . . .	75
9.4.2	Deficiencies . . . . .	76
9.5	Scalability . . . . .	76
9.5.1	Implementation . . . . .	76
9.5.2	Deficiencies . . . . .	77
<b>10</b>	<b>Conclusion</b>	<b>78</b>
10.1	Future Work . . . . .	78
10.1.1	System Management with High-level Goals . . . . .	78
10.1.2	Utilizing Linked Data in Reasoning . . . . .	78
10.2	Summary . . . . .	79
<b>A</b>	<b>Appendix SPARQL Update Generating Indicator Facets</b>	<b>85</b>

<b>B Appendix SPARQL Update Generating Relations Between Facet Values and Instances</b>	<b>86</b>
<b>C Appendix SPARQL Update for Creating a New Rule</b>	<b>86</b>
<b>D Appendix SPARQL Updates for Modifying and Removing Rules</b>	<b>87</b>

## Abbreviations

3GPP	3rd Generation Partnership Project
ANR	Automatic Neighbor Relation
BTS	Base Transceiver Station
CM	Configuration Management
CQI	Channel Quality Indicator
DL	Description Logic
DMS	Domain Management System
eNB	E-UTRAN Node B
EPC	Evolved Packet Core
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FM	Fault Management
FOL	First-order Logic
KPI	Key Performance Indicator
KR	Knowledge Representation
LTE	Long Term Evolution
MLB	Mobile Load Balancing
MLN	Markov Logic Network
MRO	Mobility Robustness Optimization
NMS	Network Management System
OAM	Operation, Administration and Maintenance
OPEX	Operating Expense
OWL	Web Ontology Language
PM	Performance Management
RDF	Resource Description Framework
RET	Remote Electrical Tilt
RLF	Radio Link Failure
SKOS	Simple Knowledge Organization System
SNR	Signal to Noise Ratio
SON	Self-Organizing Network
SPARQL	The SPARQL Protocol and RDF Query Language
Txp	Transmit Power
UE	User Equipment
UMTS	Universal Mobile Telecommunications System

Part I

# Background

# 1 Introduction

## 1.1 Overview

Mobile networks have rapidly become a crucial part of our everyday life and yet their significance to us will increase in the future. Already seven billion mobile phones have been subscribed globally while the majority (65 %) of new mobile phones sold are smart phones [1]. The growth is expected to be even larger in the future. Especially mobile data traffic (downlink) is expected to explode compared with the current situation. [1][2][3]

Growing complexity of telecommunication networks requires more automation from the network management layer. Currently researched and gradually standardized technology in telecommunication field is Self-Organizing Networks (SON) which already solves automatically some management tasks in a limited context. In addition to the SON technology, there is also effort to create fully automated management systems (autonomic computing systems) where human intervention is minimized [3][4][5].

As the complexity and level of automation rises in network management systems, they will also become less understandable to human. However, it is crucial that human operator monitors and controls the system behaviour even though the autonomic system could solve the majority of management tasks. The human operator is needed for example to verify the functionality and handle fault situations of the system. Thus, an autonomic computing system needs special attention on a user interface construction because the complex functionality of the system needs to be transparent to the user. [6]

## 1.2 Motivation

This thesis will be part of a project which aims to improve automation in a mobile network management system by using statistical relational learning [7]. Implementation is done with Markov Logic Network (MLN) [8] which has a weighted rule base for modelling the uncertainty and logical relations of a mobile (cellular) network. With respect to network performance measurements, an MLN reasoner is used to infer optimal configurations to cells and to learn better weights in the rule base.

Besides the MLN reasoning system, an important aspect in this project is to demonstrate the behaviour of the reasoner to a human user. First, it is important that another researcher or developer learns and understands how the reasoner works in order to enhance the functionality of the reasoner. Second, the automated manage-

ment system should be under human control and thus the human operator should also understand how the reasoner behaves.

The thesis focuses on creating a case study about the demonstrator. More precisely, the case study is about building a semantic search interface which provides interactive information exploration for the user. The search interface is built with faceted browsing activities and with visualization methods which support user's information exploration. Data management in this case study is handled with Semantic Web technologies to provide flexible facet creation and representation of relational data (network- and MLN-related data).

Case study is based on five research questions which relate to the functionality of the semantic search interface:

1. What information is needed?
2. Which visualization methods support the information need?
3. Which facets support the information need?
4. What other functionality supports the information need?
5. How to provide scalability for the search interface?

The first research question, the information need, is analysed before the following three questions (visualization methods, facets, and other functionality) can be examined. The final research question, scalability, relates to data model design which enables the extension and modification of the demonstrator.

### 1.3 Outline

This thesis is divided into following parts: background, design and preliminaries, implementation, and discussion. Background part (I) introduces relevant background information for the case study, such as telecommunication network technologies (including automated management systems), knowledge engineering, and user-centric information exploration.

Design and preliminaries part (II) describes the system design including the underlying MLN reasoning system and the design of the demonstrator. Moreover, this part analyses some tools and libraries that can be used in the data management and in the GUI.

Implementation part (III) presents the application by first presenting data processing methods in the demonstrator and then describing the graphical user interface with practical examples.

Finally, discussion part (IV) evaluates the implementation with respect to research questions and then summarises the case study.



## 2 Telecommunication Network Technologies

This section gives an overview to mobile network technologies. First, the notion of the mobile network is presented by describing its basic components, introducing the Long Term Evolution (LTE) standard and explaining the process in mobile network management. Second, an automation technology for mobile network management, Self-Organizing Network (SON), is introduced. Finally, there is a discussion about future trends in SON and generally in automatic mobile network management.

### 2.1 Overview

#### 2.1.1 Mobile Network

A mobile network is run by a network operator and it is used for a wireless communication between user equipments (UE) such as mobile phones and tablets. A connection from a UE to a mobile network is enabled with Base Transceiver Stations (BTS) which operator has located densely in order to provide better coverage areas for customers. The coverage areas that a BTS creates are called cells. Basically, cells are generated with BTS antennas and thus the shape and size of cell areas are dependent on (directional) antenna parameters such as tilt (antenna angle), transmission power, and beam width. [9]

A UE is typically connected to one cell at a time. By default, it is connected to a cell with the highest signal to noise ratio (SNR). Whenever SNR is decreasing under some threshold value, for example when a user is moving to the edge of a cell area, the connection is transferred to a neighbouring cell with higher SNR. [9] This action is called handover and it is provided by allowing neighbouring cells to have overlapping areas. Figure 1 illustrates a simple case in which a UE is moving away from the cell it is connected to (Cell1) and finally does a handover (to Cell2) at the overlapping area of the two cells.

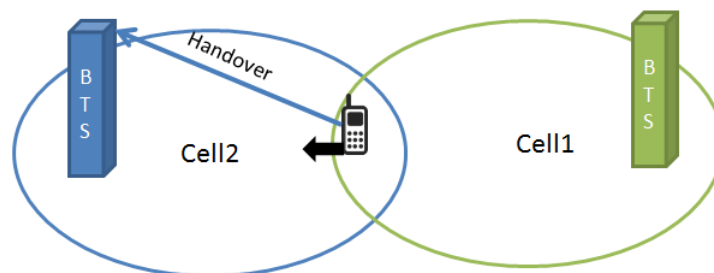


Figure 1: UE handover when moving from one cell area to another

### 2.1.2 Long Term Evolution (LTE)

LTE is a 4G communication standard and the latest standard for a radio access network. It is evolved from the 3G technology, Universal Mobile Telecommunications System (UMTS). Compared with UMTS, LTE offers features such as a higher data rate, lower transmission latency, lower cost per bit, simplified network architecture, and packet optimized radio access networks [3][10]. Unlike UMTS, LTE does not support a circuit-switched domain but instead uses a packet-switched domain also for voice calls (voice over IP). [9][10]

In figure 2 can be seen a simplified LTE network architecture. As discussed in the previous section, UEs are interacting with base stations which are called eNodeBs or eNBs in LTE network. eNBs are communicating with X2 interface with each other and together they are forming a network called Evolved Universal Terrestrial Radio Access Network (E-UTRAN). An S1 interface connects eNBs to an IP-based core network, Evolved Packet Core (EPC). An EPC contains components for UE-related data handling such as authentication and IP address allocation. [10] [11]

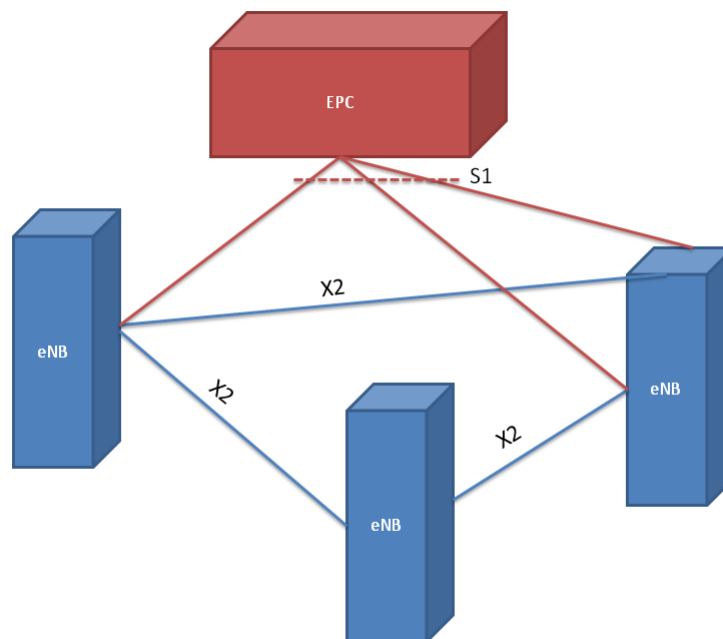


Figure 2: LTE network architecture [11]

### 2.1.3 Network Management

Management in a mobile network is handled with an architecture called operation, administration and maintenance architecture (OAM) which is used to control network behaviour. OAM can be divided into two levels: a domain management system

(DMS) and network management system (NMS). A DMS monitors the states of the vendor-specific network elements (eNBs and EPC components) and controls their configurations, such as tilt angles, transmission powers, and beam widths. Although eNBs can communicate with each other directly via X2 interface, DMS can control only network elements under its own domain. [3]

An NMS is an integration of various vendor-specific DMSs and with the NMS it is possible to control network elements in multiple domains. In figure 3 is shown the OAM architecture with interfaces between DMS and NMS (Itf-N) and between DMS and network elements (Itf-S). [3]

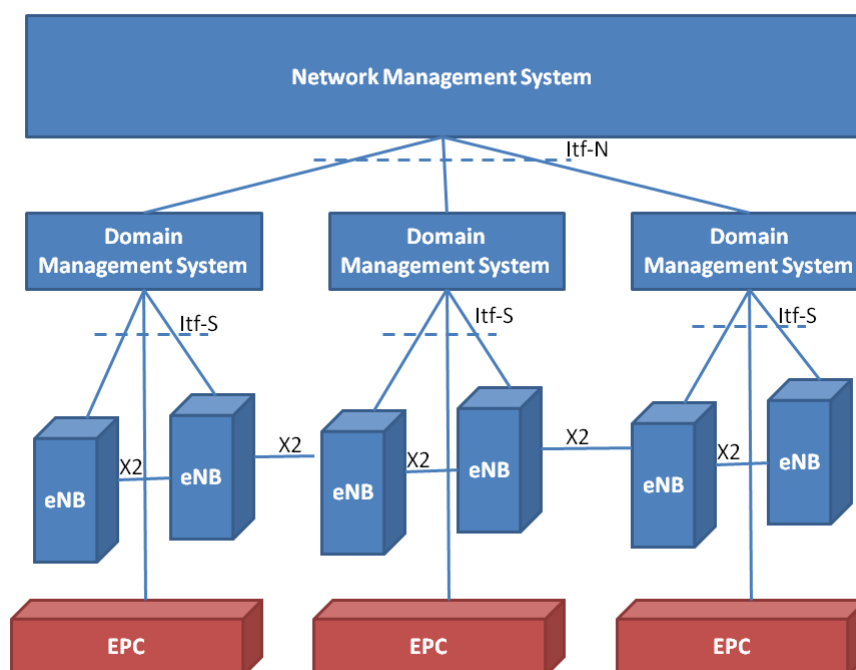


Figure 3: OAM architecture [3]

Data produced by network elements can be divided into performance-, fault- and configuration management data (PM, FM, and CM). PM is used in monitoring and analysing the performance of network elements such as signal quality, energy consumption, and handover performance. FM is operating with network problems including alarms and faults. CM provides data about configurations and it enables remote configuration to network element parameters. [3]

Measuring and analysing the network performance is a complex task as there is huge amount of data and correlations with other data elements. Therefore, PM data is usually processed with algorithms to more informative measurements. These measurements are called key performance indicators (KPIs) which give an overview of cell states. For example, one KPI describes number of radio link failures (RLF) per interval and another KPI describes channel quality (CQI) for a cell. [3] Both

CQI and RLF are used in this project for network performance analysis.

## 2.2 Self-Organizing Network (SON)

Along with LTE, Self-Organizing Network (SON) is a new standard which defines automation for network management. SON aims to reduce operating expenses by automating management tasks. [12] Also, the benefits of SON is improvement in network performance by having proactive actions and faster reactions to incidents. [13]

This section introduces the SON concepts by first giving an overview of the technology and then more detailed description of three different categories of SON functions. Finally, there is a description of SON coordination which is a technology and research topic examining the coordination of SON functions.

### 2.2.1 SON Overview

SON is designed as a set of functions which solve some tasks in the network. The functions are related to configurations (self-configuration), optimizations (self-optimization), and failure recoveries (self-healing). [3] Self-configuration functions are used when new network elements (eNBs) are deployed to the network. Self-optimization functions are used in monitoring the network and improving network performance with respect to some criteria, for example energy saving or cell load balancing. Self-healing functions aims to detect, diagnose, and recover from failure situations.

SON functions are located in several places in the network, for example in NMSs, DMSs, and eNBs. SON architecture is called centralized if SON functions are located in management layers (DMS and NMS) and distributed if they are in eNBs and EPC components. However, a practical solution is a hybrid architecture in which centralized and distributed SON functions run simultaneously. [12][3]

### 2.2.2 Self-configuration

Self-configuration contains automatic processes for network element deployment with a plug-and-play behaviour. When an eNB is powered up, auto-connectivity and auto-commissioning functions will provide an automatic IP address allocation, authentication, and configuration with the OAM system. Moreover, the functions provide a connection establishment with the EPC layer and with other eNBs. [3]

After eNB is successfully installed, new cells need to be linked to adjacent cells with neighbor relations. Automated Neighbor Relation (ANR) function is used to detect and update neighbour cells with respect to UE measurements. The inner logic of ANR function depends on vendor, as every vendor has their own local ANR implementation inside their domain network. [14]

### 2.2.3 Self-optimization

The mobile network requires continuous monitoring and parameter reconfiguration as the environment is constantly changing. Self-optimization functions solve proactively potential incidents and other cases in the network. For example, rapidly changing traffic load might require reconfiguration in cell parameters in order to have better coverage areas, handover success ratios and channel qualities. [3] ANR is also used in self-optimization because coverage areas and handover behaviour may change during run time and therefore relevant neighbour relations need to be maintained. [15]

One important SON function in optimization is Coverage and Capacity Optimization (CCO) which optimizes cell coverage areas. Basically, it adjusts the antenna tilts and transmission powers so that gaps in coverage areas are avoided. [15] However, too much overlap between cells leads to interference and thus CCO has to find optimal solution between ideal coverage area and minimal interference. [3]

Mobility Robustness Optimization (MRO) and Mobility Load Balancing (MLB) functions optimize handover functionality between cells. The MRO optimizes handover parameters by minimizing call drops and radio link failures but avoiding unnecessary handovers such as the ping-pong effect (where a UE is frequently having handovers between a cell pair). The MLB function balances cell loads by enabling handovers in cell borders from an overloaded cell to a cell with free space. As well as the MRO, also the MLB configures handover parameters to provide its functionality. [3]

Yet another important optimization function is energy saving function which is used to save overall energy consumption and operating expenses in the mobile network by switching some cells on and others off. However, while minimizing the energy consumption, the energy saving function has to avoid holes in coverage areas and consider quality of service (user satisfaction). [3] Energy saving function is suitable in time periods when overall traffic is known to be slight for example at night time.

### 2.2.4 Self-healing

Failures in network elements are found with an alarm system where KPIs are monitored and alarms are raised if KPIs have abnormal values. Self-healing functions can resolve incidents related to network failures. When a self-healing function is

implemented for a specific alarm, it is possible to recover from that without manual work. [16]

One use case for self-healing is cell outage management, which includes functionality for detecting, recovering and compensating cell outage. After an outage is detected with alarms or KPI measurements, the first action is to try recovery actions such as software restart or re-configuration. If cell outage persists even after recovery actions, compensation is needed. The objective of compensation is to minimize degradation in network performance by adapting the parameters of neighbour cells so that they fill the hole in coverage area caused by the defective cell. [17]

Another use case for self-healing is the self-recovery of a network element software. In this use case, a software failure is detected in a cell and it can be healed by loading either the latest backup or initial status of the software. Action results are verified and if the problem still occurs, incident is further escalated to the operator. [16]

### 2.2.5 Coordination and Co-design of SON Functions

Using SON functions in the network system helps in automating some tasks. However, there can be situations where several SON functions act concurrently, which might lead to unwanted results. For example, some anomaly can be caused by several reasons and thus it might also trigger several SON functions as a solution [3]. Although individual SON functions improve the network performance, the impact might be opposite when they are running concurrently.

One aspect in conflict prevention is to have co-design between SON functions. Co-design aims at disjoint SON functions by minimizing the number of shared parameters and having policies of how to co-operate with other SON functions. For example, SON functions could be designed so that only one SON function can be triggered with respect to a specific network state. However, as the number of SON functions increases, it becomes difficult to avoid all possible conflicts with a proactive design. [3]

In practice, conflicts can not be fully avoided with co-design and therefore automatic conflict detection and resolution is needed in the run-time environment. [3] SON coordinator is a technology which tries to solve these issues. The coordinator detects conflicts by monitoring network parameters and configuring and controlling SON function activities. SON function monitoring may include KPI measurements of cells that are influenced in a given impact time and impact area. The coordinator can also give priority values for SON functions to solve which function is executed in a conflict. [3][18]

In addition to the network configuration, the SON coordinator has an interface to

a network operator to enable a human user to define high-level objectives for the coordinator. A policy function is used to convert high-level objectives to specific parameters which the SON coordinator is in turn using to define SON function activities. Figure 4 shows an overview of the SON coordinator and its interfaces to the network configuration and operator. [18]

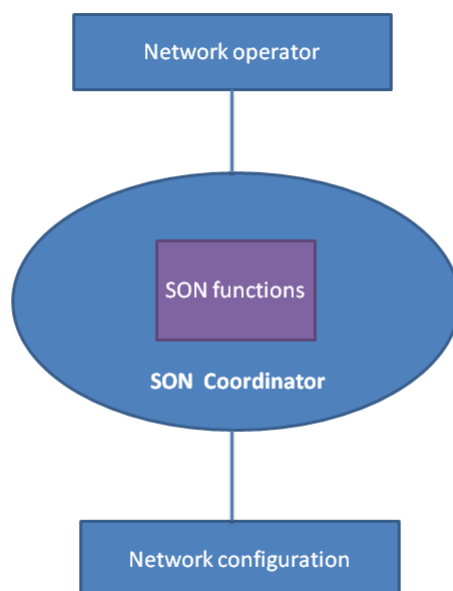


Figure 4: SON coordinator architecture [18]

## 2.3 SON Evolution

### 2.3.1 Challenges in SON

Currently, SON functions and the SON coordinator are forming a system which has capabilities to automatically maintain the mobile network in limited situations. However, the human operator still has a significant amount of domain knowledge which neither SON functions nor the SON coordinator can access. Thus, a crucial part in SON evolution would be knowledge acquisition from the operator to the system. [3]

In addition to knowledge acquisition, the current system has limitations in the expressiveness of knowledge representation. To provide more complex concepts, knowledge should be represented with a formal knowledge model such as an ontology. [3]

A practical challenge in SON is also the current multi-vendor system which operator handles with the NMS. Although SON functions have been standardized, every vendor has its own implementations of SON functions which adds complexity to network operations.

### 2.3.2 SON Future

The effort in network operation has been in the development of an autonomous closed-loop SON system in which human is not involved in the network configuration. Closed-loop system requires some high-level goals of the operator which can be translated into SON parameters and further to network actions. [17][19][20]

Besides high-level goals, a decision support system (DSS) has been envisioned which would assist the operator in making decisions related to network planning. With respect to the network status, the system could recommend new locations for base stations and calculate resource costs for quality of service. [19]

Adaption of machine learning techniques and knowledge engineering to SON functions have also been studied in research literature. For example, a future self-configuration function could deploy a cell to the network with parameter values that have been learned from a similar context. [3]

In addition to machine learning, a semantic knowledge model would provide a better understanding for the system about the context. If the network (elements, objectives and actions) were semantically defined, it would enable the system to infer relationships and to make decisions with respect to the semantic relations. [3]

### 2.3.3 Vision of Cognitive Network

The basic idea of cognitive network management architecture is related to IBM's general vision of autonomic computing [4]. The idea is the same with mobile network management: user-defined high-level goals are input to a closed-loop management system. [3] This vision without SON functions is also researched in telecommunication projects [21][22] and books [3]. These projects suggest that instead of SON functions, the management system will make dynamic configurations directly to network elements. Generally, these architectural visions include components for network data processing, data analysis, high-level goals, a knowledge base, and reasoning tasks. [3][21][22]



## 3 Knowledge Engineering

As this thesis aims to build a semantic search interface which demonstrates behaviour of a statistical reasoner, some background about knowledge engineering is presented. This section introduces some concepts related to knowledge representation, Semantic Web technologies, and statistical reasoning. After that, there is discussion about LTE- and SON-related research projects which use some knowledge engineering techniques in their systems.

### 3.1 Methodology

#### 3.1.1 Knowledge Representation

Knowledge representation (KR) is a subfield of artificial intelligence (AI) and it is a paradigm for making a formal and structured representation for collected information. With a formal representation of knowledge it is possible to express deeper meanings of concepts, for example by adding logical relationships between concepts with semantics. [23]

KR can also be described with five distinct roles which characterize KR from different views [24]:

- *Surrogate*, meaning that KR is a substitute of a real world object. However, it is always an incomplete description of the real object. [24]
- *Set of ontological commitments*, meaning that the description of a concept should be focused on relevant aspects with respect to the domain. [24]
- *Fragmentary theory of intelligent reasoning*, which is defined with three parts: 1) how reasoning is defined in its context 2) what can be reasoned and 3) what reasoning tasks are recommended. [24]
- *Medium for efficient computation*, meaning that knowledge should be organized in a way that relevant (recommended) reasoning will be efficient. [24]
- *Medium for human expression*, meaning that KR acts as a communication tool which human uses to express knowledge to computer. [24]

As reasoning is an important part of knowledge representation, it has a relation to formal logic systems, for example to first-order logic (FOL) which is a formal and compact way to describe logical relations. [23] Another family of logic systems is description logics (DL) which are fragments of FOL. [25]

### 3.1.2 Ontology

Ontology in computer science means a formal description of domain knowledge. In ontology, concepts are organized into a structure with relationships such as hierarchies of classes, value restrictions, disjointness statements, and specification of logical relationships. [26]

An ontology described in DL can be divided into two parts: terminological (TBox) part and assertion (ABox) part. TBox defines concepts and relationships (axioms and properties) between concepts. Thus, TBox contains the structural design and rules of concepts. For example, TBox in a mobile network ontology could have a concept `Cell` which has subclasses `Macrocell`, `Microcell`, and `Picocell` with respect to the cell type. Moreover, some properties can be defined to concepts, for example `Cell` can be defined to belong to only one base station.

ABox completes the ontology by defining instances of the TBox concepts. Example above could have instances `Microcell11234` which belongs to a base station `eNB5678`.

## 3.2 Semantic Web

The Semantic Web uses knowledge representation in order to describe resources and their relationships on the web. This subsection gives first an overview of the Semantic Web and after that the most significant parts are presented.

### 3.2.1 Semantic Web Overview

The Semantic Web is a vision of a more intelligent Internet where machines can have more detailed understanding about contents on the web. Technologies and standards related to the Semantic Web are developed by A World Wide Web Consortium (W3C). [26]

W3C defines Semantic Web technologies which enable knowledge representation and reasoning for web resources. Resources are defined using Uniform Resource Identifiers (URIs) which define resources globally. Resource Description Framework (RDF), RDF Schema (RDFS), and Web Ontology Language (OWL) are the core of knowledge representation on the Semantic Web and they are discussed in more detail in following sections. [26]

### 3.2.2 RDF and RDFS

RDF is a data model for defining relations between data resources in the web. An RDF statement includes three components: subject, predicate, and object. Subject is the resource to be defined, predicate is its property, and object is a value of a property. A simple example of an RDF statement is described below. The statement describes two cell instances and a neighbor relation between them.

```
PREFIX: c: <http://.../cell-ontology#> .
PREFIX: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
c:C1 rdf:type c:Cell .
c:C2 rdf:type c:Cell .
c:hasNeighbor rdf:type rdf:Property .
c:C1 c:hasNeighbor c:C2 .
```

From the example can be seen that all triple components are globally unique as they are defined with namespaces: cell-related components belong to a namespace `http://.../cell-ontology#` and RDF-related components to `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.

RDF has semantic limitations because it provides semantics only for simple triple statements. However, with RDFS it is possible to increase its expressive power with constructs such as classes, hierarchical properties and value restrictions. The example below shows how cells and neighbor relations between cells can be defined together with RDF and RDFS. As an addition to the statement above, `hasNeighbor` property has domain and range types specified.

```
c:Cell rdf:type rdfs:Class .
c:hasNeighbor rdf:type rdf:Property .
c:hasNeighbor rdfs:domain c:Cell .
c:hasNeighbor rdfs:range c:Cell .
c:C1 hasNeighbor c:C2 .
```

The example above misses statements of what type of classes `c1` and `c2` are. However, because of RDFS definitions about `hasNeighbor` property (domain and range), instances `c:C1` and `c:C2` can be reasoned to be cells (`c:Cell`).

### 3.2.3 OWL

OWL provides more expressive formal definitions for classes and relations, for example equality, disjointness, and cardinality. By using OWL, one can express characteristics

of relation properties such as functionality, symmetricity, and transitivity. These characteristics help to make more reasoning in an incomplete ontology. [27]

The example below shows how OWL can enrich the previous examples (presented in section 3.2.2) by adding semantics to property characteristics. In addition to earlier statements, a machine can now infer that **C2** is also neighbor of **C1**. Moreover, functional property for **belongsToEnb** indicates that a cell can belong to only one eNB. Therefore, if there is a relation from one cell to two eNBs, the reasoner either marks the eNBs equivalent or raises an error due to a conflict.

```
c:hasNeighbor rdf:type owl:SymmetricProperty .
c:ENB rdf:type rdfs:class .
c:belongsToEnb rdf:type owl:FunctionalProperty .
c:belongsToEnb rdfs:domain c:Cell .
c:belongsToEnb rdfs:range c:ENB .
```

OWL originally had three sublanguages defined: OWL Full, OWL DL and OWL Lite. OWL Full is the widest language with respect to the expressive power but it is not logically decidable. OWL DL has a correspondence to a description logic. It has the same elements as OWL Full but restricts the usage of the elements making it decidable and compatible with DL reasoning. OWL Lite is a subset of OWL DL which provides easy but constrained usage of OWL DL. [27]

The current OWL specification, OWL 2, has been published in 2009 and it is slightly enhancing the expressivity of the original OWL. For example, property chaining and new property characteristics (such as asymmetricity, reflexivity, and disjointness) are available in OWL 2. Moreover, OWL 2 DL is divided into three sublanguages: EL, QL and RL. Each of the sublanguages emphasise different aspects in logical reasoning. [28]

### 3.2.4 SKOS

Simple Knowledge Organisation System (SKOS) is a vocabulary (less formal ontology) used for representing concepts with simple taxonomic properties. SKOS properties enable designing simple hierarchical structures to the ontology. [29] The example below defines hierarchical structure for concepts **Cell** and **MicroCell** which both belong to scheme **MobileNetwork**.

```
c:MobileNetwork a skos:ConceptScheme .
c:Cell a skos:Concept .
c:MicroCell a skos:Concept .
```

```

c:Cell skos:inScheme c:MobileNetwork .
c:MicroCell skos:inScheme c:MobileNetwork .
c:Cell skos:narrower c:MicroCell .
c:MicroCell skos:broader c:Cell .

```

Similar structure, as example above shows, can be achieved with OWL but SKOS might be more suitable in situations where a lightweight solution is desired and where complex reasoning tasks are not needed.

### 3.2.5 SPARQL

SPARQL is a query language used for RDF data and it is an effective tool for retrieving subgraphs from an ontology or combining data from multiple ontologies. [30] Following example shows a simple SPARQL query. First, ontology namespace is defined as a prefix and then query requests all instances having `Cell` type.

```

PREFIX c: <http://example.com/>
SELECT ?cell
WHERE {
?cell rdf:type c:Cell .
}

```

## 3.3 Markov Logic Network

The Markov Logic Network (MLN) is a model that combines the FOL from logic and the Markov network from probability theory. With the MLN, uncertain and contradictory information can be represented by adding numerical weights to FOL clauses to indicate their priority in a rule base. MLN provides statistical inference and it is suitable in contexts where conflicting rules exist. For example, in a mobile network might be a conflict in which SON functions have contradictory goals such as energy saving and quality of service. With statistical reasoning these situations might be possible to resolve and produce an optimal (or balanced) outcome. [8]

The MLN knowledge base is constructed as pairs of  $(F, w)$  where  $F$  is a FOL statement and  $w$  its numerical weight. [8] Equation 1 shows a simple example of representing an MLN statement. The statement indicates that if a cell has low channel quality (CQI), transmission power (Txp) should be increased. The weight for this statement is 0.9.

$$(\forall cell(Cqi(cell, low) \implies Txp(cell, increase)), 0.9) \quad (1)$$

The MLN rule base allows also certain rules to be inserted. These rules can be used as axioms for the uncertain rules. The example above could have the axiom that Txp cannot be increased and decreased at the same time, as the equation below shows.

$$!Txp(cell, decrease) \vee !Txp(cell, increase). \quad (2)$$

When the MLN reasoner gets evidence data from the environment (CQI values for cells in this case), it can do the reasoning and produce action probabilities for individual cells. With respect to evidence data, an MLN can also learn better weights for the rules in the rule base.

### 3.4 Knowledge Systems in Telecommunication Field

This subsection introduces research projects which utilize knowledge engineering methods presented earlier such as ontology development, OWL-DL reasoning, and MLN reasoning.

#### 3.4.1 RDF Dataset for Mobile Network

OpenMobileNetwork<sup>1</sup> is a research project where WiFi-, cell-, and UE-related data is collected into an ontology and published with the Linked Open Data (LOD) principles. Data is gathered from smart phones, WiFi access points, and from other open source projects such as OpenCellID<sup>2</sup>. The data is then converted into RDF triples and published for open use. [31]

OpenMobileNetwork aims to give another aspect to network management by having cell-related data open and integrated into the LOD cloud datasets [32]. Thus, several other datasets can be utilized among it, for example DBpedia<sup>3</sup> or LinkedGeoData<sup>4</sup> for location data. Moreover, a research paper [31] presents an event-related power management scenario where a machine can query and predict peaks in data traffic by examining events linked to locations. For example, if machine queries an event in a certain location, transmission power could be automatically increased for nearby cells in that place at the right time. [31]

---

<sup>1</sup><http://datahub.io/dataset/openmobilenetwork>

<sup>2</sup><http://opencellid.org/>

<sup>3</sup><http://dbpedia.org/about>

<sup>4</sup><http://linkedgeodata.org/>

### 3.4.2 OWL Reasoner for SON Conflict Analysis

Improvement for conflict analysis in SON coordination with OWL-DL reasoning is researched in [33]. This approach presents a semantic description for SON functions and network elements. The objective of the approach is to use a DL reasoner in detecting logical conflicts between active SON functions. [33]

Semantic modelling provides the detection of indirect logical conflicts. For example, if the actions of two SON functions are not conflicting by default but their combination causes an unwanted result to network performance, semantic reasoner might be able to discover this conflict by understanding the logical relationships in the context.

The knowledge base for conflict detection contains a static SON function ontology including basic function information about its type, input, and output. In run-time, dynamic data about function locations and impact areas are aggregated with the ontology and a rule base is then used to identify whether there is a conflict between functions in the network. [33]

### 3.4.3 MLN Reasoner in SON Verification

Markov Logic Network is researched and adapted in SONVer which is a demonstration system for verifying configuration changes made by SON functions. By monitoring KPIs and configuration history, SONVer detects and diagnoses anomalies in the network. MLN is utilized in diagnosis phase by investigating if an abnormal value in a cell is caused by a configuration change. [34]

## 4 Visualization and Interaction Methods

This section gives a literature review to visualization and interaction technologies which are relevant to this thesis. First, visualization techniques and guidelines are introduced. Second, there is discussion about the exploratory search paradigm which aims to support user's knowledge acquisition with a functional search interface. Finally, a practical implementation of exploratory search, faceted search, is presented.

### 4.1 Visualization Techniques

Information visualization is an effective way to present a large set of information. A proper visualization helps the user in making decisions and it prevents misinterpretations from the data or information set. [35] Shneiderman [36] has divided visualization techniques into to seven types: 1-dimensional, 2-dimensional, 3-dimensional, multi-dimensional, temporal, tree and network. However, these are just guidelines and in reality one can use a combination of several types such as a network graph where nodes are positioned by coordinates and coloured by their type.

In [36] a guideline for visual exploration is defined. The guideline advices to consider different phases in visual exploration:

- Overview
- Zoom-in and filter
- Details-on-demand

*Overview* is considered as a starting point for the user and it gives a general view of the collection. In the overview visualization, items can be grouped to achieve a compact representation of the collection. With *zooming* and *filtering* user can investigate particular areas for more detailed information. The *details* of items should be available (for example as a pop-up window) to provide a detailed description of an item. [36]

### 4.2 Exploratory Search

#### 4.2.1 Overview

Exploratory search is a combination of focused search (executing queries) and browsing. It can be seen as an attempt to first maximize the amount of relevant search



results (recall) and then browse through the result set to find interesting things. However, traditional search engines are focusing on providing only small amount of relevant documents as they are trying to minimize irrelevant documents (precision). Thus, exploratory search has different goals in search activities as what traditional search engines have. [37][38]

Exploratory search is useful when a user has an information need for one or several of the following tasks [37]:

- Define the goal of retrieval task
- Define ways to achieve the goal
- Familiarize with the domain

Exploratory search benefits a user who is unsure about the goal and wants to define her information need better. Also, the user might know what to look for but need to explore ways to achieve the goal. Finally, if the user knows the goal and methods to achieve it, she still might need to familiarize herself with the topic to be able to get right answers. [37]

Visualization and interaction are important supporting actions in the search process as they help users to understand the information and to discover relevant aspects and relations. Moreover, a system can interact with user for example by helping in making better queries (for instance by recommending search terms [39]), offering result filtering or by grouping results into meaningful categories. [37]

Exploratory search behaviour depends highly on the context where it is used. For example, in general web search (searching with a search engine) query recommendation and query correction are important aspects. However, search scenarios related to network management rather need effective result filtering to get a heterogeneous dataset reduced into relevant items.

#### 4.2.2 Search Activities

Generally, a search process can be divided into three classes [38]:

- Lookup
- Learn
- Investigate

The latter two, learn and investigate, are considered as exploratory search activities whereas lookup is considered as a traditional search activity. *Lookup* aims to give precise results to a well-specified query such as database queries or text search inside a document. [38]

The second activity, *learning search*, is an iterative search process where the user makes several queries and compares the results to learn different aspects of the domain. In learning search the user not only executes queries but uses browsing activities to learn related things in the collection and thus gets deeper understanding with respect to her information need. [38]

*Investigative search* is a long-term search activity which can last for several weeks. In this activity, the user aims to make deductions and conclusions from the retrieved information over time. For instance, in investigative search the user can fill her knowledge gaps with new relations or she can make forecasts from the gathered information. [38]

## 4.3 Faceted Search

### 4.3.1 Overview

The faceted search is a technology which supports the goals of exploratory search. [37] Basic idea in faceted search is in filtering search results with some criteria in order to investigate characteristics of the result items. Facets are chosen from the available metadata and are utilized by grouping attributes to disjoint groups. Each group can have either a hierarchical or a flat structure. For example, event instances could have a time-related hierarchical facet with values month, week, day, and hour. [40] Facet values can be defined to be either disjunctive (allowing multiple selections) or conjunctive (allowing one selection). [41] For example, the event facet could be designed as disjunctive to allow events to last several hours.

In figure 5 faceted search is demonstrated with Amazon's<sup>5</sup> product search. Search results is first retrieved with keyword *Faceted* and after that the collection is reduced by selecting *Books* category. Items can be further reduced by choosing any of the visible facets which are listed in the left column. Amazon's search interface hides facets with no results and shows the amount of items for available facets.

---

<sup>5</sup><http://www.amazon.com/>

1-60 of 529 results for Books - "Faceted" Sort by Relevance

Show results for

< Any Category

**Books**

- General (20)
- Reference (85)
- Computers & Technology (46)
- Computer Programming Language & Tool (5)
- Christian Books & Bibles (4)
- Software Development (3)
- Religion & Spirituality (22)
- See more

Refine by

**International Shipping**

Ship to Finland

**Eligible for Free Shipping**

Free Shipping by Amazon

**New Releases**

- Last 30 days (6)
- Last 90 days (21)
- Coming Soon

**Book Format**

- Hardcover (108)
- Paperback (322)
- Kindle Edition (11)
- Audio CD

**Book Language**

- English (408)
- Polish

**Author**

- Dave Thomas

Book Format: Hardcover | Paperback | Kindle Edition

<p><b>Faceted Search</b> (Synthesis Lectures on Information Concepts, Retrieval, and S) Jun 29, 2009            \$22.72 ✓Prime            Paperback            ★★★★★ - 3</p>	<p><b>Images of Set: Changing Impressions of a multi-faceted God</b> Jan 1, 2013            \$22.50 ✓Prime            Paperback            Only 4 left in stock - order soon.            ★★★★★ - 3</p>	<p><b>Growing Wings Self-Discovery Workbook-Vol.2: 18 Workshops to a Better Life: Exploring the Multi-Faceted Self...</b> Mar 9, 2015            \$20.69 ✓Prime            Paperback            ★★★★★ - 1</p>
<p><b>Faceted Chamber in the Moscow Kremlin</b> Jan 1, 1978            \$4.36            Hardcover</p>	<p><b>Faceted Gems - A Historical Article on the Methods and Equipment Used in Lapidary</b> Mar 17, 2015            \$26.45 ✓Prime</p>	<p><b>"Denver" Jack Geyer, The Boxer: His Multi-faceted Life</b> Jan 30, 2015            \$0.00 - \$14.00 kindleunlimited            Paperback, Kindle Edition</p>

Figure 5: The faceted search interface of Amazon.com with results for keyword *Faceted* and for selecting category *Books*

### 4.3.2 Determination of Facets

In order to avoid information overload, the number of facets should be reduced to only those which are the most relevant to user. Some useful aspects (with respect to the result set) to be considered in extracting the relevant facets are listed below [41]:

- High coverage
- High entropy
- Aggregation of similar facets

*High coverage* ensures that facet information is available for the majority of items on the result set. *High entropy* means that facet values should be uniformly distributed or at least so that one facet value does not contain the majority of the results. The *aggregation of similar facets* is useful when two or more facets are mainly sharing the same results and thus they can be combined as one facet.

A more user-centric method for facet design is card sorting which can be used to define relevant facets with respect to end-user behaviour. In card sorting, test users are given the result set as a pile of cards where every card defines an item with a title and description. Test users are asked to group the items as they like either by freely defining categories for them (open card sorting) or by setting the items into predefined categories (closed card sorting). [42][43]

### 4.3.3 Semantic Faceted Search

Semantic faceted search is a faceted search interface which utilizes Semantic Web technologies in the search interface construction. Thus, an ontology is used for a data model and SPARQL for query processing. [44] By means of Semantic Web technologies, the interface construction can be simplified with respect to the facet architecture and result set processing. [45] Moreover, OWL axioms [27] and semantic reasoning eases the maintenance of frequently updated dataset as reasoner can enhance the quality of an incomplete dataset. Moreover, with semantic classification vocabularies (such as OWL and SKOS) facets can be easily designed to have a desired structure. [45]

In semantic faceted search, SPARQL is a natural way to process queries. With given constraints it is rather straightforward to dynamically construct a query which processes desired result set. As an example scenario, the user may look for cells that have been configured recently and that have many neighbors. User selects from time facet cells with configuration time smaller than *one hour ago* and then reduces results to cells which have many neighbors. In SPARQL, this search task can be processed with the query shown below. Conditions (facet selections) can be added and removed inside WHERE clause every time a facet value is selected or deselected.

```
PREFIX c: <http://example.com/cell_ontology/>
SELECT ?cell
WHERE {
    ?cell c:latestConfTime ?conf .
    FILTER (?time > "2015-04-30T00:00")
    ?cell c:amountOfNeighbors ?amount .
    FILTER (?amount > 6)
}
```

### 4.3.4 Fuzzy Faceted Search

In uncertain contexts or in contexts where a non-numerical simplification for facet value representation is needed, a fuzzy or crisp set can be used instead. With fuzzy logic, elements are defined to have a grade of membership to every facet value. The numerical values of elements are transformed into fuzzy set grades with membership functions. [46] For example, facet values for cell neighbor amount could be defined with fuzzy sets *few*, *average* and *many*. If a cell has six neighbors, the amount of neighbors could be transformed with membership functions into fuzzy values:  $f_{FEW}(6) = 0$ ,  $f_{AVG}(6) = 0.7$ , and  $f_{MANY}(6) = 0.3$ . Now, the facet system can retrieve items which match best to the criteria (such as to selection *average*) and rank the results with respect to membership grades.

A special case (and simplification) of fuzzy sets are crisp sets in which class boundaries are strict. Elements in crisp sets are allowed to belong to only one class without a grade of membership. [44]

Naturally, a mapping from numerical values to fuzzy or crisp values has to be designed by an expert to prevent misinterpretation by the end-user. The example shown in section 4.3.3 can be converted into a SPARQL query with crisp values as shown in the example below.

```
PREFIX c: <http://example.com/cell_ontology/>
SELECT ?cell
WHERE {
    ?cell c:latestConfTime c:Recently .
    ?cell c:amountOfNeighbors c:Many .
}
```



## Part II

# Design and Preliminaries

## 5 System Design

This section describes the underlying system as well as the design plan for the demonstrator. The whole system (including the system architecture, its components and main functionality) is introduced briefly to give an overview of this project. After that, the underlying statistical reasoner, Markov Logic Network (MLN) reasoner, is explained. With respect to the reasoner functionality, some use cases are analysed for the demonstrator and after that, the architecture of the demonstrator is described. Finally, components of the demonstrator is explained by introducing a plan for the presentation layer (data representation in the graphical user interface) and for the data access layer (ontology design for the network- MLN- and facet-related data).

### 5.1 System Description

The main idea of the MLN reasoning system is to provide more effective network management. As the real SON functions can neither handle uncertain information nor make logical inferences, this project presents a statistical reasoner, MLN reasoner, which can assist SON functions in making optimal decisions.

Figure 6 shows the system architecture defining the communication between system components. The reasoning system is considered a combination of a custom SON function and the MLN reasoner. The reasoning system reads performance management (PM) data and configuration management (CM) data from the LTE simulator. The reasoning system uses this data to produce action proposals and to learn better weights for the rules in the rule base. The proposed actions are sent to the SON function which in turn transforms proposals with high probability into network configurations.

The demonstrator has an interface with the reasoning system where it receives the rule base, action proposals, and the same network data that the reasoner uses in its functionality. The demonstrator can manipulate the rule base and send updated rules back to the reasoner.

### 5.2 MLN Reasoner

As described in section 3.3, the Markov Logic Network (MLN) reasoner uses probabilistic logic to infer actions needed in the network. Rules in this system are defined to have a specific structure which is represented in equation 3.



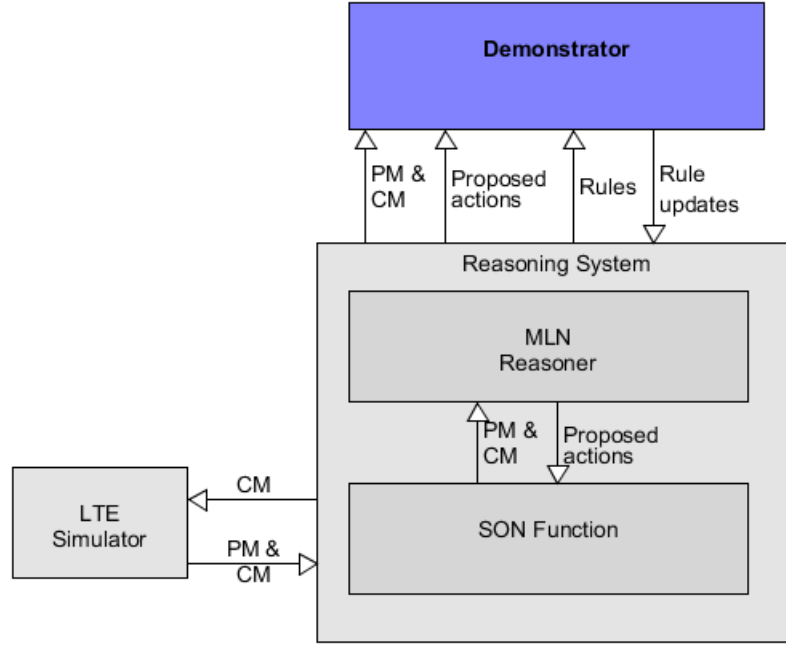


Figure 6: System architecture

$$w \text{ Context} \implies (\text{Objective} \iff \text{Action}) \quad (3)$$

Every rule clause consists of three classes: context, objective and action. *Context* defines the current network state, including network topology (neighbor relations between cells) and crisp values for key performance indicators (KPIs). *Action* is a configuration change and *objective* desired change for a KPI value. All three parts can include several predicates which need to be satisfied.

Equation 4 describes an example rule. The rule states that when cell  $c$  has a *low*  $Cqi$ , its  $Txp$  power and remote electrical tilt ( $Ret$ ) angle should be increased in order to achieve increment in the  $Cqi$ . Priority (weight) for this rule is set to 0.3.

$$0.3 \ I(t, c, Cqi, Low) \implies [O(t, c, Cqi, Inc) \iff (A(t, c, Txp, Inc) \wedge A(t, c, Ret, Inc))] \quad (4)$$

The MLN reasoner is executed regularly and weights are updated after every reasoning round. As a reasoning output, it calculates action proposals for every individual cell. The example below shows the proposed actions and their probability for cell 5.

A(5,Ret,Inc) 0.391011

A(5,Ret,Dec) 0.339016  
 A(5,Txp,Inc) 0.887961  
 A(5,Txp,Dec) 0.0610439

### 5.3 Use Case Analysis

With respect to the logic of the MLN reasoning system, some use cases need to be examined for the demonstrator. The demonstrator is targeted especially for research and development purposes. However, this system can also be considered as a potential product in the future network management and thus, the design should also help a human operator to understand the behaviour of the MLN reasoning system.

Figure 7 describes a use case diagram for the demonstrator. Use cases can be divided into two groups: knowledge acquisition (Acquire knowledge) and knowledge transfer (Update knowledge model). In knowledge acquisition, all the relevant information is related to the behaviour of the reasoner. This includes understanding the content and structure of the rules to see which rules are emphasized in the network after weight learning. One important aspect is to investigate relations between current cell states (input) and action proposals (output) which reveal the causes and consequences for the reasoning outcome. Moreover, the user should be allowed to evaluate how successful proposed actions have been in the network (output and impact). Thus, the user needs to explore anomalous events in KPIs and their relation to configurations.

In addition to knowledge acquisition, the user should have tools to interact with the MLN reasoner in order to understand the behaviour better. Therefore, there are use cases for updating the MLN model. First, the user can change rule weight to denote the significance or insignificance of a rule. The demonstrator should also allow creation, modification, and removal of the rules.

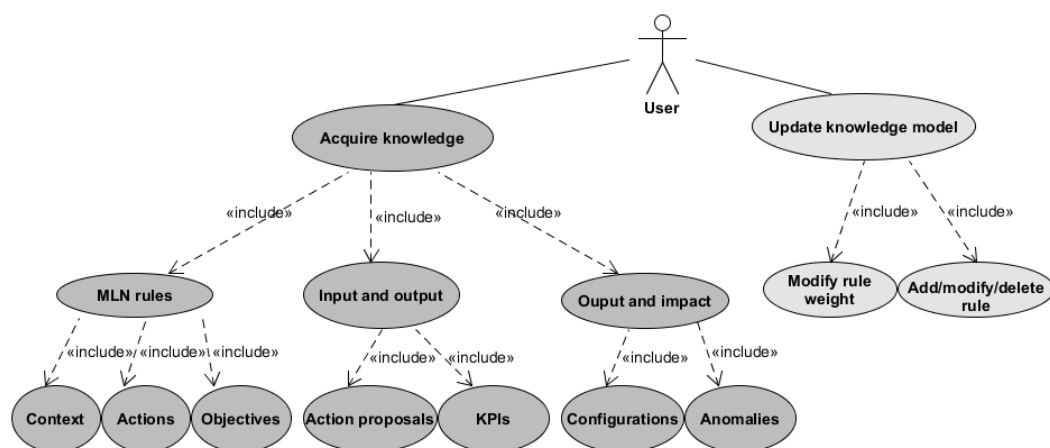


Figure 7: Use case diagram for the demonstrator

## 5.4 Demonstrator Architecture

This subsection gives a brief overview of the architectural design aspects of the demonstrator.

### 5.4.1 Overview

The demonstrator allows the user to explore and control the functionality of the MLN reasoning system and it is designed with respect to use cases discussed earlier. To provide knowledge acquisition, a graphical user interface (GUI) is designed to assist user with visualisation and interaction methods. The GUI is implemented as a HTML 5 application.

The explorative search paradigm is considered to answer the information need of the user. A practical implementation of the explorative search is a faceted search interface which is adapted to the demonstrator. Faceted browsing enables the user to explore available information and to gain more knowledge from the context. With respect to the use case analysis, the demonstrator is designed to have separate facet views for every knowledge acquisition use case. Different facet views provides more freedom and different perspectives for the user in information exploration.

For data management, the demonstrator uses an ontology for the knowledge base and SPARQL for querying it. As the objective is to explore relations from MLN- and network-related data, Semantic Web technologies provide a natural way for link creation between concepts and terms [26], too. The semantic approach simplifies the development and maintenance of multiple facet views [44]. In addition, a Java application is implemented to retrieve and process the data into the ontology.

### 5.4.2 Architecture

Figure 8 describes the components of the demonstrator. Three facet views are defined as cell states, cell events, and MLN rules. By default, every facet view is visualized as a table view. However, optional visualizations are available to provide deeper understanding about the data. For example, cell states could have a network graph and cell events a timeline. In addition to facet views, GUI has a view for MLN rule modification which is implemented as a form.

Views interact with an ontology which stores data as RDF. The ontology has a SPARQL engine as an interface for GUI and REST interface for the reasoning system. The REST component retrieves the relevant reasoner- and network-related data. Data is then processed into the ontology which GUI components can query to get

the facet-, cell- and MLN-related data.

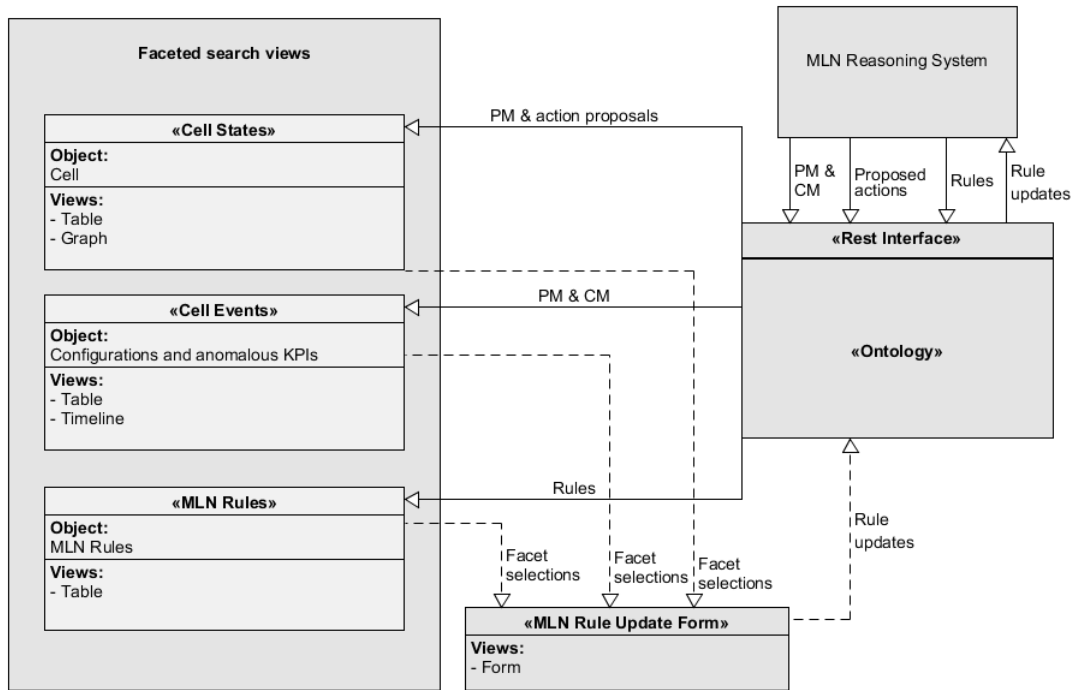


Figure 8: UI Design Architecture

## 5.5 Demonstrator Views

The following subsections describe the views of the GUI. With respect to the knowledge acquisition use cases (analysed in 5.3), a view for every case is designed: cell status view describes the input and output of the MLN reasoner, cell event view describes the output and its impact to the network, and rule view describes the structure of the uncertain rules in the MLN rule base. Every view description is divided into data representation, visualization, and facets. However, this section contains only design aspects concerning the demonstrator views. The implementation of the views (such as the actual visualizations) is presented later in section 7.

### 5.5.1 Cell Status View

The cell status view is designed to illustrate the relation between the input (performance data) and the output (action proposals) of the MLN reasoner. Thus, this view should provide a visualization and a faceted search functionality which support interactive searching with respect to the cell-related measurements and action proposals.

## Data Representation

A cell state includes attributes such as amount of neighbors, KPIs (monitored by the MLN reasoning system), and MLN action proposals.

Cell-related measurements are simplified by grouping values into crisp sets. For example, KPI values are grouped into *poor*, *medium* and *good*, and the thresholds for the crisp sets correspond to values used in the MLN reasoner. A parameter-specific action proposal is represented by a combination of action direction (increase or decrease) and action probability. The amount of neighbors is represented as a numerical value.

## Visualization

The view for cell states is designed to have two visualizations: table and network graph. The example below shows how the table view is constructed. A table row represents information about cell attributes with one column for the amount of neighbors, one for every KPI value and one for every action proposal.

```
Cell(NeighborAmount,KPI_0,KPI_1,...,Action_0,Action_1,...)
```

The network graph is provided as an alternative visualization: the network topology can be observed from a graph visualization. The graph represents nodes as cells and neighbor relations as links. Node color indicates value for a KPI and node size indicates the probability of an action proposal.

## Facets

Cell facets are generated from every KPI represented in the view. Crisp KPI values are used as facet values (such as low, medium, and high). KPIs are intuitive for facets as they are relevant part of the network status and also part of the MLN reasoner input.

In addition to KPIs, neighbor amount is used as a facet. Neighbor amount is considered to give user a perspective to network topology and it might be useful to examine cell characteristics also from this point of view.

### 5.5.2 Cell Event View

The cell event view aims to demonstrate what impact does the action proposals, which is turned into network configurations, have to the network performance. This can be done by investigating the relation between cell events which are defined here either as a parameter configuration or as an anomaly in the network performance.

#### Data Representation

A cell event defined with attributes time, type (identifier for a parameter or indicator), direction of change, amplitude of change, previous value, and current value. Time attribute corresponds to the timestamp of the simulator meaning the latest monitoring time. In simulation time, the interval of monitoring is 15 minutes. Therefore, the event time can be interpreted as: "happened inside 15 minutes before the timestamp".

In event data, crisp values are used for the direction of change and amplitude of change. The direction is defined either as a *decrease* or *increase* and amplitudes are defined as *low*, *medium*, or *high*. Crisp values for amplitudes are set manually to every event object as they behave differently. The idea is that an expert could later configure the thresholds to correspond crisp sets more accurately.

#### Visualization

As in the cell status view, the event view is designed to have two visualizations: table and timeline. A table row indicates an event with attributes shown in the example below.

```
Event(Time,Parameter,Indicator,Impact,Amplitude,PrevVal,CurrentVal)
```

Event objects are divided into separate columns with respect to their type (parameter or indicator). The direction of change can be visualized with an icon and amplitude either with an icon or with a numerical value (in percentages).

Timeline visualization provides investigation of interesting events that have temporal relation. For example, the user might discover that a particular configuration is a root cause for KPI anomalies. On the other hand, visualization might also reveal which KPI anomaly has triggered a configuration.

## Facets

Event facets are generated from time and impact. Both of the facets are constructed as hierarchical facets. Time has three stages: day, hour and minute. Impact is structured as direction (*increase* or *decrease*) and amplitude (*low*, *medium* or *high*).

### 5.5.3 Rule View

The rule view demonstrates the content of MLN reasoner. Therefore, the uncertain rules, which are the body of the MLN rule base, are represented in this view.

## Data Representation

The third view represents the uncertain rules of the MLN reasoner with respect to the structure explained in section 5.2. Thus, a rule is divided into context with KPIs and their values (*low/high*), actions with parameters and their change (*dec* or *inc*) and objectives with KPIs and their desired change (*dec* or *inc*). Also, rule weight is represented to user.

## Visualization

Rules are designed to be visualized only in a table view as no alternative visualization was found to enhance user's information exploration. The table view is represented with a following structure:

```
Rule(Context, Objectives, Actions, Weight)
```

In the rule table, every column indicates one rule class. As one rule class might contain several predicates, these are listed inside a table cell.

## Facets

Facets are generated from rule classes and from their parameter or indicator combination. For instance, context would have distinct facets for KPIs with crisp values *high* and *low*. Objective facets are generated for KPIs with crisp impact values *decrease* and *increase*. Similarly, for actions there are facets for every parameter with crisp impact values.

### 5.5.4 MLN Rule Updates

After user has examined faceted search views described above, she might come up with new rules or with new ideas to modify existing rules. Rule update is implemented with a form structure so that user does not have to learn specific syntax to express first-order logic (FOL). However, expert users are allowed to create a rule with MLN syntax. Thus, a validation for processing an MLN rule is needed.

The system assists user in filling the form by utilizing the facet selections she has made. For example, if user has selected *low Cqi* value as facet value in cell status view, the system prefills this selection to the context part of the rule ( $I(t, c, Cqi, low)$ ).

## 5.6 Demonstrator Ontology

This subsection explains the ontology used as a data model for the GUI. The ontology is stored as a whole but logically it can be split into three sub-ontologies: network, MLN reasoner, and facet ontologies. The ontology is also introduced in these parts. Each sub-ontology has a description and examples of the TBox (rules and structure) and ABox (instances) of the content.

### 5.6.1 Network Ontology

Network ontology describes the network context with respect to network data retrieved from the background system. The ontology aims to model the network data so that it is both logical (the structure is understandable to a domain expert) and functional for the application (the structure and properties support faceted browsing). The network ontology is built as an OWL 2 ontology.

#### TBox

Figure 9 gives an overview of how network-related classes are represented in the ontology. The most fundamental class in the ontology is `Cell` which have a symmetric neighbor property (`cellHasNeighbor` property between `Cell` instances) and properties to different types of `Parameter` and `Indicator` classes. Subclasses of `Parameter` and `Indicator` are bound to `Cell` with subproperties of `cellHasParameter` and `cellHasIndicator`.

`Indicator` and `Parameter` can have relation to `Event` which describes a change in the object. `Event` has a relation to `Impact` which indicates the direction (`Increase`



or Decrease) and magnitude of the change.

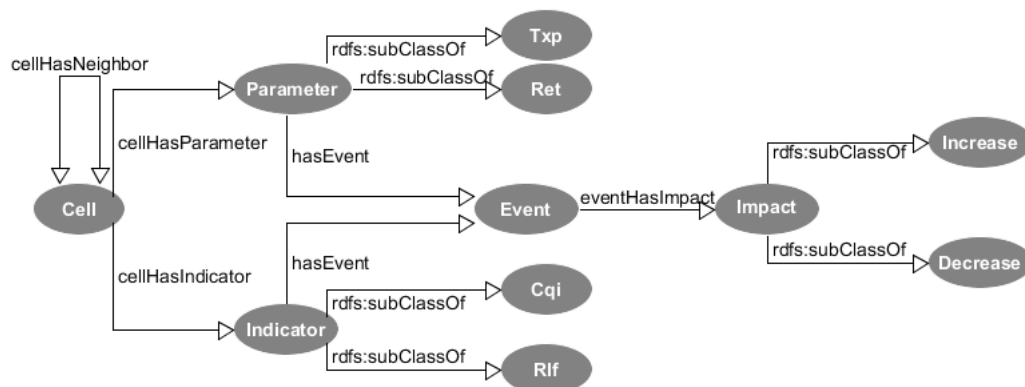


Figure 9: TBox example of network ontology

Figure 9 describes object properties only between instances. Classes have data properties such as `hasValue` to define numerical values for certain objects. The mostly used object property axioms are domain, range and functional relations. Functional relations are defined in case the domain is restricted to have only one such relation such as `Event` class has functional properties `eventBelongsTo` (inverse of `hasEvent`) and `eventHasImpact`. Defining axioms for object properties might not directly improve the current use (faceted search) of the demonstrator but they enable OWL 2 DL reasoning and more versatile use of the ontology in later projects.

## ABox

Figure 10 demonstrates how instances are represented in the ABox part of the network ontology. The example focuses on representing cell element `Cell12` and relevant object properties to it. The red color is used in the graph to separate instances from classes (which are colored as grey). All cell instances in the ontology have relation to the same indicators (`Cqi` and `Rlf`) and to parameters `Txp` and `Ret` (remote electrical tilt). Instances `Cqi12` and `Txp12` have relations to `Event` instances which both have decreasing impacts on their objects.

### 5.6.2 MLN Ontology

The MLN ontology is an OWL 2 model representing the content of the MLN reasoner. Moreover, this ontology is built using properties which bind MLN-related classes to network-related classes.

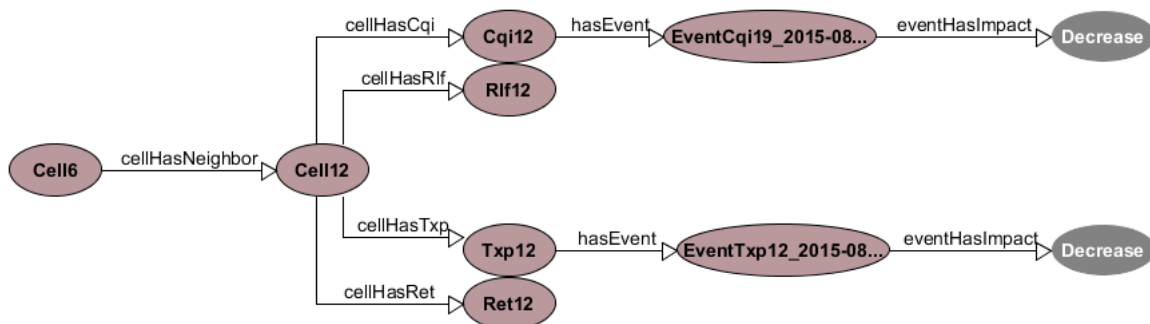


Figure 10: ABox example of the network ontology

## TBox

Figure 11 describes the TBox of the MLN ontology. As the figure shows, TBox has two main classes: **MLNRule** (rule base) and **ActionProposal** (reasoner outcome). **MLNRule** can further be divided into rule classes with types of **MLNContext**, **MLNAction**, and **MLNObjective** (corresponding to the structure and formula 3 described in 5.2).

The figure also describes the most relevant object properties between classes. As it can be seen, MLN classes are bound to network classes such as **Parameter**, **Impact**, and **Indicator**. For example, **MLNAction** class has relation to **Parameter** and **Impact** to define the content of an action predicate. Similarly, **MLNObjective** has relation to **Indicator** and **Impact**. **MLNContext** has relation to **Indicator** but the magnitude of the KPI value (**Indicator**) is described with a fuzzy value (**High** or **Low**).

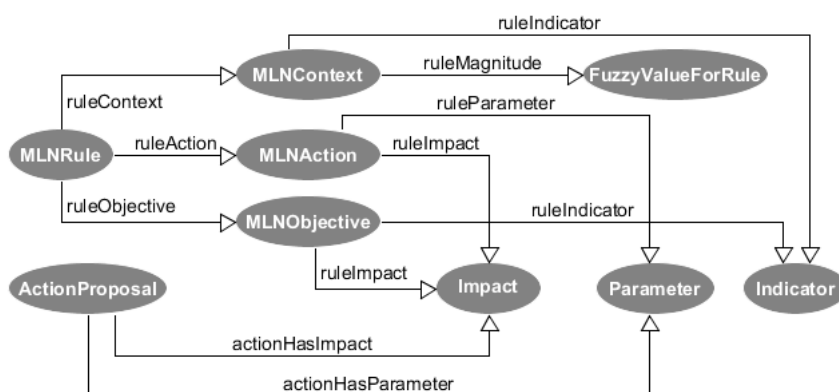


Figure 11: TBox example of the MLN ontology

## ABox

The MLN rule presented in equation 4 can be described in the MLN ontology as figure 12 shows. The MLNRule instance *Rule192* has four parts: one Context (*Rule192IndicatorCqiLow*), one Objective (*Rule192ObjectiveCqiInc*) and two Action instances (*Rule192ActionTxpInc* and *Rule192ActionRetInc*). As it can be seen, property values for these instances are network-related classes such as *Ret*, *Txp*, *Increase*, and *Cqi*. Rule weight is defined as a data property value for MLNRule instance.

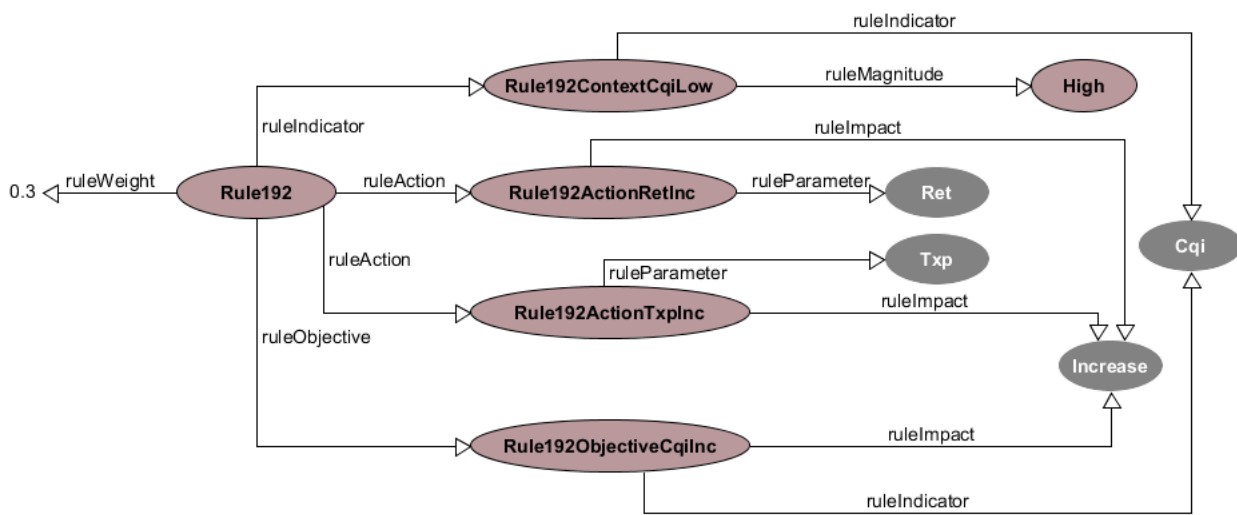


Figure 12: ABox example of a rule in the MLN ontology

Figure 13 shows an ABox example of an action proposal. Like the rule classes, action proposals are also described with property relations to the subclasses of *Indicator* and *Parameter*. Moreover, as an action is proposed for an individual cell, the cell instance and action instance are bound to each other with *cellHasActionProposal* property.

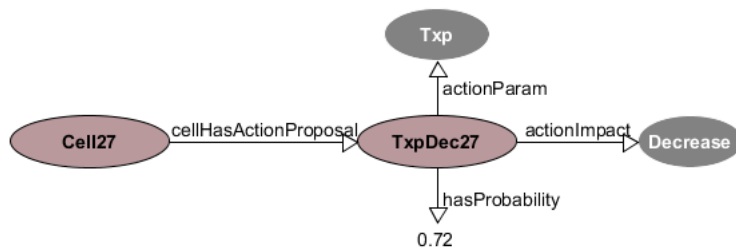


Figure 13: ABox example of an action proposal

### 5.6.3 Facet Ontologies

The facet system is built using SKOS vocabulary [29], because it provides a simple vocabulary to describe hierarchical structures. This way facet-related semantics is separated from other parts of the ontology and it is easier to design SPARQL queries related to facets.

#### TBox

Figure 14 demonstrates how SKOS elements `skos:ConceptScheme` and `skos:Concept` define facets and facet values and how SKOS property `skos:inScheme` describes their relation. Also, SKOS properties `skos:broader` and `skos:narrower` describe hierarchical relations inside a facet (between facet values).

Some concepts use data properties `fuzzyMin` and `fuzzyMax` to define numerical ranges for crisp terms. For example, the indicator- and impact-related facets use numerical ranges which are predefined into the ontology.

#### ABox

In figure 14 is presented one facet from every view: current CQI value from cell states, event time from cell events, and the objective of a CQI value from rules.

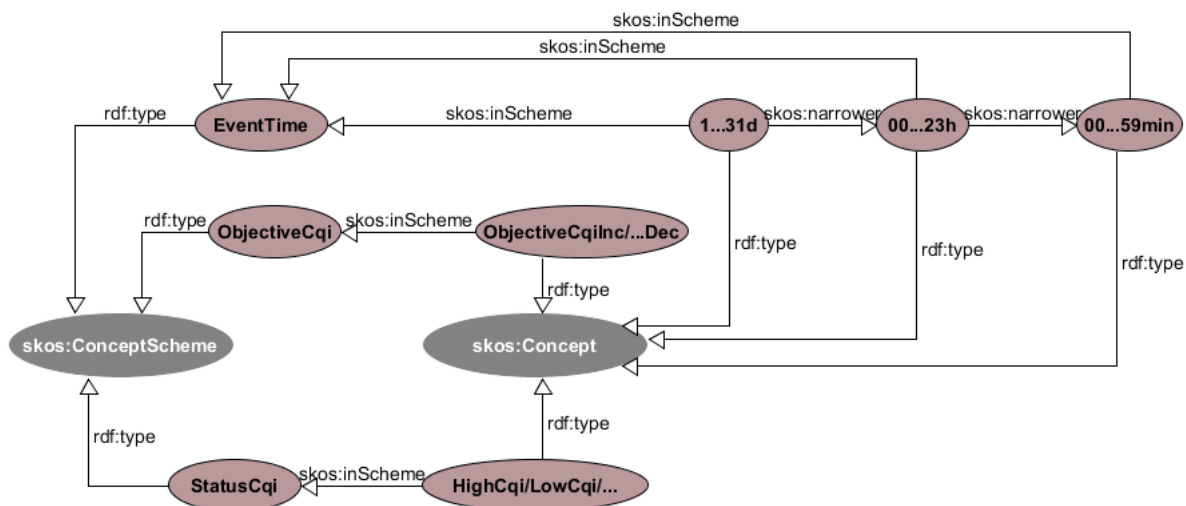


Figure 14: The facet ontology with an example facet from every facet view

Every `skos:Concept` has an object property which binds the facet instance to an

ontology instance. For example, cell instance `c:Cell127` can be bound to an RLF facet value with the following statement:

```
c:Cell127 c:facetRlf c:HighRlf
```

Similarly, hierarchical facet values are bound to ontology instances with separate properties. For example, an event instance has three facet values with respect to event time:

```
c:Event123 c:facetDay t25  
c:Event123 c:facetHour t25_12  
c:Event123 c:facetMinute t25_12_05
```

## 6 Tools and Libraries for the Demonstrator

The earlier section defined the requirements and abstract design of the demonstrator. This section discusses about tools and libraries that can be used in the implementation. The examined components is divided into ontology- and GUI-related sections.

### 6.1 Ontology Management

#### 6.1.1 SPARQL Engine

To handle ontology data, the demonstrator needs a triple store and SPARQL engine which supports both SPARQL queries (for facets) and SPARQL updates (for rule updates). One solution for a SPARQL engine is Fuseki which is part of Apache Jena framework<sup>6</sup>. It provides SPARQL endpoints for querying and updating RDF data over HTTP protocol. [47]

Another SPARQL server is NanoSparqlServer<sup>7</sup> which is built on top of a Blazegraph graph database<sup>8</sup>. As Fuseki, also NanoSparqlServer offers SPARQL endpoints that can be queried over HTTP and that support both SPARQL queries and SPARQL updates. [48]

#### 6.1.2 DL Reasoners for OWL 2 Ontology

Because the OWL 2 ontology instances are frequently updated, there is a need for an OWL 2 description logic (DL) reasoner to check whether the ontology is consistent with respect to the axioms (TBox). Although the facet system might work correctly even though the ontology would be inconsistent, it is desirable to keep the ontology consistent to enable use of OWL 2 DL reasoning tasks in later projects. For example, a DL reasoner can be used to infer new assertions (such as missing object properties) to the ontology.

A requirement for the DL reasoner is that it should be open source and usable in Java applications as the backend is implemented with Java. Two well-known DL reasoners which fulfils these conditions are HermIT<sup>9</sup> and Pellet<sup>10</sup>.

---

<sup>6</sup><https://jena.apache.org/index.html>

<sup>7</sup>[http://wiki.bigdata.com/wiki/index.php/NanoSparqlServer#REST\\_API](http://wiki.bigdata.com/wiki/index.php/NanoSparqlServer#REST_API)

<sup>8</sup><http://www.blazegraph.com/>

<sup>9</sup><http://hermit-reasoner.com/>

<sup>10</sup><https://github.com/Complexible/pellet>

Performance of the reasoners has been researched in [49] where their suitability are tested for several OWL ontologies and for different inference tasks. Although benchmarking is done for OWL and not for OWL 2, it gives an overview how well the reasoners perform in different contexts. Comparing Pellet with HermIT, HermIT seems to be a very fast reasoner generally but slower in loading the ontology and in checking the consistency. However, Pellet is slightly faster in ontology loading and consistency checks but it has variance in general reasoning performance with different ontologies. Especially ABoxes with large sizes can be challenging for Pellet. [49]

## 6.2 Graphical User Interface

The GUI is implemented as an HTML5 application and therefore Javascript is used to program GUI functionality. This section examines some Javascript tools for different purposes such as table sorting, network graph functionality, timeline functionality and facet browsing.

### 6.2.1 Sortable Table

In a table view, it is intuitive to offer column-specific sorting functionality. Tablesorter<sup>11</sup> is a jQuery plugin which offers sorting functionality. Plugin detects automatically many data types (such as strings, numbers, and dates) and sorts them correctly without extra definitions. However, the plugin allows custom sorting functions which are needed in this project because some data might be represented differently as they should be sorted. For instance, fuzzy values should not be sorted in lexicographical order but with some other criteria.

### 6.2.2 Network Graph

In addition to table view, the cell network is visualized as an interactive network graph. Network graph should allow interactive functionality such as facet selections (node filtering). It should also provide some other interaction defined in Schneiderman's guidelines [36] (section 4.1), for example zooming and details-on-demand (clicking a node). Moreover, graph layout should be customizable as different colors and sizes are used to describe indicators and MLN action proposals.

Force Layout<sup>12</sup> is a D3.js package which generates a force-directed graph structure

---

<sup>11</sup><http://tablesorter.com/docs/>

<sup>12</sup><https://github.com/mbostock/d3/wiki/Force-Layout>

from given JSON file. The graph structure is then mapped to SVG elements (nodes as circles and links as lines) for final visualization. [50] D3.js does not offer interaction functions but with other plugins and custom work some interaction can be implemented into the graph.

Vis.js has a graph component which has basic functionality for graph drawing and interaction. [51] The library is easy to use and it has several tutorial examples which can be utilized in other projects. For a large network (many thousand edges) the graph becomes slow but with a smaller graph it works well.

Sigma.js<sup>13</sup> is a Javascript library which is focused on robust graph drawing. It offers basic interaction tools such as zoom-in, details and node filtering. [52] For a more complex interaction it needs additional plugins or custom implementation. By default, Sigma.js does not offer any layout (such as force-directed layout) or clustering features but some plugins are implemented also for these purposes. However, due to its efficient graph rendering, Sigma.js is stated to be suitable for projects which require basic interaction and dynamic rendering. [53]

Cytoscape.js<sup>14</sup> is another graph visualization tool for Javascript. Its advantage is compatibility with jQuery and diverse assortment of interaction tools such as box selection, panning and pinch-to-zoom. [53][54] It also provides an API with graph theory functions. Therefore, Cytoscape does not only have network graph rendering but it also has graph analysis and algorithms such as breadth-first search and PageRank [54] implemented as Javascript functions.

### 6.2.3 Timeline

The main purpose of the timeline is to visualize KPI anomalies and configurations and temporal relations between them. As the cell states are monitored frequently (every 15 minutes in simulation time), time should be represented at least in minute detail. Moreover, timeline layout should be horizontal and it must provide representation of concurrent events as the events from one simulation round have the same timestamp.

Vis.js has a timeline component which provides interactivity, concurrent events, and scalable time representation (from years to fractions of a second). Moreover, timeline area can be divided into subgroups and therefore different anomaly types (as decrease and increase) and configurations can be separated in the timeline. [51] The timeline component of Vis.js has the same advantage as the network graph component: the ease of use and amount of useful examples which can be utilized.

Although D3.js does not have any interaction tools by default, there are some timeline

---

<sup>13</sup><https://github.com/jacomyal/sigma.js>

<sup>14</sup><https://github.com/cytoscape/cytoscape.js>



examples<sup>15</sup> which can be adapted to this purpose. However, these timeline examples seem to need custom work for adapting to this context. For example, none of the timelines have a view for events with minute detail.

One popular timeline<sup>16</sup> library is TimelineJS (used by several newspapers). It has dynamic scaling for events and has much interactivity for event details. TimelineJS emphasizes event descriptions whereas the timeline itself is shown rather small below the description. Therefore, this component is suitable especially for contexts where user wants to focus on details of individual events more than investigate the timeline itself.

#### 6.2.4 Faceted Browsing

Facets need to be implemented with Javascript functions which query and process the SPARQL statements into HTML select boxes. Some Javascript/CSS tool could be utilized to style select boxes and some tool could be used to add functionality to the select boxes, for example real-time text search.

Chosen<sup>17</sup> is a jQuery plugin which provides select box styling. It adds text search functionality which filters option values in real time with respect to text input. It also provides functionality for handling multiple selections: every selected value is presented to user and selection can be removed by clicking it.

---

<sup>15</sup><https://github.com/mbostock/d3/wiki/Gallery>

<sup>16</sup><https://github.com/NUKnightLab/TimelineJS>

<sup>17</sup><https://github.com/harvesthq/chosen>



Part III

# Implementation

## 7 Graphical User Interface

The graphical user interface (GUI) is implemented with respect to the design of the demonstrator views described in section 5.5. This section presents first the structure of the GUI to give an overview of the application. After that, the implementation of three facet views (cell states, cell events, and MLN rules) is introduced. Every facet view section includes use case tests as practical examples of their functionality.

### 7.1 GUI Overview

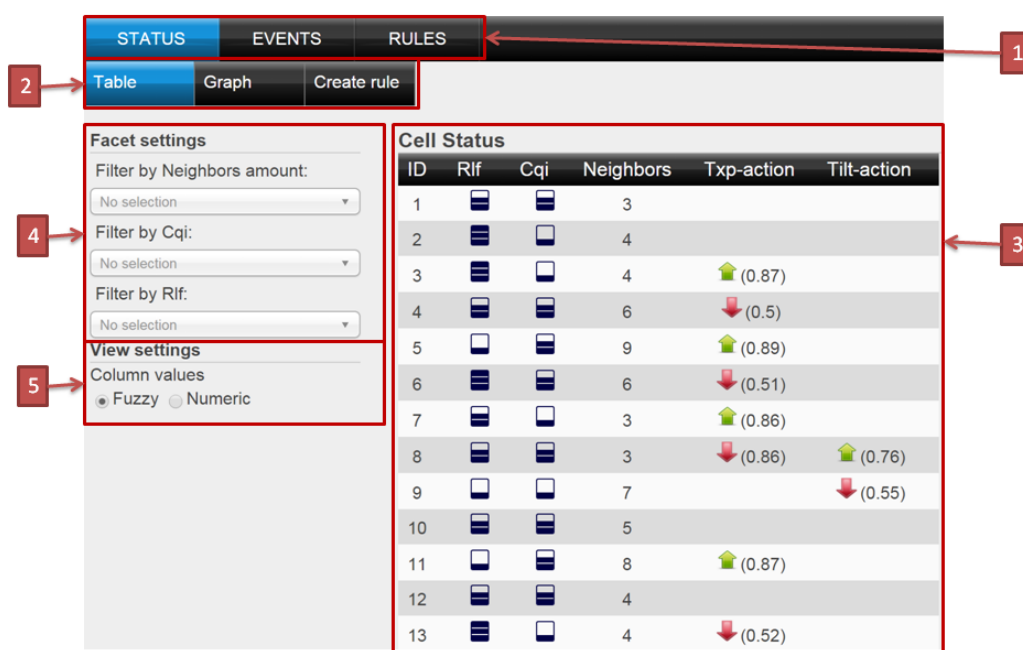


Figure 15: GUI overview (numbered parts are explained below)

Figure 15 depicts the main parts of the GUI:

1. First level navigation
2. Second level navigation
3. Content
4. Facet settings

## 5. View settings

With *first level navigation* the user can navigate between different faceted search views. Each view has a *second level navigation* which is implemented inside every view as a single page application (inside the same HTML page with dynamic content updates). *Content* visualizes the view-specific information (in the figure can be seen information about cell states) with a visualization selected from the second level navigation.

In *facet settings* user can select facet values to give filtering conditions for the result set. After every facet value selection, the content part is updated dynamically. Below facet settings there are *view settings* that are used if the content visualization provides some alternative options.

For an end-user this GUI structure provides an intuitive use of the application as the GUI has a layout similar to a traditional web page: navigation can be found from the header, additional information (such as facets and view settings) from the sidebar, and the content is located in the middle of the page. This also helps the end-user to understand the content of the application. For example, a new user can observe from the navigation bar what are the main parts of the GUI (status, events, and rules).

## 7.2 Cell States

### 7.2.1 Description

The cell status view is the default view in the GUI and it provides information about the input (cell states) and output (action proposals) of the MLN reasoning system. As defined in 5.5.1, this view has two visualizations: table and network graph. Also, the view has facets for neighbor amount and for every indicator monitored by the MLN reasoning system (currently there is only RLF and CQI).

The table view for cell states can be seen in figure 15. Action proposals are defined with numeric values for probabilities and arrow icons for action directions. Indicator value is presented with an icon corresponding to fuzzy values *undefined*, *low*, *medium*, and *high*. Cell ID and the amount of neighbors are defined with numeric values.

Figure 16 shows a graph visualization for cell states. The graph visualizes cells as nodes and neighbor relations as edges. Nodes are located with respect to their eNB coordinates to illustrate their actual positions. The node color indicates an intra cell group (belonging to the same eNB) or an action proposal for a parameter. Action proposals are color-coded as red (decrease), green (increase) or grey (no action). The

node size indicates magnitude of an indicator value or the amount of neighbors. Both the node color and size can be changed to another from the view settings.

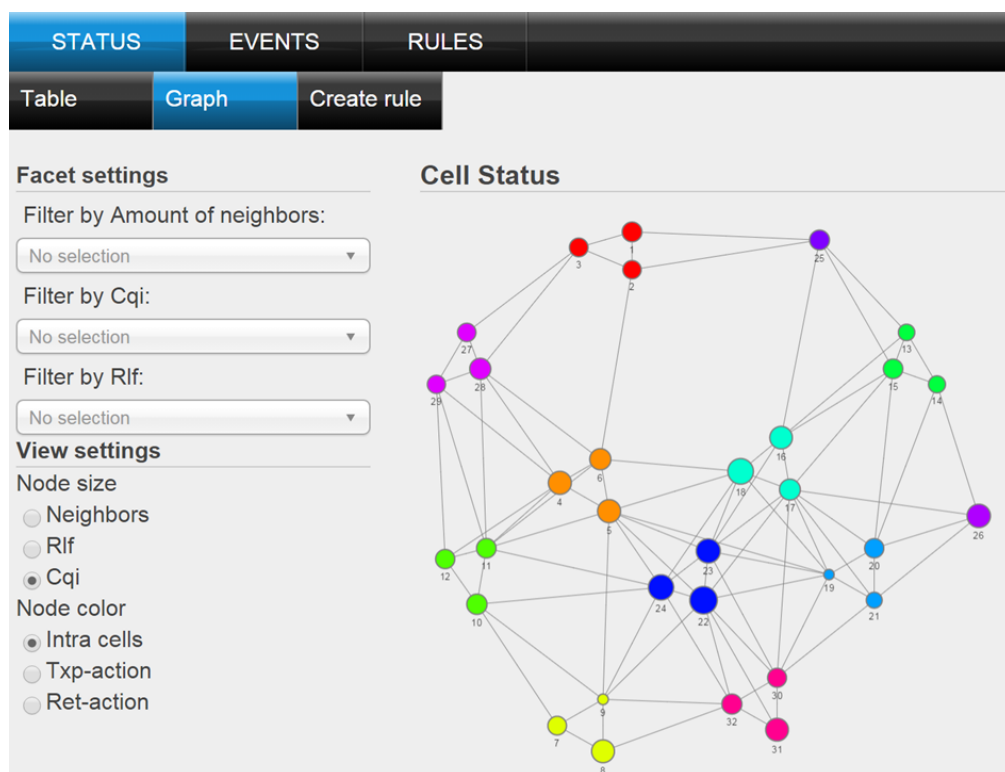


Figure 16: Network graph for cell states

Rule creation is handled with a form shown in figure 17. With the form, the user can create rules with respect to the defined rule structure (5.2). Form selections are transformed into MLN syntax and presented in a text field (as the figure shows). An expert user can modify the text field in order to create more expressive rules.

### 7.2.2 Use Case: Selecting Facets

A use case for the cell status view consists of following phases:

- Select values from facets
- Verify the result set in the table and graph
- Verify prefilled predicates in the rule creation form

Selected facet values are chosen as: *low Cqi*, *high Rlf* and *average* amount of *neighbors*. These selections reduce the result set from 32 cells to five cells. Visualizations of the result set as table and graph can be seen in figure 18.

**Facet settings**

Filter by Amount of neighbors:  
Average

Filter by Cqi:  
LowCqi

Filter by Rlf:  
HighRlf

**View settings**

**Create MLN rule**

0  $((t,c,Rlf,High) \wedge ! (t,c,Cqi,Low)) \Rightarrow [() \Leftarrow ()]$

Create rule Reset

**Select context:**

Rlf Cqi  
HighValue x LowValue x

**Select actions:**

Ret Txp  
No selection No selection

**Select objectives:**

Rlf Cqi  
No selection No selection

**Select weight:**

Weight  
No selection

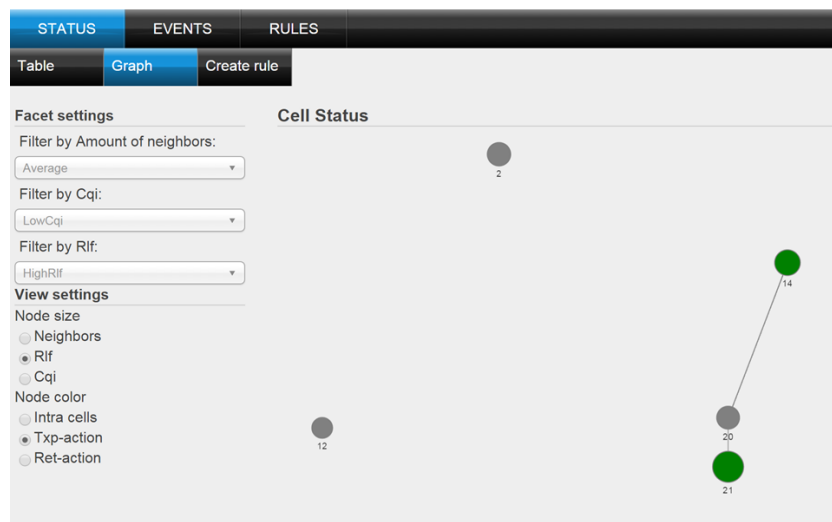
Figure 17: Rule creation form with prefilled facet selections

Facet selections *low Cqi* and *high Rlf* are mapped to values in the rule creation form, as it can be seen in figure 17.

Rule creation functionality is the same in the cell status and rule view. The rule creation form is available A use case for rule creation is described later in section 7.4.3.

STATUS		EVENTS		RULES			
Table		Graph		Create rule			
<b>Facet settings</b>		<b>Cell View - Table</b>					
Filter by Amount of neighbors:		ID	Rlf	Cqi	Neighbors	Txp-action	Ret-action
Average		2	■	■	4		
Filter by Cqi:		12	■	■	4		
LowCqi		14	■	■	4	▲ (0.82)	▲ (0.75)
Filter by Rlf:		20	■	■	6		▼ (0.66)
HighRlf		21	■	■	5	▲ (0.9)	
<b>View settings</b>							
Column values							
<input checked="" type="radio"/> Fuzzy <input type="radio"/> Numeric							

(a) Table after facet selections



(b) Graph after facet selections

Figure 18: Result set after facet selections visualized as a table and graph



## 7.3 Cell Events

### 7.3.1 Description

The cell event view describes configurations and anomalies monitored from the network. This view attempts to enable user to investigate relation between the output of the reasoner (configurations) and its impact to the network (anomalies in the network performance). Figure 19 shows network events listed in a table. First column indicates the time and date when the event is monitored. Event object type can be read either from *Parameter* or *Indicator* column which contain the object type type (Txp or RET for a parameter and CQI or RLF for an indicator) and cell ID to which the object belongs to.

The direction of an event is visualized with a colored arrow (up or down) which indicates increasing or decreasing (down) impact. In last three columns there is numeric data about relative impact magnitude (measured in percentages), previous object value, and current object value.

Time	Parameter	Indicator	Impact	Delta	From	To
06:30:00 07.07.2015		Rlf19	↑	300 %	1	4
06:30:00 07.07.2015		Rlf4	↑	136 %	22	52
07:30:00 07.07.2015		Cqi9	↓	-49 %	0.54	0.27
07:30:00 07.07.2015		Rlf32	↑	122 %	9	20
07:30:00 07.07.2015	Txp9		↓	-33 %	46	31
07:45:00 07.07.2015		Rlf19	↑	500 %	1	6

Figure 19: Table for events

The timeline visualizes events as nodes with the same colored arrow icons described in the table view. The timeline is organized into parts to separate object types from each other. Furthermore, parameters and indicators are grouped with distinct background colors. In figure 20 can be seen a timeline visualization from the same data presented in figure 19 in table view.

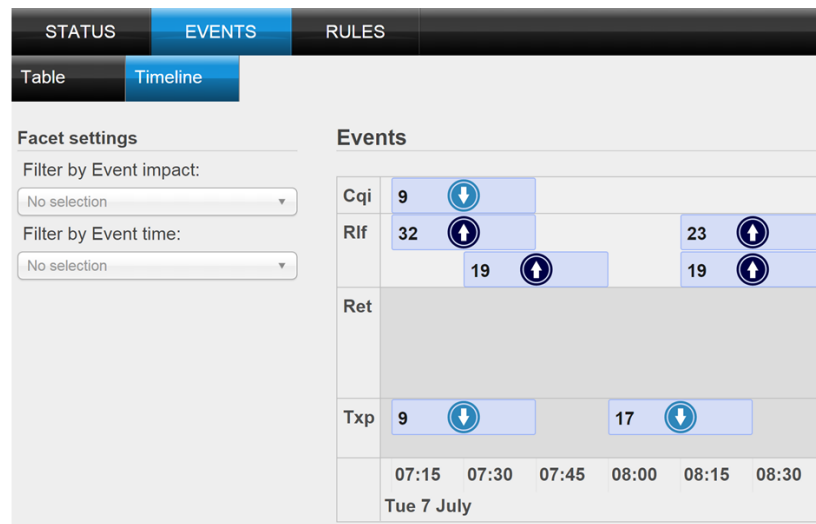


Figure 20: Timeline for events

### 7.3.2 Use Case: Selecting Facets

A facet selection use case for event view has two phases:

- Select values from facets
- Verify the result set in the table and timeline

Facets are chosen as: 07.07.2015 10 : 30 and *Decrease*. As it can be seen in figure 21, three events fulfil these conditions: one *Txp* event and two *Cqi* events.

**Facet settings**

Filter by Event impact: Decrease

Filter by Event time: 7.7.2015 10h 30min

**Events**

Time	Parameter	Indicator	Impact	Delta	From	To
10:30:00 07.07.2015	Cqi17		-45 %	0.36	0.2	
10:30:00 07.07.2015	Cqi24		-61 %	0.6	0.24	
10:30:00 07.07.2015	Txp24		-65 %	46	16	

(a) Table after facet selections

**Facet settings**

Filter by Event impact: Decrease

Filter by Event time: 7.7.2015 10h 30min

**Events**

Cqi	24									
	17									
Rlf										
Ret										
Txp	24									
	10:15	10:30	10:45	11:00	11:15	11:30	11:45			
	Tue 7 July									

(b) Timeline after facet selections

Figure 21: Result set after facet selections visualized as a table and timeline

## 7.4 Rules

### 7.4.1 Description

The rule view provides faceted search interface and modification for the uncertain rules of the MLN reasoner. Thus, the user can investigate the structure of the rules and manipulate the rule base in order to learn its logic.

Facets are generated from rule classes so that every class combined with its parameters or indicators are presented as facets. Facet selections are bound to corresponding values in the rule creation form.

The table is the only visualization of the rule items. As it can be seen in figure 22, the table columns are formed from the rule weight and from the rule classes (context, objective and action). Rule classes are further divided into individual predicates which are listed inside the rule class field.

Figure 22 shows that above the table is presented an option to remove a set of rules with respect to current facet selections. There are also two buttons after every rule item: one for changing rule weight and another for removing the rule.

The screenshot shows a web interface for managing rules. At the top, there are tabs for 'STATUS', 'EVENTS', and 'RULES'. Below the 'RULES' tab, there are two buttons: 'Table' (selected) and 'Create rule'. On the left, there are 'Facet settings' with several dropdown menus for filtering rules based on different criteria like 'MLNAction-Ret', 'MLNAction-Txp', 'MLNContext-Cqj', 'MLNContext-Rlf', 'MLNObjective-Cqj', and 'MLNObjective-Rlf'. In the center, there is a section titled 'Remove rules with following predicates (584 rules):' with a text input containing a logical expression and a 'Remove rules' button. Below this is a table titled 'Rules' with the following columns: 'ruleContext', 'ruleObjective', 'ruleAction', and 'ruleWeight'. Each row in the table represents a rule and includes two buttons: 'Change weight' and 'Remove'.

ruleContext	ruleObjective	ruleAction	ruleWeight		
I(t,c',Rlf,High)	O(t,c,Cqi,Inc)	A(t,c',Ret,Inc)	0.8	Change weight	Remove
I(t,c,Cqi,High)		A(t,c',Txp,Inc)			
I(t,c',Rlf,High)	O(t,c,Cqi,Dec)	A(t,c',Ret,Inc)	0.68	Change weight	Remove
I(t,c,Cqi,Low)	O(t,c,Rlf,Dec)	A(t,c',Txp,Dec)			
I(t,c,Rlf,High)					
I(t,c',Rlf,High)	O(t,c,Cqi,Inc)	A(t,c',Ret,Inc)	0.63	Change weight	Remove
I(t,c,Cqi,Low)		A(t,c',Txp,Dec)			
I(t,c',Cai,Low)	O(t,c,Cai,Dec)	A(t,c',Ret,Inc)	0.61	Change weight	Remove

Figure 22: Rules visualized as a table

### 7.4.2 Use Case: Selecting Facets

A facet selection use case for the rules has three phases:

- Select values from facets
- Verify results in the table
- Verify the prefilled rule creation form

Facets are chosen as *high Cqi* and *high Rlf* for context, *decrease Cqi* and *decrease Rlf* for objective, and *increase Ret* and *decrease Txp* for action. Facet selections reduce the result set from over 1000 items to eight items. However, figure 23 illustrates an issue concerning rule facets: a rule might have two facet values inside the same facet which causes an incorrect facet count in the facet box. For example, when choosing a facet value *high Cqi*, the result set is correctly filtered to items having predicate  $I(t, c, Cqi, High)$  or  $I(t, c', Cqi, High)$ . However, a rule can contain both  $I(t, c', Cqi, Low)$  and  $I(t, c, Cqi, High)$  (marked with red box inside the table), which causes this issue. The figure also shows the incorrect facet count (marked with another red box inside facet settings) which is a consequence of this issue. The incorrect facet count indicates that although the result set is filtered with respect to the selection *high Cqi*, there are still items also with facet value *low Cqi*.

Figure 23: Defect in facet selection: facet values do not have a distinct count

Figure 24 verifies that facet selections made in the rule view are bound to the rule creation form values. Moreover, the figure shows that the MLN rule text field above the form is automatically filled with respect to the form values.

### 7.4.3 Use Case: Creating a Rule

A use case for creating a rule has following actions:

- Select values from the rule creation form

- Modify the input field and submit the rule
- Verify the result

Figure 25 verifies that after a valid MLN rule is submitted to the server and the page is reloaded, the new rule appears in the table view. If the user creates an invalid MLN rule, it causes a SPARQL update error which is shown to user with a popup box.

One defect in the rule creation is that the SPARQL update script, which creates a new rule, do not create facet values for the rule. Thus, the rule is present in the default view (no facet selections) but it disappears if any facet is selected (as it does not contain any facet information).

#### 7.4.4 Use Case: Updating Weights

One use case for the rule view is updating a weight for an individual rule. This use case consists of two phases described below.

- Update weight value for a rule
- Verify the result

Figure 24: Prefilled form with respect to facet selections

**Create MLN rule**

5.0 [(I(t,c,Cqi,High)) => [(O(t,c,Cqi,Inc)) <=> (A(t,c,Txp,Inc))]]

Create rule Reset

Select context:

Cqi Rlf

Select objectives:

Cqi Rlf

Select actions:

Txp Ret

Select weight:

Weight

(a) Form before creation

**Rules**

ruleContext	ruleObjective	ruleAction	ruleWeight	
I(t,c,Cqi,High)	O(t,c,Cqi,Inc)	A(t,c,Txp,Inc)	5	Change weight Remove
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Ret,Dec) A(t,c,Txp,Dec)	2.5	Change weight Remove
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Txp,Dec)	0.97	Change weight Remove
I(t,c,Cqi,Low) I(t,c,Rlf,High)	O(t,c,Cqi,Inc) O(t,c,Rlf,Inc)	A(t,c,Ret,Dec) A(t,c,Txp,Inc)	0.85	Change weight Remove
				Change weight Remove

(b) Table after creation

Figure 25: Rule creation form and table view after creation

Weight is updated for the uppermost rule in figure 26. Weight value is set from 1.17 to 2.5. As figure 26 shows, the value is correctly updated to the rule table.

#### 7.4.5 Use Case: Removing Individual Rules

Removing an individual rule consists of two actions:

- Remove a specific rule
- Verify the result

Rules			
ruleContext	ruleObjective	ruleAction	ruleWeight
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Ret,Dec) A(t,c,Txp,Dec)	1.17
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Txp,Dec)	0.97
I(t,c,Cqi,Low)	O(t,c,Cqi,Inc)	A(t,c,Ret,Dec)	0.85
I(t,c,Rlf,High)	O(t,c,Rlf,Inc)	A(t,c,Txp,Inc)	

(a) Before update

Rules			
ruleContext	ruleObjective	ruleAction	ruleWeight
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Ret,Dec) A(t,c,Txp,Dec)	2.5
I(t,c,Cqi,Low)	O(t,c,Cqi,Dec)	A(t,c,Txp,Dec)	0.97
I(t,c,Cqi,Low)	O(t,c,Cqi,Inc)	A(t,c,Ret,Dec)	0.85
I(t,c,Rlf,High)	O(t,c,Rlf,Inc)	A(t,c,Txp,Inc)	

(b) After update

Figure 26: Table view before and after submitting new weight for a rule

Figure 27 shows that after confirmation is done to rule removal, rule is vanished from the view.

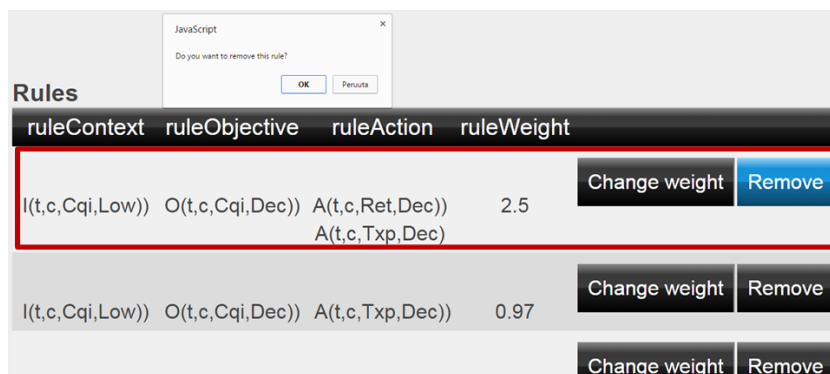
#### 7.4.6 Use Case: Removing a Set of Rules

Removing a set of rules is based on the current facet selections which define the conditions for the rule set. This use case has three phases:

- Select a facet value
- Remove rules with respect to facet condition
- Verify the result

Figure 28 illustrates how a set of rules is removed. Facet value *increase Txp* is selected and this selection is visualized above the rule table. After clicking the *Remove rules* button and confirming the action, rules are removed from the ontology





ruleContext	ruleObjective	ruleAction	ruleWeight	Change weight	Remove
I(t,c,Cqi,Low))	O(t,c,Cqi,Dec))	A(t,c,Ret,Dec)) A(t,c,Txp,Dec)	2.5	Change weight	Remove
I(t,c,Cqi,Low))	O(t,c,Cqi,Dec))	A(t,c,Txp,Dec))	0.97	Change weight	Remove
				Change weight	Remove

(a) Before removal



ruleContext	ruleObjective	ruleAction	ruleWeight	Change weight	Remove
I(t,c,Cqi,Low))	O(t,c,Cqi,Dec))	A(t,c,Txp,Dec))	0.97	Change weight	Remove
I(t,c,Cqi,Low)) I(t,c,Rlf,High))	O(t,c,Cqi,Inc) O(t,c,Rlf,Inc))	A(t,c,Ret,Dec)) A(t,c,Txp,Inc)	0.85	Change weight	Remove
I(t,c',Rlf,High))	O(t,c,Cqi,Inc))	A(t,c',Ret,Inc))	0.8	Change weight	Remove

(b) After removal

Figure 27: Table view before and after removing a rule

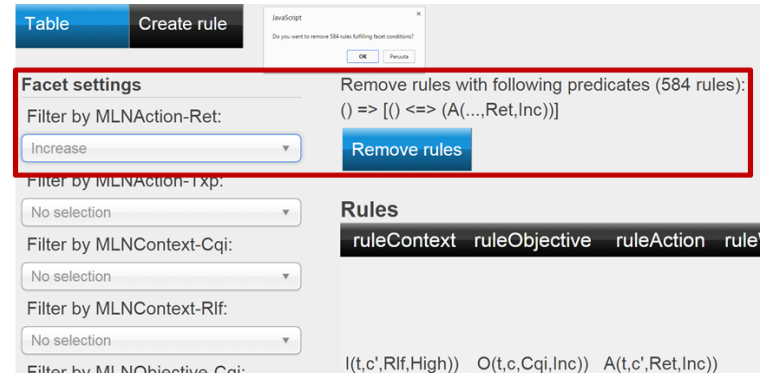
and from the result set. Also, all the facet selections are automatically cleared after removal. The figure illustrates that after the action, no facet selection for *increase Txp* exists as every rule that contains this value has been removed.

## 7.5 Plugins for the GUI

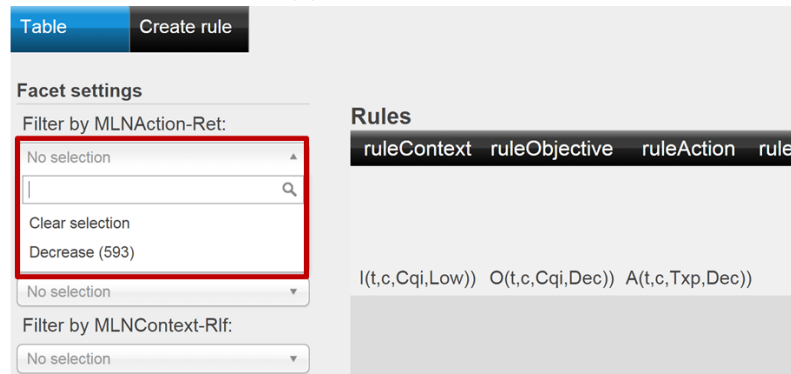
This application has several Javascript/jQuery plugins adapted for different purposes. Suitable plugins were presented and analyzed in section 6.2. From those alternatives, following plugins were adapted to the GUI:

For select boxes and table sorting, no alternative plugins were introduced as their functionalities are rather simple. Also, *chosen.js* and *tablesorter.js* were considered suitable because they are jQuery plugins, easy to use, and have the required functionality.

For visualization plugins, there were some alternatives for both the network graph and timeline. For both visualizations, *vis.js* plugin was chosen to simplify implementation



(a) Before removal



(b) After removal

Figure 28: Table view before and after removing a set of rules

Table 1: Plugins for the GUI

Purpose	Tool
Facets (select boxes)	<i>chosen.js</i>
Table (sort by column)	<i>tablesorter.js</i>
Network graph (visualization)	<i>vis.js</i>
Timeline (visualization)	<i>vis.js</i>

and to unify the visual look in the GUI. The plugin provides the same data model and data processing methods for network graph and timeline visualization.

## 8 Data Management

This section presents the implementation of the data management. First section gives an overview of the data sequence in the demonstrator. After that, subsections describe the implementation of how the network- and MLN-related data is processed into the ontology, how the ontology is kept consistent, and how it is queried.

### 8.1 Data Sequence Diagram

The demonstrator gets both network- and MLN-related data from the MLN reasoning system. Figure 29 explains the data sequence of the demonstrator starting from data retrieval (from the MLN reasoning system) and ending in a rule update (from the user interface). A Java component, named here as the Ontology Processor, retrieves the data from the MLN reasoning system in four phases as PM (`sendPM`), CM (`sendCM`), MLN rules (`sendRules`) and reasoning outcome (`sendActionProposals`) are sent separately. The Ontology Processor parses (with methods `PMtoRDF`, `CMtoRDF`, `rulesToRDF` and `actionsToRDF`) retrieved data objects into an OWL 2 ontology (Network- and MLN sub-ontologies). When all four data objects are retrieved and parsed into the ontology, a reasoner checks the consistency of the current ontology.

The ontology is sent to a SPARQL server (`updateOntology`) which re-creates facets (`generateFacets`) with respect to the new ontology. Finally, updated ontology can be queried by the UI (for example `getItems` and `getFacets`). As user modifies the rule base a SPARQL update is sent to SPARQL server (`updateRuleBase`) in order to save changes into the ontology. Moreover, MLN reasoning system can execute SPARQL queries to retrieve the updated rule base (with a method `getRuleBase`).

### 8.2 Input Data to RDF

Network-related performance and configuration data (PM and CM) are retrieved as two separate JSON objects. The Ontology Processor parses JSON objects to OWL 2 ontology instances (ABox) which are appended to the predefined TBox ontology. Following example summarises the JSON input for the PM data. The input consists of an array of cells defined with and id ("`key`"). Every cell has KPI values, such as "`throughput`", "`rlf`", "`cqi`", and "`load`".

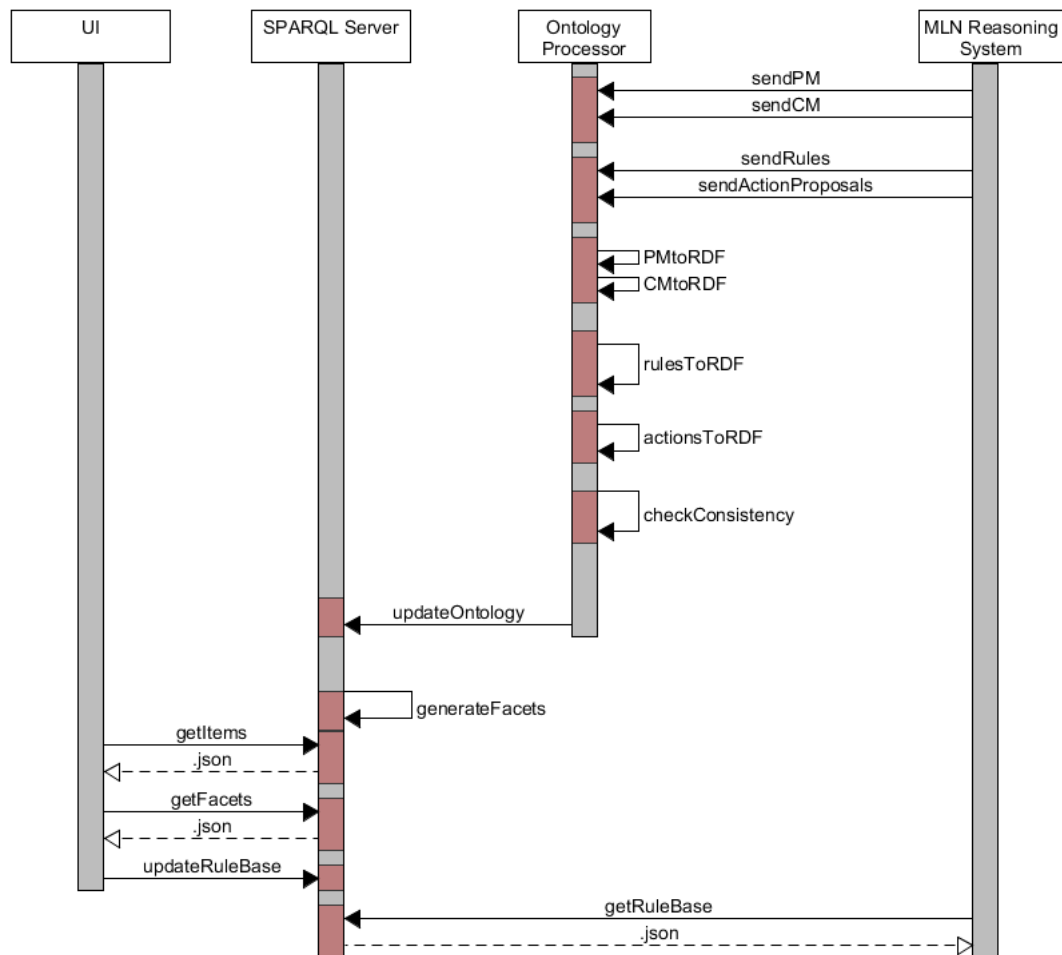


Figure 29: Data sequence in the demonstrator

```
[ {
  "key" : 12,
  "value" : {
    "throughput" : 10388140,
    "rlf" : 114,
    "cqi" : 0.7,
    "load": 0.8,
  }
  ...
}
```

An example below gives an insight of the CM data retrieved as JSON. Again, the input file contains an array of cells. Every cell contains configuration data as "tilt", "txPower", and "beamSteer". Moreover, every cell contains an array defining its current cell neighbors.

```
[ {
  "key" : 21,
  "value" : {
    "tilt" : 1.0,
    "txPower" : 46.0,
    "beamSteer" : 0.0,
    "neighbors" : [ {
      "id" : 22
    }, {
      "id" : 6
    }, {
      "id" : 23
    }
  ]
  ...
}
```

The MLN files parsed in the demonstrator consist of the rule base and reasoning outcome (action proposals). An example of the rule base is described below. As it can be seen, the file contains predicate definitions (for example `A(time, cell, param, delta)`), certain rules (for example `!0(a1, a2, a3, Dec) v !0(a1, a2, a3, Inc).`), and uncertain rules (for example `0.158837 (I(t, c, Cqi, High)) => [(0(t, c, Cqi, Inc)) <=> (A(t, c, Txp, Inc))]`). The rule base contains approximately 1300 uncertain rules.

```
//predicate declarations
A(time, cell, param, delta)
E(time, cell, param, extremum)
```

```

I(time,cell,kpi,level)
N(cell,cell,neighborship)
O(time,cell,kpi,delta)

!E(a1,a2,a3,Min) v !E(a1,a2,a3,Max) .

!A(a1,a2,a3,Dec) v !E(a1,a2,a3,Min) .

!A(a1,a2,a3,Inc) v !E(a1,a2,a3,Max) .

!A(a1,a2,a3,Dec) v !A(a1,a2,a3,Inc) .

!O(a1,a2,a3,Dec) v !O(a1,a2,a3,Inc) .

!N(a1,a1,a2) .

!N(a1,a2,Inter) v !N(a1,a2,Intra) .

N(a1,a2,a3) v !N(a2,a1,a3) .

0.158837 (I(t,c,Cqi,High)) => [(O(t,c,Cqi,Inc)) <=> (A(t,c,Txp,Inc))]
0.267003 (I(t,c,Cqi,High)) => [(O(t,c,Cqi,Inc)) <=> (A(t,c,Txp,Dec))]
0.0499781 (I(t,c,Cqi,High)) => [(O(t,c,Cqi,Inc)) <=> (A(t,c,Ret,Inc))]
0.123257 (I(t,c,Cqi,High)) => [(O(t,c,Cqi,Inc)) <=> (A(t,c,Ret,Dec))]
.....

```

The MLN reasoning outcome is retrieved as an other file which is described below. The file contains action proposal probability for every cell and for every action. In the current system there are four possible actions for every cell: increase or decrease the remote electrical tilt (RET) or the transmission power (Txp).

```

A(1,Ret,Inc) 0.40401
A(1,Ret,Dec) 0.364014
A(1,Txp,Inc) 0.359014
A(1,Txp,Dec) 0.362014
A(2,Ret,Inc) 0.364014
A(2,Ret,Dec) 0.380012

```

### 8.3 Consistency Checking of the Ontology

To ensure that the ontology stays consistent after updates, an OWL 2 DL reasoner is executed after the ontology is reloaded with new instances. Pellet<sup>18</sup> is adapted to the system for this purpose because it has functionality to check the ontology consistency and to give an explanation for inconsistencies. The example below shows the explanation which Pellet reasoner gives for an inconsistency when `Indicator` and `Parameter` are defined to be disjoint but they are sharing (`rdfs:range`) an object property (`hasEvent`). The consistency report is logged to a file and expert user can investigate the report in order to analyse if the TBox need modification.

```
c:EventRlf52015-06-27_19_04_40_242 c:eventBelongsToIndicator c:Rlf5 .

[] a owl:AllDisjointClasses ;
owl:members (c:Cell c:Event c:Impact c:Indicator c:Parameter c:Value) .

c:eventBelongsToIndicator rdfs:range c:Indicator .

c:Rlf5 c:hasEvent c:EventRlf52015-06-27_19_04_40_242 .

c:eventHasEffect rdfs:range c:Parameter ;
owl:inverseOf c:hasEvent .
```

### 8.4 SPARQL Server

After data is processed to the ontology and consistency is checked, the ontology is uploaded to Fuseki server. From the two SPARQL engines discussed in section 6.1.1, Fuseki is chosen for this system. The main reason for selecting Fuseki is the earlier experience of using it and thus its suitability for this purpose is already known in practice.

### 8.5 Creating Facets

After a new ontology is uploaded to the SPARQL server, SPARQL update scripts are automatically executed to create new facets and facet values from the uploaded ontology. Every facet has two SPARQL update queries. The first query is generating `ConceptScheme`, `Concept` (related to `ConceptScheme`), and facet properties (related to `Concept` instances). The second query binds ontology instances to `Concept` instances.

<sup>18</sup><https://github.com/complexible/pellet>

Appendix A shows how facets, facet values, and facet properties are generated for every indicator facet. For example, the query generates a `ConceptScheme` instance `c:StatusCqi` which has `Concept` instances `c:HighCqi` and `c:LowCqi`. Along with `c:HighCqi` and `c:LowCqi` a facet property `c:facetCqi` is generated.

Appendix B shows how the generated facet values are bound to cell instances. `DELETE` statement is not needed in SPARQL updates as new ABox always replaces the earlier in the SPARQL server. Thus, earlier facet data is cleared when ontology is re-uploaded to the server.

## 8.6 SPARQL Patterns for the GUI

SPARQL server is queried by the user to retrieve the view- and facet-specific data and the server is updated with modified rules every time user manipulates the rules. This section first describes the general structure of SPARQL query patterns (result set query and facet query) and then explains how modifications to the rule base are done with SPARQL update statements.

### 8.6.1 Result Set

The basic idea of retrieving result sets for every view is defined below. The example is describing the result set query for cell status view but by replacing `c:Cell` (first line in `SELECT` statement) with `c:Event` or `c:MLNRule` and `c:facetCellProperty` (third line) with `c:facetEventProperty` or `c:facetRuleProperty` one can retrieve result sets also for cell events and MLN rules. The last two lines inside the statement indicate that user has made facet selections for low CQI and high RLF. These selections filter the results to only those fulfilling the conditions.

```
SELECT ?label ?predLabel ?objLabel
{
  ?s a c:Cell .
  ?s skos:prefLabel ?label .
  ?predLabel rdfs:subPropertyOf c:facetCellProperty .
  ?s ?predLabel ?facetval .
  ?facetval skos:prefLabel ?objLabel .
  ?s <http://www...#facetCqi> <http://...#LowCqi> .
  ?s <http://www...#facetRlf> <http://...#HighRlf> .
}
ORDER BY ?label
```



The statement above returns facet information for table view: `?label` defines the object (cell, event or MLNRule), `?predLabel` the column title (facet property), and `?objLabel` the column value (facet value). In addition to facet information, every view needs extra information for example action proposals for cell states. Thus, this SPARQL query pattern needs to be filled with additional data for every table view.

### 8.6.2 Facets

Facets and facet values are queried with another SPARQL query pattern as the example below describes. The example is for facets in the cell status view but it can be modified to match with events and rules by changing `c:facetCellProperty` (fourth line in `Select` statement) to either `c:facetEventProperty` or `c:facetRuleProperty`. The query has low Cqi selected as a facet value and thus facet-related data is filtered with this condition.

`SELECT` parameters return facet values as `?facet`, facet labels as `?facetLabel`, facet scheme as `?scheme` and the number of occurrences for a facet value as `?count`. `?prop` value is a facet property, for example `c:facetRlf`.

```
SELECT ?facet ?facetLabel ?scheme ?prop ?order (count(?facet) as ?count)
WHERE {
  ?facet skos:inScheme ?conScheme .
  ?conScheme skos:prefLabel ?scheme .
  ?facet c:facetOrder ?order .
  ?prop rdfs:subPropertyOf c:facetCellProperty .
  ?s ?prop ?facet .
  ?facet skos:prefLabel ?facetLabel .
  ?s <http://www...#facetCqi> <http://www...#LowCqi> .
}
group by ?facet ?facetLabel ?altLabel ?scheme ?prop ?min
order by ?scheme ?order
```

### 8.6.3 Rule Updates

Rule updates are done with SPARQL update statements which have the following functionality:

- Create rule
- Update rule weight

- Remove a rule
- Remove set of rules

Rule creation differs from the other rule modification tasks because the input needs to be parsed from MLN syntax to RDF triples. Parsing and validation are done with the same string parser which is used for rule parsing in the Ontology Processor. After an MLN rule is parsed to triples, it is inserted into the server with a statement described in appendix C.

Functionalities for latter rule modification tasks are more straightforward as user actions can be mapped straight to RDF triples. Appendix D shows examples of how a rule weight is updated, single rule is removed and set of rules is removed.

Part IV  
Discussion

## 9 Evaluation

This section evaluates the implementation described in sections 8 and 7. The evaluation is based on the research questions defined in the introduction section:

- What information is needed?
- Which visualization methods support the information need?
- Which facets support the information need?
- What other functionality supports the information need?
- How to provide scalability for the search interface?

Evaluation of every topic is divided into implementation and deficiencies parts. Implementation sections focus on discussing the positive and negative aspects of the demonstrator with respect to the research questions. Deficiencies sections discuss issues which are missing from the demonstrator.

### 9.1 Information Quality

Information offered in the demonstrator can be divided into three abstract types: the input and output of the MLN reasoner (cell states and action proposals), the impacts of the reasoner output (configurations and anomalies) and content of the rule base. These three information types are presented in separate faceted search views to bring diverse perspectives to explore the reasoning system.

#### 9.1.1 Implementation

The cell status view describes the cell states and action proposals of the reasoner. Section 7.2 clarifies that this information is available in the view as the latest MLN-monitored KPI values and action proposals are available. In addition to KPIs and action proposals, the amount of neighbors and neighbor relations are presented to provide additional information about cell characteristics. Although neighbor relation might not be relevant information in the demonstration, it is still considered as an informative aspect in describing cell states. For example, the amount of neighbors might reveal some pattern which occurs only with cells with many (or few) neighbors.

The cell event view provides user verification for the causes and consequences of configurations made by the reasoning system. As section 7.3 describes, current implementation is revealing temporal relationships between configurations and anomalies. However, in order to make investigations related to the reasoning system, the user needs to assume that all configurations are actually executed by the reasoning system (no manual or SON function configuration). This is because the demonstrator does not recognize the type of the configuration (who or what has made it).

In the rule base representation, clauses for uncertain rules are split into rule classes, as shown in MLN ontology description 5.6.2 and in view description 7.4. The current representation of the rule base can be seen sufficient because it is not seen reasonable to split or hide rule parts any further without losing relevant information.

### 9.1.2 Deficiencies

In the cell event view, historical action proposals should be presented and there should be a relation between action proposals and configurations. This relation would provide the user information about action proposals (type and probability) that causes successful or unsuccessful configurations. As it can be seen from the data sequence diagram (figure 29), this approach would need changes to data processing, as the current system is not collecting historical data from action proposals.

Another issue in the cell event view is that the user might be interested to see information about cell neighbor relations (without changing to the cell status view) to verify relations between configurations and anomalies. The use case in section 7.3.2 shows that configuration in cell 24 most probably affects to its own *Cqi* value but possibly also to *Cqi* value of cell 17. Knowing that cell 17 and 24 are not neighbors reveals that cell 17 is not likely affected by this configuration.

An information gap in the demonstrator is that MLN reasoner-related information is not provided about certain rules or settings data (thresholds, KPIs, parameters, etc.). Both of these would be helpful for the user to get better understanding about the rule base.

## 9.2 Facets

### 9.2.1 Implementation

The use case for cell states 7.2.2 shows that especially KPI-related facets demonstrate well the relationship between action proposals and cell states. Moreover, the use case shows that KPI facets support rule creation as they are bound to *Context* fields

in the rule creation form (figure 17). As discussed earlier, information about the amount of neighbors is not seen as an important aspect here and therefore the facet might be unnecessary in this context.

The use case 7.3.2 for cell events shows that relations between configurations and anomalies can be revealed with time and impact facets. For example, in the use case the user can verify that *Txp* for cell 24 decreases its own *Cqi*. However, a minor issue with the time facet is that there are no facet values for months. In the current system the issue can be avoided by keeping the simulation inside a month (simulation time can be manipulated to start at the beginning of a month). However, with respect to a more consistent system and longer simulations, also months and years should be included in the time facet.

In the rule view, facets are chosen as a combination of rule classes (*context*, *objectives* and *actions*) and their objects (*Cqi*, *Rlf*, *Txp*, and *Ret*). As the structure of the rules is fixed, these facets support well browsing activities for the rule base. The use case 7.4.2 shows that all the facets in this view support binding to the rule creation form which eases user's work load in rule creation.

However, rule facets might encounter an issue as the use case 7.4.2 demonstrates. System produces incorrect facet information when rule includes cell or time variables with opposite facet values (as there is *low* and *high* *Cqi* values for *MLNContextCqi* in the use case). Disjointness could be solved by adding facet information about different cell and time values or by limiting facet values to match only cell variables which indicate the cell itself (*c*) and not its neighbors (*c'*).

### 9.2.2 Deficiencies

Cell states should have facets for action proposals as the objective of the view is to demonstrate relation between the input and output of the reasoner. Therefore, the demonstrator should also enable user to explore items with respect to output values.

In addition to time and impact facets in the cell event view, there should be an object facet. User might want to filter result set with respect to different object types such as parameter or indicator or with their subtypes as *Txp*, *Ret*, *Cqi* or *Rlf*. For example, a facet structure with two levels could be useful: the first level for the object type (parameter or indicator) and the second for their subtypes (*Txp*, *Ret*, *Cqi*, and *Rlf*).

In the rule view, a facet for weight values should be available. Weights could be grouped to crisp values *low*, *medium*, and *high*.

## 9.3 Visualization Methods

### 9.3.1 Implementation

A network graph was chosen as an alternative visualization for cell states. As figures 16 and 18 show, network graph visualization supports relationship discovery between cell states and action proposals. Figure 18 also shows that user can investigate the network topology after facet selections which might reveal new aspects from MLN reasoning system. Also, the observation of the cell table content has been simplified as KPI values and action proposals are visualized as icons instead of numerical values.

In cell events, the advantage of the timeline visualization is shown in figures 20 and 21 where causes and consequences can be easily observed from the timeline. In the event table, impacts are visualized with icons which again eases the information exploration.

Rules are visualized only as a table because no alternative visualization is considered to give any advantage to the rule base exploration. In the table visualization, the rules are easily observable as the rule content is grouped into rule classes (see figure 22).

### 9.3.2 Deficiencies

The network graph visualization already gives the relative positions of the nodes. However, the visualization could be enriched with spatial information so that nodes are rendered on a map. This might help user to understand why cells (and eNBs) are located in certain places.

In the timeline visualization, increasing and decreasing impacts are separated with arrows and colors (figure 20). However, there are no colors to indicate if the impact is good or poor for the system. For example, an increase in  $Cqi$  is generally considered as good impact but an increase in  $Rlf$  as poor. Categorization of impacts is problematic as there is no information available about impact qualities (need to be set manually).

## 9.4 Other Functionality

### 9.4.1 Implementation

Other functionality includes components which support user's information exploration. These components include rule base manipulation tasks such as creating a rule,

updating a rule weight, removing a single rule, and removing a set of rules. These actions are successfully implemented into the system as it can be seen in use cases 7.4.3, 7.4.4, 7.4.5, and 7.4.6. After MLN reasoning system retrieves the updated rules and uses them in the reasoning, user can then investigate what effects did the changes have to the reasoning results.

As mentioned in the rule creation use case (section 7.4.3), one minor defect is the missing facet information for the created rule. The demonstrator should bind new instances to facet values and therefore some improvement to the current SPARQL update script is needed.

## 9.4.2 Deficiencies

Use case analysis in the system design (section 5.3) stated that the system should allow modification of an existing rule. However, the current system does not have modification action implemented. Naturally, modification can already be achieved by deleting a rule and creating a new rule with desired changes.

Another functional deficiency is that the user should be able to modify the thresholds of crisp values. The MLN reasoning system could then retrieve the modified threshold values from the ontology and change its own threshold values to the reasoning system. This way user would have more tools to modify the behaviour of the reasoning system and investigate what impact the threshold values have to the reasoning output.

## 9.5 Scalability

### 9.5.1 Implementation

The scalability of the system is ensured with the architectural design described in section 5.6 and with the implementation of data management explained in section 8. Sections 8.1 and 8.2 explain briefly how the network- and MLN- related data are aggregated and published as an OWL 2 ontology. The advantage of using an ontology as a data model is that the ontology can then be queried with SPARQL queries and be modified with SPARQL updates (section 8.6). Also, query patterns described in 8.6.1 and 8.6.2 simplify the development of the faceted search view, as patterns can be reused in the development.

Another advantage of the ontological approach is the possibility to use a semantic reasoner (such as OWL 2 DL reasoner) for consistency checks and for relationship discovery. Currently, Pellet reasoner is adapted to the system and it makes a consistency report which is logged to a file (section 8.3). A reasoning task for missing



properties is not yet adapted to the system as it is not seen to bring benefit for the current implementation. However, in further projects reasoner could be used to produce new properties to the ontology.

### 9.5.2 Deficiencies

One issue related to scalability is that the ontology needs some view-specific enrichment in order to generalize the view representation in the demonstrator. As described in the query design of SPARQL result sets (section 8.6.1), only semantics that is common between the views is based on facet properties. With the view-specific enrichment, SPARQL queries could be reformulated with a more general property structure which would define view-specific attributes (columns and column values) and would make the query pattern more usable.

## 10 Conclusion

This section concludes this thesis by first discussing about possible follow-up projects and then by summarising the thesis.

### 10.1 Future Work

Two possible future works are presented in this subsection. First is discussed how high-level goals could be adapted to the demonstrator for better human-reasoner interaction. Second topic discusses how open data can be utilized in this project.

#### 10.1.1 System Management with High-level Goals

The current implementation could be enhanced by creating high-level goals which the user can use to modify the behaviour of the reasoning system. As it is discussed in sections 2.3.2 and 2.3.3, user-defined high-level goals have been seen as a crucial aspect in the future SON architecture and generally in the automatic network management. Similarly, it would be interesting to examine how the MLN reasoning system could be controlled with high-level goals.

A high-level goal can be used to define active KPIs and dynamic threshold values for the active KPIs. For example, an energy-saving goal could be mapped to KPIs *Capacity* and *Cqi*. Threshold values for these KPIs can be scaled so that lower *Cqi* value is still satisfiable whereas *Capacity* value should be higher than usual. Naturally, this approach assumes that the reasoning system has a larger collection of KPIs and that the rule base can be dynamically modified by eliminating rules with irrelevant KPIs.

#### 10.1.2 Utilizing Linked Data in Reasoning

Another research direction for this work could be examining how linked data, such as RDF-modelled linked open data (LOD), can be used to extend the MLN reasoning to other aspects which might affect the network performance. For example, weather and event data might be relevant information with respect to quality and capacity issues.

An approach that uses LOD in network management was mentioned in section 3.4.1. This project introduced a LOD dataset utilizing event-related data. Similarly, the reasoning system could also be bound to other datasets by aggregating data

from different SPARQL or REST endpoints. However, it is likely that processing heterogeneous data into reasoner data (with crisp classification, rule structure, etc.) needs plenty of planning and implementation. Moreover, LTE simulator data and other datasets would not have real correlation which of course complicates the evaluation of the system.

## 10.2 Summary

An objective of this thesis was to create a case study which examines how a semantic search interface demonstrates the behaviour of an automated network management system. The underlying network management system consisted of a Self-Organizing Network (SON) function and a Markov Logic Network (MLN) reasoner with statistical and logical reasoning capabilities.

The search interface was implemented as a faceted search interface with an ontology as a data storage and a SPARQL engine as a query tool. The case study has five research questions (described in section 1.2) which defines the main focus of the implementation. The objective is to find user's information need, useful visualization methods, relevant facets, supporting functionality (in addition to faceted browsing), and a data architecture which is scalable for later projects.

The scope of this thesis includes both data architecture (ontology and data flow) design as well as graphical user interface (visualization and interaction) design. With respect to the research questions, the implementation has two significant aspects. First, the implementation aims to show that the design plan works in practice. Second, the evaluation of the demonstrator produces new answers and thoughts to the research questions.

Considering the objective of the thesis, the evaluation of the demonstrator is rather critical in order to reveal the relevant aspects which answer the research questions. Section 9 gives a rather accurate evaluation of the positive, negative, and the missing aspects of the current demonstrator. Therefore, the thesis can be seen to provide an encompassing case study for semantic search interface implementation in this context.

Moreover, as the search interface enables the user to acquire knowledge from the reasoning system, the user might discover new ideas and enhancements for the reasoning system itself. As section 10.1 shows, some new interesting research topics have already come up from this case study.

## References

- [1] Ericsson. Ericsson Mobility Report. Technical report, Ericsson, 07 2014. <http://www.ericsson.com/res/docs/2014/ericsson-mobility-report-june-2014.pdf>.
- [2] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper. Technical report, Cisco, 02 2015. [http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white\\_paper\\_c11-520862.html](http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html).
- [3] Seppo Hämmäläinen, Henning Sanneck, and Cinzia Sartori. *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*. Wiley Online Library, 1st edition, 2012.
- [4] Horn. Autonomic Computing: IBM’s Perspective on the State of Information Technology. *IBM Corporation*. Available at [http://www.research.ibm.com/autonomic/manifesto/autonomic\\_computing.pdf](http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf), October 2001.
- [5] Nancy Samaan and Ahmed Karmouch. Towards autonomic network management: an analysis of current and future research directions. *Communications Surveys & Tutorials, IEEE*, 11(3):22–36, 2009.
- [6] Daniel M Russell, Paul P Maglio, Rowan Dordick, and Chalapathy Neti. Dealing with ghosts: Managing the user experience of autonomic computing. *IBM Systems Journal*, 42(1):177–188, 2003.
- [7] Lise Getoor. *Introduction to statistical relational learning*. MIT Press, 2007.
- [8] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, February 2006.
- [9] C.I. Cox. *An Introduction to LTE: LTE, LTE-advanced, SAE, VoLTE and 4G Mobile Communications*. Wiley Online Library, 2014.
- [10] Stefania Sesia, Issam Toufik, and Matthew Baker. *LTE: the UMTS long term evolution, From Theory to Practice*. Wiley Online Library, 2nd edition, 2011.
- [11] Magdalena Nohrborg. Lte. <http://www.3gpp.org/technologies/keywords-acronyms/98-lt>. Accessed: 2015-02-16.
- [12] 3GPP. Telecommunication management; self-organizing networks (son); concepts and requirements. [http://www.etsi.org/deliver/etsi\\_ts/132500\\_132599/132500/12.01.00\\_60/ts\\_132500v120100p.pdf](http://www.etsi.org/deliver/etsi_ts/132500_132599/132500/12.01.00_60/ts_132500v120100p.pdf). Accessed: 2015-02-16.

- [13] Olav Østerbø and Ole Grøndalen. Benefits of Self-Organizing Networks (SON) for mobile operators. *Journal of Computer Networks and Communications*, 2012, 2012.
- [14] Lasse Laine. 3rd Generation Partnership Project Long Term Evolution. Master's thesis, Aalto University, Sep 2011.
- [15] 3GPP. Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Requirements. Technical report, 3rd Generation Partnership Project, December 2013.
- [16] 3GPP. Technical Specification Group Services and System Aspects; Telecommunication management; Self-Organizing Networks (SON); Study on self-healing. Technical report, 3rd Generation Partnership Project, December 2014. [http://www.etsi.org/deliver/etsi\\_ts/132500\\_132599/132541/12.00.00\\_60/](http://www.etsi.org/deliver/etsi_ts/132500_132599/132541/12.00.00_60/).
- [17] T Kürner, M Amirijoo, I Balan, H van den Berg, A Eisenblätter, et al. Final report on self-organisation and its implications in wireless access networks, 2010. <http://www.fp7-socrates.eu/files/Deliverables/>.
- [18] Juan Ramiro and Khalid Hamied. *Self-Organizing Networks (SON): Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE*. Wiley Online Library, 2nd edition, 2011.
- [19] A Eisenblätter, B Gonzalez Rodriguez, Fredrik Gunnarsson, T Kurner, Remco Litjens, Bart Sas, Berna Sayrac, Lars Christoph Schmelz, and Colin Willcock. Integrated self-management for future radio access networks: Vision and key challenges. In *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, pages 1–10. IEEE, 2013.
- [20] L. Jorguseski, A. Pais, F. Gunnarsson, A. Centonza, and C. Willcock. Self-organizing networks in 3gpp: standardization and future trends. *Communications Magazine, IEEE*, 52(12):28–34, December 2014.
- [21] Brendan Jennings, Sven van der Meer, Sasitharan Balasubramaniam, Dmitri Botvich, Mícheál ó Foghlú, William Donnelly, and Johannes Strassner. Towards autonomic management of communications networks. *Communications Magazine, IEEE*, 45(10):112–121, 2007.
- [22] S. Kuklinski, M. Skrocki, L. Rajewski, J. Meseguer Llopis, and Z. Wereszczynski. Garson: Management performance aware approach to autonomic and cognitive networks. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 914–918, Dec 2012.
- [23] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25, 1995.

- [24] Randall Davis, Howard Shrobe, and Peter Szolovits. What Is a Knowledge Representation? *AI Magazine*, 14(1), 1993.
- [25] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.
- [26] Grigoris Antoniou and Frank Van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.
- [27] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. Technical report, W3C, February 2004. <http://www.w3.org/TR/owl-features/>.
- [28] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview. second edition, W3C, December 2012. <http://www.w3.org/TR/owl2-overview/>.
- [29] Antoine Isaac and Ed Summers. Skos simple knowledge organization system primer. Technical report, W3C, August 2009. <http://www.w3.org/TR/skos-primer/>.
- [30] Steve Harris and Andy Seaborne. Sparql 1.1 query language. Technical report, W3C, December 2013. <http://www.w3.org/TR/sparql11-query/>.
- [31] Abdalbaki Uzun and Axel Küpper. Openmobilenetwork: extending the web of data by a dataset for mobile networks and devices. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 17–24. ACM, 2012.
- [32] Anja Jentzsch Max Schmachtenberg, Christian Bizer and Richard Cyganiak. The Linking Open Data cloud diagram 2014. <http://lod-cloud.net/>. Accessed: 2015-08-25.
- [33] Vilho Räsänen and Haitao Tang. Knowledge modeling for conflict detection in self-organized networks. In Kostas Pentikousis, Rui Aguiar, Susana Sargento, and Ramón Agüero, editors, *Mobile Networks and Management*, volume 97 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 107–119. Springer Berlin Heidelberg, 2012.
- [34] Gabriela F Ciocarlie, Chih-Chieh Cheng, Christopher Connolly, Ulf Lindqvist, Kenneth Nitz, Szabolcs Novaczki, Henning Sanneck, and Muhammad Naseer-ul Islam. Demo: SONVer: SON verification for operational cellular networks. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, pages 611–612. IEEE, 2014.
- [35] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, January 2002.

- [36] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.
- [37] Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–98, January 2009.
- [38] Gary Marchionini. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.
- [39] Jürgen Koenemann and Nicholas J Belkin. A case for interaction: a study of interactive information retrieval behavior and effectiveness. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 205–212. ACM, 1996.
- [40] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 401–408, New York, NY, USA, 2003. ACM.
- [41] Daniel Tunkelang. Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–80, 2009.
- [42] Osmo Suominen, Kim Viljanen, and Eero Hyvönen. User-centric faceted search for semantic portals. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *The Semantic Web: Research and Applications*, volume 4519 of *Lecture Notes in Computer Science*, pages 356–370. Springer Berlin Heidelberg, 2007.
- [43] Carol Righi, Janice James, Michael Beasley, Donald L Day, Jean E Fox, Jennifer Gieber, Chris Howe, and Laconya Ruby. Card sort analysis best practices. *Journal of Usability Studies*, 8(3):69–89, 2013.
- [44] Markus Holi. *Crisp, Fuzzy, and Probabilistic Faceted Semantic Search*. PhD thesis, Aalto University, 2010.
- [45] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Evgeny, Sarunas Marciuska, and Dmitriy Zheleznyakov. Towards semantic faceted search. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 219–220, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.
- [46] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.
- [47] A Seaborne. Fuseki: serving rdf data over http. [http://jena.apache.org/documentation/serving\\_data/](http://jena.apache.org/documentation/serving_data/), 2011. [Online; Accessed: 2015-07-21].

- [48] Blazegraph Wiki. Nanosparqlserver - blazegraph. <https://wiki.blazegraph.com/wiki/index.php/NanoSparqlServer>, 2015. [Online; Accessed: 2015-07-21].
- [49] Jürgen Bock, Peter Haase, Qiu Ji, and Raphael Volz. Benchmarking OWL Reasoners. In Frank van Harmelen, Andreas Herzig, Pascal Hitzler, Zuoquan Lin, Ruzica Piskac, , and Guilin Qi, editors, *Proceedings of the ARea2008 Workshop*, volume 350, <http://ceur-ws.org>, June 2008. CEUR Workshop Proceedings.
- [50] Wiki. Force layout. <https://github.com/mbostock/d3/wiki/Force-Layout>, 2015. [Online; Accessed: 2015-07-21].
- [51] Almende B.V. vis.js documentation. <http://visjs.org/docs>. [Online; Accessed: 2015-05-30].
- [52] Anonym. Sigma js. <http://sigmajs.org/>. [Online; Accessed: 2015-05-30].
- [53] Rui Wang, Yasset Perez-Riverol, Henning Hermjakob, and Juan Antonio Vizcaíno. Open source libraries and frameworks for biological data visualisation: A guide for developers. *Proteomics*, 2014.
- [54] Max Franz. Cytoscape.js. <http://js.cytoscape.org/>. [Online; Accessed: 2015-05-30].



## A Appendix SPARQL Update Generating Indicator Facets

```

INSERT {
  ?schemeUri a skos:ConceptScheme .
  ?schemeUri skos:prefLabel ?prefLabel .
  ?schemeUri a owl:namedIndividual .

  ?conceptUri a skos:Concept .
  ?conceptUri skos:inScheme ?schemeUri .
  ?conceptUri skos:prefLabel ?label .
  ?conceptUri skos:altLabel ?label .
  ?conceptUri c:facetOrder ?order .
  ?conceptUri c:fuzzyMin ?min .
  ?conceptUri c:fuzzyMax ?max .

  ?facetPropUri a owl:FunctionalProperty ,
                owl:ObjectProperty ;

                rdfs:domain c:Cell ;
                rdfs:range ?range ;
                rdfs:subPropertyOf c:facetCellProperty ;
c:propertyRelationTo ?y .
}
WHERE {
  ?y rdfs:subPropertyOf c:cellHasIndicator
  ?y rdfs:range ?range .
  ?y skos:prefLabel ?prefLabel .
  BIND (URI(CONCAT("http://...#", "Status", ?prefLabel)) AS ?schemeUri)
  ?x rdfs:subClassOf c:FuzzyValue .

  ?indType rdfs:subClassOf c:FuzzyValueForIndicator .
  ?ind a ?indType .
  ?ind skos:prefLabel ?label .
  ?ind c:fuzzyMin ?min .
  ?ind c:fuzzyMax ?max .

  BIND((round(?min*100)) AS ?order)
  BIND (URI(CONCAT("http://...#", ?label, ?prefLabel)) AS ?conceptUri)
  BIND (URI(CONCAT("http://...#", "facet", ?prefLabel)) AS ?facetPropUri)
  FILTER (contains(str(?ind), ?prefLabel))
}

```

## B Appendix SPARQL Update Generating Relations Between Facet Values and Instances

```

INSERT {
  ?cell ?facetProp ?facet .
}
WHERE {
  ?prop rdfs:subPropertyOf c:hasIndicator .
  ?cell ?prop ?ind .
  ?prop skos:prefLabel ?lab .
  BIND (URI(CONCAT("...", "facet", ?lab)) AS ?facetProp)
  BIND (URI(CONCAT("...", "Status", ?lab)) AS ?scheme)
  ?ind c:numericValue ?val .
  ?facet skos:inScheme ?scheme .
  ?facet c:fuzzyMin ?min .
  ?facet c:fuzzyMax ?max .
  FILTER( ?min < ?val) .
  FILTER( ?max >= ?val) .
}

```

## C Appendix SPARQL Update for Creating a New Rule

```

INSERT {
  <http://www...#RuleContextCqiHigh1234>
    a      <http://www...#MLNContext> ;
  <http://www...#ruleClause>
    "I(t,c,Cqi,High)" ;
  <http://www...#ruleIndicator>
    <http://www...#Cqi> ;
  <http://www...#ruleMagnitude>
    <http://www...#HighValue> .

  <http://www...#RuleObjectiveCqiDecrease1234>
    a      <http://www...#MLNObjective> ;
  <http://www...#ruleClause>
    "O(t,c,Cqi,Dec)" ;
  <http://www...#ruleImpact>

```

```

        <http://www...#Decrease> ;
    <http://www...#ruleIndicator>
        <http://www...#Cqi> .

<http://www...#RuleActionTxpDecrease1234>
    a      <http://www...#MLNAction>;
    <http://www...#ruleClause>
        "A(t,c,Txp,Dec)]" ;
    <http://www...#ruleImpact>
        <http://www...#Decrease> ;
    <http://www...#ruleParameter>
        <http://www...#Txp> .

<http://www...#Rule1234>
    a      <http://www...#MLNRule> ;
    <http://www...#ruleAction>
        <http://www...#RuleActionTxpDecrease1234> ;
    <http://www...#ruleClause>
        "(I(t,c,Cqi,High)) => [(O(t,c,Cqi,Dec)) <=> (A(t,c,Txp,Dec))]" ;
    <http://www...#ruleContext>
        <http://www...#RuleContextCqiHigh1234> ;
    <http://www...#ruleObjective>
        <http://www...#RuleObjectiveCqiDecrease1234> ;
    <http://www...#ruleWeight>
        "0.5"^^xsd:double .

}
WHERE {
}

```

## D Appendix SPARQL Updates for Modifying and Removing Rules

### Updating a Weight

```

DELETE
{
  c:Rule1234 c:ruleWeight ?weight .
}
INSERT {

```

```

c:Rule1234 c:ruleWeight "1.97"^^xsd:double .
}
WHERE {
c:Rule1234 c:ruleWeight ?weight .
}

```

## Removing a Rule

```

DELETE
{
c:Rule1234 ?p ?o .
?o ?p2 ?o2 .
}
INSERT {
}
WHERE {
c:Rule1234 ?p ?o .
OPTIONAL
{?o ?p2 ?o2 . ?p2 rdfs:subPropertyOf c:RuleProperties . }
}

```

## Removing a Set of Rules

```

DELETE
{
?s ?p ?o .
?o ?p2 ?o2 .
}
INSERT {
}
WHERE {
?s <http://www...#facetMLNActionRet> <http://www...#MLNActionRetIncrease> .
?s ?p ?o .
OPTIONAL {?o ?p2 ?o2 . ?p2 rdfs:subPropertyOf c:RuleProperties .}
}

```