RDF and RDF Schema


Jouni Tuominen
Department of Computer Science
jouni.tuominen@aalto.fi
Outline

- RDF data model
- RDF syntax
- RDF Schema (RDFS)
- RDF(S) semantics
RDF data model
RDF data model and relational databases

- Information is often available as tables in relational databases or CSV files

- RDF is a set of triples
  - n-ary information can be represented as triples

- RDF is a data model: directed named graph
RDF data model: fundamental concepts

- Resources
  - Internationalized Resource Identifiers (IRI; generalization of URI)
  - Literals
  - Blank nodes
- Statements (triples)
- Graphs
- Datasets and quads
Internationalized Resource Identifiers
for identifying resources globally
Internationalized Resource Identifier

- Used for giving an identity to a resource, so that the resource can be described and referred to
- In RDF graphs, IRIs can be given to nodes (visualized typically as ovals) and arcs
  - IRI node can be a start or end node of an arc
  - The start node of an arc is either an IRI or blank node
  - The end node can be an IRI, blank node, or literal data
  - The arc always has an IRI
- An arc attaches a property with some value to a node
- Entity – attribute – value model
Global identifiers for resources

**URI: Uniform Resource Identifier**
- Identifier that conforms syntactically to some **URI scheme**
  - E.g., ftp, http, https, mailto, urn, oid, xmpp, ...
- (Note change in nomenclature: Universal -> Uniform)

**URL: Uniform Resource Locator**
- Specialization of URI that also describes its primary access mechanism (e.g., its network location in HTTP)

**URN: Uniform Resource Name**
- Specialization of URI that only specifies its name

**IRI: Internationalized Resource Identifier**
- Generalization of URI based on Unicode character set
- URL encoding not needed
Examples

**URI: Uniform Resource Identifier**
- http://dbpedia.org/resource/Helsinki

**URL: Uniform Resource Locator**
- http://www.aalto.fi/fi/research/
- http://www.ask.com/web?qsrc=1&o=0&l=dir&q=Capital+of+Finland&qo=serpSearchTopBox
- http://urn.fi/urn:isbn:978-952-10-4171-6

**URN: Uniform Resource Name**
- urn:isbn:978-952-10-4171-6

**IRI: Internationalized Resource Identifier**
3. Syntax Components

The generic URI syntax consists of a hierarchical sequence of components referred to as the scheme, authority, path, query, and fragment.

\[
\text{URI} = \text{scheme} :: "":"" \text{hier-part} ["\?" \text{query}] ["\#" \text{fragment}]
\]

\[
\text{hier-part} = "/" / \text{authority} \text{path-ebempty} \\
/ \text{path-absolute} \\
/ \text{path-rootless} \\
/ \text{path-empty}
\]

The scheme and path components are required, though the path may be empty (no characters). When authority is present, the path must either be empty or begin with a slash ("/") character. When authority is not present, the path cannot begin with two slash characters ("//"). These restrictions result in five different ABNF rules for a path (Section 3.3), only one of which will match any given URI reference.

The following are two example URIs and their component parts:

```
foc://example.com:8042/over/there?name=ferret#nose
```

```
scheme  \____________/\__________/ \__________/ \__________/ \\
authority \____________/ \__________/ \__________/ \\
path  \______________\ \__________/ \__________/ \\
query  \__________/ \__________/ \__________/ \\
fragment \______________\ \__________/ \\
```

**IRI: IETF RFC 3987**
URI schemes

- Particular syntactic types of URIs with an agreed interpretation
- Standardized by IANA Internet Assigned Numbers Authority
  - Tens of URI schemes are available:
    - ftp, http, mailto, urn, oid, xmpp, ...
- Semantic Web advocates the use of HTTP URI/IRIs (URLs)
  - HTTP URIs not only identify things but are addresses, too
  - Type URI in a browser and you get useful info back!
Literals

data values
Literals

Literal is data encoded as a string
• ”Suomi”, ”Last waltz in Paris”

Literal value can be accompanied with a XML language tag:
• ”Suomi”@fi, ”Last waltz in Paris”@en

Literal value can be accompanied with a datatype (XML Schema)
• "-5"^^xsd:integer, "4.2E9"^^xsd:double
• Abbreviated: -5, 4.2E9
• Default datatype: ”Suomi”^^xsd:string

Visualized typically as a rectangle in an RDF graph
Blank nodes

locally identified nodes
Nodes can also be ”blank nodes”

RDF graphs may have unique blank nodes (bnodes, anonymous nodes)

- Node ID used only locally in an RDF graph
- Can be represented in RDF syntax, e.g. (Turtle), as: _:name or []
  - _:a _:cat [] [ ... ]
- No need for an IRI for external reference

Blank nodes arise from embedded descriptions

- Systems can generate new distinct IRIs automatically (Skolemization)

Considered an annoying feature but needed, too

- Must be disambiguated when combining graphs
- Names may change when writing/reading graphs
Statements

asserting information
Statement

Statement asserts a relationship (property) between two resources
• E.g., ”Helsinki is part of Finland”

Statement is represented as a triple
• <resource, property, property_value>
  <subject, predicate, object>

RDF graph = set of statements
Statement characteristics

Subject is an IRI or blank node
Predicate is an IRI (blank node is not reasonable predicate)
Object is an IRI, literal, or blank node

• Literals are used only as property values
Are binary predicates enough?

RDF uses only binary properties

- Often we use predicates with more than 2 arguments

**Example problem: referee(X, Y, Z)**

- X is the referee in a chess game between players Y and Z

**N-ary predicates can be represented by binary ones:**

- a new auxiliary resource `chessGame`
- new binary predicates for arguments: `ref`, `player1`, and `player2`
RDF graphs

set of triples
RDF graph

RDF graph = set of triples (statements)

- \(<\text{start node, arc, end node}>\) i.e. \(<\text{subject, predicate, object}>\)

Multiple graphs can be merged with the union operation of set theory
RDF graph(s) as set(s) of triples

\[
G = \{<:r1, \text{dc:author}, \text{"Väinö Linna"}>, <:r1, \text{rdf:type}, \text{:Romaani}>, <:r1, \text{dc:title}, \text{"Tuntematon sotilas"}@fi>, <:r2, \text{dc:author}, \text{"Väinö Linna"}>, <:r2, \text{rdf:type}, \text{:Romaani}>, <:r2, \text{dc:title}, \text{"Täällä Pohjantähden alla"}@fi>\}
\]

romaani (Finnish) = novel
Datasets and quads

set of graphs
Datasets, graphs, and quads in RDF 1.1

• **Dataset** consists of a set of **RDF graphs**
  - Multiple **named graphs** and at most one **unnamed (default) graph**

• Graphs are encode sets of quads, where the 4th position is a **graph IRI**
  - `<http://example.org/spiderman>`
    `<http://www.perceive.net/schemas/relationship/enemyOf>`
    `<http://example.org/green-goblin>`
    `<http://example.org/graphs/spiderman>`.
  - If the 4th member is omitted, triple belongs to the default graph
Quads

Adding the graph information into a triple can be important

• Information modularization
  -  *E.g., restricting the search only to a specific graph*

• Representing provenience information
  -  *The origin of the statement, the date of the addition into the dataset*
  -  *Used, e.g., in the Google Knowledge Graph*
  -  *Facilitates the management of contents*
Serialization

Representing graph as linear text (string)
- E.g., in a file: reading and writing

**Alternative serializations for different needs**

1. Intuitive for humans to read/write
   - *N*-triples, *Notation 3*
   - *Turtle*
   - *TriG, N-Quads*

2. XML-interpretability for machines
   - *RDF/XML*
   - *Existing XML tools available*

3. For **web** programming
   - *JSON-LD*

4. Embedding in web pages
   - *RDFa*
   - *Publishing information for, e.g., search engines*
Intuitive for humans

“Turtle family of RDF languages”
N-Triples

Triple set is serialized in the following form:

subject1 predicate1 object1 .
subject2 predicate2 object2 .
...
IRIs are enclosed in angle brackets (<>): <iri>:

<http://example.org/product2>
<http://www.w3.org/1999/02/22-rdf-syntax#type>
<http://example.org/computer> .

• For machines easy to read/write line by line
• For humans difficult to read due to redundancy
Notation 3 (N3)

- Easier to read, compact way for serializing RDF information
- Developed by Tim Berners-Lee, however no W3C recommendation status
- Namespace prefixes are first introduced in the beginning of the file, e.g.
  ```
  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax#> .
  @prefix o: <http://example.org/> .
  ```

- Then the triples are encoded, e.g.
  ```
  o:product1 rdf:type o:computer .
  ```

- Subject doesn’t have to be repeated, e.g.
  ```
  o:product3 rdf:type o:computer ;
     o:brand o:apple .
  ```

- Blank nodes
  ```
  o:product4 rdf:type [ o:brand "Nokia" ] .
  ```

- Also other syntactic sugar available
Turtle – Terse RDF Triple Language

- Extends the N-Triples notation
- Subset of Notation 3, non-valid RDF extensions discarded

Notation 3 includes at least the following syntax that is not in Turtle (not a complete list):

1. { ... }
2. is of
4. @keywords
5. => implies
6. = equivalence
7. @forAll
8. @forSome
9. <=

- E.g., used in SPARQL query patterns
- Recommended human-readable RDF notation
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>.

<http://www.w3.org/People/EM/contact#me> rdf:type contact:Person;
    contact:fullName "Eric Miller";
    contact:mailbox <mailto:em@w3.org>;
    contact:personalTitle "Dr.".
Turtle – syntactic sugar

Example

```turtle
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
    rel:enemyOf <#spiderman> ;
    a foaf:Person ;  # in the context of the Marvel universe
    foaf:name "Green Goblin" .

<#spiderman>
    rel:enemyOf <#green-goblin> ;
    a foaf:Person ;
    foaf:name "Spiderman", "Человек-паук"@ru .
```
TriG

Extends Turtle notation for representing datasets (set of graphs)

# This document contains a default graph and two named graphs.

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

# default graph

{  
  <http://example.org/bob> dc:publisher "Bob" .
  <http://example.org/alice> dc:publisher "Alice" .
  }

<http://example.org/bob>  
{  
  :a foaf:name "Bob" .
  _:a foaf:mbox <mailto:bob@oldcorp.example.org> .
  _:a foaf:knows _:b .
  }

<http://example.org/alice>  
{  
  _:b foaf:name "Alice" .
  _:b foaf:mbox <mailto:alice@work.example.org> .
  }
N-Quads

Extends N-Triples notation for representing triples with graph information (line by line)

```xml
<http://one.example/subject1> <http://one.example/predicate1> <http://one.example/object1> <http://example.org/graph3> . # comments here
# or on a line by themselves
_:subject1 <http://an.example/predicate1> "object1" <http://example.org/graph1> .
_:subject2 <http://an.example/predicate2> "object2" <http://example.org/graph5> .
```
XML-interpretability for machines

RDF/XML
RDF/XML

- XML language for serializing RDF graphs
- Originally the only RDF syntax in the RDF 1.0 recommendation
- Meant for machines, complicated for humans
  - *Existing XML tools available*
Example of RDF/XML

```xml
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ex="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
                  dc:title="RDF/XML Syntax Specification (Revised)">
    <ex:editor rdf:nodeID="abc"/>
  </rdf:Description>

  <rdf:Description rdf:nodeID="abc"
                  ex:fullName="Dave Beckett">
  </rdf:Description>
</rdf:RDF>
```
For web programming

JSON-LD
JSON-LD (JSON Linked Data)

• Human-readable notation with built-in support in programming languages/environments, such as JavaScript, Python
• See also interactive JSON-LD ”playground”
Example of JSON-LD

http://json-ld.org/playground/index.html
Embedding in web pages

RDFa
RDFa: bridging the human and data webs

- Bridge by embedding RDF in XHTML
- Software can then extract the RDF graph for the machine
- See also interactive RDFa “playground”
Example of using RDFa

Literal Properties: RDFa lets Alice connect not just one URL to another—for example to connect her blog entry URL to the Creative Commons license URL—but also to connect one URL to a string such as "The Trouble with Bob". All arrows are labeled with the corresponding property name, which is also a URL.
Another example

```
<div typeof="foaf:Person" xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <p property="foaf:name">
    Alice Birmemswick
  </p>
  <p>
    Email: <a rel="foaf:mbox" href="mailto:alice@example.com">alice@example.com</a>
  </p>
  <p>
    Phone: <a rel="foaf:phone" href="tel:+1-617-555-7332">+1 617.555.7332</a>
  </p>
</div>
```
Embedded metadata on the Web: http://schema.org

• Used by Google, Yahoo, Bing, Yandex (RDFa, Microdata, Microformats)
• Web of (meta)data is coming ...
Real-life example

RDFa 1.1 Distiller and Parser

- [http://www.w3.org/2012/pyRdfa/](http://www.w3.org/2012/pyRdfa/)
- Try it, e.g., with Svenska YLE news:
Real-life example: web page

"Krav på egen laptop leder till ojämlikhet"

Hem och Skolans verksamhetsledare Micaela Romantschk-Pietilä tror att kravet på att nya gymnasieelever själva ska bekosta bärbar dator kan leda till att en del väljer bort gymnasiet.
Real-life example: extracted RDF

```
<?xml encoding="utf-8" version="1.0"?>
<rdf:RDF
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:siooc="http://siooc.org/siooc/types#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:vcard="http://www.w3.org/2001/XMLSchema#integer"
  xmlns:subj="http://siooc.org/siooc/ns#"/>

  <html:title xmlns="sv">Sluta göra beslut som trycker</html:title>
  <siooc:subject rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
  <!-- Content encoded: -->
```

I höst förutsäges alla som börjar gymnasiet att ha en egen bärbar dator, i och med den nya elektroniska studentexamen.

Mecia Romantschuk-Pietilä, som är verksamhetsledare för Helsinckis gymnasieskolor, tror att det kommer att vara ett ovanligt samband mellan framtiden och elevernas teknikförmåga.

---

```
<do:subject rdf:resource="http://www.yao.fi/mtco/koko/p15019/">
    <do:subject rdf:resource="http://www.yao.fi/mtco/koko/p1520/">
      <do:subject rdf:resource="http://www.yao.fi/mtco/koko/p63277/">
        <do:subject rdf:resource="http://www.yao.fi/mtco/koko/p6347/">
          <do:subject rdf:resource="http://www.yao.fi/mtco/koko/p18748/">
              </do:subject>
            </do:subject>
          </do:subject>
        </do:subject>
      </do:subject>
    </do:subject>
  </do:subject>
</do:subject>
```

---

**51**
RDF data validation

• The validity of different RDF syntaxes can be checked with validators
• As part of the validation the serialized representation can be visualized as an RDF graph
• W3C RDF/XML validator
• More validators
RDF Schema (RDFS)
Why RDF Schema?

Introducing classes and individuals (instances)
- A class is a set of individuals
  - E.g., John and Mary are individuals of class Person

Introducing constraints on using properties
- **Domain** constraint: only certain classes of individuals can have certain properties
  - E.g., only persons have a social security number
- **Range** constraint: certain properties may have value of a certain class only
  - E.g., a person’s parent must be a person, too

Introducing class and property hierarchies

Introducing semantics for validating data and for reasoning
Individuals

- Individual–class relationship is expressed by property `rdf:type`
- Classes are individuals of the (meta)class `rdfs:Class`
- `rdfs:Class` is an instance/individual of itself
Class hierarchies

Classes can be organized in hierarchies

- A is a **subclass** of B if every instance of A is also an instance of B
- Then B is a **superclass** of A

A class may have multiple superclasses

- Multiple inheritance
- A subclass graph is then not a tree
Class hierarchy: rdfs:subClassOf
Properties and constraints

- Properties are individuals of the class `rdf:Property`
- A property may have domain and/or range constraints, expressed by properties `rdfs:domain` and `rdfs:range`
- Constraints: Only humans may love only animals (of any kind)
- Inheritance reasoning:
  - Property hierarchy: Since humans love they also know animals
  - Class hierarchy: Polly (a bird and therefore an animal) can be loved by humans such as Mary
Example (Semantic Web Primer 2nd ed.)
RDF(S) core classes

- rdfs:Resource
- rdfs:Class
- rdfs:Literal
- rdf:Property
- rdf:Statement
Hierarchy of RDF(S) core classes

These classes form a hierarchy in the RDF(S) specifications written by RDFS itself!
RDF(S) core properties

- rdf:type
- rdfs:subClassOf
- rdfs:subPropertyOf
- rdfs:domain
- rdfs:range
Other constructs in RDF(S)

Reification mechanism
- For adding metadata to individual triples (statements)
- "John believes Mary loves Polly"
  - :s1 rdf:type rdfs:Statement;
    rdf:subject :Mary;
    rdf:predicate :loves;
    rdf:object :Polly .
    :John :believe :s1 .

Container classes for generic data structures
- rdf:Bag Bags
- rdf:Seq Sequences
- rdf:Alt Alternatives
- rdfs:Container (superclass of containers)
Other constructs in RDF(S) (2)

Utility properties

- **rdfs:label**: human-readable label
- **rdfs:comment**: for commenting
- **rdfs:seeAlso**: related explaining resource
- **rdfs:isDefinedBy**: subproperty of **rdfs:seeAlso**

There are also some other primitives in the specifications
RDF(S) specification in RDFS

Namespace IRIs of RDF and RDFS contain the specifications for 1) classes and 2) properties

- http://www.w3.org/1999/02/22-rdf-syntax-ns#
- http://www.w3.org/2000/01/rdf-schema#

```xml
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<http://www.w3.org/2000/01/rdf-schema#> a owl:Ontology ;
dc:title "The RDF Schema vocabulary (RDFS)" .

rdfs:Resource a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Resource" ;
rdfs:comment "The class resource, everything." .

rdfs:Class a rdfs:Class ;
rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
rdfs:label "Class" ;
rdfs:comment "The class of classes." ;
rdfs:subClassOf rdfs:Resource .
```
RDF(S) semantics
RDFS semantics

Based on first-order predicate logic
RDF data can therefore be used for reasoning new data
= Adding new triples in the graph

Two ways of defining the same semantics in logic

• Axiomatic semantics by representing RDF constructs in terms of logical axioms
• Direct inference rule-based semantics
  - Simpler way
Axiomatic semantics: an example (Semantic Web Primer)

- An RDF statement (triple) $(R, P, V)$ is represented as $\text{PropVal}(P, R, V)$
- $\text{Type}(R, T)$ is a shorthand for $\text{PropVal}(\text{type}, R, T)$
- $\text{subClassOf}$ is a property:
  $\text{Type}(\text{subClassOf}, \text{Property})$
- $\text{type}$ can be applied to resources and has a class as its value:
  $\text{Type}(?r, ?c) \rightarrow (\text{Type}(?r, \text{Resource}) \land \text{Type}(?c, \text{Class}))$
- If a class $C$ is a subclass of a class $C'$, then all instances of $C$ are also instances of $C'$:
  $\text{PropVal}(\text{subClassOf}, ?c, ?c') \leftrightarrow$
  $(\text{Type}(?c, \text{Class}) \land \text{Type}(?c', \text{Class}) \land$
  $\forall ?x \, (\text{Type}(?x, ?c) \rightarrow \text{Type}(?x, ?c')))$
Semantics based on inference rules

Semantics in terms of RDF triples instead of restating RDF in terms of first-order logic

- Sound and complete inference system
  - But no need for heavy first-order logic proof system (good for scalability)

Rule system consists of inference rules of the form:

IF E contains certain triples
THEN add to E certain additional triples

where E is an arbitrary set of RDF triples
Examples of inference rules

IF E contains the triple (?x, ?p, ?y)
THEN E also contains (?p, rdf:type, rdf:Property)

IF E contains the triples (?u, rdfs:subClassOf, ?v)
and (?v, rdfs:subclassOf, ?w)
THEN E also contains the triple (?u, rdfs:subClassOf, ?w)

IF E contains the triples (?x, rdf:type, ?u)
and (?u, rdfs:subClassOf, ?v)
THEN E also contains the triple (?x, rdf:type, ?v)
Examples of inference rules (2)

Any resource $?y$ which appears as the value of a property $?p$ can be inferred to be a member of the range of $?p$

- This shows that range definitions in RDF Schema are not used to restrict the range of a property, but rather to infer the membership of the range

IF E contains the triples (?x, ?p, ?y) and (?p, rdfs:range, ?u)

THEN E also contains the triple (?y, rdf:type, ?u)
Summary

- RDF provides a foundation for representing and processing metadata
- RDF has a graph-based data model
- RDF has different syntaxes
- RDF has a decentralized philosophy
  - Incremental building of knowledge
  - Sharing and reusing metadata
Summary (2)

• RDF is domain-independent
• RDF Schema provides a mechanism for describing specific domains
  - *RDF Schema is a primitive ontology language*
• Key concepts of RDF (Schema) are
  - *Classes and instances*
  - *Type and subclass relations for class hierarchies*
  - *Property and subproperty relations for property hierarchies*
  - *Domain and range restrictions connecting properties and classes*
**What’s next on this course?**

Lectures on Wednesdays at 10.15-12.00 in lecture hall 2534-2535 (TUAS)

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.01.2016</td>
<td>Introduction, WWW, and Semantic Web</td>
</tr>
<tr>
<td>20.01.2016</td>
<td>RDF and RDF Schema</td>
</tr>
<tr>
<td>27.01.2016</td>
<td>Linked Data and SPARQL</td>
</tr>
<tr>
<td>03.02.2016</td>
<td>Ontologies and SKOS</td>
</tr>
<tr>
<td>10.02.2016</td>
<td>Web Ontology Language OWL</td>
</tr>
<tr>
<td>24.02.2016</td>
<td>Logic and Inference: Rules</td>
</tr>
<tr>
<td>02.03.2016</td>
<td>Semantic Web Infrastructure and Applications</td>
</tr>
</tbody>
</table>

Lecture schedule (changes possible)

Assignment support sessions on Thursdays at 10.15-12.00 in computer class 1521-1522 (TUAS)

<table>
<thead>
<tr>
<th>Date</th>
<th>Support for</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.01.2016</td>
<td>Assignment 1</td>
</tr>
<tr>
<td>04.02.2016</td>
<td>Assignment 1</td>
</tr>
<tr>
<td>11.02.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>25.02.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>03.03.2016</td>
<td>Assignment 2</td>
</tr>
<tr>
<td>04.03.2016</td>
<td>Assignment 3</td>
</tr>
<tr>
<td>10.03.2016</td>
<td>Assignment 3</td>
</tr>
<tr>
<td>17.03.2016</td>
<td>Assignment 3</td>
</tr>
<tr>
<td>24.03.2016</td>
<td>Assignment 3</td>
</tr>
</tbody>
</table>

04.02. DL for Assignment 1

03.03. DL for Assignment 2

24.03. DL for Assignment 3