Modelling and Aggregating Social Network Data

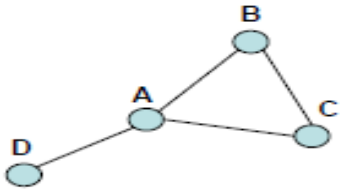
Module 3

Network Data Representation

- The most common kind of social network data can be modelled by a ***graph*** where the nodes represent individuals and the edges represent binary social relationships.
- A number of different, proprietary formats exist for serializing graphs and attribute data in machine- processable electronic documents.
- The most commonly encountered formats are ***Pajek*** and ***UCINET***.
- These are text-based formats and they can be easily edited using simple text editors.

- Researchers represented their data initially using ***Microsoft Excel spreadsheets***, which can be exported in CSV (Comma Separated Values) format.
- As this format is not specific to graph structures ,additional constraints need to be put on the content before the file is processed by graph packages.
- The visualization software packages also have proprietary formats such as the dot format used by the open source ***GraphViz*** package developed at AT&T Research.2

- The ***GraphML*** format represents an advancement in terms of both interoperability and extensibility.
- GraphML originates from the information visualization community where a shared format greatly increases the usability of new visualization methods.
- GraphML is based on XML with a schema defined in XML Schema.
- This has the advantage that GraphML files can be *edited, stored, queried, transformed etc. using generic XML tools.*




```
*Vertices 4
1 "A"
2 "B"
3 "C"
4 "D"
*Edges
1 1
1 2
1 3
1 4
2 3
```

```
dl
n = 4
labels embedded
format = edgelist
data:
A B
A C
A D
B C
```

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph id="G" edgedefault="undirected">
    <node id="a"/>
    <node id="b"/>
    <node id="c"/>
    <node id="d"/>
    <edge source="a" target="b"/>
    <edge source="a" target="c"/>
    <edge source="a" target="d"/>
    <edge source="b" target="c"/>
  </graph>
</graphml>
```

Figure 5.1. A simple graph (upper left) described in Pajek .NET, UCINET DL and GraphML formats.

- None of these formats support the *aggregation and reuse of electronic data*.
- One of the common reasons to use multiple data sources is to perform ***triangulation***
  - To use a variety of data sources and/or methods of analysis to verify the same conclusion.
- We need to recognize matching instances in the different data sources and merge the records before we can proceed with the analysis.
- These graph formats reduce social individuals and their relationships to nodes and edges, which is the only information required for the purposes of analyzing single networks

Ontological representation of social individuals

- The Friend-of-a-Friend (FOAF) ontology is an OWL based format for representing personal information and an individual's social network.
- FOAF started as an experimentation with Semantic Web technology.
- The idea of FOAF was to provide a machine processable format for representing the kind of personal information described in homepages of individuals that made the original Web successful.

- Thus FOAF has a vocabulary for describing personal attribute information found on homepages such as name and email address of the individual, projects, interests, links to work and school homepage etc.
- FOAF profiles can also contain a description of the individual's friends using the same vocabulary that is used to describe the individual himself.
- FOAF profiles can be linked together to form networks of web-based profiles.

- FOAF became the center point of interest in 2003 with the spread of Social Networking Services such Friendster, Orkut, LinkedIn etc.
- Drawbacks to centralized social networking services.
  - The information is under the control of the database owner who has an interest in keeping the information bound to the site and is willing to protect the data through technical and legal means.
  - Centralized systems do not allow users to control the information they provide on their own terms.

- FOAF profiles are created and controlled by the individual user and shared in a distributed fashion.
- FOAF profiles are posted on the personal website of the user and linked from the user's homepage with the HTML META tag.
- The distributed nature of FOAF networks means that FOAF requires a mechanism to link individual profiles and thus allow the discovery of related profiles.
- Related profiles can be discovered by crawling the FOAF network along these links, known as ***scutters*** (RDF crawlers) .
- FOAF profiles generated from user profiles at online communities only contain links to members of the same website.

- An advantage of FOAF in terms of sharing FOAF data is the relative stability of the ontology.
- The number of FOAF users means that the maintainers of the ontology are grateful to keep the vocabulary and its semantics stable.
- To facilitate adoption, terms are not added to the vocabulary any more, rather authors are encouraged to create extensions using the mechanisms of RDF, e.g. creating subclasses and sub properties and adding new properties to existing classes.

FOAF Basics
Agent
Person
name
nick
title
homepage
mbox
mbox_sha1sum
img
depiction (depicts)
surname
family_name
givenname
firstName

Personal Information
weblog
knows
interest
currentProject
pastProject
plan
based_near
workplaceHomepage
workInfoHomepage
schoolHomepage
topic_interest
publications
geekcode
myersBriggs
dnaChecksum

Online Accounts / IM
OnlineAccount
OnlineChatAccount
OnlineEcommerceAccount
OnlineGamingAccount
holdsAccount
accountServiceHomepage
accountName
icqChatID
msnChatID
aimChatID
jabberID
yahooChatID

Projects and Groups
Project
Organization
Group
member
membershipClass
fundedBy
theme

Documents and Images
Document
Image
PersonalProfileDocument
topic (page)
primaryTopic
tipjar
sha1
made (maker)
thumbnail
logo

Table 5.1. Classes and properties of the FOAF ontology.

Basic FOAF Profile

FOAF Profile Instructions

Fill out the fields directly below this text to complete a basic FOAF Profile. Optional items are available further down the page in the Tabs Section. These options include people you know, social networks to which you may belong, and other additional information described by the FOAF vocabulary.

Remember, as with all other online profiles, do not publish any information you wish to keep completely private. This is particularly applicable with informartion such as an email address - which most people do not want to give out in an effort to avoid spam.

Note: None of the information you enter in this generator is used or stored in any way. Your privacy can be assured because the file is processed using JavaScript and is entirely client-side.

Wish List Bookmarklet Instructions

In chapter 4 of the book we discuss connecting your FOAF Profile to a 'Wish List' of items you want. This is done via a Bookmarklet that gets added to your browser. To add that bookmarklet, enter your wish list link into the input box below and hit Enter. Then drag the link generated next to it into your bookmarks bar. It can then be used on the page of any product you want to add to your wish list by simply clicking it, copying the output, and adding it to your wish list.

Wish List Link: http://paprika.umw.edu/~marsha/starwars/wishlist.ttl

FOAF Profile Form

Base URI	http://paprika.umw.edu/~marsha/starwars/foaf.ttl
Title (Mr, Ms, Mrs, Dr, etc)	Jedi
First Name (Given)	Anakin
Last Name (Family/Surname)	Skywalker
Nickname	The Chosen One
Your Email Address	darthvader@example.com
Homepage	http://www.imdb.com/character/ch0000005/bio

Friends I Know | Social Networking | Additional Info

phone:
tel:8665551212
nickname:
The Chosen One
sha1sum of a personal mailbox URI name:
cc77937087f686e222bcf1194fb8c671d8591e00
homepage:
http://www.imdb.com/character/ch0000005/bio
title:
Mr.
gender:
Male
Given name:
Anikan
family_name:
Skywalker

Ontological representation of social relationships

- Ontological representations of social networks such as FOAF need to be extended with a framework for modelling and characterizing social relationships for two reasons:
  - To support the automated integration of social information on a semantical basis and
  - To capture established concepts in Social Network Analysis.

- Some of the characteristics of social relationships are.
- Sign:
  - A relationship can represent both positive and negative attitudes (like or hate).
  - The positive or negative charge of relationships is important for the study of balance within social networks, which is the subject of balance theory.

- Strength:
  - The Tie strength is a complex construct of several characteristics of social relations. (frequency, trust, complementary,)
- Provenance:
  - A social relationship may be viewed differently by the individual participants of the relationship, sometimes the degree that the tie is unreciprocated, i.e. perceived by only one member of the dyad.

- Relationship history:
  - Social relationships come into existence by some event involving two individuals.
  - Such an event may not require personal contact, but it has to involve social interaction.
  - From this event, social relationships begin a lifecycle of their own during which the characteristics of the relationship may change through interaction or the lack of.

- Relationship roles:
  - A social relationship may have a number of social roles associated with it known as relationship roles.
  - In a student/professor relationship within a university setting there is one individual playing the role of professor, while another individual is playing the role of a student.
  - Both the relationship and the roles may be limited in their interpretation and use to a certain social context .

- In the case of Web-based social networking services, a variety of ways of capturing relationship types and other attributes of relationships.
  - Orkut allows to describe the strength of friendship relations on a 5-point scale from “haven’t met” to “best friend”, while other sites may choose other scales or terms.
  - Further, various sites focus on very different kind of relationships and exchanges, e.g. LinkedIn differs from Orkut in focusing on professional exchanges.
- Ideally all users of these services would agree to single shared typology of social relations and shared characterization of relations

Conceptual model

- Social relations could be represented as n-ary predicates; however, n-ary relations are not supported directly by the RDF/OWL languages.
- There are several alternatives to n-ary relations in RDF/ OWL.
- To deal with n-ary relations we employ the technique that is known as reification: represent the relation as a class, whose instances are concrete relations of that type.

- RDF itself has a reified representation of statements: the rdf :Statement object represents the class of statements.
- This class has *three* properties that correspond to the components of a statement, *rdf :subject, rdf :predicate, rdf :object*.
- These properties are used to link the statement instance to the resources involved in the statement.

- Relationships become subclasses of the rdf :Statement class.
- New Relationship class is related to a general Parameter class by the ***hasParameter*** relationship.
- Relationship types such as Friendship are subclasses of the Relationship class, while their parameters (such as strength or frequency) are subtypes of the Parameter class.

- The two alternatives differ in the representation of parameters.
  - The first scheme borrows from the design of OWL-S for representing service parameters, as used in the specification of the profile of a Web Service.
  - The second alternative differs in that the “native” method of RDF is used for representing parameters: the generic Parameter class is defined as a subclass of rdf :Property.

- This model has the advantage that it becomes more natural to represent parameter values and restrictions on them.
- The disadvantage is that this solution is not compliant with OWL DL: declaring properties ranging on properties and creating subclasses rdf :Property are not allowed in this species of OWL.

- ***Cognitive structuring*** works by applying the generic pattern we associate with such a relationship to the actual state-of-affairs we observe.
- For example, a student/professor relationship at the Free University of Amsterdam is defined by the social context of the university and this kind of relationship may not be recognizable outside of the university.

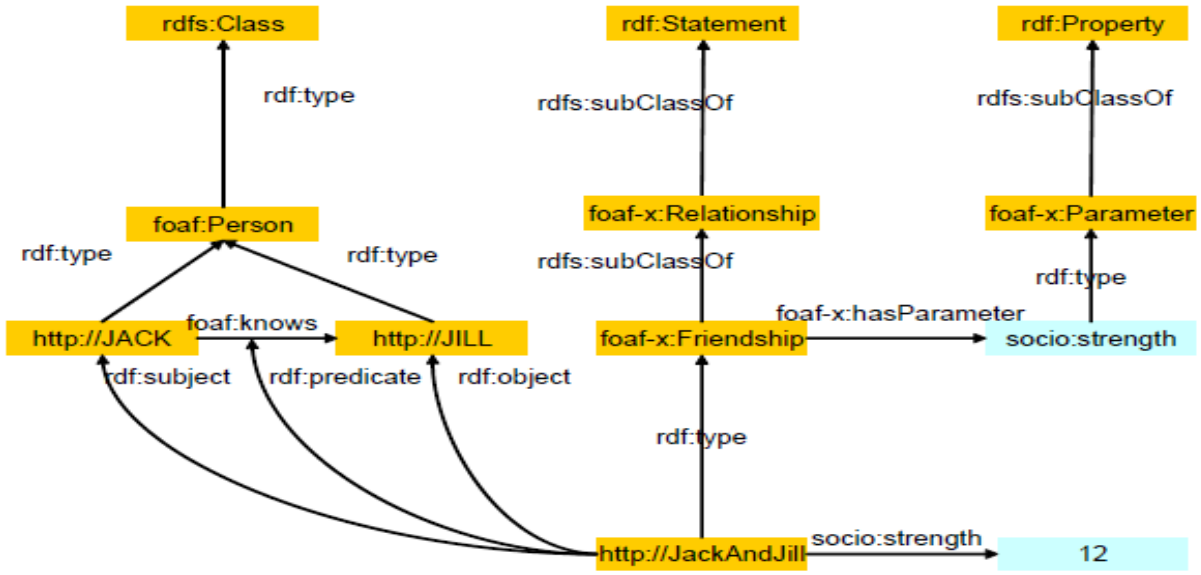


Figure 5.4. An alternative RDF representation of social relationships.

- The representation of context and the separation of the level of state-of-affairs (observations of objects and sequences of events) from the higher level of descriptions (contexts) that can be used to interpret those state-of-affairs turns out to be a common problem in the representation of much of human knowledge.
- A solution is the Descriptions and Situations ontology design pattern that provides a model of context and allows to clearly delineate these two layers of representation

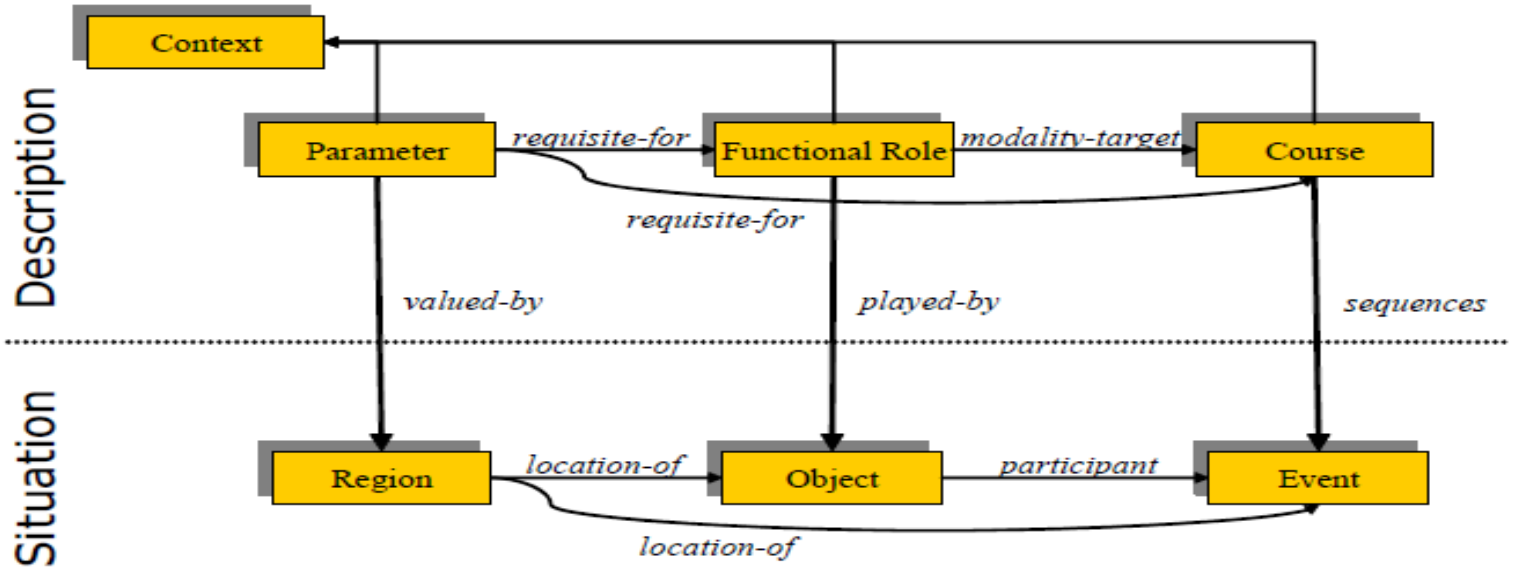


Figure 5.5. The Descriptions and Situations ontology design pattern.

- D&S is a generic pattern for modelling non-physical objects whose intended meaning results from statements, i.e. it emerges in combination with other entities.
  - For example, a norm, a plan, or a social role is usually represented as a set of statements and not as a concept.
  - Non-physical objects may change and be manipulated similar to physical entities, and are treated as first-order objects.
- D & S has been successfully applied in a wide range of real-life ontology engineering projects from representing Service Level Agreements (SLAs) to the descriptions of Web Services

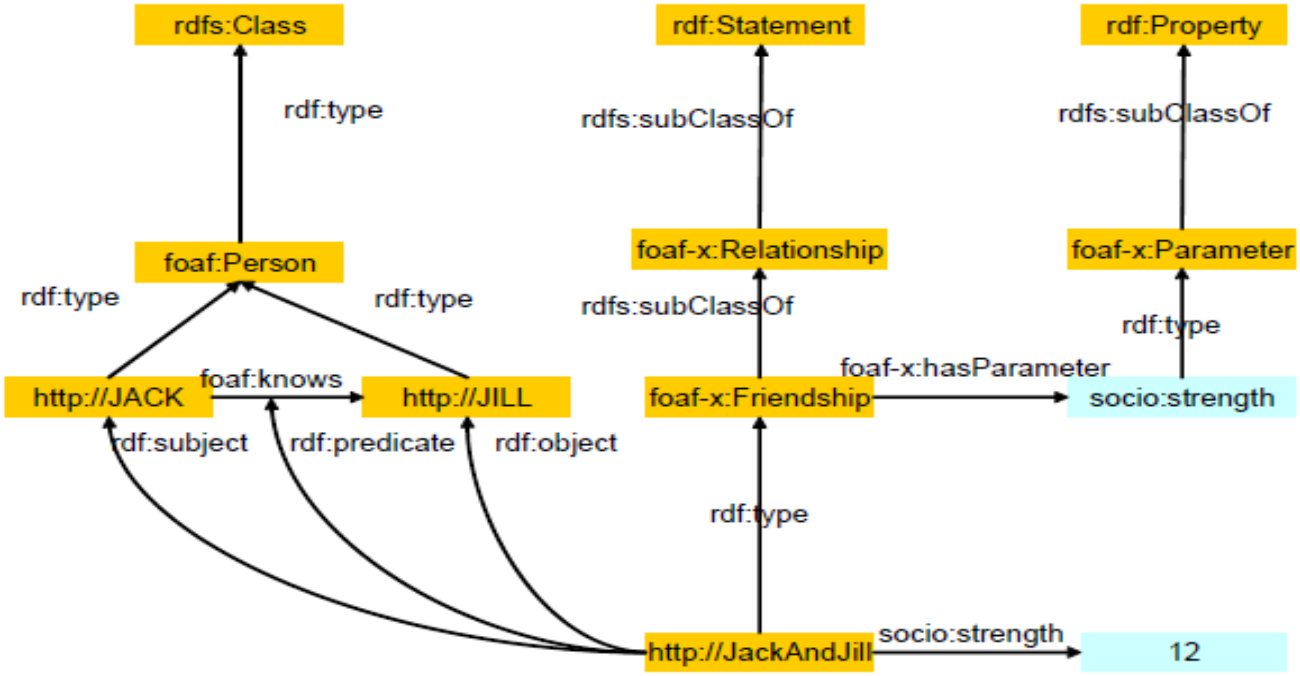


Figure 5.4. An alternative RDF representation of social relationships.

Aggregating and Reasoning with Social Network Data

- Our first step is to convert some data sets in traditional formats (relational databases, Excel sheets, XML files etc.) into an RDF-based syntax, which allows to store the data in an ontology store and manipulate it with ontology-based tools.
- Then apply *ontology mapping* to unify our data on the schema level by mapping classes (types) and properties from different schemas to a shared ontology such as FOAF

- It includes set of processes like
  1. Assign identifiers to resources
  2. On the notion of equality
  3. Determining equality
  4. Reasoning with instances
  5. Evaluating Smushing

1. Representing Identity

- One of the main advantages of RDF over other UML is the possibility to uniquely identify resources (instances, classes or properties).
- The primary mechanism for this is the assignment of URIs to resources.
- Every resource, except blank nodes is identified by a URI.
- For example, in the publishing , Standard schemes such as ISBN and ISSN numbers for books and periodicals and DOIs (Digital Object Identifiers) are widely adopted, while each publisher and online repository also maintains its own identifier scheme.

- Further, publications that are accessible online can be represented by their URL.
- All of these identifiers are *unique* (an identifier identifies a single resource), but mostly not *single* (a resource may be identified by multiple identifiers).
- This is the case with URLs (the same publication may be posted at multiple URLs), but it happens even with DOIs whose registration is centralized that the same publication is assigned two or more identifiers.

- Multiple identifiers can be represented in RDF in two separate ways.
- First, one can introduce a separate resource and use the identifiers as URIs for these resources.
- Once separate resources are introduced for the same object, the equality of these resources can be expressed using the owl:sameAs property.
- The other alternative is to chose one of the identifiers and use it as a URI.

- The first criterion means that good URIs should be unambiguous or at least should be chosen such that it is unlikely that someone else would use the same URI for something different.
- This is important because resources with the same URI but different intended meanings are likely to result in inconsistencies (a URI clash).
- The second criterion is also important because there is no way to rename resources (to reassign URIs).
- Once a URI changes the only solution is to introduce a new resource and assert its equivalence with the old resource.

2. On the Notion of Equality

- RDF and OWL (starting with OWL Lite) allow us to identify resources and to represent their (in)equality using the owl:sameAs and owl:differentFrom properties.
- These are only the basic building blocks for defining equality.
- The meaning of equality with respect to the objects of the domain depends on the domain itself and the goals of the modelling.

- The most well-known formulation of equality was given by Wilhelm Gottfried Leibniz in his *Discourse on Metaphysics*.
- The Identity of *Indiscernible* or *Leibniz-law* can be phrased as the principle of all things being identical unless we are able to distinguish (discern) them or put it differently: no two distinct objects have exactly the same properties.
- The converse of the Leibniz-law is called *Indiscernibility of Identicals*
- The two laws taken together (often also called Leibniz-law) are the basis of the definition of equality in a number of systems.

Leibniz-law

$$\forall P : P(x) \leftrightarrow P(y) \rightarrow x = y$$

$$\forall P : P(x) \leftrightarrow P(y) \leftarrow x = y$$

- This quantification provides the Leibniz-law different interpretations in open and closed worlds.
- In an open world the number of properties is unknown and thus the Leibniz-law is not useful in practice: we can never conclude that two resources are equal since we can never be certain whether there is another property out there that could allow us to distinguish them.
- In a closed world we can possibly iterate over all properties to check if two resources are equal; if we can distinguish them by some property we can assume they are equal.

- The reflexive, symmetric and transitive properties of equality follow from these definitions.
- The reflexive, symmetric and transitive properties of sameAs also follow:

$$\forall s : (s, owl{:}sameAs, s)$$

$$(s_1, owl{:}sameAs, s_2) \rightarrow (s_2, owl{:}sameAs, s_1)$$

$$(s_1, owl{:}sameAs, s_2) \wedge (s_2, owl{:}sameAs, s_3) \rightarrow (s_1, owl{:}sameAs, s_3)$$

3. Determining equality

- In OWL there are a limited set of constructs that can lead to (in)equality statements.
- *Functional* and *inverse functional properties* (IFPs) and *maximum cardinality restrictions* lead to conclude that two symbols must denote the same resource.
- Example1, the foaf :mbox property denoting the email address of a person is inverse-functional as a mailbox can only belong to a single person.

- Example 2: Consider a hypothetical *ex:hasParent* property, which has a maximum cardinality of two.
- If we state that a single person has three parents (which we are allowed to state) an OWL reasoner should conclude that at least two of them has to be the same.
- Once inferred, the equality of instances can be stated by using the *owl:sameAs* property.
- To conclude that two symbols do not denote the same object, which is expressed by the *owl:differentFrom* property.
- Example: Instances of disjoint classes must be different from each other.

4 Reasoning with instance equality

- Reasoning is the inference of new statements (facts) that necessarily follow from the set of known statements.
- What follows from a set of statements is determined by the semantics of the language.
- Every fact we add to our knowledge base has a meaning in the sense that it represents some constraint on the mapping between the symbols of our representation and some domain of interpretations.
- Every piece of additional knowledge excludes some unintended interpretation of our knowledge base.

- Description Logic versus Rule-Based Reasoners
  - Description Logic reasoners are designed to support primary tasks of classification and the consistency checking of ontologies.
  - In the case of equality reasoning this means that if we want to check whether two instances are necessarily the same, we add the opposite to the ontology and check for inconsistency.
  - If the reasoner signals inconsistency, we can be certain that the instances are the same and add that to our knowledge base.

- Description logic reasoning is very inefficient in practice as we need to perform a consistency check for every pair of instances in the ontology.

- Rule-based reasoning : The semantics of RDF(S) can be completely expressed using a set of inference rules.

- Forward versus backward chaining
  - *Forward chaining:* All consequences of the rules are computed to obtain what is called a complete materialization of the knowledge base.
  - This is done by repeatedly checking the prerequisites of the rules and adding their conclusions until no new statements can be inferred.
  - Advantage: queries are fast to execute since all true statements are readily available.
  - Disadvantage: storing the complete materialization often takes excessive amounts of space as even the simplest RDF(S) ontologies result in a large amount of inferred statements

- *Backward-chaining*, rules are executed "backwards" and on demand, i.e. when a query needs to be answered.
- While with forward chaining we check the prerequisites of rules and add all their conclusions, here we are given a conclusion and check whether it is explicitly stated or whether it could inferred from some rule, either because the prerequisites of the rule are explicitly stated to be true or again because they too could be inferred from some other rule(s).
- The drawback of backward chaining is longer query execution times.

5. Evaluating Smushing

- Smushing can be considered as either a retrieval problem or a clustering problem.
- In terms of retrieval, we try to achieve a maximum precision and recall of the theoretically correct set of mappings.
- In ontology mapping where there is typically a one-to-one mapping between elements of two ontologies.
- In the instance unification problem a single resource may be mapped to a number of other resources.
- Thus we can conceptualize this task as clustering and evaluate our clusters against the ideal clustering.

- Instance unification or smushing can be considered as an optimization task where we try to optimize an information retrieval or clustering-based measure
- Instance unification is “easier” than ontology mapping.
- In ontology mapping a local choice to map two instances may limit our future choices and thus we run the risk of ending up in a local minimum.

- In case of smushing, we only have to be aware that in cases where we have both positive and negative rules (owl:sameAs and owl:differentFrom) there is a possibility that we end in an inconsistent state (where two resources are the same as well as different).

Questions

1. Describe the ontological representation of social individuals.
2. Describe the ontological representation of social relationships.
3. Explain how the reasoning with instance equality is done in social network data?
4. What is meant by Evaluating Smushing?
5. List and explain some tools/ software used to represent network data
6. Describe how aggregating and reasoning can be done on social network data.