



Aalto University
School of Science



Web Ontology Language OWL

Background and Context

Eero Hyvönen

Aalto University, Department of Computer Science

University of Helsinki, HELDIG-centre

Semantic Computing Research Group (SeCo), <http://seco.cs.aalto.fi>

eero.hyvonen@aalto.fi

Learning objective

- Learn Web Ontology Language OWL

Contents

- OWL: background and context
- The OWL language

OWL: Background and Context

Component Technologies and Tools for the Semantic Web

Languages

- *Data exchange language:* *RDF*
- *Vocabulary/schema languages:* *SKOS, OWL*
- *Data/ontology query language:* *SPARQL*
- *Rules for reasoning:* *RIF, SWRL, SPIN, ...*

Storages and querying

- Triplestore systems (Fuseki, Sesame, Redland, Virtuoso, ...)
 - <http://en.wikipedia.org/wiki/Triplestore>

Development tools

- Ontology editors
 - *Protégé* <https://protege.stanford.edu/>
 - *TopBraid Composer* <https://www.topquadrant.com/topbraid-composer-install/>
- Software development tools
 - *Java: Apache Jena* <https://jena.apache.org/>
 - *Python: RDFLib* <https://pypi.org/project/rdflib/>

What is OWL

- Standard language (W3C) for representing vocabularies/ontologies/schemas
- Much richer than RDF Schema and SKOS (W3C recommendations for light-weight vocabularies)
- Original OWL
 - *Published as W3C recommendation on 10.2.2004*
- OWL 2 = OWL
 - *Latest W3C recommendation on 11.12.2012*
 - *Extends and replaces the old recommendation*

Requirements for Ontology Languages

- **Syntax**, convenience for expressing knowledge
- Formal **semantics**
- Sufficient **expressive** power
- Efficient **reasoning** support

OWL Syntaxes

- Based on RDF(S)
 - *Turtle and RDF/XML*
- Specific OWL/XML schema
- More user-friendly notations
 - *Functional-style syntax* *(for specifications)*
 - *Manchester syntax* *(simple notation for non-logicians)*

Formal Semantics

- Defines precise meaning for syntactic expressions
- As with RDF Schema two alternative semantics are provided
 - *Direct semantics (formal logic formulation)*
 - *RDF-based semantics (rules for inferring new triples)*
- Basis for interpreting expressions in reasoning:

`:p rdfs:range :D .`

`:x :p :y .`

\Rightarrow `:y rdf:type :D .`

- Goal: OWL can be used without being a formal logician

Sufficient Expressive Power

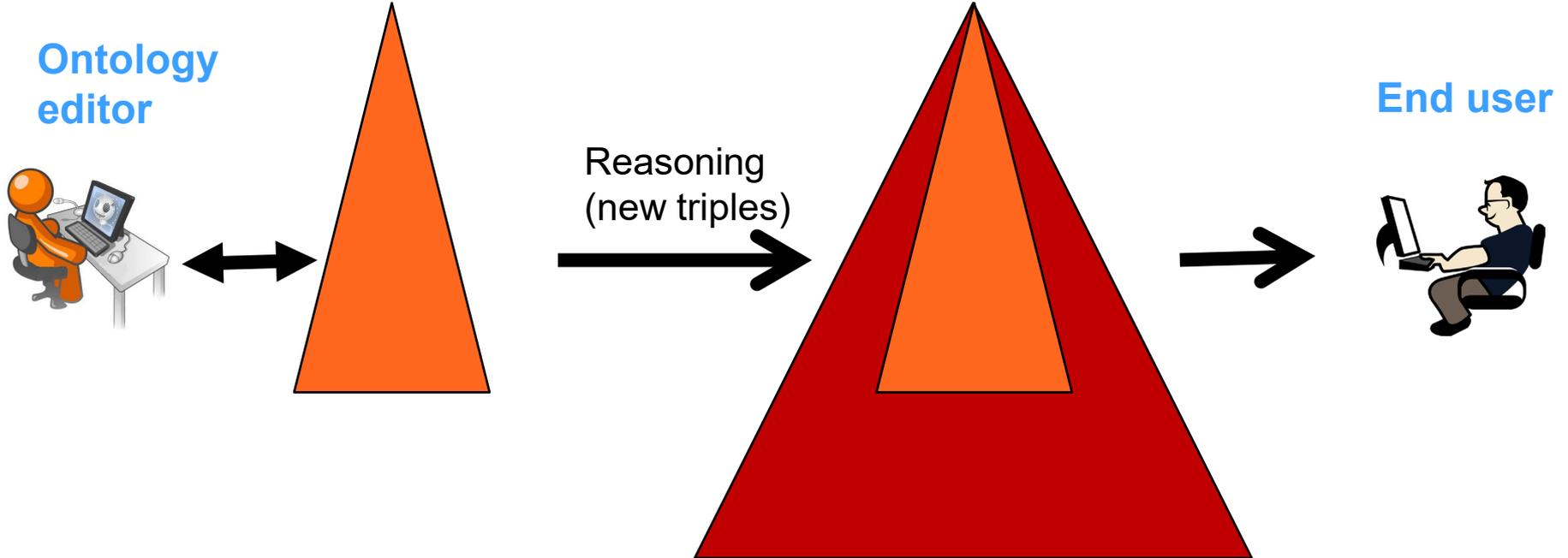
- RDF(S) semantics is limited
 - *Class membership (instance-class relations)*
 - *Class and property hierarchies*
 - *Domain and range of properties*
- OWL introduces lots of new features, e.g.
 - *Equivalence (classes) & Equality (individuals)*
 - *Disjointness (classes) & Difference (individuals)*
 - *Boolean combination of classes*
 - *Local (class-wise) scope of properties*
 - *Special relational characteristics of properties*
 - *Cardinality of properties*

Reasoning Support

- Enriching ontology and metadata with new facts
- Checking consistency
- Finding unintended relations
- Tradeoff between expressive power and efficient reasoning
 - Different versions “profiles” of OWL are available

Enriching Ontology by Reasoning

Ontology edited, (small, simple) Enriched Ontology (larger, more complex)



Protégé Editor for Editing Ontologies

OWL constructs

Plugins

The screenshot displays the Protégé editor interface for an ontology. The browser address bar shows the URL: `families (http://example.com/owl/families) : [/common/home/jwtuomin/local/svn/teaching/semweb/2016/lectures/2016-02-10-OWL/family-owl-2-demo.owl]`. The menu bar includes File, Edit, View, Reasoner, Tools, Refactor, Window, and Help. The toolbar contains various icons for editing and navigation. The main workspace shows a class hierarchy on the left and a class diagram on the right. The class hierarchy is rooted at `Thing` and includes classes such as `Adult`, `Dead`, `Female`, `Grownup`, `HappyPerson`, `Human`, `JohnsChildren`, `MyBirthdayGuests`, `NarcisticPerson`, `Orphan`, `Parent`, `Person`, `Teenager`, `YoungChild`, `Woman`, `Man`, `ChildlessPerson`, `ChildlessPerson`, `Teenager`, `Parent`, `Man`, `Mother`, `Father`, `Grandfather`, `SocialRole`, `Teenager`, and `YoungChild`. The class diagram shows the relationships between these classes, with many instances of `is-a` (subclass) relationships. The `Thing` class is the root, and many other classes inherit from it. The `Person` class is a subclass of `Human`, and `Human` is a subclass of `Person`. The `Parent` class is a subclass of `Human`, and `Man` and `Woman` are subclasses of `Parent`. The `ChildlessPerson` class is a subclass of `Parent`. The `Teenager` class is a subclass of `Parent`. The `YoungChild` class is a subclass of `Parent`. The `Grandfather` class is a subclass of `Parent`. The `HappyPerson` class is a subclass of `Person`. The `SocialRole` class is a subclass of `Person`. The `Dead` class is a subclass of `Person`. The `Orphan` class is a subclass of `Person`. The `NarcisticPerson` class is a subclass of `Person`. The `JohnsChildren` class is a subclass of `Person`. The `Female` class is a subclass of `Person`. The `Grownup` class is a subclass of `Person`. The `Adult` class is a subclass of `Person`. The `Reasoner active` checkbox is checked, and the `Show Inferences` checkbox is also checked.

<https://protege.stanford.edu/>

TopBraid Composer

- Commercial product with a free edition option
- SPIN rules for reasoning
- SPARQL querying

The screenshot displays the TopBraid Composer interface for editing an ontology. The main window is titled 'TopBraid - Family/Family-owl-2-demo.ttl - TopBraid Composer FE'. The interface is divided into several panes:

- Classes:** A tree view on the left showing the ontology's class hierarchy. The class 'Father (1)' is selected and highlighted in orange.
- Resource Form:** The central pane shows the 'Father' resource form. It includes fields for 'Name' (set to 'Father'), 'Annotations', and 'Other Properties'. The 'SocialRole' property is expanded to show 'Man' and 'Parent' as subclasses.
- Properties:** A list of properties on the right, with 'hasSSN' highlighted in orange.
- Instances:** A table at the bottom shows instances of the selected class. The instance 'John' is highlighted, with its type 'owl:NamedIndividual' and the property 'hasChild' visible.

[Resource]	rdf:type	rdfs:label	rdfs:comment
John	owl:NamedIndividual	hasChild	

<https://www.topquadrant.com/topbraid-composer-install/>

OWL 2 Web Ontology Language Document Overview (Second Edition)

W3C Recommendation 11 December 2012

This version:

<http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>

Latest version (series 2):

<http://www.w3.org/TR/owl2-overview/>

Latest Recommendation:

<http://www.w3.org/TR/owl-overview>

Previous version:

<http://www.w3.org/TR/2012/PER-owl2-overview-20121018/>

Editors:

W3C OWL Working Group (see [Acknowledgements](#))

Please refer to the [errata](#) for this document, which may include some normative corrections.

A [color-coded version of this document showing changes made since the previous version](#) is also available.

This document is also available in these non-normative formats: [PDF version](#).

See also [translations](#).

Copyright © 2012 W3C[®] (MIT, ERCIM, Keio), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents.

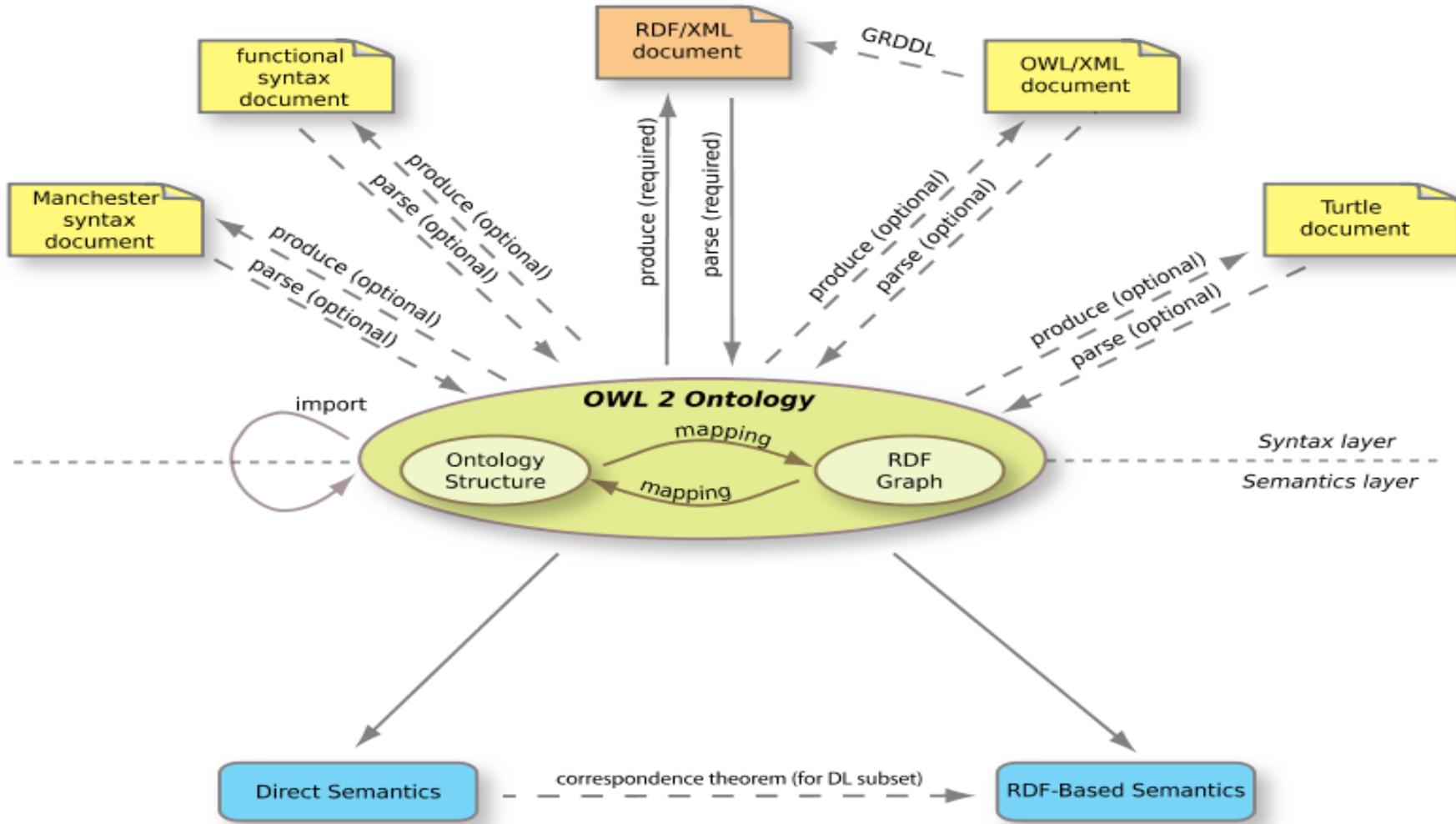


Figure 1. The Structure of OWL 2

Summary

- Background and context of OWL was specified
- See next lecture for learning the OWL language