Introduzione agli Open Data
Dati a 4 e 5 stelle

Cristiano Longo
longo@dmi.unict.it

Università di Catania

Dati a 4 e 5 stelle

Riportiamo le definizioni dell'Agenzia per l'Italia Digitale di open data a quattro e 5 stelle:

- *quattro stelle* - Dati con caratteristiche del livello precedente (licenze aperte, formato aperto, disponibili e interpretabili dalle macchine) ma esposti usando gli standard W3C RDF e SPARQL (tecnologie del *Web Semantico*);
- *cinque stelle* - Dati con caratteristiche del livello precedente ma collegati a dati esposti da altre persone e organizzazioni (*Linked Open Data*).

Dati a 4 e 5 stelle

Riportiamo le definizioni dell'Agenzia per l'Italia Digitale di open data a quattro e 5 stelle:

- *quattro stelle* - Dati con caratteristiche del livello precedente (licenze aperte, formato aperto, disponibili e interpretabili dalle macchine) ma esposti usando gli standard W3C RDF e SPARQL (tecnologie del *Web Semantico*);
- *cinque stelle* - Dati con caratteristiche del livello precedente ma collegati a dati esposti da altre persone e organizzazioni (*Linked Open Data*).

Il Web Semantico

Il *Web Semantico* nasce per associare informazioni *strutturate* alle pagine web, che solitamente sono composte da testo libero.[1]

> *[. . .] the Semantic Web approach instead develops languages for expressing information in a machine processable form.*

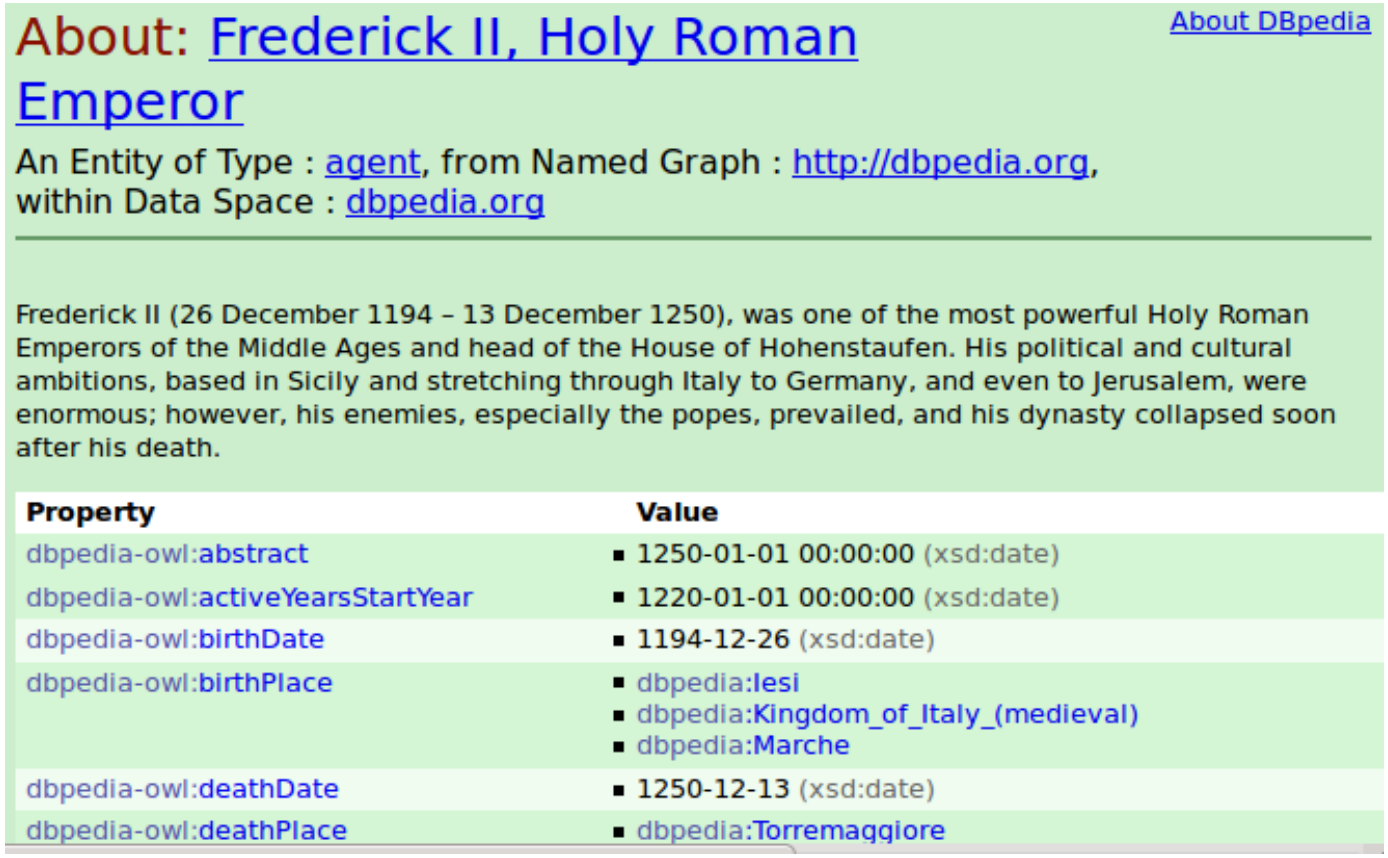
Semantic Web Roadmap, Tim Berners-Lee, 1998.

I *linguaggi di rappresentazione* usati nel Web semantico hanno una *sintassi rigorosa* e sono dotati di una *semantica formale*.

[1] Vedi *Semantic Web Roadmap*, Tim Berners-Lee, 1998.

Il Web Semantico - Esempio

Riportiamo la visualizzazione dell'entità corrispondente a Federico II di Svevia su `dbpedia.org`, il corrispondente semantico di *wikipedia.org*.



About: Frederick II, Holy Roman Emperor

About DBpedia

An Entity of Type : agent, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

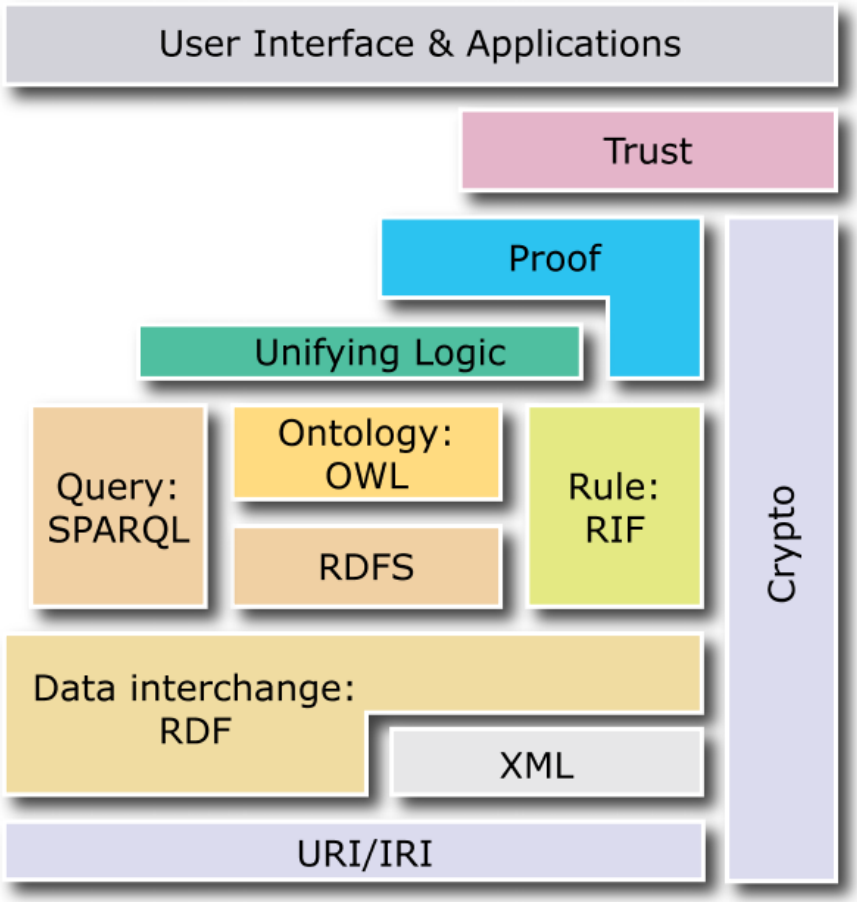
Frederick II (26 December 1194 – 13 December 1250), was one of the most powerful Holy Roman Emperors of the Middle Ages and head of the House of Hohenstaufen. His political and cultural ambitions, based in Sicily and stretching through Italy to Germany, and even to Jerusalem, were enormous; however, his enemies, especially the popes, prevailed, and his dynasty collapsed soon after his death.

Property	Value
dbpedia-owl:abstract	1250-01-01 00:00:00 (xsd:date)
dbpedia-owl:activeYearsStartYear	1220-01-01 00:00:00 (xsd:date)
dbpedia-owl:birthDate	1194-12-26 (xsd:date)
dbpedia-owl:birthPlace	dbpedia:Iesi dbpedia:Kingdom_of_Italy_(medieval) dbpedia:Marche
dbpedia-owl:deathDate	1250-12-13 (xsd:date)
dbpedia-owl:deathPlace	dbpedia:Torremaggiore

Figure : Federico II su dbpedia.org

Il Web Semantico - Tecnologie

L'insieme delle tecnologie usate nel Web Semantico costituiscono il cosiddetto *Semantic Web Stack*.



Il Web Semantico è costituito dall'insieme dei dataset codificati ed esposti attraverso queste tecnologie.

Il Web Semantico - Linked Open Data (1/2)

Nel Web Semantico gli oggetti (concreti o astratti) sono individuati attraverso *URI*.[2] Questo permette di fare riferimento alla stessa entità in dataset differenti.

Il *Linked Open Data Cloud* contiene presenti 365 dataset (fonte `http://stats.lod2.eu/`) collegati tra loro.

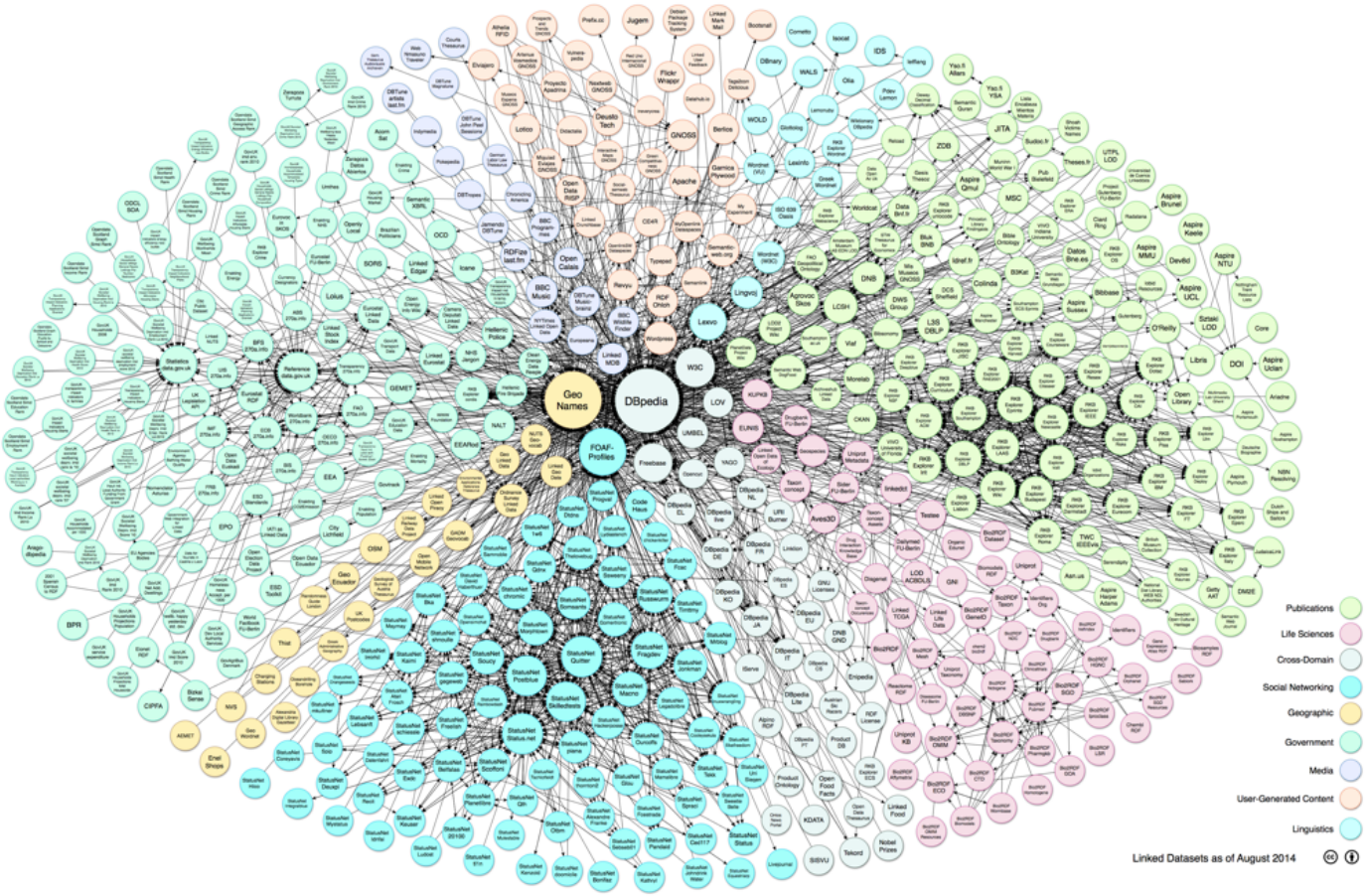


Figure : Linked Open Data Cloud

Il Web Semantico - Linked Open Data (2/2)

Alcuni dataset presenti nel Linked Open Data Cloud sono:

- *DBPedia* (`dbpedia.org`) corrispondente a `wikipedia.org`;
- *Linked Movie Database* (`http://linkedmdb.org/`) controparte sul Web Semantico di *Internet Movie Database* (`http://www.imdb.com/`);
- *Linked GeoData* (`http://linkedgeodata.org`) contiene i dati di *OpenStreetMap* (`http://www.openstreetmap.org/`);
- *AGROVOC* (`http://aims.fao.org/agrovoc`) è il dataset della FAO (`http://fao.org`);
- *Europeana* (`http://pro.europeana.eu/linked-open-data`) contiene dati su beni culturali e tradizioni Europee.

Ontologie

I linguaggi di rappresentazione usati nel Web Semantico si basano tutti sulla nozione di *ontologia*, mutuata dall'ambito dei sistemi di rappresentazione della conoscenza.

Una *ontologia* è una descrizione *parziale* del mondo. Essa è costituita da un insieme finito di *affermazioni* dei seguenti tipi:

Constraints: impongono dei vincoli *semantici* sul dominio di conoscenza che si va a rappresentare. La notazione richiama quella insiemistica;

$$HumanBeing \sqsubseteq Mortal$$

Property Assertions: impongono una relazione tra due elementi del dominio;

$$Alice\ motherOf\ Bob$$

Class Assertions: indicano l'appartenenza di un elemento ad un insieme.

$$HumanBeing(Socrate)$$

Ontologie

I linguaggi di rappresentazione usati nel Web Semantico si basano tutti sulla nozione di *ontologia*, mutuata dall'ambito dei sistemi di rappresentazione della conoscenza.

Una *ontologia* è una descrizione *parziale* del mondo. Essa è costituita da un insieme finito di *affermazioni* dei seguenti tipi:

Constraints: impongono dei vincoli *semantici* sul dominio di conoscenza che si va a rappresentare. La notazione richiama quella insiemistica;

$$HumanBeing \sqsubseteq Mortal$$

Property Assertions: impongono una relazione tra due elementi del dominio;

$$Alice\ motherOf\ Bob$$

Class Assertions: indicano l'appartenenza di un elemento ad un insieme.

$$HumanBeing(Socrate)$$

Ontologie

I linguaggi di rappresentazione usati nel Web Semantico si basano tutti sulla nozione di *ontologia*, mutuata dall'ambito dei sistemi di rappresentazione della conoscenza.

Una *ontologia* è una descrizione *parziale* del mondo. Essa è costituita da un insieme finito di *affermazioni* dei seguenti tipi:

Constraints: impongono dei vincoli *semantici* sul dominio di conoscenza che si va a rappresentare. La notazione richiama quella insiemistica;

$$HumanBeing \sqsubseteq Mortal$$

Property Assertions: impongono una relazione tra due elementi del dominio;

$$Alice\ motherOf\ Bob$$

Class Assertions: indicano l'appartenenza di un elemento ad un insieme.

$$HumanBeing(Socrate)$$

Ontologie

I linguaggi di rappresentazione usati nel Web Semantico si basano tutti sulla nozione di *ontologia*, mutuata dall'ambito dei sistemi di rappresentazione della conoscenza.

Una *ontologia* è una descrizione *parziale* del mondo. Essa è costituita da un insieme finito di *affermazioni* dei seguenti tipi:

Constraints: impongono dei vincoli *semantici* sul dominio di conoscenza che si va a rappresentare. La notazione richiama quella insiemistica;

$$HumanBeing \sqsubseteq Mortal$$

Property Assertions: impongono una relazione tra due elementi del dominio;

$$Alice\ motherOf\ Bob$$

Class Assertions: indicano l'appartenenza di un elemento ad un insieme.

$$HumanBeing(Socrate)$$

Ontologie - Sintassi

Riportiamo la definizione formale per la sintassi delle ontologie.

Siano N_C, N_P, N_I tre insiemi infiniti, numerabili e a due a due disgiunti di nomi di *classe*, *proprietà* e *individuo*, rispettivamente.

Una *ontologia* è un insieme finito di asserzioni dei seguenti tipi:

(Constraints)	$C \sqsubseteq D$
	$P \sqsubseteq Q$
	$\mathrm{dom}(P) \sqsubseteq C$
	$\mathrm{range}(P) \sqsubseteq C$
(Class Assertions)	$C(a)$
Property Assertions	$a\,P\,b$ (equivalente $P(a, b)$)

dove $C, D \in N_C$, $P, Q \in N_P$ e $a, b \in N_I$.

Si noti che la grammatica per i vincoli ivi riportata è *minimale*. Esistono linguaggi di rappresentazione che permettono di esprimere vincoli più complessi.

Ontologie - Sintassi

Segue una ontologia presentata a titolo di esempio

$$\begin{aligned}\mathcal{O} = \quad \{ \quad & \mathit{Woman} \sqsubseteq \mathit{Human}, \\ & \mathit{Man} \sqsubseteq \mathit{Human}, \\ & \mathit{Woman} \sqsubseteq \mathit{Female}, \\ & \mathit{Man} \sqsubseteq \mathit{Male}, \\ & \mathit{Woman}(\mathit{Alice}), \\ & \mathit{Man}(\mathit{Bob}), \\ & \mathit{Alice\ relative\ Bob}, \\ & \mathit{Alice\ child\ Charlie}\}\end{aligned}$$

Ontologie - L'Editor Protégé

Protégé[3] è una suite per la modellazione di ontologie del Web Semantico, disponibile sia in versione web, che in versione installabile localmente (*Protégé Desktop*).

Permette di definire gerarchie di classi (tab *Classes*) a partire dalla classe radice `Thing`. Per ogni classe è possibile definire *label* e *comment* come annotazioni.

Analogamente è possibile definire gerarchie di proprietà (tab *Object Poperties*) a partire dalla proprietà radice `topObjectProperty` e di definire annotazioni per le proprietà. Inoltre, è possibile imporre dei vincoli di dominio e codominio per le proprietà.

Infine è possibile definire degli individui (tab *Individuals*) associarli a delle classi di appartenenza e metterli in relazione tra loro attraverso delle proprietà.

[3] `http://protege.stanford.edu/products.php`

Ontologie - L'Editor Protégé

Protégé[3] è una suite per la modellazione di ontologie del Web Semantico, disponibile sia in versione web, che in versione installabile localmente (*Protégé Desktop*).

Permette di definire gerarchie di classi (tab *Classes*) a partire dalla classe radice `Thing`. Per ogni classe è possibile definire *label* e *comment* come annotazioni.

Analogamente è possibile definire gerarchie di proprietà (tab *Object Poperties*) a partire dalla proprietà radice `topObjectProperty` e di definire annotazioni per le proprietà. Inoltre, è possibile imporre dei vincoli di dominio e codominio per le proprietà.

Infine è possibile definire degli individui (tab *Individuals*) associarli a delle classi di appartenenza e metterli in relazione tra loro attraverso delle proprietà.

[3] `http://protege.stanford.edu/products.php`

Ontologie - L'Editor Protégé

Protégé[3] è una suite per la modellazione di ontologie del Web Semantico, disponibile sia in versione web, che in versione installabile localmente (*Protégé Desktop*).

Permette di definire gerarchie di classi (tab *Classes*) a partire dalla classe radice `Thing`. Per ogni classe è possibile definire *label* e *comment* come annotazioni.

Analogamente è possibile definire gerarchie di proprietà (tab *Object Poperties*) a partire dalla proprietà radice `topObjectProperty` e di definire annotazioni per le proprietà. Inoltre, è possibile imporre dei vincoli di dominio e codominio per le proprietà.

Infine è possibile definire degli individui (tab *Individuals*) associarli a delle classi di appartenenza e metterli in relazione tra loro attraverso delle proprietà.

[3] `http://protege.stanford.edu/products.php`

Ontologie - L'Editor Protégé

Protégé[3] è una suite per la modellazione di ontologie del Web Semantico, disponibile sia in versione web, che in versione installabile localmente (*Protégé Desktop*).

Permette di definire gerarchie di classi (tab *Classes*) a partire dalla classe radice `Thing`. Per ogni classe è possibile definire *label* e *comment* come annotazioni.

Analogamente è possibile definire gerarchie di proprietà (tab *Object Poperties*) a partire dalla proprietà radice `topObjectProperty` e di definire annotazioni per le proprietà. Inoltre, è possibile imporre dei vincoli di dominio e codominio per le proprietà.

Infine è possibile definire degli individui (tab *Individuals*) associarli a delle classi di appartenenza e metterli in relazione tra loro attraverso delle proprietà.

[3] `http://protege.stanford.edu/products.php`

Ontologie - Interpretazioni

Per definire la semantica delle ontologie è necessario prima introdurre il concetto di interpretazione.

Una *interpretazione* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ è una coppia $\Delta^{\mathcal{I}}$, $\cdot^{\mathcal{I}}$ dove:

- $\Delta^{\mathcal{I}}$ è un insieme non vuoto;
- $\cdot^{\mathcal{I}}$ è una funzione (polimorfa) che associa
  - ad ogni nome di concetto in N_C un sottoinsieme di $\Delta^{\mathcal{I}}$,
  - ad ogni nome di proprietà in N_P una relazione su $\Delta^{\mathcal{I}}$,
  - ad ogni nome di individuo in N_I un elemento di $\Delta^{\mathcal{I}}$.

Consideriamo ad esempio la seguente ontologia

$$\mathcal{O} = \{ \mathit{Woman} \sqsubseteq \mathit{Human}, \mathit{Man} \sqsubseteq \mathit{Human}, \mathit{Woman} \sqsubseteq \mathit{Female}, \mathit{Man} \sqsubseteq \mathit{Male}, \mathit{Woman}(\mathit{Alice}), \mathit{Man}(\mathit{Bob}), \mathit{Alice\ relative\ Bob}, \mathit{Alice\ child\ Charlie} \}$$

Una possibile interpretazione $\mathcal{I}$ è la seguente:

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \mathbb{N} \\
\mathit{Alice}^{\mathcal{I}} &= 0 \\
\mathit{Bob}^{\mathcal{I}} &= 1 \\
\mathit{Charlie}^{\mathcal{I}} &= 2 \\
\mathit{Human}^{\mathcal{I}} &= \{0, 1, 2\} \\
\mathit{Male}^{\mathcal{I}} &= \{1, 2\} \\
\mathit{Female}^{\mathcal{I}} &= \{0\} \\
\mathit{Man}^{\mathcal{I}} &= \{1, 2\} \\
\mathit{Woman}^{\mathcal{I}} &= \{0\}
\end{aligned}$$

NB: è sufficiente prendere in considerazione ai nostri fini i simboli che compaiono nell'ontologia.

Ontologie - Interpretazioni

Per definire la semantica delle ontologie è necessario prima introdurre il concetto di interpretazione.

Una *interpretazione* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ è una coppia $\Delta^{\mathcal{I}}$, $\cdot^{\mathcal{I}}$ dove:

- $\Delta^{\mathcal{I}}$ è un insieme non vuoto;
- $\cdot^{\mathcal{I}}$ è una funzione (polimorfa) che associa
  - ad ogni nome di concetto in N_C un sottoinsieme di $\Delta^{\mathcal{I}}$,
  - ad ogni nome di proprietà in N_P una relazione su $\Delta^{\mathcal{I}}$,
  - ad ogni nome di individuo in N_I un elemento di $\Delta^{\mathcal{I}}$.

Consideriamo ad esempio la seguente ontologia

$$\mathcal{O} = \{ \; \mathit{Woman} \sqsubseteq \mathit{Human}, \mathit{Man} \sqsubseteq \mathit{Human}, \mathit{Woman} \sqsubseteq \mathit{Female}, \mathit{Man} \sqsubseteq \mathit{Male}, \\ \mathit{Woman}(\mathit{Alice}), \mathit{Man}(\mathit{Bob}), \mathit{Alice\ relative\ Bob}, \mathit{Alice\ child\ Charlie} \}$$

Una possibile interpretazione $\mathcal{I}$ è la seguente:

$$\begin{array}{lcl} \Delta^{\mathcal{I}} & = & \mathbb{N} \\ \mathit{Alice}^{\mathcal{I}} & = & 0 \\ \mathit{Bob}^{\mathcal{I}} & = & 1 \\ \mathit{Charlie}^{\mathcal{I}} & = & 2 \\ \mathit{Human}^{\mathcal{I}} & = & \{0, 1, 2\} \\ \mathit{Male}^{\mathcal{I}} & = & \{1, 2\} \\ \mathit{Female}^{\mathcal{I}} & = & \{0\} \\ \mathit{Man}^{\mathcal{I}} & = & \{1, 2\} \\ \mathit{Woman}^{\mathcal{I}} & = & \{0\} \end{array}$$

NB: è sufficiente prendere in considerazione ai nostri fini i simboli che compaiono nell'ontologia.

Ontologie - Soddisfacibilità

La semantica formale di cui sono equipaggiate le ontologie abilita l'esecuzione automatica di *reasoning tasks*. Quello fondamentale è la verifica di *Soddisfacibilità*, che permette di controllare che una ontologia non sia autocontraddittoria.

La nozione di soddisfacibilità per le ontologie è definita come segue. Sia $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ una interpretazione.

$$\begin{array}{rcl}
\mathcal{I} \text{ soddisfa } C \sqsubseteq D & \iff & C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } P \sqsubseteq Q & \iff & P^{\mathcal{I}} \subseteq Q^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } \mathrm{dom}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(x \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } \mathrm{range}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(y \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } C(a) & \iff & a^{\mathcal{I}} \in C^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } a\, P\, b & \iff & [a^{\mathcal{I}}, a^{\mathcal{I}}] \in P^{\mathcal{I}}
\end{array}$$

per ogni $C, D \in N_C$, $P, Q \in PNames$, $a, b \in N_I$.

$\mathcal{I}$ soddisfa una ontologia $\mathcal{O}$ se e solo se $\mathcal{I}$ soddisfa tutti i vincoli e le asserzioni in $\mathcal{O}$.

Una ontologia $\mathcal{O}$ è detta *soddisfacibile* (o anche *consistente*) se e solo se esiste una interpretazione $\mathcal{I}$ che la soddisfa.

Ontologie - Soddisfacibilità

La semantica formale di cui sono equipaggiate le ontologie abilita l'esecuzione automatica di *reasoning tasks*. Quello fondamentale è la verifica di *Soddisfacibilità*, che permette di controllare che una ontologia non sia autocontraddittoria.

La nozione di soddisfacibilità per le ontologie è definita come segue. Sia $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ una interpretazione.

$$\begin{array}{rcl}
\mathcal{I} \text{ soddisfa } C \sqsubseteq D & \iff & C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } P \sqsubseteq Q & \iff & P^{\mathcal{I}} \subseteq Q^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } \mathrm{dom}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(x \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } \mathrm{range}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(y \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } C(a) & \iff & a^{\mathcal{I}} \in C^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } a\, P\, b & \iff & [a^{\mathcal{I}}, a^{\mathcal{I}}] \in P^{\mathcal{I}}
\end{array}$$

per ogni $C, D \in N_C$, $P, Q \in PNames$, $a, b \in N_I$.

$\mathcal{I}$ soddisfa una ontologia $\mathcal{O}$ se e solo se $\mathcal{I}$ soddisfa tutti i vincoli e le asserzioni in $\mathcal{O}$.

Una ontologia $\mathcal{O}$ è detta *soddisfacibile* (o anche *consistente*) se e solo se esiste una interpretazione $\mathcal{I}$ che la soddisfa.

Ontologie - Soddisfacibilità

La semantica formale di cui sono equipaggiate le ontologie abilita l'esecuzione automatica di *reasoning tasks*. Quello fondamentale è la verifica di *Soddisfacibilità*, che permette di controllare che una ontologia non sia autocontraddittoria.

La nozione di soddisfacibilità per le ontologie è definita come segue. Sia $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ una interpretazione.

$$
\begin{array}{rcl}
\mathcal{I} \text{ soddisfa } C \sqsubseteq D & \iff & C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } P \sqsubseteq Q & \iff & P^{\mathcal{I}} \subseteq Q^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } \mathrm{dom}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(x \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } \mathrm{range}(P) \sqsubseteq C & \iff & (\forall [x, y] \in P^{\mathcal{I}})(y \in C^{\mathcal{I}}) \\
\mathcal{I} \text{ soddisfa } C(a) & \iff & a^{\mathcal{I}} \in C^{\mathcal{I}} \\
\mathcal{I} \text{ soddisfa } a\, P\, b & \iff & [a^{\mathcal{I}}, a^{\mathcal{I}}] \in P^{\mathcal{I}}
\end{array}
$$

per ogni $C, D \in N_C$, $P, Q \in PNames$, $a, b \in N_I$.

$\mathcal{I}$ soddisfa una ontologia $\mathcal{O}$ se e solo se $\mathcal{I}$ soddisfa tutti i vincoli e le asserzioni in $\mathcal{O}$.

Una ontologia $\mathcal{O}$ è detta *soddisfacibile* (o anche *consistente*) se e solo se esiste una interpretazione $\mathcal{I}$ che la soddisfa.

Ontologie - Soddisfacibilità - Esempi (1/2)

Consideriamo la seguente ontologia

$$\mathcal{O} = \{ \begin{array}{l} \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \\ \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\} \end{array}$$

Consideriamo la seguente interpretazione (di Herbrandt) $\mathcal{I}$:

$$\begin{array}{rcl} \mathit{Socrate}^{\mathcal{I}} & = & \mathit{Socrate} \\ \mathit{Plato}^{\mathcal{I}} & = & \mathit{Plato} \\ \mathit{Human}^{\mathcal{I}} & = & \{\mathit{Socrate}, \mathit{Plato}\} \\ \mathit{teacherOf}^{\mathcal{I}} & = & \{[\mathit{Socrate}, \mathit{Plato}]\} \end{array}$$

$\mathcal{I}$ soddisfa $\mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human} \iff (\forall [x, y] \in \mathit{teacherOf}^{\mathcal{I}})(x \in \mathit{Human}^{\mathcal{I}})$

$\mathcal{I}$ soddisfa $\mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato} \iff [\mathit{Socrate}^{\mathcal{I}}, \mathit{Plato}^{\mathcal{I}}] \in \mathit{teacherOf}^{\mathcal{I}}$

Quindi $\mathcal{I}$ soddisfa $\mathcal{O}$.

Quindi $\mathcal{O}$ è soddisfacibile.

Ontologie - Soddisfacibilità - Esempi (1/2)

Consideriamo la seguente ontologia

$$\mathcal{O} = \{ \begin{array}{l} \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \\ \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\} \end{array}$$

Consideriamo la seguente interpretazione (di Herbrandt) $\mathcal{I}$:

$$\begin{array}{rcl} \mathit{Socrate}^{\mathcal{I}} & = & \mathit{Socrate} \\ \mathit{Plato}^{\mathcal{I}} & = & \mathit{Plato} \\ \mathit{Human}^{\mathcal{I}} & = & \{\mathit{Socrate}, \mathit{Plato}\} \\ \mathit{teacherOf}^{\mathcal{I}} & = & \{[\mathit{Socrate}, \mathit{Plato}]\} \end{array}$$

$$\begin{array}{rcl} \mathcal{I} \text{ soddisfa } \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human} & \iff & (\forall [x, y] \in \mathit{teacherOf}^{\mathcal{I}})(x \in \mathit{Human}^{\mathcal{I}}) \\ \mathcal{I} \text{ soddisfa } \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato} & \iff & [\mathit{Socrate}^{\mathcal{I}}, \mathit{Plato}^{\mathcal{I}}] \in \mathit{teacherOf}^{\mathcal{I}} \end{array}$$

Quindi $\mathcal{I}$ soddisfa $\mathcal{O}$.

Quindi $\mathcal{O}$ è soddisfacibile.

Ontologie - Soddisfacibilità - Esempi (1/2)

Consideriamo la seguente ontologia

$$\mathcal{O} = \{ \quad \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \\ \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\}$$

Consideriamo la seguente interpretazione (di Herbrandt) $\mathcal{I}$:

$$\begin{aligned} \mathit{Socrate}^{\mathcal{I}} &= \mathit{Socrate} \\ \mathit{Plato}^{\mathcal{I}} &= \mathit{Plato} \\ \mathit{Human}^{\mathcal{I}} &= \{\mathit{Socrate}, \mathit{Plato}\} \\ \mathit{teacherOf}^{\mathcal{I}} &= \{[\mathit{Socrate}, \mathit{Plato}]\} \end{aligned}$$

$$\begin{aligned} \mathcal{I} \text{ soddisfa } \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human} &\iff (\forall [x, y] \in \mathit{teacherOf}^{\mathcal{I}})(x \in \mathit{Human}^{\mathcal{I}}) \\ \mathcal{I} \text{ soddisfa } \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato} &\iff [\mathit{Socrate}^{\mathcal{I}}, \mathit{Plato}^{\mathcal{I}}] \in \mathit{teacherOf}^{\mathcal{I}} \end{aligned}$$

Quindi $\mathcal{I}$ soddisfa $\mathcal{O}$.

Quindi $\mathcal{O}$ è soddisfacibile.

Ontologie - Soddisfacibilità - Esempi (2/2)

Consideriamo la seguente ontologia

$$\mathcal{O} = \{ \quad \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \\ \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\}$$

La seguente interpretazione (di Herbrandt) $\mathcal{I}_1$ NON soddisfa $\mathcal{O}$:

$$\begin{aligned} \mathit{Socrate}^{\mathcal{I}_1} &= \mathit{Socrate} \\ \mathit{Plato}^{\mathcal{I}_1} &= \mathit{Plato} \\ \mathit{Human}^{\mathcal{I}_1} &= \{\textbf{Plato}\} \\ \mathit{teacherOf}^{\mathcal{I}_1} &= \{[\mathit{Socrate}, \mathit{Plato}]\} \end{aligned}$$

Infatti $\mathcal{I}_1$ non soddisfa $\mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}$ perchè *Socrate* non è nell'insieme *Human* (interpretati con $\mathcal{I}_1$).

Ontologie - Implicazione

Date due Ontologie $\mathcal{O}$ e $\mathcal{O}'$, si dice che $\mathcal{O}$ *implica* $\mathcal{O}'$ se e solo se tutte le interpretazioni che soddisfano $\mathcal{O}$ soddisfano anche $\mathcal{O}'$.

La verifica di implicazione può essere utilizzata per ricavare tutte le *conseguenze logiche* di una ontologia.

È facile verificare che, date $\mathcal{O}$ e $\mathcal{O}'$ sotto, $\mathcal{O}'$ contiene tutte le asserzioni che sono conseguenze logiche di $\mathcal{O}$.

$$\mathcal{O} = \{ \quad \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \quad \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\} \quad \Longrightarrow \mathcal{O}' = \{\mathit{Human}(\mathit{Socrate})\}$$

Ontologie - Implicazione

Date due Ontologie $\mathcal{O}$ e $\mathcal{O}'$, si dice che $\mathcal{O}$ *implica* $\mathcal{O}'$ se e solo se tutte le interpretazioni che soddisfano $\mathcal{O}$ soddisfano anche $\mathcal{O}'$.

La verifica di implicazione può essere utilizzata per ricavare tutte le *conseguenze logiche* di una ontologia.

È facile verificare che, date $\mathcal{O}$ e $\mathcal{O}'$ sotto, $\mathcal{O}'$ contiene tutte le asserzioni che sono conseguenze logiche di $\mathcal{O}$.

$$\mathcal{O} = \{ \quad \begin{array}{l} \mathrm{dom}(\mathit{teacherOf}) \sqsubseteq \mathit{Human}, \\ \mathit{Socrate}\ \mathit{teacherOf}\ \mathit{Plato}\} \end{array} \quad \Longrightarrow \mathcal{O}' = \{\mathit{Human}(\mathit{Socrate})\}$$

Ontologie - Reasoning con Protégé

L'editor Protégé fornisce la possibilità di eseguire il reasoning attraverso il menù *Reasoner*. Le asserzioni inferite verranno mostrate in evidenza.

Ontologie nel Web Semantico

Le ontologie usate nel Web Semantico sono caratterizzate dai seguenti punti:

- nomi di classi, proprietà ed individui sono *IRI*

$$IRI = N_C \cup N_P \cup N_I,$$

- sono presenti dei tipi di dato *concreti*.

IRI nel Web Semantico

Nell'ambito del Web Semantico, tutti gli oggetti reali o concreti sono identificati attraverso IRI. As esempio

`http://dbpedia.org/resource/Leonardo_da_Vinci`

è la IRI usata nell'ontologia `dbpedia.org` per indicare Leonardo da Vinci, e ancora

`http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619`

indica la *Mona Lisa* nell'ontologia del progetto *Europeana*.

LodLive[4] e *LodView*[5] sono due servizi online che permettono di navigare il Linked Open Data Cloud ed esaminare le informazioni in esso contenute in merito ad uno specifico elemento a partire dalla (da una delle) IRI assegnate a quell'elemento.

[4] `http://lodlive.it`
[5] `http://lodview.it`

IRI nel Web Semantico

Nell'ambito del Web Semantico, tutti gli oggetti reali o concreti sono identificati attraverso IRI. As esempio

`http://dbpedia.org/resource/Leonardo_da_Vinci`

è la IRI usata nell'ontologia `dbpedia.org` per indicare Leonardo da Vinci, e ancora

`http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619`

indica la *Mona Lisa* nell'ontologia del progetto *Europeana*.

LodLive[4] e *LodView*[5] sono due servizi online che permettono di navigare il Linked Open Data Cloud ed esaminare le informazioni in esso contenute in merito ad uno specifico elemento a partire dalla (da una delle) IRI assegnate a quell'elemento.

[4] `http://lodlive.it`

[5] `http://lodview.it`

IRI nel Web Semantico - Namespaces

Nelle ontologie del Web Semantico le IRI possono essere abbreviate con il meccanismo dei *namespace* (prefissi), mutuato da XML.

Ad ogni ontologia spesso è assegnato un *base prefix*, che solitamente coincide con la IRI alla quale è possibile scaricare l'ontologia stessa. Essa viene usata come prefisso per ottenere le IRI degli oggetti dell'ontologia nel caso in cui all'oggetto sia assegnata una IRI *incompleta*. Ad esempio, se il base prefix dell'ontologia $\mathcal{O}$ è `http://example.org/`,

$$Alice \quad \Longrightarrow \texttt{http://example.org/Alice}.$$

È possibile specificare degli ulteriori *prefix* come coppie

$$< prefixname > \quad \longrightarrow \quad < prefixuri >$$

e *abbreviare* delle IRI nell'ontologia con la sintassi
$< prefixname >:< IRIspecificpart >$.

Se ad esempio nell'ontologia è definito il prefisso

$$ex2 \quad \longrightarrow \quad \texttt{http://example2.org/}$$

le IRI `ex2:Alice` verranno espanse in `http://example2.org/Alice` .

IRI nel Web Semantico - Namespaces

Nelle ontologie del Web Semantico le IRI possono essere abbreviate con il meccanismo dei *namespace* (prefissi), mutuato da XML.

Ad ogni ontologia spesso è assegnato un *base prefix*, che solitamente coincide con la IRI alla quale è possibile scaricare l'ontologia stessa. Essa viene usata come prefisso per ottenere le IRI degli oggetti dell'ontologia nel caso in cui all'oggetto sia assegnata una IRI *incompleta*. Ad esempio, se il base prefix dell'ontologia $\mathcal{O}$ è `http://example.org/`,

$$Alice \quad \Longrightarrow \texttt{http://example.org/Alice}.$$

È possibile specificare degli ulteriori *prefix* come coppie

$$< prefixname > \quad \longrightarrow \quad < prefixuri >$$

e *abbreviare* delle IRI nell'ontologia con la sintassi
$< prefixname >:< IRIspecificpart >$.

Se ad esempio nell'ontologia è definito il prefisso

$$ex2 \quad \longrightarrow \quad \texttt{http://example2.org/}$$

le IRI `ex2:Alice` verranno espanse in `http://example2.org/Alice` .

IRI nel Web Semantico - Namespaces

Nelle ontologie del Web Semantico le IRI possono essere abbreviate con il meccanismo dei *namespace* (prefissi), mutuato da XML.

Ad ogni ontologia spesso è assegnato un *base prefix*, che solitamente coincide con la IRI alla quale è possibile scaricare l'ontologia stessa. Essa viene usata come prefisso per ottenere le IRI degli oggetti dell'ontologia nel caso in cui all'oggetto sia assegnata una IRI *incompleta*. Ad esempio, se il base prefix dell'ontologia $\mathcal{O}$ è `http://example.org/`,

$$Alice \quad \Longrightarrow \texttt{http://example.org/Alice}.$$

È possibile specificare degli ulteriori *prefix* come coppie

$$< prefixname > \quad \longrightarrow \quad < prefixuri >$$

e *abbreviare* delle IRI nell'ontologia con la sintassi
$< prefixname >:< IRIspecificpart >$.

Se ad esempio nell'ontologia è definito il prefisso

$$ex2 \quad \longrightarrow \quad \texttt{http://example2.org/}$$

le IRI `ex2:Alice` verranno espanse in `http://example2.org/Alice` .

Gestione degli IRI in Protégé

Nell'editor Protégé la IRI assegnata ad ogni elemento (classe, proprietà, individuo) è visibile come popup che compare passando sull'elemento col mouse.

Nel tab *Active Ontology* è possibile specificare la IRI dell'ontologia, che andrebbe usata come base prefix. Inoltre è possibile indicare alcuni metadati relativi all'ontologia stessa (autore, versione, ...).

Nella sotto-tab *Ontology Prefixes* è possibile definire ulteriori prefissi.

Gestione degli IRI in Protégé

Nell'editor Protégé la IRI assegnata ad ogni elemento (classe, proprietà, individuo) è visibile come popup che compare passando sull'elemento col mouse.

Nel tab *Active Ontology* è possibile specificare la IRI dell'ontologia, che andrebbe usata come base prefix. Inoltre è possibile indicare alcuni metadati relativi all'ontologia stessa (autore, versione, ...).

Nella sotto-tab *Ontology Prefixes* è possibile definire ulteriori prefissi.

Gestione degli IRI in Protégé

Nell'editor Protégé la IRI assegnata ad ogni elemento (classe, proprietà, individuo) è visibile come popup che compare passando sull'elemento col mouse.

Nel tab *Active Ontology* è possibile specificare la IRI dell'ontologia, che andrebbe usata come base prefix. Inoltre è possibile indicare alcuni metadati relativi all'ontologia stessa (autore, versione, ...).

Nella sotto-tab *Ontology Prefixes* è possibile definire ulteriori prefissi.

Letterali

I letterali vengono usati per rappresentare tipi di dato *concreti*, come ad esempio stringhe di testo, numeri, date, ...

Grazie ai letterali è possibile ad esempio esprimere affermazioni dei seguenti tipi:

- Il cognome di Mario è *Rossi*;
- Cristiano è nato il giorno *22 Marzo 1979*;
- L'Empire State Building è alto *380 metri*.

Datatype

Per introdurre i Letterali è necessario fornire prima la definizione di *datatype* (vedi `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-Datatypes` e `http://www.w3.org/TR/xmlschema11-2/`).

Un *datatype* è caratterizzato da tre componenti:

- un *lexical space*, ossia un insieme di stringhe (finite) di caratteri nella codifica *UNICODE*;
- un *value space*, che è un insieme non meglio specificato e numerabile di *valori* (interi, date, Booleani, ...);
- un *lexical-value mapping* che associa ad ogni stringa nel lexical space un elemento nel value space.

I datatype vengono di solito indicati con delle IRI.

Datatype - Esempio 1 : `xsd:integer`

Il datatype `xsd:integer` (dove `xsd` è l'abbreviazione per il namespace `http://www.w3.org/2001/XMLSchema#`) è definito come segue:

- il *lexical space* di `xsd:integer` è costituito da tutte le sequenze finite di cifre da 0 a 9, possibilmente precedute dal carattere "-" o "+";
- il *value space* è l'insieme dei numeri interi;
- il valore di una stringa nel lexical space di `xsd:integer` si ottiene considerando le cifre presenti nella stringa come cifre del corrispondente numero in base 10, e moltiplicando il numero così ottenuto per -1 nel caso in cui la stringa inizi con il carattere "-".

Datatype - Esempio 2 : `xsd:string`

Il datatype `xsd:string` è definito come segue:

- il *lexical space* di `xsd:string` comprende tutte le sequenze di caratteri (UNICODE) di zero o più caratteri;
- il *value space* di `xsd:string` coincide col suo lexical space;
- il *lexical-value mapping* associa ogni stringa nel lexical space con se stessa (indetitá).

Altri esempi di datatype

Riportiamo alcuni datatype (mutuati da XML Schema) di uso comune.

`xsd:boolean`	true, false
`xsd:decimal`	Arbitrary-precision decimal numbers
`xsd:integer`	Arbitrary-size integer numbers
`xsd:double`	64-bit floating point numbers incl. $\pm$Inf, $\pm$0, NaN
`xsd:float`	32-bit floating point numbers incl. $\pm$Inf, $\pm$0, NaN
`xsd:date`	Dates (yyyy-mm-dd) with or without timezone
`xsd:time`	Times (hh:mm:ss.sss...) with or without timezone
`xsd:dateTime`	Date and time with or without timezone
`xsd:dateTimeStamp`	Date and time with required timezone
`xsd:duration`	Duration of time
`xsd:byte`	-128...+127 (8 bit)
`xsd:short`	-32768...+32767 (16 bit)
`xsd:int`	-2147483648...+2147483647 (32 bit)
`xsd:long`	-9223372036854775808...+9223372036854775807 (64 bit)
`xsd:unsignedByte`	0...255 (8 bit)
`xsd:unsignedShort`	0...65535 (16 bit)
`xsd:unsignedInt`	0...4294967295 (32 bit)
`xsd:unsignedLong`	0...18446744073709551615 (64 bit)
`xsd:positiveInteger`	Integer numbers > 0
`xsd:nonNegativeInteger`	Integer numbers ≥ 0
`xsd:negativeInteger`	Integer numbers < 0
`xsd:nonPositiveInteger`	Integer numbers ≤ 0
`xsd:hexBinary`	Hex-encoded binary data

Letterali - Definizione

Formalmente, i letterali sono definiti come segue (vedi `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-Graph-Literal`).

Un *letterale* è costituito da un *datatype* t e da una stringa di caratteri nel lexical space di t (la cosiddetta *lexical form* del letterale).

Se un letterale è di tipo `xsd:string`, ad esso può essere associato un *language tag* ad indicarne la *lingua*. Per i valori che questo attributo può assumere fare riferimento allo *IANA Language Subtag Registry*.

Letterali - Esempi

Seguono alcuni esempi di letterali:

lexical form	**data type**	**language tag**
"380"	`xsd:integer`	-
"March-22-1979"	`xsd:date`	-
"Rossi"	`xsd:string`	-
"Parigi"	`xsd:string`	it
"Paris"	`xsd:string`	en

Nel caso in cui si ometta l'indicazione del tipo di dato, il letterale si assume essere di tipo `xsd:string`.

Letterali - Notazione (1/3)

Per indicare i letterali spesso si usano le seguenti notazioni

"lexform"^^`type`

$< lexform, type >$

ove *lexform* e *type* sono la lexical form e il data type del letterale, rispettivamente.

Alcuni esempi:

"380"^^`xsd:integer`	$<$ "380", $xsd : integer >$
"March-22-1979"^^`xsd:date`	$<$ "$March - 22 - 1979$", $xsd : date >$
"Rossi"^^`xsd:string`	$<$ "$Rossi$", $xsd : string >$

Letterali - Notazione (2/3)

Nel seguito indicheremo con

- $\mathcal{D}$ l'insieme di tutti i possibili data type, e con
- $\mathcal{L}$ l'insieme dei letterali.

Si assume per convenienza che i data type dei letterali in $\mathcal{L}$ siano tutti contenuti in $\mathcal{D}$.

Letterali - Notazione (3/3)

Inoltre, dato un letterale $l \in \mathcal{L}$, indicheremo con

- *datatype*(l) il tipo di dato di l (ovviamente *datatype*(l) $\in \mathcal{D}$), e con
- *lexform*(l) la lexical form di l.

Seguono alcuni esempi:

datatype("380" ^^`xsd:integer`) = `xsd:integer`
datatype("March-22-1979" ^^`xsd:date`) = `xsd:date`
datatype("Rossi" ^^`xsd:string`) = `xsd:string`

lexform("380" ^^`xsd:integer`) = "380"
lexform("March-22-1979" ^^`xsd:date`) = "March-22-1979"
lexform("Rossi" ^^`xsd:string`) = "Rossi"

Confronto tra Letterali

Due letterali sono uguali se e solo se sono uguali le loro lexical form, se hanno lo stesso tipo e se sono uguali i loro language tag, ove presenti. Di conseguenza due letterali possono essere diversi anche avendo lo stesso *valore*. Ad esempio i due seguenti letterali hanno entrambi come valore l'intero 1 ma sono diversi:

"1"^^`xsd:integer`
"01"^^`xsd:integer`.

Ontologie - Datatype Properties

È possibile definire delle proprietà che abbiano un oggetto di tipo concreto (su Protégé la tab *Datatype properties*).

È possibile vincolare una proprietà ad avere come codominio solo letterali con un certo data type.

Alcuni esempi vincoli di questa natura sono

$$\text{range}(\textit{surname}) \sqsubseteq \texttt{xsd:string}$$
$$\text{range}(\textit{hasbirth}) \sqsubseteq \texttt{xsd:date}$$

con $\textit{surname}, \textit{hasbirth} \in N_P$ e `xsd:string`, `xsd:date` data type.