Introduzione agli Open Data
Dati a 4 e 5 stelle
Interrogazioni e Vocabolari

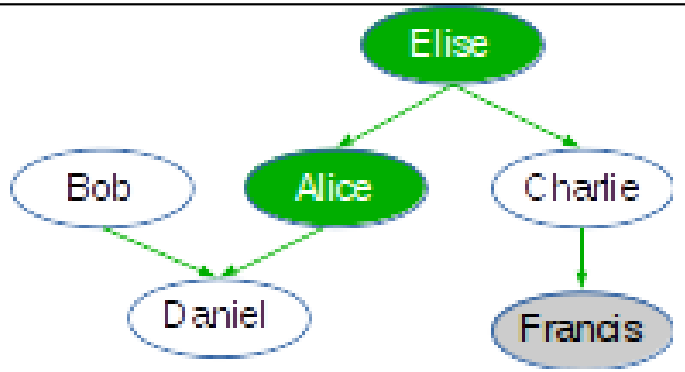
Cristiano Longo
longo@dmi.unict.it

Università di Catania

Interrogazioni

Il metodo più immediato per ottenere informazioni da una ontologia è il *Conjunctive Query Answering*. Consideriamo ad esempio la seguente ontologia:

$$\begin{aligned}\mathcal{O} = \quad & \{Female(Elise), Female(Alice), Male(Bob),\\ & Male(Charlie), Male(Daniel),\\ & Alice\ childOf\ Elise, Charlie\ childOf\ Elise,\\ & Daniel\ childOf\ Alice, Daniel\ childOf\ Bob,\\ & Francis\ childOf\ Charlie\}\end{aligned}$$



Alcune interrogazioni che è possibile effettuare con il conjunctive query answering sono:

- "Trova tutti gli individui maschi."
- "Chi sono gli individui con almeno un figlio maschio?"
- "Chi sono i figli di *Alice*?"
- "Chi sono gli individui con almeno un figlio maschio ed una femmina?"
- "Chi sono gli individui maschi con almeno un figlio maschio?"

Query Congiuntive

Sia $V = \{x, y, z, ...\}$ l'insieme infinito, numerabile e disgiunto da N_C, N_P e N_I delle *variabili*. Le *formule atomiche* sono espressioni dei due seguenti tipi:

$$C(x), \quad x\, P\, y$$

con $x, y \in N_I \cup V$, $C \in N_C$ e $P \in N_P$.

Una *query congiuntiva* è una congiunzione finita di formule atomiche $T_1 \wedge \ldots \wedge T_n$. Alcuni esempi ($x$ e y variabili):

$$Male(x); \quad y\ childOf\ x \wedge Male(y)$$

Query Congiuntive

Sia $V = \{x, y, z, ...\}$ l'insieme infinito, numerabile e disgiunto da N_C, N_P e N_I delle *variabili*. Le *formule atomiche* sono espressioni dei due seguenti tipi:

$$C(x), \quad x\, P\, y$$

con $x, y \in N_I \cup V$, $C \in N_C$ e $P \in N_P$.

Una *query congiuntiva* è una congiunzione finita di formule atomiche $T_1 \wedge \ldots \wedge T_n$. Alcuni esempi ($x$ e y variabili):

$$Male(x); \quad y\ childOf\ x \wedge Male(y)$$

Query Congiuntive - Soluzione

Una soluzione per una query congiuntiva $Q = T_1 \wedge \ldots \wedge T_n$ rispetto all'ontologia $\mathcal{O}$ è una sostituzione σ che associa ad ogni variabile in Q un nome di individuo in $\mathcal{O}$ in maniera tale che le formule atomiche $\sigma T_1, \ldots, \sigma T_n$ compaiano tutte in $\mathcal{O}$.

Consideriamo la query Q e l'ontologia $\mathcal{O}$.

$$Q = y \; childOf \; x \wedge Male(y)$$

$$\begin{aligned} \mathcal{O} = \quad & \{Female(Elise), Female(Alice), Male(Bob), \\ & Male(Charlie), Male(Daniel), \\ & Alice \; childOf \; Elise, Charlie \; childOf \; Elise, \\ & Daniel \; childOf \; Alice, Daniel \; childOf \; Bob, \\ & Francis \; childOf \; Charlie\} \end{aligned}$$

x	y
Elise	*Charlie*
Alice	*Daniel*
Bob	*Daniel*

Query Congiuntive - Soluzione

Una soluzione per una query congiuntiva $Q = T_1 \wedge \ldots \wedge T_n$ rispetto all'ontologia $\mathcal{O}$ è una sostituzione σ che associa ad ogni variabile in Q un nome di individuo in $\mathcal{O}$ in maniera tale che le formule atomiche $\sigma T_1, \ldots, \sigma T_n$ compaiano tutte in $\mathcal{O}$.

Consideriamo la query Q e l'ontologia $\mathcal{O}$.

$$Q = y \; childOf \; x \; \wedge \; Male(y)$$

$$\begin{aligned} \mathcal{O} = \quad & \{Female(Elise), Female(Alice), Male(Bob), \\ & Male(Charlie), Male(Daniel), \\ & Alice \; childOf \; Elise, Charlie \; childOf \; Elise, \\ & Daniel \; childOf \; Alice, Daniel \; childOf \; Bob, \\ & Francis \; childOf \; Charlie\} \end{aligned}$$

x	y
Elise	*Charlie*
Alice	*Daniel*
Bob	*Daniel*

Il protocollo SPARQL

Le basi di conoscenza presenti sul Web Semantico usualmente mettono a disposizione uno *SPARQL endpoint* che permette di interrogarle e, ove permesso, di modificarle.

Knowledge Base	**Endpoint IRI**
Europeana	`http://europeana.ontotext.com/sparql`
CNR	`http://data.cnr.it/sparql/`
Camera dei Deputati	`http://dati.camera.it/sparql`
DBPedia	`http://dbpedia.org/sparql`

Table : Alcuni endpoint sparql

Il *protocollo SPARQL* (vedi `http://www.w3.org/TR/sparql11-protocol/`) è basato sul protocollo HTTP le richieste SPARQL vengono inviate agli endpoint come richieste GET o POST e l'endpoint risponde con un *esito*.

Le richieste si suddividono in richieste di *query* o *update*.
In caso di richieste di tipo query effettuate con successo, la risposta alla chiamata GET o POST conterrà anche tutte le *soluzioni* dell'interrogazione in uno dei formati XML, JSON o CSV (il formato di risposta va specificato nella richesta).

Il protocollo SPARQL

Le basi di conoscenza presenti sul Web Semantico usualmente mettono a disposizione uno *SPARQL endpoint* che permette di interrogarle e, ove permesso, di modificarle.

Knowledge Base	**Endpoint IRI**
Europeana	`http://europeana.ontotext.com/sparql`
CNR	`http://data.cnr.it/sparql/`
Camera dei Deputati	`http://dati.camera.it/sparql`
DBPedia	`http://dbpedia.org/sparql`

Table : Alcuni endpoint sparql

Il *protocollo SPARQL* (vedi `http://www.w3.org/TR/sparql11-protocol/`) è basato sul protocollo HTTP le richieste SPARQL vengono inviate agli endpoint come richieste GET o POST e l'endpoint risponde con un *esito*.

Le richieste si suddividono in richieste di *query* o *update*.
In caso di richieste di tipo query effettuate con successo, la risposta alla chiamata GET o POST conterrà anche tutte le *soluzioni* dell'interrogazione in uno dei formati XML, JSON o CSV (il formato di risposta va specificato nella richesta).

Il protocollo SPARQL

Le basi di conoscenza presenti sul Web Semantico usualmente mettono a disposizione uno *SPARQL endpoint* che permette di interrogarle e, ove permesso, di modificarle.

Knowledge Base	**Endpoint IRI**
Europeana	`http://europeana.ontotext.com/sparql`
CNR	`http://data.cnr.it/sparql/`
Camera dei Deputati	`http://dati.camera.it/sparql`
DBPedia	`http://dbpedia.org/sparql`

Table : Alcuni endpoint sparql

Il *protocollo SPARQL* (vedi `http://www.w3.org/TR/sparql11-protocol/`) è basato sul protocollo HTTP le richieste SPARQL vengono inviate agli endpoint come richieste GET o POST e l'endpoint risponde con un *esito*.

Le richieste si suddividono in richieste di *query* o *update*.
In caso di richieste di tipo query effettuate con successo, la risposta alla chiamata GET o POST conterrà anche tutte le *soluzioni* dell'interrogazione in uno dei formati XML, JSON o CSV (il formato di risposta va specificato nella richesta).

SPARQL Query Language

Le richieste di tipo *query* vanno specificate nel linguaggio denominato *SPARQL Query*. La specifica di questo linguaggio è disponibile all'indirizzo

`http://www.w3.org/TR/sparql11-query/` .

Una *query SPARQL* ha la seguente sintassi

```
BASE <iriBase>
PREFIX p1 : <iriP1>
...
PREFIX pn : <iriPn>

SELECT ?x1 ... ?xm WHERE { GraphPattern }
```

dove:

- la sezione *BASE* $<$ *iriBase* $>$ è opzionale;
- $p1, \ldots, pn$ sono prefissi di namespace ($n \geq 0$, se $n = 0$ non è presente alcun prefisso);
- $<$ *iriP*1 $>, \ldots, <$ *iriPn* $>$ sono IRI;
- $?x1, \ldots, ?xm$ sono *variabili* ($m > 0$);
- *GraphPattern* è un *graph pattern* di uno dei tipi che vedremo in seguito.

SPARQL Query Language - Triple Pattern

Il tipo di graph pattern più basilare è quello dei *triple pattern*. Un triple pattern è una espressione di uno dei seguenti tipi:

$$s\ p\ o\,, \quad s\ \mathtt{a}\ o$$

ove s e p possono essere IRI o variabili, a è una parola riservata, e o può essere una IRI, una variabile o un letterale.

Le formule atomiche nelle query congiuntive possono essere facilmente tradotte in triple patterns:

$C(?a) \quad \Longrightarrow \quad ?a\ \mathtt{a}\ C \quad | \quad ?a\ \mathtt{rdf:type}\ C$ (le due formulazioni sono equivalenti)

$?a\ P\ ?b \quad \Longrightarrow \quad ?a\ P\ ?b.$

Si noti che la proprietà `rdf:type` viene utilizzata per esprimere l'appartenenza nel linguaggio RDF.

A differenza delle query congiuntive, in SPARQL le variabili possono comparire anche al posto del nome della classe o della proprietà. Ad esempio i seguenti due sono triple pattern validi:

$?C(a)$ "Trova tutte le classi cui appartiene a"

$?a\ ?p\ ?b$ "Trova tutte le triple nell'ontologia".

SPARQL Query Language - Triple Pattern

Il tipo di graph pattern più basilare è quello dei *triple pattern*. Un triple pattern è una espressione di uno dei seguenti tipi:

$$s\ p\ o\,, \quad s\ \mathtt{a}\ o$$

ove s e p possono essere IRI o variabili, a è una parola riservata, e o può essere una IRI, una variabile o un letterale.

Le formule atomiche nelle query congiuntive possono essere facilmente tradotte in triple patterns:

$C(?a) \implies ?a\ \mathtt{a}\ C \quad | \quad ?a\ \mathtt{rdf:type}\ C$ (le due formulazioni sono equivalenti)

$?a\ P\ ?b \implies ?a\ P\ ?b.$

Si noti che la proprietà `rdf:type` viene utilizzata per esprimere l'appartenenza nel linguaggio RDF.

A differenza delle query congiuntive, in SPARQL le variabili possono comparire anche al posto del nome della classe o della proprietà. Ad esempio i seguenti due sono triple pattern validi:

$?C(a)$ "Trova tutte le classi cui appartiene a"

$?a\ ?p\ ?b$ "Trova tutte le triple nell'ontologia".

SPARQL Query Language - Triple Pattern

Il tipo di graph pattern più basilare è quello dei *triple pattern*. Un triple pattern è una espressione di uno dei seguenti tipi:

$$s\ p\ o\,, \quad s\ \mathtt{a}\ o$$

ove s e p possono essere IRI o variabili, a è una parola riservata, e o può essere una IRI, una variabile o un letterale.

Le formule atomiche nelle query congiuntive possono essere facilmente tradotte in triple patterns:

$C(?a) \implies ?a\ \mathtt{a}\ C \quad | \quad ?a\ \mathtt{rdf:type}\ C$ (le due formulazioni sono equivalenti)

$?a\ P\ ?b \implies ?a\ P\ ?b.$

Si noti che la proprietà `rdf:type` viene utilizzata per esprimere l'appartenenza nel linguaggio RDF.

A differenza delle query congiuntive, in SPARQL le variabili possono comparire anche al posto del nome della classe o della proprietà. Ad esempio i seguenti due sono triple pattern validi:

$?C(a)$ "Trova tutte le classi cui appartiene a"

$?a\ ?p\ ?b$ "Trova tutte le triple nell'ontologia".

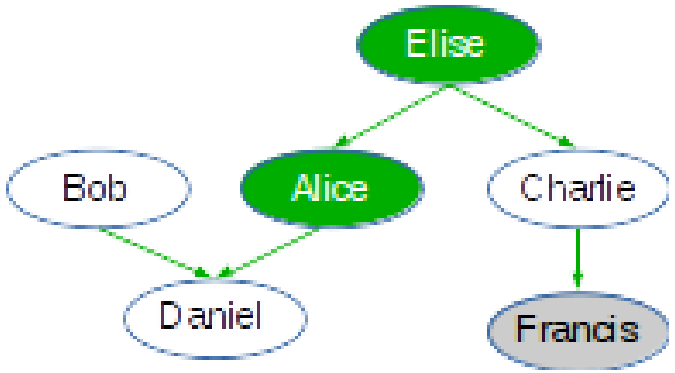
SPARQL Query Language - Soluzioni

Analogamente a quanto accade nell'ambito del conjunctive query answering, le *soluzioni* per un triple pattern T rispetto a una ontologia $\mathcal{O}$ sono tutte le sostituzioni σ che associano ad ogni variabile presente in T una IRI o un letterale in modo tale che σT sia presente in $\mathcal{O}$.

Il risultato di una query Q, avente come pattern un triple pattern T e come variabili nella select $x_1, \ldots, x_n$, rispetto all'ontologia $\mathcal{O}$ è l'insieme delle sostituzioni σ, ristrette alle variabili $x_1, \ldots, x_n$, tali che σT appartiene ad $\mathcal{O}$.

SPARQL Query Language - Triple Pattern - Esempio

Consideriamo ad esempio la seguente ontologia (base prefix `http://ex.org/`)



$$\begin{aligned}\mathcal{O} = \quad & \{Female(Elise), Female(Alice), Male(Bob),\\ & Male(Charlie), Male(Daniel),\\ & Alice\ childOf\ Elise, Charlie\ childOf\ Elise,\\ & Daniel\ childOf\ Alice, Daniel\ childOf\ Bob,\\ & Francis\ childOf\ Charlie\}\end{aligned}$$

Consideriamo inoltre la query Q definita come segue.

```
BASE <http://ex.org/>

SELECT ?x ?y WHERE { ?x childOf ?y . }
```

Le soluzioni di Q rispetto a $\mathcal{O}$ sono

?x	?y
<http://ex.org/Eliza>	<http://ex.org/Giorgia>
<http://ex.org/Eliza>	<http://ex.org/Francis>

SPARQL Query Language - Basic Graph Pattern

Un *basic graph pattern* è un insieme di triple pattern, separati dal carattere ".". Una sostituzione è una soluzione per un basic graph pattern se e solo se è una soluzione per tutti i triple pattern che compaiono in esso.

Un esempio di query con il basic graph pattern è il seguente:

> *"Trova tutte le persone con almeno un figlo maschio"*

```
BASE <http://ex.org/>

SELECT ?x  WHERE {
  ?y childOf ?x .
  ?y a Male
}
```

Da una query congiuntiva si può ottenere il basic graph pattern corrispondente come segue:

- applicare la trasformazione vista in precedenza ai triple pattern che lo costituiscono;
- sostituire i caratteri "." con l'operatore logico di congiunzione $\wedge$.

Ad esempio

$$C(?x) \wedge ?x\ P\ b \quad \Longrightarrow \quad ?x\ \mathrm{a}\ C\ .\ ?x\ P\ b\ .$$

SPARQL Query Language - Basic Graph Pattern

Un *basic graph pattern* è un insieme di triple pattern, separati dal carattere ".". Una sostituzione è una soluzione per un basic graph pattern se e solo se è una soluzione per tutti i triple pattern che compaiono in esso.

Un esempio di query con il basic graph pattern è il seguente:

> *"Trova tutte le persone con almeno un figlo maschio"*

```
BASE <http://ex.org/>

SELECT ?x  WHERE {
  ?y childOf ?x .
  ?y a Male
}
```

Da una query congiuntiva si può ottenere il basic graph pattern corrispondente come segue:

- applicare la trasformazione vista in precedenza ai triple pattern che lo costituiscono;
- sostituire i caratteri "." con l'operatore logico di congiunzione $\wedge$.

Ad esempio

$$C(?x) \wedge ?x\ P\ b \quad \Longrightarrow \quad ?x\ \mathrm{a}\ C\ .\ ?x\ P\ b\ .$$

SPARQL Query Language - Filtri sui letterali

È possibile specificare dei filtri sui letterali che compaiono nei pattern. Essi sono dei predicati, che dipendono dal tipo di dato dei letterali.[1] Permettono di selezionare tutte le soluzioni nelle quali ad una determinata variabile viene associato un letterale che soddisfa un predicato.

La seguente query ad esempio permette di ottenere tutti gli individui con un figlio il cui nome inizia con la lettera 'R'.

```
BASE <http://ex.org/>

SELECT ?x  WHERE {
  ?y childOf ?x .
  ?y fullName ?name .
  FILTER regex(?name, "^R.*") .
}
```

[1] http://www.w3.org/TR/sparql11-query/#OperatorMapping

SPARQL Query Language - Optional Pattern Matching

Dati due graph pattern P_1 e P_2, il seguente è un *optional graph pattern*:

$$P_1 \texttt{OPTIONAL} P_2.$$

Le sostituzioni che sono soluzioni per questo pattern sono quelle che

- sono soluzioni per P_1 e P_2, oppure
- sono soluzioni per P_1

sostitu
SPC Data,[2] tra le altre informazioni, contiene l'elenco di tutte le Pubbliche Amministrazioni italiane. Le seguenti query permettono di ricavare la IRI dell'individuo corrispondente al comune di Catania.[3]

```
SELECT ?x WHERE{
  ?x ?p "Comune di Catania"
}
```

```
PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
SELECT ?x WHERE{
  ?x ?p ?l .
  FILTER (fn:contains(fn:lower-case(?l), "comune di catania"))
}
```

La URL ottenuta come risultato è *deferenceable*. Visitandola col browser si ottiene una pagina web dove: vengono visualizzati tutti gli oggetti e i letterali con in quali è in relazione, e le relative proprietà; tutte le classi cui appartiene, che sono collegate all'individuo con la proprietà `rdf:type`.

Da questo dataset è possibile ad esempio ottenere tutti i comuni di una regione o ~~l'elenco dei comuni e dei rispettivi sindaci~~.

[2] http://spcdata.digitpa.gov.it/index.html