Rappresentazione personalizzata dell'informazione nel Web Semantico*

Matteo Golfarelli, Andrea Proli, and Stefano Rizzi

DEIS, University of Bologna, Viale Risorgimento 2, Bologna, Italy
{golfare, proliand}@csr.unibo.it, srizzi@deis.unibo.it

Sommario Un problema ancora irrisolto, in relazione allo sviluppo di un ambiente interattivo per la visualizzazione e la navigazione delle informazioni nel contesto del Web Semantico, consiste nell'inviduare il compromesso ideale tra potere espressivo e indipendenza dal dominio. In questo articolo introduciamo M-FIRE, un framework configurabile che permette di istanziare sistemi di visualizzazione e navigazione attraverso la definizione di metafore personalizzate: le metafore (1) guidano il processo che consente di ottenere una rappresentazione visiva di un dato insieme di informazioni, e (2) determinano quali interrogazioni generare in risposta alle azioni compiute dall'utente sulla rappresentazione ottenuta.

1 Introduzione

Il funzionamento del Web Semantico è basato sulla possibilità di definire in modo formale la semantica di un insieme di informazioni e, conseguentemente, di automatizzarne l'elaborazione. Il consorzio W3C ha prodotto una serie di linguaggi standard per la rappresentazione della conoscenza che supportano elaborazioni più o meno sofisticate (e computazionalmente complesse): questi linguaggi rappresentano gli strati della cosiddetta "Semantic Web Tower", in cui ogni strato è definito come un'*estensione semantica* [1] dello strato sottostante. Alla base della torre (escludendo il ruolo puramente sintattico esercitato da XML) si trova RDF, che consente di formulare delle asserzioni (*statement*) riguardo un insieme di *risorse* e le *proprietà* da cui sono descritte.

Sebbene la sintassi di RDF sia interpretabile da parte di un essere umano, la maggior parte degli utenti del Web Semantico non ha familiarità con essa e nemmeno con la semantica di concetti astratti che fanno parte del vocabolario di OWL, quali 'restrizione', 'specializzazione', 'cardinalità' e così via: per questo, gli utenti dovrebbero disporre di strumenti in grado di (1) tradurre insiemi di asserzioni RDF in rappresentazioni grafiche facili da interpretare, e (2) tradurre le azioni compiute su di esse in interrogazioni sul documento sorgente.

Un problema rilevante, nella costruzione di un ambiente interattivo per la visualizzazione e la navigazione di documenti RDF, consiste nell'individuare il compromesso ideale tra *potere espressivo* e *indipendenza dal dominio applicativo.*

* This work was partially funded by MIUR under the Wisdom COFIN Project.

[1] http://www.w3.org/TR/rdf-mt/

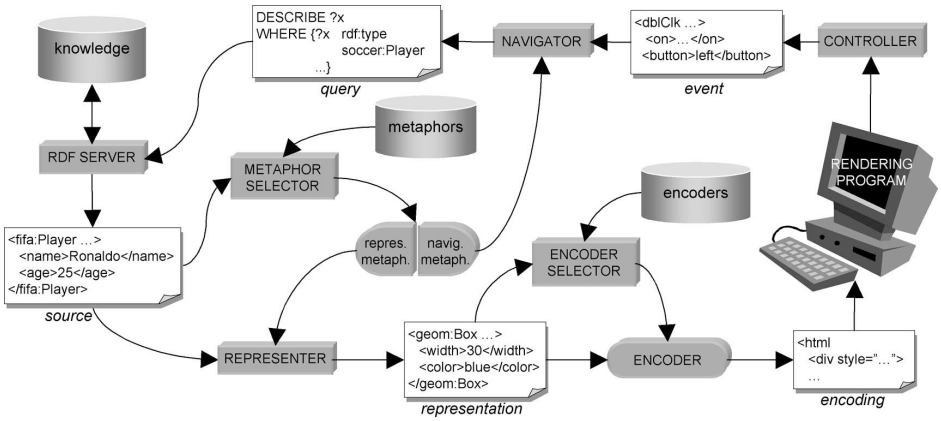

Figura 1. Architettura funzionale di M-FIRE

Un alto potere espressivo è dato dalla capacità di definire, per ciascuna categoria di utenti, le rappresentazioni grafiche e le primitive di navigazione che risultano più intuitive e più semplici da utilizzare, mentre l'indipendenza dal dominio applicativo garantisce la capacità di raggiungere tale obiettivo indipendentemente dal contenuto specifico dei documenti rappresentati. In parole povere, lo scopo è quello di massimizzare la *flessibilità*, istanziando rappresentazioni specifiche tramite un'applicazione generica.

La maggior parte degli strumenti esistenti privilegiano l'indipendenza dal dominio e, di conseguenza, rappresentano risorse e proprietà nella forma grafica più vicina alla forma astratta con cui esse sono definite: le rappresentazioni generate da questi strumenti, solitamente, ricalcano il modello concettuale di un documento RDF inteso come un grafo, in cui gli elementi visivi rappresentano entità – classi, proprietà, relazioni di specializzazione e di istanziamento – che sono familiari ad una categoria di utenti molto ristretta, ma non agli esperti del dominio applicativo. Al contrario, i formalismi visuali specifici per un singolo dominio risentono di una minore riusabilità, ma permettono, attraverso l'impiego di costrutti grafici specializzati, di generare una rappresentazione delle entità descritte in una forma più vicina a quella con cui vengono percepite dagli utenti.

Un possibile approccio al problema consiste nel disaccoppiare il meccanismo che trasforma un documento RDF in una sua rappresentazione dai criteri che guidano il processo stesso di rappresentazione. In questo articolo presentiamo M-FIRE, un framework configurabile e originale per istanziare in modo semplice sistemi di visualizzazione e di navigazione per documenti RDF, basato sull'adozione di *metafore* personalizzate. Le metafore guidano il processo che consente di ottenere una rappresentazione visuale di un dato documento, e definiscono come generare le interrogazioni in risposta alle azioni compiute dall'utente. Questo consente di realizzare un *browsing semantico* facendo affidamento su rappresentazioni intuitive di informazioni complesse, con cui l'utente possa interagire tramite semplici operazioni.

L'architettura funzionale di M-FIRE è riassunta in Figura 1. Innanzitutto, presupponiamo di disporre di un *Server RDF* che, data un'interrogazione in un linguaggio supportato (SPARQL, RDQL, RQL, ARQ, ...), restituisca un documento RDF, che chiameremo *documento sorgente*. Un modulo chiamato *metaphor selector* analizza il documento sorgente e, in base al suo contenuto, restituisce la *metafora* più opportuna. Diversi criteri possono guidare la scelta della metafora migliore, tra cui il vocabolario utilizzato nel documento sorgente e l'eventuale profilo dell'utente finale. La metafora comprende una *metafora di rappresentazione* e una *metafora di navigazione*. La metafora di rappresentazione viene data in input al *representer*, che applica le direttive in essa contenute per generare un *documento di rappresentazione*; quest'ultimo descrive in una forma astratta, indipendente da qualsiasi linguaggio di formattazione, la rappresentazione grafica che deve essere prodotta. Quindi, un programma opportunamente scelto (*encoder*) traduce il documento di rappresentazione in una forma concreta, detta *encoding* (ad esempio, un documento HTML, VRML o SVG), che possa essere data in input al programma di visualizzazione utilizzato dall'utente finale (ad esempio, un browser Web). La scelta dell'encoder migliore per tradurre un dato documento di rappresentazione compete all'*encoder selector*. Anche in questo caso, il vocabolario del documento di rappresentazione potrebbe essere un valido criterio di scelta: alcuni costrutti grafici non sono supportati da certi formati – per esempio, HTML non fornisce una primitiva per il disegno di cerchi – per cui la presenza nel documento di rappresentazione di uno di questi costrutti precluderebbe la scelta dei formati che non lo supportano e dei relativi encoder.

Una volta completata la visualizzazione dell'encoding da parte del programma in uso, l'utente può interagire con essa. Gli eventi generati dall'utente vengono catturati da un modulo chiamato *event controller*, che ne produce una descrizione astratta nella forma di un documento RDF. La descrizione dell'evento riporta soltanto il tipo di azione effettuata dall'utente (clic, doppio clic, drag&drop, ...) e l'insieme di asserzioni sulla cui rappresentazione l'utente ha agito, senza alcun riferimento alla natura di tale rappresentazione. Essa viene data in input al *navigator*, insieme alla metafora di navigazione precedentemente scelta. Così come, per un dato documento sorgente, la metafora di rappresentazione guida il representer nel produrre un documento di rappresentazione, allo stesso modo la metafora di navigazione indica al navigator quale interrogazione formulare in corrispondenza di un certo evento. L'interrogazione ottenuta viene quindi inoltrata al server RDF e il processo si ripete.

Questo articolo ha l'obiettivo di illustrare come venga realizzato, in M-FIRE, il meccanismo che consente di tradurre un documento RDF in una visualizzazione grafica, approfondendo in particolare gli aspetti di maggior rilievo legati alle fasi di rappresentazione e di encoding.

2 Stato dell'arte

Attualmente esistono diversi strumenti general-purpose [7,4,2,6,3,1] che evidenziano la semantica di alcune risorse fondazionali appartenenti al vocabolario di

RDF(S) o di OWL ed offrono agli utenti alcune primitive di navigazione, perlopiù con l'effetto di espandere i nodi di un grafo. Questa limitazione è ragionevole, dal momento che i formalismi in questione devono essere in grado di rappresentare il più ampio numero possibile di documenti RDF, indipendentemente dal dominio applicativo. D'altra parte, falliscono nell'intento di fornire agli utenti una rappresentazione intuitiva delle entità descritte, mancando di conseguenza l'obiettivo primario del nostro approccio.

Jambalaya [5] costituisce un passo in avanti significativo verso lo sviluppo di strumenti altamente configurabili, in quanto la semantica di un insieme di informazioni viene visualizzata tenendo conto di parametri non banali specificati dall'utente. In Jambalaya, la primitiva grafica di contenimento viene utilizzata, di default, per codificare la semantica sia delle relazioni di sussunzione che di quelle di istanziamento; tuttavia, agli utenti è consentito modificare questo comportamento associando qualsiasi proprietà alla primitiva di contenimento. Il nostro approccio generalizza quello di Jambalaya, in quanto permette di rappresentare qualsiasi struttura semantica per mezzo di qualsiasi struttura grafica.

Nel contesto dell'ambiente visuale IsaViz[2] per il browsing e l'editing di documenti RDF, un linguaggio chiamato Graph Stylesheets (GSS) viene definito per associare informazioni di stile ai nodi e agli archi di un grafo RDF. Rispetto al nostro approccio, GSS ricopre al contempo tre ruoli: quello delle metafore di rappresentazione, perchè permette di associare stili grafici ad alcuni pattern semantici; quello di un vocabolario per i documenti di rappresentazione, in quanto comprende un insieme predefinito di costrutti grafici; infine, quello di un formato di encoding perché esiste un programma di visualizzazione che ne interpreta direttamente il contenuto (IsaViz). Il nostro approccio mira a disambiguare questi aspetti introducendo una netta separazione tra le diverse funzionalità, e d'altro canto generalizza quello di GSS in quanto il vocabolario grafico non è limitato a priori.

FRESNEL[3] è un semplice vocabolario per specificare quali parti di un grafo RDF devono essere visualizzate e come, utilizzando linguaggi di formattazione già esistenti quali CSS. Esistono diverse anaologie tra M-FIRE e FRESNEL, ma anche differenze significative. Innanzitutto, il paradigma di rappresentazione di FRESNEL è incentrato sulle *risorse*, mentre quello di M-FIRE è incentrato sulle *asserzioni*: vale a dire che in FRESNEL le rappresentazioni sono generate per i singoli oggetti, mentre in M-FIRE sono associate a (insiemi di) triple RDF. L'approccio basato su asserzioni è più generale: rappresentare l'esistenza di una risorsa X equivale a rappresentare l'asserzione X rdf:type rdf:Resource (viceversa, in generale, per ottenere l'equivalente della rappresentazione di un'asserzione tramite la rappresentazione di una singola risorsa occorre un processo di reificazione). Inoltre, al contrario di quanto avviene in FRESNEL, in M-FIRE i criteri di navigazione (definiti dalle metafore di navigazione) sono mantenuti separati da quelli di rappresentazione (definiti dalle metafore di rappresentazione).

[2] http://www.w3.org/2001/11/IsaViz/
[3] http://www.w3.org/2005/04/fresnel-info/

3 Rappresentazione

La *rappresentazione* è il processo attraverso il quale viene generato un *documento di rappresentazione* per un dato documento sorgente; il documento di rappresentazione descrive un insieme di oggetti grafici, ciascuno dei quali rappresenta un insieme di asserzioni appartenenti al documento sorgente. Sia il documento sorgente che il documento di rappresentazione sono dei *documenti RDF*, secondo la definizione (semplificata) che segue.

Definizione 1 *Un'*asserzione *è una tripla* $\langle \text{sogg}, \text{pred}, \text{ogg} \rangle$ *dove il* soggetto *sogg è una* risorsa *identificata da un* IRI*, il* predicato *pred è una* proprietà *anch'essa identificata da un IRI, e l'*oggetto *ogg è o una risorsa identificata da un IRI o un* literal.[4] *Un* documento RDF *d è un insieme di asserzioni, e il suo* vocabolario $Voc(d)$ *è l'unione degli insiemi di IRI e di literal che figurano nelle asserzioni che lo compongono.*

I criteri che guidano il processo di rappresentazione sono definiti all'interno di una *metafora di rappresentazione* che esprime l'associazione tra strutture semantiche di un certo tipo (*pattern RDF*) e strutture grafiche di un certo tipo (*template RDF*). Le seguenti definizioni sono necessarie ai fini di un'analisi più approfondita.

Definizione 2 *Un* pattern asserzionale *è un'asserzione in cui* nomi di variabile *possono sostituire IRI e literal nel ruolo di soggetto, oggetto e predicato. Indichiamo l'insieme dei nomi di variabile che compaiono in un pattern asserzionale sp come* $Var(sp)$*. Un* pattern RDF *p è un insieme di pattern asserzionali, e l'insieme dei nomi di variabile che compaiono in p è dato da* $Var(p) = \bigcup_{sp_i \in p} Var(sp_i)$*. Dati un pattern RDF p e un documento RDF d, diciamo che d* corrisponde *a p se e solo se esiste un* binding*, ossia una funzione* $\beta : Var(p) \rightarrow Voc(d)$*, tale che d possa essere ottenuto sostituendo in p ogni nome di variabile v con* $\beta(v)$*.*

Definizione 3 *Un* template asserzionale *è un pattern asserzionale in cui* template di risorsa *possono figurare nel ruolo di soggetto e oggetto al posto di IRI e literal. Un template di risorsa è un nome che designa un ruolo astratto all'interno del pattern, che può essere istanziato da diverse risorse. Indichiamo l'insieme dei template di risorsa che figurano in un template asserzionale st come* $Tem(st)$*. Un* template RDF *t è un insieme di template asserzionali, e l'insieme dei template di risorsa che compaiono in t è dato da* $Tem(t) = \bigcup_{st_i \in t} Tem(st_i)$*. Dati un template RDF t e un documento RDF d, diciamo che d* istanzia *t se e solo se esistono due funzioni* $\tau^T : Tem(t) \rightarrow Voc(d)$ *e* $\tau^V : Var(t) \rightarrow Voc(d)$ *tale che d possa essere ottenuto da t sostituendo ogni template di risorsa r con* $\tau^T(r)$ *e ogni nome di variabile v con* $\tau^V(v)$*.*

In concreto, una metafora di rappresentazione definisce quale template RDF istanziare per ogni insieme di asserzioni nel documento sorgente che corrisponde a un certo pattern RDF. Formalmente:

[4] http://www.w3.org/RDF

Definizione 4 *Una metafora di rappresentazione* elementare *è una coppia* $\mathrm{mr} = \langle \mathrm{p}, \mathrm{t} \rangle$*, in cui p è un pattern RDF, t è un template RDF e ogni variabile in t compare anche in p (*$Var(t) \subseteq Var(p)$*).*

Una metafora di rappresentazione elementare $mr = \langle p, t \rangle$ viene applicata ad un documento sorgente d dapprima identificando gli insiemi di asserzioni S_i in d che corrispondono a p e i relativi binding β_i, quindi creando un'istanza del template t per ciascun S_i tale che (1) τ_i^V coincida con β_i e (2) per ogni template di risorsa t, $\tau_i^T(t) \neq \tau_j^T(t)$. L'unione delle istanze di t generate per ciascun S_i costituisce il *documento di rappresentazione*.

Example 1. Nel linguaggio di definizione delle metafore, i nomi di variabile sono preceduti da un punto interrogativo (?x, ?y, ?z) e i template di risorsa da un carattere '#' (#b, #f); La clausola FOR PATTERN dichiara il pattern RDF da utilizzare per la corrispondenza, mentre la clausola GENERATE dichiara il template RDF da istanziare. La metafora di rappresentazione elementare SoccerPlayerAsBox crea un documento di rappresentazione in cui, per ogni giocatore di calcio che milita in una squadra italiana, figura un rettangolo blu contenente il nome di quel giocatore:

```
METAPHOR SoccerPlayerAsBox
GENERATE { #b rdf:type geom:Box . #b geom:hasColor Blue . #b geom:hasText ?z . }
FOR PATTERN { ?x rdf:type soccer:Goalscorer . ?x misc:playsIn ?y .
              ?x misc:hasName ?z . ?y misc:hasNation geo:italy . }
```

La metafora di rappresentazione elementare NationAsFlag, invece, rappresenta una nazione tramite un'immagine con bordo sottile che ne raffigura la bandiera:

```
METAPHOR NationAsFlag
GENERATE { #f rdf:type geom:Img . #f geom:src ?z . #f geom:border "thin" . }
FOR PATTERN { ?x rdf:type geo:Nation . ?x geo:hasFlag ?y . ?y misc:hasFile ?z }
```

□

Le metafore di rappresentazione possono essere riutilizzate in modo da comporre metafore più complesse. Una metafora di rappresentazione *composta* estende una o più metafore di rappresentazione unendo secondo un certo criterio i documenti RDF da queste generati. Più precisamente, l'unione avviene istanziando un particolare template RDF ogniqualvolta i documenti istanziati dalle metafore componenti soddisfino un certo predicato di collegamento, espresso per mezzo di un pattern RDF.

Example 2. La metafora di rappresentazione composta SoccerPlayerWithNationality estende le regole di rappresentazione elementari illustrate nell'Esempio 1 in modo da creare un documento di rappresentazione in cui, per ogni giocatore di calcio che milita in una squadra italiana, figura un rettangolo blu contenente il nome di quel giocatore collegata tramite una linea alla rappresentazione della sua nazione di provenienza.

```
METAPHOR SoccerPlayerWithNationality
EXTENDS SoccerPlayerAsBox P, NationAsFlag N
GENERATE { #l rdf:type geom:Line . #l geom:from #P.b . #l geom:to #N.i . }
FOR PATTERN { ?P.x soccer:comesFrom ?N.x . }
```

La clausola EXTENDS dichiara la lista delle metafore componenti e i relativi alias. Come si evince dall'esempio, nel pattern RDF e nel template RDF di una metafora composta è possibile fare riferimento, rispettivamente, alle variabili (?P.x, ?N.x) e ai template di risorsa (#P.b, #N.i) utilizzati all'interno delle metafore componenti. □

4 Encoding

Il documento di rappresentazione descrive in forma astratta la visualizzazione che deve essere prodotta: il vocabolario del documento di rappresentazione non è fissato a priori, sicché la forma concreta più idonea ad implementare la visualizzazione descritta deve essere determinata analizzandone il contenuto. In linea di principio, diversi formati potrebbero prestarsi alla codifica di un documento di rappresentazione: due cerchi colorati connessi da una linea tratteggiata potrebbero essere realizzati sia in GraphML che in SVG; una tabella contenente nomi e foto di persone potrebbe essere implementata in SVG o in HTML.

La Figura 2 mostra la visualizzazione di due documenti ottenuti applicando diverse metafore allo stesso documento sorgente e fornendo in input ad un encoder HTML i documenti di rappresentazione risultanti.

Un punto cruciale per quanto concerne il processo di encoding riguarda la traduzione delle *annotazioni semantiche*. Le annotazioni semantiche vengono generate dal representer mettendo in corrispondenza insiemi di asserzioni all'interno del documento di rappresentazione con gli insiemi di asserzioni che essi rappresentano all'interno del documento sorgente. Se il processo di rappresentazione può essere visto come una mappatura dal dominio concettuale a quello grafico, le annotazioni semantiche sono le informazioni di mappatura che consentono di risalire dal dominio grafico a quello concettuale. È di fondamentale importanza incorporare queste annotazioni all'interno dell'encoding: in caso contrario risulterebbe impossibile, da parte dell'event controller, generare una descrizione delle azioni compiute dall'utente che comprenda un riferimento all'informazione rappresentata; a sua volta, questo precluderebbe la possibilità di formulare delle primitive di navigazione per mezzo di una metafora.

5 Conclusioni

In questo articolo abbiamo presentato M-FIRE, un approccio originale alla visualizzazione e alla navigazione di documenti RDF basato sull'utilizzo di metafore per istanziare presentazioni differenziate delle stesse informazioni. Riteniamo che il punto di forza di M-FIRE sia la sua capacità di offrire agli utenti una visualizzazione espressiva e specializzata senza perdere in riusabilità, il che costituisce

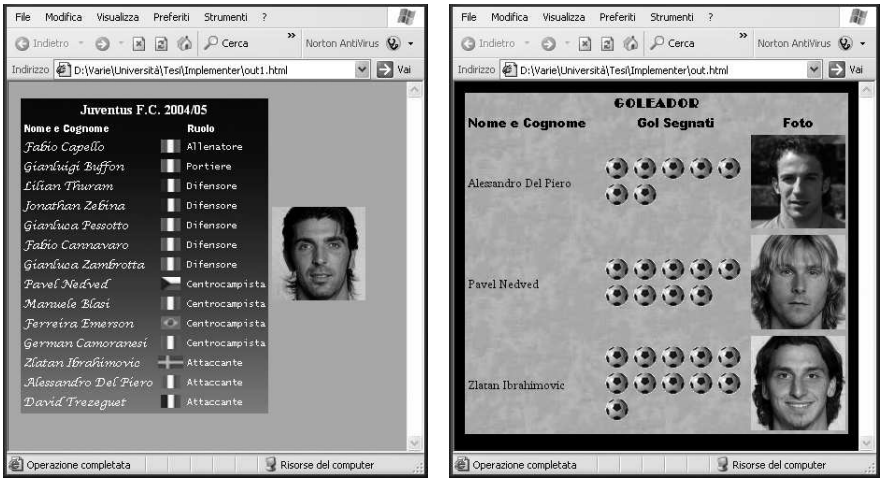

Figura 2. Output generato da un encoder HTML per documenti di rappresentazione ottenuti applicando metafore differenti allo stesso documento sorgente

un passo in avanti significativo rispetto allo stato dell'arte. Anche le primitive di navigazione sono definite per mezzo di metafore, completando così il framework in un approccio unificato alla fruizione della conoscenza.

Per provare l'efficacia di M-FIRE è stato implementato, utilizzando la libreria open-source Jena, un prototipo del metaphor selector, del representer, dell'encoder selector, e sono stato sviluppati un encoder per HTML ed uno per GraphML.

Riferimenti bibliografici

1. J. Domingue, E. Motta, and O. Garcia. *Knowledge Modelling in WebOnto and OCML: A User Guide*. Knowledge Media Institute, Milton Keynes, UK, 1999.
2. A. Farquhar, R. Fikes, W. Pratt, and J. Rice. Collaborative ontology construction for information integration, 1995.
3. G. W. Furnas. Generalized fisheye views. *SIGCHI Bull.*, 17(4):16–23, 1986.
4. F. V. Harmelen et al. Ontology-based information visualisation. In *Proc. Workshop on Visualisation of the Semantic Web*, pages 546–554, 2001.
5. M. Storey et al. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In *Proc. Workshop on Interactive Tools for Knowledge Capture*, Victoria, Canada, 2001.
6. B. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proc. 10th Knowledge Acquisition Workshop*, Banff, Canada, 1996.
7. R. Volz, D. Oberle, S. Staab, and B. Motik. KAON SERVER - a semantic web management system. In *Proc. WWW*, Budapest, Hungary, 2003.