Ingegneria della conoscenza

Appunti dalle lezioni 2004-05 – Versione 1.2, 11.04.2005

Marco Colombetti, Politecnico di Milano

Parte I: Le logiche descrittive e il linguaggio OWL

1. L'ingegneria della conoscenza

Il termine *ingegneria della conoscenza* (*knowledge engineering*, KE) è stato coniato da E. Feigenbaum (1977). Intesa come ramo applicativo dell'intelligenza artificiale, la KE si occupa del progetto, della realizzazione e del mantenimento di *sistemi basati su conoscenze* (*knowledge based system*, KBS) e in particolare di *sistemi esperti* (vedi ad es. Stefik, 1995). In generale, un sistema basato su conoscenze è un sistema informatico in grado di sfruttare le informazioni contenute in una *base di conoscenze* (*knowledge base*, KB) mediante procedure automatiche di *ragionamento*.

In prima approssimazione, la conoscenza può essere definita come *informazione disponibile per l'azione* (Dretske, 1981). È solo grazie alle loro conoscenze che gli esseri umani possono agire, e in particolare agire in modo *razionale*, ovvero basando le loro azioni su ragioni. Anche nei computer risiedono grandi quantità di informazioni, ma solo una piccola parte di queste può essere sfruttata dagli stessi computer per agire: basta pensare alla miriade di documenti contenuti del web, la maggior parte dei quali può essere compresa e utilizzata soltanto dagli esseri umani.

L'obiettivo di questo corso è esplorare la possibilità di automatizzare certe azioni che oggi richiedono l'intervento umano, perché i computer non riescono a sfruttare appieno le informazioni disponibili.

1.2 Tipi di conoscenze

Dicevamo che la conoscenza è informazione disponibile per l'azione. Una prima distinzione va tracciata fra *conoscenze dichiarative* e *conoscenze procedurali*, ovvero fra *conoscere* (*knowing that*) e *saper fare* (*knowing how*)[1]. Si *conosce* la storia romana e si *sa* nuotare ("so nuotare" in inglese è "I *can* swim", ovvero "*sono capace* di nuotare"). "Conoscere" o "sapere" il Java significa conoscere il linguaggio e saper programmare con il linguaggio.

In questo corso ci concentriamo sul conoscere, sul *knowimg that*. A sua volta, il *knowing that* può essere diviso in tre parti:

- *conoscenze terminologiche*: conoscenza del lessico di una lingua; ad esempio, so che "madre" significa "donna con almeno un figlio";
- *conoscenze nomologiche*: conoscenza di regolarità, di leggi generali che regolano il mondo; ad esempio so che le madri sono sempre più anziane dei loro figli, che le madri in genere amano i loro figli, e così via;
- *conoscenze fattuali*: conoscenza di fatti particolari; ad esempio, so che Marco è figlio di Laura.

Un caso solo apparentemente diverso è costituito dalla conoscenza di *individui* (nel senso di oggetti individuali, concreti o astratti, viventi o non viventi, animati o inanimati); ad esempio, "conosco Barbara", "conosco la Nona di Beethoven". Anche in questo caso si tratta di conoscenze fattuali, più precisamente di un insieme spesso molto ampio di conoscenze fattuali relative a uno specifico individuo (Barbara, la Nona di Beethoven).

---

[1] Questa distinzione è stata analizzata per la prima volta da Gilbert Ryle (1940).

1.3 Fonti di conoscenza

Le conoscenze di un soggetto provengono da varie fonti, fra cui le più ovvie sono:

- *l'esperienza diretta*, ovvero l'interazione del soggetto con il suo ambiente;
- *il ragionamento*: deduttivo (dalle premesse alle conclusioni), abduttivo (dagli effetti osservati alle possibili cause), induttivo (dai fatti specifici alle regole generali);
- *la comunicazione*, ovvero l'uso di sistemi di segni e in particolare del linguaggio verbale per trasferire informazioni da un soggetto a un altro.

Qualunque sia la fonte, poi, è essenziale la funzione della *memoria*, nel senso di capacità di conservare nel tempo elementi di conoscenza e soprattutto di reperirli con efficienza quando occorre farne uso.

1.4 Il ruolo della conoscenza nei sistemi informatici

Le conoscenze servono per:

- *interpretare la realtà*: capire che cosa è successo (ad es., ipotizzare le cause di un evento che si è verificato);
- *prevedere l'evoluzione della realtà*: prevedere con approssimazione accettabile quali eventi si verificheranno nel futuro;
- *agire in modo razionale modificando la realtà*: costruire piani d'azione per raggiungere determinati obiettivi.

A noi interessano in particolare gli usi adatti a un sistema informatico. Di questo argomento si è sempre occupata l'intelligenza artificiale: è quindi utile tracciare una storia minima di quest'area dell'informatica.

Prima fase (1956–1965): l'intelligenza artificiale e la risoluzione automatica di problemi

L'intelligenza artificiale (IA o AI, da *artificial intelligence*) nasce nel 1956 con l'obiettivo di progettare sistemi in grado di risolvere problemi automaticamente. All'inizio l'attenzione è rivolta soprattutto alla ricerca di metodi per la risoluzione automatica di problemi mediante strategie per tentativi ed errori.

Seconda fase (1965–1975): la rappresentazione delle conoscenze

Ci si rende presto conto che la capacità di risolvere problemi dipende in modo essenziale dalle conoscenze a disposizione. Il problema principale diventa rappresentare le conoscenze in modo tale che un programma possa effettivamente utilizzarle; nasce quindi un grande interesse per i metodi di *rappresentazione delle conoscenze*. Si sviluppa una controversia fra i sostenitori delle *rappresentazioni dichiarative* e i sostenitori delle *rappresentazioni procedurali* (vedi il par. 1.2). In ambedue i casi, comunque, l'approccio adottato è *simbolico*, nel senso che le conoscenze sono rappresentate tramite strutture dati contenenti simboli (come ad es. le formule della logica).

Terza fase (1975–1985): l'ingegneria della conoscenza e i sistemi esperti

Ci si sposta verso applicazioni realistiche e d'interesse industriale. Aumenta l'investimento economico nel settore e si sviluppa l'ingegneria della conoscenza. Rappresentazioni dichiarative e procedurali convivono negli stessi sistemi basati su conoscenze.

Quarta fase (1985–1995): la crisi e la ricerca di strade alternative

Crolla la fiducia del mercato nei sistemi basati su conoscenze, anche a causa di un incauto *overselling* e dell'impossibilità di soddisfare le attese. Si indagano approcci alternativi, denominati spesso "subsimbolici" o "non simbolici", in contrasto con l'approccio simbolico tradizionale (ad es. le reti neurali, la computazione evolutiva e così via, noti anche come metodi di *soft computing*). Questi metodi si rivelano più adatti dei metodi simbolici per certi tipi di problemi (ad es. l'analisi di segnali, l'apprendimento automatico e così via), ma in molte applicazioni non eliminano la necessità di rappresentare conoscenze simboliche.

Quinta fase (1995–): il web e le ontologie

Il grande successo del web riporta in prima linea, in forma nuova, la questione della rappresentazione simbolica delle conoscenze. In questo campo i problemi più sentiti sono l'*interoperabilità delle applicazioni* e la *gestione delle risorse in rete*. Il concetto fondamentale è quello di *ontologia*, che ha cominciato a precisarsi negli anni novecentottanta e che tratteremo in dettaglio in questo corso.

2. L'ingegneria della conoscenza e il web semantico

Con i metodi dell'ingegneria della conoscenza sono stati sviluppati diversi tipi di applicazioni d'interesse industriale. A partire dal 1975 e per tutti gli anni novecentottanta le applicazioni più diffuse sono stati i sistemi esperti. Un sistema esperto è un sistema software in grado di risolvere problemi utilizzando conoscenze specialistiche; applicazioni tipiche sono l'analisi di grandi volumi di dati, la diagnosi dei guasti di impianti industriali, la diagnosi medica, la configurazione di sistemi informatici complessi, il progetto industriale, la pianificazione della produzione e così via.

In certi casi l'adozione di sistemi esperti ha permesso di automatizzare attività complesse là dove tecniche informatiche più tradizionali avevano fallito. Tuttavia i sistemi esperti non sono mai diventati prodotti di grande diffusione, e questo per almeno due ragioni:

- lo sviluppo di un sistema esperto è un'impresa lunga e costosa, che può essere affrontata soltanto per applicazioni di nicchia in cui sia possibile investire una notevole quantità di risorse;
- i sistemi esperti richiedono una notevole competenza anche per l'uso e soprattutto per il mantenimento, e proprio perché si basano su conoscenze specialistiche non possono essere facilmente trasferiti da un'area applicativa a un'altra.

Oggi la tecnologia dei sistemi esperti continua a rivestire interesse in alcuni settori applicativi, ma è improbabile che possa dar luogo a prodotti di vasta diffusione.

A partire dagli anni novecentonovanta ha cominciato a profilarsi un altro settore di grande interesse per l'ingegneria della conoscenza: il web semantico (Daconta *et al.*, 2003; Antoniou, van Harmelen, 2004). Prima di occuparci di questo concetto tracceremo una breve descrizione di alcuni fattori che stanno influenzando il progresso dell'informatica.

2.1 Alcuni sviluppi recenti dell'informatica

Da più di un decennio assistiamo a una significativa rivoluzione nel campo dell'informatica, cui le tecnologie del web stanno dando un notevole contributo. A grandi linee i punti centrali di questa rivoluzione sono: la centralità dei dati, l'interoperabilità delle applicazioni e l'utilizzo delle tecnologie del web in reti di diverse tipologie.

La centralità dei dati

Qualsiasi applicazione informatica è costituita da dati e programmi; le due componenti non hanno però lo stesso peso in applicazioni di tipo differente.

Consideriamo come primo esempio un'applicazione per la simulazione di sistemi dinamici. In un caso del genere i dati si possono ridurre a un vettore che rappresenti lo stato iniziale del sistema e all'assegnamento di un valore a un piccolo numero di parametri del sistema. Sulla base di questi dati è possibile condurre complesse simulazioni per indagare, ad esempio, l'evoluzione nel tempo dello stato del sistema. Per ottenere risultati di qualità, naturalmente, sono estremamente critici gli algoritmi utilizzati per la simulazione; ci troviamo quindi di fronte a un tipo di applicazione che potremmo denominare *centrata sul processo*.

Un esempio molto diverso è costituito dalle applicazioni che si fondano sull'utilizzo di basi di dati. In questo caso l'informazione utile risiede essenzialmente nei dati, e i processi si limitano spesso a reperire i dati necessari e a fornirne una presentazione adeguata. Le applicazioni di questo tipo possono essere denominate *centrate sui dati*.

Naturalmente non avrebbe senso chiedersi quale tipo di applicazione informatica sia più importante: a seconda delle necessità si utilizzerà un sistema centrato sul processo o centrato sui dati. È vero però che le applicazioni centrate sui dati sono sempre più diffuse in termini percentuali: lo stesso web si presenta essenzialmente come un colossale archivio di dati interrogato quotidianamente da milioni di utenti. Non è quindi scorretto affermare che negli ultimi anni l'interesse dell'informatica si è gradualmente spostato dai sistemi centrati sul processo ai sistemi centrati sui dati.

L'interoperabilità "interna"

Per molti anni le aziende hanno sviluppato e utilizzato applicazioni "chiuse", che operano su dati per lo più in formato proprietario. Questo tipo di applicazioni crea problemi già all'interno della singola azienda, perché in una situazione del genere è difficile far interagire applicazioni distinte; a maggior ragione è arduo combinare applicazioni appartenenti ad aziende diverse. All'interno di una singola azienda è possibile raggiungere un buon livello d'interoperabilità utilizzando le tecnologie basate su XML; in particolare le tecnologie del web basate su XML consentono di sviluppare intranet aziendali integrando dati e *servizi web* (*web services*, WS) rivolti verso l'interno dell'azienda. Quando però si ha l'esigenza di superare i confini di un'intranet queste tecnologie si rivelano spesso insufficienti.

L'interoperabilità "esterna" e i sistemi aperti

Da qualche anno è nato un forte interesse per le applicazioni "aperte", in grado cioè di interoperare con applicazioni appartenenti ad aziende diverse (all'interno di un'extranet che connetta aziende federate) o con applicazioni comunque distribuite su internet. Ad esempio, i servizi web possono essere utilizzati anche all'interno di un'intranet o di un'extranet, ma è chiaro che le loro maggiori potenzialità si rivelano al livello di internet.

Per realizzare applicazioni aperte a livello di internet le tecnologie basate su XML sono oggi necessarie ma non sufficienti. Per essere realmente "aperto" un sistema informatico deve poter essere *scoperto* e *utilizzato* da un'applicazione remota senza bisogno dell'intervento umano: a questo scopo XML non è sufficiente, perché codifica e standardizza la sintassi dei dati (ovvero la loro struttura), ma non la loro semantica (ovvero, il significato dei dati). Ad esempio, un documento XML può codificare il prezzo di un prodotto con la stringa

```
<prezzo>85</prezzo>,
```

mentre un altro documento può codificare la stessa informazione con la stringa

```
<costo>85</costo>.
```

Non è ovviamente possibile per un'applicazione informatica scoprire che si tratta di due diverse codifiche della stessa informazione, a meno che non sia disponibile la conoscenza del fatto che "prezzo" e "costo" sono sinonimi, almeno nel contesto dell'applicazione in oggetto (si tratta di un elemento di conoscenza terminologica, vedi il par. 1.2). Non è azzardato a questo punto passare alla seguente conclusione: l'interoperabilità in un ambiente aperto richiede che le applicazioni possano accedere a un repertorio di conoscenze comuni e siano in grado di sfruttare tali conoscenze in modo autonomo (ovvero senza intervento umano). Questo fatto porta oggi in primo piano sia le tecnologie basate su XML, sia i modelli e le tecniche sviluppati nell'ultimo trentennio dall'ingegneria della conoscenza.

2.2 Il web semantico

Come è noto, l'invenzione del web è attribuita a Tim Berners-Lee. Nella concezione di Berners-Lee, tuttavia, il web come lo conosciamo oggi è solo il primo gradino di una scala che prevede al gradino successivo il *web semantico*. L'idea è semplice: perché possano essere elaborati in modo completamente automatico, non è sufficiente che i dati sul web abbiano una struttura chiaramente definita (come è possibile fare utilizzando XML): è anche necessario che i dati abbiano un significato condiviso dalla comunità che li utilizza.

Per consentire lo sviluppo del web in questa direzione il W3C ha proposto e sta tuttora elaborando un insieme di tecnologie standardizzate (RDF, RDFS, OWL, ...) che analizzeremo in dettaglio durante il corso.

2.3 Il triangolo aristotelico

L'idea del web semantico è dunque centrata sull'assegnamento di un *significato* ai dati presenti sul web. Il concetto di significato, a sua volta, chiama in causa il rapporto fra *linguaggio* e *realtà*. Come tutti sappiamo, molti termini del linguaggio corrispondono a entità del mondo reale: ad esempio, il termine "seggiola" (un sostantivo dell'italiano) può corrispondere all'insieme di tutte le seggiole, oppure a una seggiola specifica (un individuo membro dell'insieme delle seggiole). La relazione fra linguaggio e realtà (e quindi fra il termine "seggiola" e l'insieme delle seggiole concrete) è detta *relazione semantica*.

Secondo una tradizione antica (che nella storia del pensiero occidentale si può far risalire ad Aristotele), il rapporto fra linguaggio e realtà non è diretto, ma è mediato dai *concetti*: chi parla l'italiano riesce a collegare la parola "seggiola" alle seggiole reali proprio perché possiede un concetto di seggiola; In altre parole, la relazione fra linguaggio e realtà è mediata dalla *mente*. Più precisamente, dobbiamo distinguere fra:

- il *termine* "seggiola" dell'italiano ("chair" in inglese e così via);
- l'*insieme di tutte le seggiole* che esistono nella realtà (l'*estensione* del termine "seggiola");
- il *concetto* di seggiola così come è rappresentato nella mente degli esseri umani (l'*intensione* del termine "seggiola").

La relazione fra queste entità è schematizzata nei triangoli riportati nella figura 2.1.

Il gergo dell'ingegneria della conoscenza spesso confonde i tre vertici del triangolo: così, una definizione di "seggiola" sarà indifferentemente denominata "termine", "concetto" o "classe". Questo abuso linguistico è innocuo purché si tenga in mente la distinzione appena introdotta; a rigore, comunque, nei sistemi informatici sarebbe preferibile parlare sempre di *termini*, dato che un gli oggetti informatici sono comunque simboli appartenenti a un linguaggio formale.

3. I sistemi per la rappresentazione delle conoscenze

Come abbiamo visto, il nostro obiettivo principale è rendere certe conoscenze disponibili ai sistemi informatici; il problema, quindi, è come rappresentare le conoscenze in un formato che sia *computer readable* (nel senso che un computer possa "leggere" le conoscenze e soprattutto utilizzarle per eseguire certi compiti d'interesse applicativo). Dato che questo corso non ha una prospettiva storica, ci piazzeremo subito all'interno della problematica delle applicazioni al web (vedi il par. 1.4), e concentreremo la nostra attenzione soltanto sulle rappresentazioni *simboliche* e *dichiarative*. Il paradigma di riferimento per le rappresentazioni dichiarative è la *logica simbolica* (o *formale*), con particolare riguardo alla *logica dei predicati del primo ordine* (*first order logic*, FOL).

Figura 2.1. Triangoli aristotelici

In FOL si assume che tutte le rappresentazioni riguardino un insieme non vuoto (e per il resto arbitrario) di individui, detto *universo*[2]. Di questi individui possiamo rappresentare *proprietà* oppure *relazioni* che li leghino fra loro; da questo punto di vista, un fatto è semplicemente il sussistere di una proprietà di un determinato individuo ("Barbara è bionda") oppure il sussistere di una relazione fra più individui ("Alberto è più alto di Barbara", "Alberto ha dato il suo cellulare a Barbara", ecc.). Nei sistemi di rappresentazione che esamineremo si fanno assunzioni analoghe, con l'ulteriore limitazione che le relazioni prese in considerazione sono esclusivamente binarie.

Ogni metodo per la rappresentazione delle conoscenze si basa su due componenti principali: un *linguaggio per la rappresentazione di conoscenze* e un insieme di *procedure di ragionamento*.

3.1 Linguaggi di rappresentazione

Un linguaggio per la rappresentazione di conoscenze è un linguaggio formale, con sintassi testuale o grafica, le cui *espressioni* sono utilizzate per rappresentare elementi di conoscenza.

Come esempio vogliamo rappresentare il significato del termine "madre" come "donna con almeno un figlio". Per utilizzare FOL parafrasiamo la definizione di "madre" come:

(*x* è una madre) se e solo se (*x* è una donna ed esiste almeno un *y* tale che *y* è figlio di *x*).

Possiamo esprimere questa definizione usando una notazione testuale, come ad esempio il linguaggio di FOL solitamente presentato nei libri di logica[3]:

$\forall x$ (MADRE(x) $\leftrightarrow$ DONNA(x) $\wedge$ $\exists y$ Figlio(x,y)),

dove:

$\forall x$ ...	per ogni *x*, ... (il *quantificatore universale*)
$\exists y$ ...	esiste un *y* tale che ... (il *quantificatore esistenziale*)
... $\leftrightarrow$ ...	... se e solo se ... (il *connettivo bicondizionale*)
... $\wedge$ ...	... e ... (il *connettivo di congiunzione*)
MADRE(x), DONNA(x)	due *predicati monoargomentali*, che esprimono proprietà di *x*
Figlio(x,y)	un *predicato biargomentale*, che esprime una relazione binaria fra *x* e *y*

Più spesso, nelle applicazioni informatiche si incontrano notazioni testuali puramente ASCII, che non utilizzano simboli matematici come $\forall$ e $\exists$. Ad esempio in CL (*Common Logic Standard*, una proposta di standard ISO per la logica predicativa, `http://cl.tamu.edu/`) la stessa definizione apparirebbe come:

```
(forall (?x)
  (iff (MADRE ?x)
       (and (DONNA ?x)
            (exists (?y) (Figlio ?x ?y)))))
```

Le espressioni possono anche essere scritte in un formato adatto allo scambio dei dati fra applicazioni software: tipicamente si tratta di documenti strutturati secondo un opportuno DTD o schema XML. Ad esempio, seguendo il DTD proposto per CL, alla formula precedente corrisponde il documento XML di seguito riportato:

---

[2] Nei testi di logica ciò che qui chiamiamo *universo* è generalmente detto *dominio*. Preferiamo utilizzare il termine "universo" per evitare confusioni con il *dominio di un ruolo* (vedi il par. 5).

[3] Per i simboli di FOL si è usato il font Symbol.

```
<formula>
  <forall>
    <varlist>
      <var>x</var>
    </varlist>
    <iff>
      <atom>
        <con>MADRE</con>
        <var>x</var>
      </atom>
      <and>
        <atom>
          <con>DONNA</con>
          <var>x</var>
        </atom>
        <exists>
          <varlist>
            <var>y</var>
          </varlist>
          <atom>
            <con>Figlio</con>
            <var>x</var>
            <var>y</var>
          </atom>
        </exists>
      </and>
    </iff>
  </forall>
</formula>
```

Indipendentemente dalla notazione prescelta (classica, ASCII, XML, ...) occorrerà comunque che le espressioni utilizzate abbiano un significato definito in modo rigoroso e privo di ambiguità.

Nella logica, la semantica delle espressioni è definita utilizzando modelli insiemistici e la stessa via può essere seguita per le altre notazioni utilizzate. Alternativamente è possibile assegnare una traduzione dalle notazioni non classiche alla notazione classica e contare poi sul fatto che la notazione classica ha una semantica precisamente definita.

3.2 Ragionamento automatico

Nel contesto in cui ci stiamo muovendo, per “ragionamento” s’intende il *ragionamento deduttivo* o *deduzione*. Una deduzione è un processo che fa passare da alcune espressioni (dette *premesse* o *ipotesi*) a un’espressione (detta *conclusione* o *tesi*), in modo tale da *conservare l’eventuale verità delle premesse*: in altre parole, se le premesse sono vere, lo sarà anche la conclusione.

Ad esempio, dati come premesse

la definizione di “madre”,

il fatto che Laura è una donna,

il fatto che Marco è figlio di Laura,

si può dedurre come conclusione che

Laura è una madre.

In FOL la deduzione può essere rappresentata in vari modi, ad esempio tramite la prova seguente (il metodo di prova utilizzato è noto come “calcolo della deduzione naturale”; l’ultima riga della prova rappresenta la conclusione):

1. $\forall x$ (MADRE(x) $\leftrightarrow$ DONNA(x) $\wedge$ $\exists y$ Figlio(x,y)) premessa
2. DONNA(laura) premessa
3. Figlio(laura,marco) premessa
4. MADRE(laura) $\leftrightarrow$ DONNA(laura) $\wedge$ $\exists y$ Figlio(laura,y) da 1, per eliminazione di $\forall$
5. $\exists y$ Figlio(laura,y) da 3, per introduzione di $\exists$
6. DONNA(laura) $\wedge$ $\exists y$ Figlio(laura,y) da 2 e 3, per introduzione di $\wedge$
7. MADRE(laura) da 4 e 6, per eliminazione di $\leftrightarrow$

Utilizzando FOL si possono esprimere conoscenze molto articolate e, almeno in linea di principio, eseguire in modo automatico ragionamenti complessi. C’è però un problema: in FOL la procedura di deduzione (detta anche *procedura di prova* o *calcolo*) non è una procedura di decisione, ma soltanto di *semidecisione*. Ciò significa che:

- quando la conclusione è deducibile delle premesse, la procedura termina in un numero finito di passi producendo una prova;
- quando invece la conclusione non è deducibile delle premesse, la procedura può non terminare.

In parole povere, un sistema informatico può “andare in ciclo” se tenta di dedurre da un insieme di premesse una conclusione che in effetti non è deducibile delle premesse.

L’indecidibilità di FOL dipende dalla notevole espressività del linguaggio: come è naturale, più un linguaggio di rappresentazione è espressivo, più sono problematiche le procedure di ragionamento basate su di esso. Molte ricerche nel campo dei linguaggi di rappresentazione delle conoscenze hanno l’obiettivo di identificare un sottolinguaggio di FOL tale che: il linguaggio sia comunque abbastanza espressivo per le applicazioni; la deduzione si basi su una procedura di decisione (che quindi termini in ogni caso dopo un numero finito di passi, sia quando la conclusione è deducibile dalle premesse, sia quando non lo è); la procedura di decisione abbia complessità computazionale accettabile (ovvero richieda una quantità accettabile di risorse di calcolo). I sistemi di questo tipo hanno preso il nome di *logiche descrittive* (*description logic*, DL). Le DL utilizzano una sintassi semplificata rispetto a FOL. Ad esempio, le tre premesse

1. $\forall x$ (MADRE(x) $\leftrightarrow$ DONNA(x) $\wedge$ $\exists y$ Figlio(x,y))
2. DONNA(laura)
3. Figlio(laura,marco)

in logica descrittiva verrebbero rappresentate come[4]:

1. MADRE $\equiv$ DONNA $\sqcap$ $\exists$Figlio
2. DONNA(laura)
3. Figlio(laura,marco)

Nel mondo delle applicazioni si usano spesso rappresentazioni di tipo grafico: ad esempio, l’espressione MADRE $\equiv$ DONNA $\sqcap$ $\exists$Figlio si può rappresentare come indicato nella figura 3.1. In alternativa è possibile utilizzare una notazione testuale ASCII, come

```
(define-concept MADRE (and DONNA (some Figlio))).
```

Si è inoltre diffusa l’abitudine di rappresentare le espressioni direttamente con la sintassi XML utilizzata per lo scambio di dati fra applicazioni software.

[4] Per i simboli delle DL si è utilizzato il font Lucida Math Symbol.

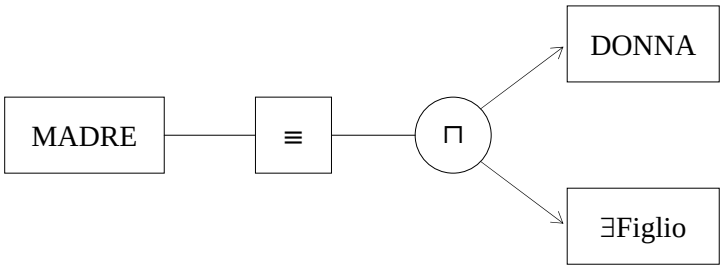

Figura 3.1. Forma grafica di un'espressione logico-descrittiva.

3.3 Le rappresentazioni

In svariate aree dell'informatica si utilizzano sistemi di rappresentazione per scopi ed usi differenti. Sono ad esempio sistemi di rappresentazione le basi di dati, i diagrammi UML, i documenti XML. Al di là delle ovvie differenze, tutti i sistemi di rappresentazione condividono alcune caratteristiche comuni, che cercheremo qui di analizzare.

Il primo concetto da analizzare è la distinzione fra *realtà* e *rappresentazione*. Ogni rappresentazione, infatti, è necessariamente la rappresentazione di qualcosa, e questo "qualcosa" è un frammento della realtà interessante per qualche applicazione. A sua volta, una rappresentazione può essere analizzata a tre livelli distinti:

- Livello 1: *modello concreto*. Il modello concreto di un frammento di realtà è la rappresentazione dei fatti che sussistono nel frammento di realtà (conoscenze fattuali sul frammento di realtà); ad esempio, "Marco Colombetti è un docente di una facoltà di ingegneria del Politecnico di Milano".
- Livello 2: *modello concettuale*. Il modello concettuale di un frammento di realtà è una rappresentazione dei concetti utilizzati per formulare il modello concreto (conoscenze terminologiche e nomologiche); ad esempio, per esprimere il fatto dell'esempio precedente è necessario definire i concetti di persona (dotato di nome e cognome), di docente, di facoltà e così via.
- Livello 3: *metamodello*. Il metamodello di un sistema di rappresentazione è la specifica degli strumenti formali utilizzabili per definire il modello concettuale e il modello concreto; ad esempio: il linguaggio di FOL, oppure il formalismo degli schemi E-R, oppure il formalismo del diagramma delle classi UML, e così via.

Nella figura 3.2 presentiamo i tre livelli per alcune tipologie di rappresentazione.

	basi di dati	*specifiche SW*	*logica*	*RDF*	*DL*
metamodello	il formalismo E-R	il formalismo dei diagrammi UML	la logica predicativa del primo ordine	il formalismo N3 (triple RDF)	una particolare DL
modello concettuale	un modello concettuale E-R	un diagramma delle classi	una teoria del primo ordine	un documento RDFS	una TBox
modello concreto	una base di dati	le istanze delle classi	il "diagramma" di un modello (ovvero, le formule atomiche prive di variabili che descrivono un particolare modello)	un documento RDF	un'ABox

Figura 3.2. I tre livelli delle rappresentazioni.

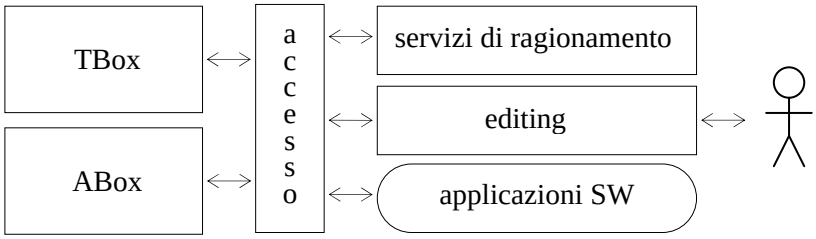

Figura 3.3. L'architettura di un KRS.

3.4 I sistemi per la rappresentazione delle conoscenze

In che cosa, allora, un KRS si distingue, poniamo, da una rappresentazione nel campo delle basi dei dati o di UML? La differenza principale sta in questo:

- nei campi delle basi di dati e della specifica del software, i modelli concettuali (E-R, UML ecc.) rappresentano un passo importante nello sviluppo di un sistema, ma non fanno parte del prodotto finale;
- nei KRS, invece, i modelli concettuali sono parte integrante del prodotto finale, in quanto sono disponibili e vengono utilizzati a run time dai sistemi software.

Un KRS, infatti, è costituito da cinque componenti principali (fig. 3.3):

- una *terminological box* (*TBox*), che rende disponibile e accessibile la rappresentazione formale del modello concettuale di un frammento di realtà;
- un'*assertion box* (*ABox*), che rende disponibile e accessibile la rappresentazione formale del modello concreto di un frammento di realtà;
- un'*interfaccia di accesso*, che consente l'accesso alle conoscenze contenute nella TBox e nella ABox da parte di applicazioni software;
- un insieme di *servizi di ragionamento* (*reasoning services*), ovvero un'applicazione software in grado di dedurre nuove conoscenze dalle conoscenze contenute nella TBox e nella ABox;
- un'*interfaccia di editing*, ovvero un'applicazione software che consente la gestione dei contenuti della TBox e della ABox da parte di un operatore umano.

Nei KRS odierni, l'interoperabilità con le varie applicazioni software è assicurata dall'adozione sistematica di standard basati su XML. Ciò consente, in particolare, di utilizzare strumenti sviluppati e mantenuti da fornitori differenti. In questo corso faremo riferimento alla libreria Jena come interfaccia di accesso, a Protégé 2000 come interfaccia di editing e a Racer come supporto per i servizi di ragionamento.

4. Primi elementi di logica descrittiva

Come si è già detto, gli odierni linguaggi per la definizione di ontologie sono basati su logiche descrittive (DL); ad esempio, il linguaggio DAML+OIL è basato su una DL detta $\mathcal{SHIQ}$, mentre il linguaggio OWL (standard W3C) è basato su una DL detta $\mathcal{SHOIN}(\mathrm{D}_n)$ (vedi il par. 7). A sua volta ogni DL si caratterizza per l'utilizzo di un certo numero di *operatori logici*, scelti da un repertorio di operatori possibili. In questo paragrafo introdurremo alcuni operatori di uso comune.

4.1 Termini, equivalenze e sussunzioni

Il linguaggio formale di una DL serve a definire *termini* che descrivono concetti. Ad esempio, l'espressione

DONNA

è un *termine atomico*, che intuitivamente significa “donna”. L’espressione

(4.1) PERSONA ⊓ FEMMINA

è un *termine complesso*, che si legge “persona e femmina” o “persona intersezione femmina” e intuitivamente significa “persona di genere femminile”.

Nelle DL è d’uso indicare generici termini atomici con le lettere *A* e *B*, e generici termini arbitrari (cioè atomici o complessi) con le lettere *C* e *D*. Inoltre i termini sono spesso chiamati *concetti* (perché descrivono concetti) o *classi* (perché denotano insiemi di oggetti della realtà).

L’espressione

(4.2) DONNA ≡ PERSONA ⊓ FEMMINA,

che intuitivamente significa “donna equivale a persona femmina”, è detta *equivalenza terminologica*. In generale un’equivalenza terminologica

$C \equiv D$,

che si legge “*C* equivale a *D*”, esprime appunto l’equivalenza fra i due termini *C* e *D*. Nel caso particolare in cui si abbia

$A \equiv C$

(con *A* atomico) si parla di *definizione terminologica*. Dunque la 4.2 è una definizione terminologica: più precisamente si tratta della definizione del termine DONNA a partire dai termini PERSONA e FEMMINA. L’espressione

(4.3) RAGAZZA ⊑ DONNA,

che si legge “ragazza è sussunto da donna” e intuitivamente significa “una ragazza è una donna”, è detta *sussunzione terminologica*. In generale una sussunzione terminologica, $C \sqsubseteq D$, esprime il fatto che ogni individuo descritto dal termine *C* è anche descritto dal termine *D*. Come è lecito attendersi, l’equivalenza $C \equiv D$ coincide con la doppia sussunzione $C \sqsubseteq D$ e $D \sqsubseteq C$.

4.2 Ontologie

Chiameremo *enunciati terminologici*, le espressioni che esprimono equivalenze o sussunzioni fra termini. Enunciati di questo genere possono essere assunti come *assiomi terminologici*, in modo analogo a quanto si fa nelle teorie del primo ordine: un assioma terminologico è semplicemente un enunciato terminologico assunto come vero. Chiamiamo poi *terminologia* od *ontologia* un insieme finito di assiomi terminologici; un’ontologia non è altro, quindi, che una teoria del primo ordine esprimibile in una DL (per una grammatica degli enunciati terminologici utilizzati in queste note si veda l’Appendice I).

La funzione di un’ontologia è definire le relazioni di equivalenza e di sussunzione che sussistono fra un certo numero di termini. In particolare le ontologie assegnano un significato non ambiguo a un certo numero di termini atomici, in base al significato di altri termini; spesso, infatti, un termine atomico viene definito in funzione di altri termini, i quali a loro volta acquistano un significato grazie a ulteriori definizioni terminologiche, e così via. Questo processo, tuttavia, non può essere continuato all’infinito: dato che ogni ontologia è finita, prima o poi si arriva a termini privi di una definizione, che vanno considerati come *primitivi*. Ogni ontologia, quindi, si basa su un insieme più o meno grande di termini atomici primitivi, di cui non si dà alcuna definizione.

4.3 Semantica dei termini e degli enunciati terminologici

Termini ed enunciati terminologici sono espressioni di un linguaggio formale. Per utilizzare correttamente tali espressioni è necessario attribuire a tali espressioni un significato.

Per le espressioni di una DL è possibile specificare una semantica formale, che associa a ogni termine e a ogni enunciato terminologico un’interpretazione definita in modo insiemistico (per una definizione della semantica formale secondo queste linee si veda l’Appendice II). In queste note seguiremo una

strada diversa: assoceremo una semantica ai termini in modo indiretto, fornendo una traduzione di termini ed enunciati nel linguaggio della logica predicativa del primo ordine (FOL). Non si deve però dimenticare una cosa: mentre a un termine o enunciato terminologico è sempre possibile associare una formula di FOL, non è vero l'inverso: le DL sono infatti sottoinsiemi propri di FOL, e quindi esistono formule di FOL che non corrispondono ad alcun termine o enunciato terminologico (fig. 5.1).

Traduzione di termini

Intuitivamente, ogni termine atomico o complesso esprime un *predicato monàdico*, ovvero una proprietà che ciascun individuo di un universo prefissato può possedere o non possedere. In FOL i predicati sono rappresentati da formule dotate di *esattamente una variabile libera* (con una o più occorrenze nella formula); pertanto, la traduzione di un termine atomico o complesso in FOL è costituita da una formula con esattamente una variabile libera. D'ora in avanti, se C è un generico termine indicheremo con $[C]_x$ la sua traduzione in FOL, in cui occorre x come unica variabile libera.

Ogni termine atomico A si traduce con una formula del tipo $A(x)$, dove $A(–)$ è un simbolo predicativo a un argomento e x è una variabile. Naturalmente è possibile utilizzare come traduzione qualsiasi *variante alfabetica* di $A(x)$, come ad esempio $A(y)$. Stabilita questa regola abbiamo le traduzioni:

$[\text{DONNA}]_x = \text{DONNA}(x)$,

$[\text{PERSONA}]_x = \text{PERSONA}(x)$,

$[\text{FEMMINA}]_x = \text{FEMMINA}(x)$.

Consideriamo ad esempio la traduzione del termine DONNA nella formula predicativa DONNA(x). Dato un universo prefissato di individui, ogni termine determina un'*estensione*, definita come l'insieme di tutti gli individui dell'universo cui il termine si applica. Ad esempio, se l'universo coincide con la totalità degli individui presenti in un'aula durante una lezione, l'estensione in quell'universo del termine DONNA è costituito da tutte le donne presenti nell'aula. Ora, questi individui sono appunto quei possibili valori della variabile x che rendono vera la formula predicativa DONNA(x). L'idea generale è questa: dato un termine atomico A, la semantica del termine ci deve consentire di identificare l'estensione di A nell'universo; tale estensione è rappresentata da tutti e soli gli individui dell'universo che rendono vera la formula $A(x)$ quando siano assunti come valori della x. Considerazioni analoghe varranno poi per i termini complessi.

Nelle DL sono solitamente utilizzati due termini atomici predefiniti, $\top$ (*top*) e $\bot$ (*bottom*), che corrispondono rispettivamente al *termine universale* (che denota la totalità degli individui esistenti nell'universo) e al *termine vuoto* (che denota l'insieme vuoto di individui). Questi due termini si traducono in FOL rispettivamente con una formula che sia vera di ogni possibile valore della x, e con una formula che sia falsa per ogni possibile valore della x; ad esempio:

$[\top]_x = (x = x)$,

$[\bot]_x = (x \neq x)$.

L'operatore d'intersezione $\sqcap$ corrisponde al connettivo booleano di congiunzione; pertanto il termine complesso 4.1 si traduce come:

$$[\text{PERSONA} \sqcap \text{FEMMINA}]_x = [\text{PERSONA}]_x \wedge [\text{FEMMINA}]_x$$
$$= \text{PERSONA}(x) \wedge \text{FEMMINA}(x),$$

dove l'operatore '$\wedge$' è il connettivo di congiunzione.

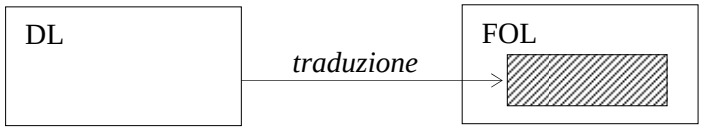

Figura 5.1 Traduzione da DL a FOL.

Traduzione di enunciati terminologici

Gli enunciati terminologici si traducono in FOL come formule chiuse (ovvero, prive di variabili libere) quantificate universalmente. Nel seguito indicheremo rispettivamente con $[C \sqsubseteq D]$ e $[C \equiv D]$ le traduzioni in FOL di $C \sqsubseteq D$ e di $C \equiv D$. Le regole di traduzione sono le seguenti:

$[C \sqsubseteq D] = \forall x\,([C]_x \rightarrow [D]_x)$,

$[C \equiv D] = \forall x\,([C]_x \leftrightarrow [D]_x)$,

dove gli operatori '$\rightarrow$' e '$\leftrightarrow$' sono rispettivamente il connettivo condizionale e il connettivo bicondizionale. Ad esempio, gli enunciati 4.3 e 4.2 si traducono come

$$[\text{RAGAZZA} \sqsubseteq \text{DONNA}] = \forall x\,(([\text{RAGAZZA}]_x \rightarrow [\text{DONNA}]_x)$$
$$= \forall x\,(\text{RAGAZZA}(x) \rightarrow \text{DONNA}(x)),$$
$$[\text{DONNA} \equiv \text{PERSONA} \sqcap \text{FEMMINA}] = \forall x\,(([\text{DONNA}]_x \leftrightarrow [\text{PERSONA} \sqcap \text{FEMMINA}]_x)$$
$$= \forall x\,(\text{DONNA}(x) \leftrightarrow \text{PERSONA}(x) \wedge \text{FEMMINA}(x))^5.$$

4.4 Negazione e disgiunzione

Gli uomini sono il complemento delle donne rispetto alla totalità delle persone. Utilizzando l'operatore $\neg$ di complemento (o negazione) è quindi possibile definire:

$\text{UOMO} \equiv \text{PERSONA} \sqcap \neg\text{FEMMINA}$,

la cui traduzione in FOL è

$[\text{UOMO} \equiv \text{PERSONA} \sqcap \neg\text{FEMMINA}] = \forall x\,(\text{UOMO}(x) \leftrightarrow \text{PERSONA}(x) \wedge \sim\text{FEMMINA}(x))$,

dove $\sim$ è il connettivo booleano di negazione. Un altro operatore utile è l'operatore $\sqcup$ di unione (o disgiunzione), che ci permette ad esempio di definire gli esseri viventi come unione di vegetali e animali:

$\text{VIVENTE} \equiv \text{VEGETALE} \sqcup \text{ANIMALE}$.

La traduzione è

$[\text{VIVENTE} \equiv \text{VEGETALE} \sqcup \text{ANIMALE}] = \forall x\,(\text{VIVENTE}(x) \leftrightarrow \text{VEGETALE}(x) \vee \text{ANIMALE}(x))$,

dove $\vee$ è il connettivo booleano di disgiunzione (*or* non esclusivo). Si noti che gli operatori $\neg$, $\sqcap$ e $\sqcup$ formano un'algebra booleana, con $\top$ come top e $\bot$ come bottom; in particolare:

$\neg\top$ equivale a $\bot$,

$\neg\neg C$ equivale a C,

$\neg(C \sqcap D)$ equivale a $\neg C \sqcup \neg D$,

$\neg(C \sqcup D)$ equivale a $\neg C \sqcap \neg D$.

5. I ruoli

Oltre ai termini corrispondenti a predicati con un argomento (detti, come abbiamo visto, concetti o classi), le DL utilizzano termini corrispondenti a predicati a due argomenti, che esprimono relazioni binarie fra individui della realtà; tali termini vengono detti *ruoli*, o *proprietà*, o *attributi*, o *relazioni*.

---

[5] Per ridurre le coppie di parentesi nelle formule di FOL assumiamo la seguente gerarchia di precedenza fra i connettivi binari: $\wedge > \vee > \rightarrow > \leftrightarrow$; inoltre gli operatori unari (il connettivo $\sim$ e i quantificatori) si applicano sempre alla più piccola sottoformula seguente.

5.1 I quantificatori

Un esempio di enunciato terminologico che utilizza un ruolo è

(5.1) MADRE $\sqsubseteq \exists$Figlio,

che intuitivamente significa "ogni madre ha un figlio". L'espressione

(5.2) $\exists$Figlio

è un termine complesso, formato dal *quantificatore esistenziale* '$\exists$' e dal *ruolo* Figlio. Anche in questo caso abbiamo una regola di traduzione in FOL che ci fornisce una formula dotata di un'unica variabile libera:

$$[\exists \text{Figlio}]_x = \exists y\ \text{Figlio}(x,y),$$

dove Figlio(–,–) è un predicato a due argomenti, che esprime una relazione binaria fra elementi dell'universo. Si noti che la formula FOL $\exists y$ Figlio(x,y) contiene *due* variabili, x e y: la variabile y, tuttavia, è *vincolata* dal quantificatore esistenziale $\exists$ di FOL, e pertanto x è l'unica variabile libera della formula. A questo punto è possibile dare la traduzione della 5.1:

$$\begin{aligned}[\text{MADRE} \sqsubseteq \exists\text{Figlio}] &= \forall x\ ([\text{MADRE}]_x \rightarrow [\exists\text{Figlio}]_x) \\ &= \forall x\ (\text{MADRE}(x) \rightarrow \exists y\ \text{Figlio}(x,y)).\end{aligned}$$

In alcune DL il quantificatore esistenziale può essere *qualificato*; ad esempio, il termine

(5.3) $\exists$Figlio.FEMMINA

denota l'insieme di tutti gli individui dell'universo che hanno almeno una figlia. La traduzione in FOL di un termine della forma $\exists R.C$, dove R è un ruolo e C è un termine arbitrario, è:

$$[\exists R.C]_x = \exists y\ (R(x,y) \wedge [C]_y).$$

Ad esempio il termine 5.2 si traduce come:

$$\begin{aligned}[\exists\text{Figlio.FEMMINA}]_x &= \exists y\ (\mathit{Figlio}(x,y) \wedge [\text{FEMMINA}]_y) \\ &= \exists y\ (\mathit{Figlio}(x,y) \wedge \text{FEMMINA}(y)).\end{aligned}$$

È facile verificare che l'espressione $\exists R$ è equivalente all'espressione $\exists R.\top$; infatti:

$$[\exists R.\top]_x = \exists y\ (R(x,y) \wedge [\top]_y) = \exists y\ (R(x,y) \wedge (y = y)) = \exists y\ R(x,y).$$

Al di là della semantica formale, che ovviamente definisce il significato delle espressioni DL in modo rigoroso e privo di ambiguità, è importante sviluppare la capacità di leggere i termini in modo intuitivo ma corretto. Un errore comune consiste nel leggere i termini 5.2 e 5.3 rispettivamente come "esiste un figlio" ed "esiste un figlio femmina". Queste due letture sono errate, in quanto portano a trattare un termine come se esprimesse una *proposizione*; non bisogna invece dimenticare che un termine rappresenta un *predicato monàdico*, o se si preferisce l'insieme degli individui che ne costituisce l'estensione. Pertanto il termine 5.2 va letto come "x ha almeno un figlio" oppure come "l'insieme degli individui che hanno almeno un figlio", e il termine 5.3 come "x ha almeno una figlio femmina" oppure come "l'insieme degli individui che hanno almeno un figlio femmina".

Le DL prevedono anche l'uso del quantificatore universale. Ad esempio, il termine

(5.4) $\forall$Figlio.FEMMINA

denota l'insieme di tutti gli individui dell'universo i cui figli (se esistono) sono tutti femmine. La traduzione in FOL di un termine della forma $\forall R.C$, dove R è un ruolo arbitrario e C è un termine arbitrario, è:

$$[\forall R.C]_x = \forall y\ (R(x,y) \rightarrow [C]_y).$$

Ad esempio il termine 5.4 si traduce come:

$$\begin{aligned}[\forall\text{Figlio.FEMMINA}]_x &= \forall y\ (\mathit{Figlio}(x,y) \rightarrow [\text{FEMMINA}]_y) \\ &= \forall y\ (\mathit{Figlio}(x,y) \rightarrow \text{FEMMINA}(y)).\end{aligned}$$

Anche in questo caso valgono le considerazioni già fatte per il quantificatore esistenziale: il termine 5.4 non equivale alla proposizione “tutti i figli sono femmine”, bensì al predicato monàdico “tutti i figli di *x* sono femmine” o alla sua estensione “l’insieme degli individui i cui figli sono tutti femmine”. Una seconda osservazione importante è che la quantificazione universale, come in FOL, *non* presuppone l’esistenza di almeno un individuo con le proprietà specificate. Ad esempio l’insieme definito dal termine 5.4 comprende sia gli individui dell’universo i cui figli sono tutti femmine, sia gli individui dell’universo *che siano privi di figli*. In altre parole la lettura più completa del termine 5.4 è: “l’insieme degli individui i cui figli, se esistono, sono tutti femmine”.

Mentre il quantificatore esistenziale si usa sia nella forma semplice $\exists R$ (vedi la 5.1), sia nella forma qualificata $\exists R.C$ (vedi la 5.3), il quantificatore universale si usa solo nella forma qualificata $\forall R.C$. Infatti il termine $\forall R$, equivalente come abbiamo visto all’inizio del paragrafo a $\forall R.\top$, coinciderebbe semplicemente con $\top$, nel senso che sarebbe banalmente verificato per ogni individuo dell’universo, in quanto la formula FOL che traduce $[\forall R.\top]_x$,

$$\forall y\,(R(x,y) \rightarrow (y = y)).$$

è banalmente vera per ogni valore di *x* nell’universo degli individui. Si noti infine che:

$\neg\exists R.C$ equivale a $\forall R.\neg C$,

$\neg\exists R$ equivale a $\forall R.\bot$,

$\neg\forall R.C$ equivale a $\exists R.\neg C$.

5.2 Il ruolo inverso

Dato un ruolo *R*, che esprime una relazione binaria $R(x,y)$, è spesso utile poter esprimere il *ruolo inverso* R^-, che esprime semplicemente la relazione $R(y,x)$. Avremo quindi:

$$[\exists R^-.C]_x = \exists y\,(R(y,x) \wedge [C]_y),$$

$$[\forall R^-.C]_x = \forall y\,(R(y,x) \rightarrow [C]_y).$$

Un esempio di uso del ruolo inverso è mostrato nel prossimo sottoparagrafo.

5.3 Dominio e codominio

I ruoli esprimono relazioni binarie fra individui dell’universo. In generale, però, tali relazioni hanno senso solo limitatamente a certi sottoinsiemi dell’universo; ad esempio, il ruolo Figlio mette in relazione fra loro due *persone*, mentre non ha senso se applicato, poniamo, alle pietre o alle nuvole. A un ruolo *R* si associano quindi due insiemi di individui, detti il *dominio* e il *codominio* del ruolo, che rappresentano gli insiemi di individui sui cui pensiamo variare le variabili *x* e *y* nell’espressione $R(x,y)$.

Il dominio *C* e il codominio *D* di un ruolo *R* si possono definire nel modo seguente:

$$\top \sqsubseteq \forall R^-.C,$$

$$\top \sqsubseteq \forall R.D.$$

Infatti:

$$[\top \sqsubseteq \forall R^-.C] = \forall x\,((x = x) \rightarrow \forall y\,(R(y,x) \rightarrow [C]_y)) = \forall x\,\forall y\,(R(y,x) \rightarrow [C]_y)),$$

$$[\top \sqsubseteq \forall R.D] = \forall x\,((x = x) \rightarrow \forall y\,(R(x,y) \rightarrow [D]_y)) = \forall x\,\forall y\,(R(x,y) \rightarrow [D]_y)).$$

Ad esempio, possiamo specificare nel modo seguente il fatto che il ruolo Proprietario mette in relazione una persona con un bene:

$$\top \sqsubseteq \forall \text{Proprietario}^-.\text{PERSONA},$$

$$\top \sqsubseteq \forall \text{Proprietario}.\text{BENE}.$$

5.4 Vincoli di cardinalità

Nella maggior parte delle DL è possibile esprimere sui ruoli vincoli di *cardinalità semplice*, come:

$\leq nR,\ \geq nR,$

oppure di *cardinalità qualificata*, come:

$\leq nR.C,\ \geq nR.C,$

dove n è una costante naturale arbitraria (un intero senza segno). Ad esempio, un individuo con almeno tre figlie si definisce come:

(5.4) GEN-3-F $\equiv \geq$3Figlio.FEMMINA.

Ancora una volta, richiamiamo l'attenzione sul fatto che il termine

$\geq$3Figlio.FEMMINA

non rappresenta la proposizione "esistono almeno tre figlie", bensì il predicato "x ha almeno tre figlie" o la sua estensione "l'insieme degli individui che hanno almeno tre figlie". Le espressioni di questo genere si traducono in FOL nel modo seguente:

$[\leq nR]_x = \exists_{\leq n} y\ R(x,y),$

$[\geq nR]_x = \exists_{\geq n} y\ R(x,y),$

$[\leq nR.C]_x = \exists_{\leq n} y\ (R(x,y) \wedge [C]_y),$

$[\geq nR.C]_x = \exists_{\geq n} y\ (R(x,y) \wedge [C]_y).$

Pertanto, l'enunciato 5.4 si traduce come:

$$[\text{GEN-3-F} \equiv \geq 3\text{Figlio.FEMMINA}] = \forall x\ ([\text{GEN-3-F}]_x \leftrightarrow [\geq 3\text{Figlio.FEMMINA}]_x)$$
$$= \forall x\ (\text{GEN-3-F}(x) \leftrightarrow \exists_{\geq 3} y\ (\text{Figlio}(x,y) \wedge \text{FEMMINA}(y)))$$

Si ricorda che i quantificatori $\exists_{\leq n} x$ e $\exists_{\geq n} x$ si possono definire nel modo seguente nella logica predicativa del primo ordine dotata di uguaglianza:

$\exists_{\leq n} x\ \varphi(x) =_{\text{def}} \forall x_0\ \forall x_1 \ldots \forall x_n\ (\varphi(x_0) \wedge \varphi(x_1) \wedge \ldots \wedge \varphi(x_n) \rightarrow x_0 = x_1 \vee \ldots \vee x_{n-1} = x_n),$

$\exists_{\geq n} x\ \varphi(x) =_{\text{def}} \exists x_1 \ldots \exists x_n\ (\varphi(x_1) \wedge \ldots \wedge \varphi(x_n) \wedge x_1 \neq x_2 \wedge \ldots \wedge x_{n-1} \neq x_n).$

Pertanto, la traduzione completa dell'enunciato 5.4 è:

$$\forall x\ (\text{GEN-3-F}(x) \leftrightarrow \exists y_1\ \exists y_2\ \exists y_3\ (\text{Figlio}(x,y_1) \wedge \text{FEMMINA}(y_1) \wedge \text{Figlio}(x,y_2) \wedge \text{FEMMINA}(y_2)$$
$$\wedge\ \text{Figlio}(x,y_3) \wedge \text{FEMMINA}(y_3) \wedge y_1 \neq y_2\ \wedge\ y_1 \neq y_3\ \wedge\ y_2 \neq y_3).$$

La cardinalità esatta, semplice o qualificata, si può ora esprimere nel modo seguente:

$=nR \quad =_{\text{def}} \leq nR \sqcap \geq nR,$

$=nR.C \quad =_{\text{def}} \leq nR.C \sqcap \geq nR.C.$

Infine si noti che:

$\geq 1R\{.C\}$ equivale a $\exists R\{.C\},$

$\leq 0R\{.C\}$ equivale a $\neg\exists R\{.C\}.$

5.5 I ruoli funzionali

Com'è noto, una *relazione funzionale* è una relazione binaria tale che ogni elemento del dominio è in relazione con *al più un* elemento del codominio. Nelle DL, un ruolo che esprima una relazione funzionale è detto *ruolo funzionale*. Consideriamo ad esempio il ruolo Marito, il cui dominio e il cui codominio si possono definire come segue:

(5.5) $\top \sqsubseteq \forall \text{Marito}^{-}.\text{DONNA},$

(5.6) $\top \sqsubseteq \forall \text{Marito.UOMO}.$

Il ruolo Marito è funzionale perché, nella nostra cultura, ogni donna ha al più un marito (per volta, s'intende). Per esprimere questo fatto è sufficiente affermare che la classe delle donne coincide con la classe degli individui che hanno al più un marito:

(5.7) DONNA $\sqsubseteq \leq 1$Marito.

Si noti che la 5.7 non elimina la necessità di definire comunque il dominio di Marito con la 5.5. Infatti la 5.7 da sola non escluderebbe la possibilità che il dominio di Marito sia più ampio dell'insieme delle donne; questa possibilità è però esclusa dalla 5.5.

Una relazione funzionale è una *funzione* se ogni elemento del dominio è in relazione con *almeno un* elemento del codominio (ovvero, se la relazione è totale sul dominio). Dato che ogni funzione è anche una relazione funzionale, ne segue che ogni funzione mette in relazione ciascun elemento del dominio (detto *argomento* della funzione) con *esattamente un* elemento del codominio (detto *valore* della funzione corrispondente all'argomento). Nelle DL, un ruolo che sia una funzione è detto *ruolo funzione* o semplicemente *funzione*. Ad esempio, è una funzione il ruolo Padre,

$\top \sqsubseteq \forall$Padre^{-}.PERSONA,

$\top \sqsubseteq \forall$Padre.UOMO,

che associa a ogni persona il relativo padre. Il fatto che questo ruolo sia una funzione si esprime come

PERSONA $\sqsubseteq =1$Padre.

Infine, un ruolo R si dice *inversamente funzionale* (*inversamente funzione*) se R^{-} è un ruolo funzionale (un ruolo funzione).

5.6 Sussunzione fra ruoli e proprietà simmetrica

In molte DL è consentito esprimere sussunzioni ed equivalenze fra ruoli con espressioni della forma:

$R \sqsubseteq S$,

$R \equiv S$.

Ad esempio, è possibile dire che un Figlio è una specie di Parente,

Figlio $\sqsubseteq$ Parente,

o che Figlio è l'inverso di Genitore:

Figlio $\equiv$ Genitore^{-}.

La semantica FOL di queste espressioni è ovvia:

$[R \sqsubseteq S] = \forall x\, \forall y\, (R(x,y) \rightarrow S(x,y))$,

$[R \equiv S] = \forall x\, \forall y\, (R(x,y) \leftrightarrow S(x,y))$.

La sussunzione fra ruoli consente poi di imporre a un ruolo R la *proprietà simmetrica* con l'assioma

$R \sqsubseteq R^{-}$.

5.7 Composizione di ruoli e proprietà transitiva

In alcune DL è possibile costruire ruoli complessi utilizzando l'operatore $\circ$ di composizione. Dati due ruoli R ed S, visti come relazioni binarie, il ruolo composto $R \circ S$ rappresenta la relazione:

$(R \circ S)(x,y) = \exists z\, (R(x,z) \wedge S(z,y))$.

La composizione di ruoli è molto utile, ma spesso non viene ammessa perché è problematica per la decidibilità della logica. Se si ha a disposizione l'operatore di composizione di ruoli è possibile imporre a un ruolo R la *proprietà transitiva* con l'assioma:

$(R \circ R) \sqsubseteq R$.

Molte DL (tra cui $\mathcal{SHOIN}(\mathrm{D}_n)$, su cui si basa il linguaggio OWL), pur non consentendo la composizione di ruoli, prevedono comunque la possibilità di *dichiarare* che un ruolo è transitivo:

$Tr(R)$.

Si noti che tale dichiarazione va considerata a tutti gli effetti come un assioma terminologico. Va infine osservato che anche in assenza dell'operazione di composizione di ruoli è comunque possibile scrivere termini equivalenti alle *composizioni quantificate*, in quanto:

$\forall(R \circ S).C$ equivale a $\forall R.(\forall S.C)$,

$\exists(R \circ S).C$ equivale a $\exists R.(\exists S.C)$.

Infatti:

$$
\begin{aligned}
[\forall(R \circ S).C]_x &= \forall y\,((R \circ S)(x,y) \rightarrow [C]_y) \\
&\leftrightarrow \forall y\,(\exists z\,(R(x,z) \wedge S(z,y)) \rightarrow [C]_y) \\
&\leftrightarrow \forall y\,\forall z\,(R(x,z) \wedge S(z,y) \rightarrow [C]_y) \\
&\leftrightarrow \forall z\,\forall y\,(R(x,z) \wedge S(z,y) \rightarrow [C]_y) \\
&\leftrightarrow \forall z\,\forall y\,(R(x,z) \rightarrow (S(z,y) \rightarrow [C]_y)) \\
&\leftrightarrow \forall z\,(R(x,z) \rightarrow \forall y\,(S(z,y) \rightarrow [C]_y)) \\
&= \forall z\,(R(x,z) \rightarrow [\forall S.C]_z) \\
&= [\forall R.(\forall S.C)]_x \\
[\exists(R \circ S).C]_x &= \exists y\,((R \circ S)(x,y) \wedge [C]_y) \\
&\leftrightarrow \exists y\,(\exists z\,(R(x,z) \wedge S(z,y)) \wedge [C]_y) \\
&\leftrightarrow \exists y\,\exists z\,(R(x,z) \wedge (S(z,y) \wedge [C]_y)) \\
&\leftrightarrow \exists z\,\exists y\,(R(x,z) \wedge (S(z,y) \wedge [C]_y)) \\
&\leftrightarrow \exists z\,(R(x,z) \wedge \exists y\,(S(z,y) \wedge [C]_y)) \\
&= \exists z\,(R(x,z) \wedge [\exists S.C]_z) \\
&= [\exists R.(\exists S.C)]_x
\end{aligned}
$$

6. Nominali, termini enumerativi e domini concreti

6.1 Nominali e termini enumerativi

Nelle ontologie è spesso possibile utilizzare anche nomi d'individui *a*, *b*, *c*, ..., detti comunemente *nominali*. Dati *n* nominali a_1, ..., a_n è possibile definire il *termine enumerativo*

$\{a_1, ..., a_n\}$.

Ad esempio:

(6.1) COLORE-RGB ≡ {rosso, verde, blu}.

L'interpretazione in FOL di un termine enumerativo è la seguente:

$[\{a_1, ..., a_n\}]_x = (x = a_1 \vee ... \vee x = a_n)$.

Con ciò, la semantica della 6.1 è data da:

[COLORE-RGB ≡ {rosso, verde, blu}] =

$\forall x$ (COLORE-RGB(x) ↔ x = rosso ∨ x = verde ∨ x = blu).

L'operatore '{...}' è spesso chiamato *one-of*, perché enumera tutti i valori che la variabile x può assumere.

6.2 Unicità dei nomi ed altre assunzioni

Va sottolineato che (contrariamente a quanto avviene in FOL, ma analogamente a quanto avviene nelle basi di dati) nelle DL si assume a volte (ma non sempre!) l'*unicità dei nomi* (*unique name assumption*): ciò significa che due nominali distinti non possono fare riferimento allo stesso individuo dell'universo. Quest'assunzione è discutibile in quanto è irrealistica nell'ambito del web, che costituisce uno dei contesti applicativi più interessanti per le DL; tuttavia essa è spesso incorporata negli strumenti per il ragionamento (come Racer) e di ciò è necessario tenere conto. In termini logici, se si utilizzano n nominali $a_1, ..., a_n$ l'assunzione di unicità del nome equivale alle $n(n–1)/2$ asserzioni

$a_1 \neq a_2, \quad a_1 \neq a_3, \quad ..., \quad a_{n-1} \neq a_n,$

o più concisamente:

(6.2) $\neq(a_1, ..., a_n)$.

Nel seguito *non* assumeremo in generale l'unicità dei nomi: quando occorra l'assunzione verrà introdotta esplicitamente con un'asserzione della forma 6.2.

Come abbiamo già detto, l'unicità dei nomi viene assunta nelle basi di dati. Altre due assunzioni, sempre tipiche delle basi di dati, sono l'*assunzione di chiusura del dominio* (*domain closure assumption*) e l'*assunzione del mondo chiuso* (*closed world assumption*). L'assunzione di chiusura del dominio (o meglio di *chiusura dell'universo*, nella terminologia adottata in questo corso) consiste nell'ipotesi che l'universo di tutti gli individui contenga soltanto gli individui cui si fa riferimento con un nominale presente nel sistema (ovvero, esistono soltanto gli individui che hanno un nome). Questa assunzione *non viene mai adottata* nel campo delle DL. Dell'assunzione del mondo chiuso, anch'essa estranea alle DL, parleremo nel paragrafo 7.

6.3 Domini concreti

Nelle applicazioni delle DL è spesso importante rappresentare valori costanti, ad esempio numeri interi (con o senza il segno), numeri *floating point*, caratteri, stringhe; gli insiemi di valori di questo genere vengono chiamati *domini concreti*. In queste note considereremo tali valori alla stregua di nominali appartenenti a insiemi denotati da termini atomici come NATURAL, INTEGER, FLOAT, CHARACTER, STRING. Si noti che nelle DL non sono però disponibili le operazioni tipicamente associati a tali insiemi.

Come esempio, consideriamo il ruolo Età, che associa a ogni persona uno e un solo numero naturale (l'età espressa in anni):

$\top \sqsubseteq \forall \text{Età}^{-}.\text{PERSONA},$

$\top \sqsubseteq \forall \text{Età}.\text{NATURAL},$

$\text{PERSONA} \sqsubseteq =1\text{Età}.$

7. TBox e ABox

Come si è detto nel paragrafo 3, un sistema di rappresentazione delle conoscenze consta di una TBox e di una ABox. La TBox contiene assiomi terminologici e definisce un'ontologia; l'ABox contiene invece conoscenze fattuali espresse sotto forma di *asserzioni*.

Nelle DL si possono esprimere diversi tipi di conoscenze fattuali. Sono innanzitutto ammesse asserzioni del tipo

(7.1) $C(a)$,

dove C è un termine arbitrario e a è un nominale, oppure

(7.2) $R(a,b)$,

dove R è un ruolo ed a, b sono nominali (non necessariamente distinti).

Esempi:

MADRE(laura),

DONNA ⊓ ∃Figlio(laura) (dove C = DONNA ⊓ ∃Figlio),

Figlio(laura,marco).

La semantica FOL di $C(a)$ si ottiene dalla traduzione FOL $[C]_x$, sostituendo tutte le occorrenze della variabile x con il nominale a. Indicheremo con $sost(a,x,[C]_x)$ il risultato di tale sostituzione; ad esempio:

[MADRE(laura)] = $sost$(laura,x,[MADRE]$_x$) = $sost$(laura,x,MADRE(x)) = MADRE(laura),

[DONNA ⊓ ∃Figlio(laura)] = $sost$(laura,x,[DONNA ⊓ ∃Figlio]$_x$)

= $sost$(laura,x, DONNA(x) ∧ ∃y Figlio(x,y)),

= DONNA(laura) ∧ ∃y Figlio(laura,y),

La semantica della formula DL $R(a,b)$ è invece data semplicemente dalla formula FOL $R(a,b)$. Ad esempio si ha:

[Figlio(laura,marco)] = Figlio(laura,marco).

Inoltre, nelle DL che come OWL non assumono l'unicità dei nomi l'ABox può contenere asserzioni del tipo

$a = b$,

$a \neq b$,

ed eventualmente abbreviazioni del tipo $\neq(a_1, ..., a_n)$.

Abbiamo ora occasione di discutere l'assunzione del mondo chiuso (vedi il par. 6.2), tipica delle basi di dati e di molti sistemi d'intelligenza artificiale. Tradotta nei nostri termini, l'assunzione del mondo chiuso suonerebbe come segue:

- tutto ciò che è esplicitamente asserito nell'ABox è vero;
- tutto ciò che non è esplicitamente asserito nell'ABox è falso.

Questa assunzione *non* viene adottata nelle DL. Si noti tra l'altro che un'assunzione del genere presuppone la *conoscenza completa* del mondo dell'applicazione: infatti, se tutto ciò che si asserisce è vero e tutto ciò che non si asserisce è falso, non è possibile assumere rispetto a un fatto una posizione neutrale (non si sa se sia vero o falso). La semantica dell'ABox delle DL è invece compatibile con una situazione di *conoscenza parziale*: di alcune asserzioni si sa che sono vere, di altre che sono false, di altre ancora non si sa nulla. Supponiamo ad esempio che nell'universo del discorso siano presenti tre persone, di nome Laura, Marco e Michiko. Potremmo sapere che Laura è una donna e che Marco non lo è, ma non sapere se Michiko sia o non sia una donna. Potremmo quindi asserire nell'ABox

DONNA(laura),

¬DONNA(marco),

e non dire nulla su Michiko, salvo magari che

PERSONA(michiko).

Un'ultima osservazione: l'asserzione ¬DONNA(marco) non va analizzata come

¬(DONNA(marco)),

perché una simile forma non è consentita nell'ABox, bensì come

(¬DONNA)(marco);

abbiamo così un'asserzione della forma $C(a)$, con

C = ¬DONNA,

a = laura.

8. I linguaggi per le ontologie

Come abbiamo già detto, le DL sono linguaggi di rappresentazione adatti alla definizione di ontologie. Fino ad ora, le DL più utilizzate per questo scopo sono note come $\mathcal{SHIQ}$ e $\mathcal{SHOIN}(\mathrm{D}_n)$; vediamo ora che cosa significhino questi acronimi:

- la lettera $\mathcal{S}$ indica la possibilità di scrivere enunciati di equivalenza e di sussunzione utilizzando i termini
  $\top,\ \bot,\ \neg C,\ C \sqcap D,\ C \sqcup D,\ \forall R.C,\ \exists R.C,$
  nonché assiomi di transitività dei ruoli,
  $Tr(R)$;
- la lettera $\mathcal{H}$ (da *role hierarchy*) indica la possibilità di definire relazioni di inclusione fra ruoli:
  $R \sqsubseteq S$;
- la lettera $\mathcal{O}$ (da *one of*) indica la possibilità di definire termini per enumerazione:
  $\{a_1,...,a_n\}$;
- la lettera $\mathcal{I}$ indica la possibilità di definire il ruolo inverso:
  R^-;
- la lettera $\mathcal{N}$ indica la possibilità di definire cardinalità non qualificate:
  $\leq nR,\ \geq nR,\ =nR$;
- la lettera $\mathcal{Q}$ indica la possibilità di definire cardinalità qualificate:
  $\leq nR.C,\ \geq nR.C$ e $=nR.C$;
- infine, D_n indica la possibilità di utilizzare domini concreti.

In queste note abbiamo utilizzato una logica $\mathcal{SHOIN}(\mathrm{D}_n)$ o al più $\mathcal{SHOIQ}(\mathrm{D}_n)$.

Il linguaggio OWL

Il linguaggio OWL (*Web Ontology Language*) è lo standard attualmente proposto dal W3C (`www.w3.org`) per la definizione di ontologie per il web semantico. Sviluppato a partire da DAML+OIL (un linguaggio con logica $\mathcal{SHIQ}$, a sua volta basato sui linguaggi OIL e DAML-ONT), OWL prevede tre livelli di complessità crescente:

- *OWL Lite*, semplice da usare e implementare ma scarsamente espressivo;
- *OWL DL*, che implementa la logica descrittiva $\mathcal{SHOIN}(\mathrm{D}_n)$, abbastanza espressivo ma comunque decidibile e dotato di procedure di ragionamento di complessità nota, approfonditamente studiate e ormai ben ottimizzate;
- *OWL Full*, che consente di definire rappresentazioni che vanno al di là della logica predicativa del primo ordine; OWL Full è molto espressivo ma non decidibile e quindi problematico dal punto di vista della meccanizzazione del ragionamento.

Nel seguito, salvo diversa indicazione, ci occuperemo esclusivamente di OWL DL.

Un'ontologia OWL si articola in una TBox e un'ABox, ambedue rappresentate come grafi RDF e quindi come insiemi di triple RDF. Diversi costrutti di RDFS sono direttamente adottati da OWL (con eventuali limitazioni d'uso in OWL DL e completa libertà d'uso in OWL Full); OWL introduce inoltre costrutti propri, non presenti in RDFS, comunque rappresentati come triple RDF.

La rappresentazione RDF delle ontologie OWL presenta grandi vantaggi per l'interoperabilità delle applicazioni, ma è difficilmente leggibile e alquanto confusiva a causa delle numerose varianti consentite da RDF. D'altra parte gli strumenti per la definizione di ontologie utilizzano in genere interfacce grafiche che nascondo all'utente la rappresentazione RDF. Per questi motivi non adotteremo la sintassi di RDF in questi appunti, ma continueremo a utilizzare i simboli tipici delle DL. Per facilitare l'utilizzo di OWL, tuttavia, nel resto di questo paragrafo introdurremo e useremo sistematicamente la terminologia specifica di questo linguaggio, che spesso si discosta dalla terminologia più diffusa nel campo delle DL.

8.1 Descrizioni di classi

Nel linguaggio OWL i termini sono denominati *descrizioni di classi*, gli operatori per la definizione di termini sono denominati *costruttori di classi* e i ruoli sono denominati *proprietà*. OWL prevede sei tipi di descrizioni di classi:

- identificatore,
- enumerazione,
- restrizione di proprietà,
- intersezione,
- unione,
- complemento.

Identificatore

Ogni descrizione di classe descrive una risorsa di tipo `owl:Class`. Nel caso più semplice la descrizione consta di un *identificatore* della classe (un URI), corrispondente a un termine atomico delle DL. OWL prevede due identificatori predefiniti per la *classe universale* e la *classe vuota*:

- `owl:Thing`, corrispondente a $\top$;
- `owl:Nothing`, corrispondente a $\bot$.

Ogni individuo è istanza della classe `owl:Thing`.

Enumerazione

Una classe finita può essere descritta dall'*enumerazione* (`owl:oneOf`) di tutti gli individui che le appartengono:

$\{a_1,...,a_n\}$.

Ogni nominale (ovvero a_1, ... , a_n, ...) va interpretato come un URI.

Restrizioni di proprietà

In OWL sono dette *restrizioni di proprietà* le descrizioni di classi corrispondenti ai termini $\exists R.C$, $\forall R.C$, $\forall R.\{a\}$, $\leq nR$, $\geq nR$ e $=nR$. Più precisamente:

- la restrizione $\exists R.C$ è denominata `owl:someValuesFrom`;
- la restrizione $\forall R.C$ è denominata `owl:allValuesFrom`;
- la restrizione $\forall R.\{a\}$ è denominata `owl:hasValue`;
- la restrizione $\leq nR$ è denominata `owl:maxCardinality`;
- la restrizione $\geq nR$ è denominata `owl:minCardinality`;
- la restrizione $=nR$ è denominata `owl:cardinality`;

Si noti che le restrizioni di cardinalità non consentono di specificare una classe di appartenenza per i valori della proprietà; s'intende quindi che tali valori appartengono al *range* (codominio) della proprietà, che può essere definito a parte (vedi il par. 8.4). Inoltre, nelle restrizioni $\forall R.C$ e $\exists R.C$ la C può essere una generica descrizione di classe OWL oppure un *tipo di dati* RDF.

Intersezione

Una classe può essere descritta come *intersezione* (`owl:intersectionOf`) di un numero finito di classi:

$C_1 \sqcap ... \sqcap C_n$.

Unione

Una classe può essere descritta come *unione* di (`owl:unionOf`) un numero finito di classi:

$C_1 \sqcup ... \sqcup C_n$.

Complemento

Una classe può essere descritta come *complemento* di (`owl:complementOf`) un'altra classe:

$\neg C$.

8.2 Assiomi di classe

In OWL le definizioni terminologiche della TBox sono dette *assiomi di classe*. OWL prevede tre tipi distinti di assiomi di classe:

- assiomi di sottoclasse
- assiomi di equivalenza
- assiomi di disgiunzione.

Sottoclassi

Fra due descrizioni di classi, C e D, si può definire una relazione di sottoclasse (`rdfs:subClassOf`). Questa relazione, direttamente importata da RDFS, corrisponde alla sussunzione terminologica nelle DL:

$C \sqsubseteq D$.

Equivalenza

Fra due descrizioni di classi, C e D, si può definire una relazione di equivalenza (`owl:equivalentClass`), che corrisponde all'equivalenza terminologica nelle DL:

$C \equiv D$.

Disgiunzione

Fra due descrizioni di classi, C e D, si può dichiarare una relazione di disgiunzione (`owl:disjointWith`), che corrisponde all'equivalenza:

$C \sqcap D \equiv \bot$.

8.3 Le proprietà

Coerentemente con quanto previsto da RDFS, in OWL anche le *proprietà* (corrispondenti ai ruoli nelle DL) possono essere viste come particolari classi. Come tali le proprietà possono avere sottoproprietà ed essere combinate con vari costruttori.

Così come ogni classe di individui è una risorsa di tipo `owl:Class`, tutte le proprietà sono risorse di tipo `rdf:Property`. In OWL le proprietà possono essere risorse di tipo `owl:ObjectProperty` (*proprietà di individui*, cioè fra elementi di classi OWL) oppure `owl:DatatypeProperty` (*proprietà di dati* appartenenti a tipi di dati RDFS). Nel seguito salvo diversa indicazione ci occuperemo soltanto di proprietà di individui.

In OWL si possono specificare i seguenti aspetti relativi alle proprietà:

- identificatore di proprietà,
- dominio e codominio (*range*),
- sottoproprietà,
- proprietà equivalente,
- proprietà inversa.

Identificatore di proprietà

Un *identificatore di proprietà* corrisponde a un ruolo e sarà quindi indicato con R o S.

Dominio e codominio (range)

Di una proprietà possono essere specificati il *dominio* (`rdfs:domain`) e il *codominio* (`rdfs:range`). Tali specifiche sono equivalenti alle sussunzioni:

$\top \sqsubseteq \forall R^{-}.C$,

$\top \sqsubseteq \forall R.C$.

Sottoproprietà

Una proprietà può essere definita come *sottoproprietà* di (`rdfs:subPropertyOf`) un'altra proprietà:

$R \sqsubseteq S$.

Proprietà equivalente

Una proprietà può essere definita come *equivalente* (`owl:equivalentProperty`) a un'altra proprietà:

$R \equiv S$.

Proprietà inversa

Data una proprietà R si può definire la *proprietà inversa* (`owl:inverseOf`) R^{-}.

Sulle proprietà è possibile definire vincoli globali di cardinalità, ovvero:

- funzionalità,
- funzionalità inversa.

Funzionalità

Una proprietà R è *funzionale* (`owl:FunctionalProperty`) se

$\top \sqsubseteq \leq 1R$.

Funzionalità inversa

Una proprietà R è *funzionale inversa* (`owl:InverseFunctionalProperty`) se

$\top \sqsubseteq \leq 1R^{-}$.

Simmetria e transitività

Infine, è possibile specificare alcune caratteristiche formali delle proprietà, e in particolare:

- simmetria,
- transitività.

In OWL è possibile dichiarare che una proprietà è simmetrica (`owl:SymmetricProperty`) o che una proprietà è transitiva (`owl:TransitiveProperty`).

8.4 Individui e fatti

In OWL le asserzioni dell'ABox sono detti *fatti*. Introduciamo qui due tipi di fatti:

- fatti relativi all'appartenenza di un individuo a una classe o ai valori di una proprietà di un individuo;
- fatti relativi all'identità di uno o più individui.

Appartenenza a una classe e valori di una proprietà

In OWL è possibile specificare che un individuo a appartiene a una classe C,

$C(a)$,

oppure che una proprietà R di un individuo a ha valore b,

$R(a,b)$.

Identità

Il linguaggio OWL non assume che gli individui abbiano nome unico. Quindi è possibile asserire che due nomi fanno riferimento allo stesso individuo (`owl:sameAs`):

$a = b$.

Analogamente è possibile asserire che due nomi fanno riferimento al individui distinti (`owl:differentFrom`):

$a \neq b$.

È anche possibile asserire che *n* individui sono tutti distinti fra loro (`owl:AllDifferent`).

9. Servizi di ragionamento

L'aspetto che distingue nettamente una base di conoscenze da una base di dati è la possibilità di condurre *ragionamenti* in modo automatico. Poiché una base di conoscenze, **KB**, è costituita da una TBox, **T**, e da un'ABox, **A**, scriveremo in generale

$\mathbf{KB} = \langle \mathbf{T}, \mathbf{A} \rangle$.

Nel contesto della logica, quando si parla di "ragionamento" ci si riferisce sempre a ragionamenti di tipo deduttivo, o più semplicemente *deduzioni* (non prenderemo quindi in considerazione i ragionamenti di tipo induttivo o abduttivo, menzionati nel par. 1.3). In generale, quindi, un ragionamento è per noi un procedimento che porta a verificare se un enunciato *X* (ad esempio la sussunzione o l'equivalenza di due termini) è *conseguenza logica* di una base di conoscenze.

9.1 Conseguenza logica

Intuitivamente un enunciato *X* è conseguenza logica di una base di conoscenze **KB** quando *X* è necessariamente vero in ogni situazione concreta in cui siano veri gli assiomi terminologici e le asserzioni contenuti in **KB**. Più precisamente, un enunciato *X* è conseguenza logica di una base di conoscenze **KB** quando *X* è vero in ogni *modello* (nel senso di FOL) degli assiomi terminologici e delle asserzioni contenuti in **KB**. In tal caso scriviamo

$\mathbf{KB} \models X$,

che si legge: **KB** *implica logicamente* (*entails*) *X*, o *X è conseguenza logica di* **KB**.

Consideriamo ad esempio la TBox **T** contenente le definizioni seguenti:

T1. $\text{GENITORE} \equiv \text{PERSONA} \sqcap \exists \text{Figlio}$

T2. $\top \sqsubseteq \forall \text{Figlio}^{-}.\text{PERSONA}$

T3. $\top \sqsubseteq \forall \text{Figlio}.\text{PERSONA}$

T4. $\text{DONNA} \equiv \text{PERSONA} \sqcap \text{FEMMINA}$

T5. $\text{UOMO} \equiv \text{PERSONA} \sqcap \neg\text{FEMMINA}$

T6. $\text{MADRE} \equiv \text{GENITORE} \sqcap \text{FEMMINA}$

T7. $\text{PADRE} \equiv \text{GENITORE} \sqcap \neg\text{FEMMINA}$

Il contenuto di **T** implica logicamente che certi enunciati, pur non essendo contenuti esplicitamente in **T**, sono necessariamente veri sotto l'ipotesi che sia vero il contenuto di **T**. Ad esempio:

- ogni madre è una persona nonché una donna;
- ogni padre è una persona nonché un uomo;
- la classe delle madri e la classe dei padri sono disgiunte.

Questi enunciati possono essere scritti formalmente nel modo seguente:

MADRE $\sqsubseteq$ PERSONA,

MADRE $\sqsubseteq$ DONNA,

PADRE $\sqsubseteq$ PERSONA,

PADRE $\sqsubseteq$ UOMO,

MADRE $\sqcap$ PADRE $\equiv \bot$.

Per segnalare che questi enunciati sono conseguenze logiche di **T** scriviamo

T $\models$ MADRE $\sqsubseteq$ PERSONA,

e analogamente per gli altri enunciati.

Dobbiamo ora distinguere fra tre diversi concetti:

- *compito di ragionamento* (*reasoning task*): è caratterizzato dal tipo di enunciati che si desidera dedurre da una base di conoscenze;
- *procedura di ragionamento*: è l'algoritmo che consente la deduzione degli enunciati;
- *servizio di ragionamento*: è un servizio effettivamente implementato da uno strumento e messo a disposizione delle applicazioni che accedono alla base di conoscenze.

9.2 Compiti di ragionamento relativi alle TBox

Per quanto riguarda le TBox, i compiti di ragionamento più significativi per le applicazioni sono i seguenti:

- *sussunzione*: data una TBox **T**, stabilire se una sussunzione $C \sqsubseteq D$ è conseguenza logica di **T**, ovvero stabilire se
  $\mathbf{T} \models C \sqsubseteq D$;
- *equivalenza*: data una TBox **T**, stabilire se un'equivalenza $C \equiv D$ è conseguenza logica di **T**, ovvero stabilire se
  $\mathbf{T} \models C \equiv D$;
- *soddisfacibilità*: data una TBox **T**, stabilire se un termine C è *soddisfacibile*, nel senso che l'insieme degli individui dell'universo che soddisfano il termine non è necessariamente vuoto;
- *disgiunzione*: data una TBox **T**, stabilire se due termini C e D sono *disgiunti*, nel senso che l'intersezione degli insiemi di individui dell'universo che soddisfano i due termini è necessariamente vuota.

Riduzione dei compiti di ragionamento alla sussunzione

Si vede facilmente che i quattro compiti di ragionamento fondamentali per le TBox possono essere ridotti alla sola sussunzione. Infatti:

- *equivalenza*: dimostrare $\mathbf{T} \models C \equiv D$ equivale a dimostrare che
  $\mathbf{T} \models C \sqsubseteq D$ e $\mathbf{T} \models D \sqsubseteq C$;
- *soddisfacibilità*: dimostrare che C è soddisfacibile equivale a dimostrare che *non è insoddisfacibile*, ovvero che:
  $\mathbf{T} \not\models C \sqsubseteq \bot$;
- *disgiunzione*: dimostrare che C e D sono *disgiunti* equivale a dimostrare che
  $\mathbf{T} \models C \sqcap D \sqsubseteq \bot$.

Dunque un'unica procedura di ragionamento, in grado di stabilire se vale o non vale la sussunzione fra due termini arbitrari, può essere utilizzata per realizzare tutti e quattro i compiti di ragionamento fondamentali per le TBox. Questa è la strada che si segue per implementare i servizi di ragionamento per le DL poco espressive.

Riduzione dei compiti di ragionamento alla soddisfacibilità

I quattro compiti di ragionamento fondamentali per le TBox possono anche essere ridotti alla sola soddisfacibilità. Infatti:

- *sussunzione*: dimostrare $\mathbf{T} \models C \sqsubseteq D$ equivale a dimostrare che:

  $\mathbf{T} \models$ "$C \sqcap \neg D$ è insoddisfacibile";
- *equivalenza*: dimostrare $\mathbf{T} \models C \equiv D$ equivale a dimostrare che

  $\mathbf{T} \models$ "$C \sqcap \neg D$ è insoddisfacibile" e $\mathbf{T} \models$ "$\neg C \sqcap D$ è insoddisfacibile";
- *disgiunzione*: dimostrare che *C* e *D* sono *disgiunti* equivale a dimostrare che

  $\mathbf{T} \models$ "$C \sqcap D$ è insoddisfacibile".

Dunque un'unica procedura di ragionamento, in grado di stabilire se un termine è o non è soddisfacibile, consente di realizzare i quattro compiti di ragionamento fondamentali per le TBox. Questa è la strada che si segue per implementare i servizi di ragionamento per le DL molto espressive, come ad esempio $\mathcal{SHOIN}(\mathrm{D}_n)$.

9.3 La procedura SAT

Per le DL decidibili (come $\mathcal{SHOIN}(\mathrm{D}_n)$) si può formulare una procedura che prende in ingresso una TBox arbitraria **T** e un termine arbitrario *C* e, in un numero finito di passi, stabilisce se *C* è o non è soddisfacibile (tenendo conto ovviamente delle definizioni terminologiche di **T**). Nelle sue versioni più diffuse questa procedura, che chiameremo SAT, è basata sul cosiddetto *metodo dei tableaux* da tempo studiato e applicato nell'ambito di FOL. Qui ci limitiamo a dare un'idea intuitiva del funzionamento della procedura basandoci su un semplice esempio.

Supponiamo di voler mostrare che, assumendo la TBox **T** del paragrafo 9.1, vale la sussunzione

$\mathrm{MADRE} \sqsubseteq \exists \mathrm{Figlio}$,

ovvero che il termine

$\mathrm{MADRE} \sqcap \neg\exists \mathrm{Figlio}$

è insoddisfacibile (in altre parole, non può esistere un individuo che sia una madre e non abbia figli). La procedura SAT parte dall'ipotesi che il termine *sia* soddisfacibile, e sviluppa un ragionamento sulla base di questa ipotesi iniziale. A grandi linee i passi del ragionamento sono i seguenti:

- Se $\mathrm{MADRE} \sqcap \neg\exists \mathrm{Figlio}$ è soddisfacibile, allora può esistere un individuo nell'universo cui il termine si applica. Introducendo un nominale, *a*, per identificare questo individuo possiamo scrivere

  1. $\mathrm{MADRE} \sqcap \neg\exists \mathrm{Figlio}(a)$.
- Da 1, poiché l'intersezione si distribuisce rispetto all'argomento possiamo dedurre che:

  2.1 $\mathrm{MADRE}(a)$,

  2.2 $\neg\exists \mathrm{Figlio}(a)$.
- Da 2.1, applicando la definizione T6, possiamo dedurre che:

  3. $\mathrm{GENITORE} \sqcap \mathrm{FEMMINA}(a)$.
- Da 3, poiché l'intersezione si distribuisce rispetto all'argomento possiamo dedurre che:

  4.1 $\mathrm{GENITORE}(a)$,

  4.2 $\mathrm{FEMMINA}(a)$.
- Da 4.1, applicando la definizione T1, possiamo dedurre che:

  5. $\mathrm{PERSONA} \sqcap \exists \mathrm{Figlio}(a)$.
- Da 5, poiché l'intersezione si distribuisce rispetto all'argomento possiamo dedurre che:

  6.1 $\mathrm{PERSONA}(a)$,

  6.2 $\exists \mathrm{Figlio}(a)$.

- L'asserzione 6.2 è in contraddizione con la 2.2. Ne segue che l'ipotesi iniziale, che il termine MADRE ⊓ ¬∃Figlio fosse soddisfacibile, va rifiutata. Quindi il termine è insoddisfacibile.

Dato il rapporto che lega insoddisfacibilità e sussunzione, questa deduzione ci permette di stabilire che:

$\mathbf{T} \models \text{MADRE} \sqsubseteq \exists\text{Figlio}$.

9.4 Compiti di ragionamento relativi all'ABox

Ci occuperemo ora dei servizi di ragionamento che coinvolgono non soltanto assiomi terminologici della TBox, ma anche asserzioni dell'ABox. Come abbiamo già notato le *asserzioni* contenute in un'ABox possono essere *basate su termini* o *basate su ruoli*; ovvero, le asserzioni possono assumere una delle due forme seguenti:

$C(a)$, dove C è un termine *arbitrario* ed a è un nominale;

$R(a,b)$, dove R è un ruolo e a, b sono nominali.

Ad esempio le espressioni seguenti sono possibili asserzioni:

(8.1) PERSONA ⊓ ¬FEMMINA(marco);

Figlio(bob,cecilia).

Va sottolineato che, in linea di principio, si potrebbero limitare le asserzioni basate su termini ai soli termini atomici. In altre parole, le asserzioni basate su termini potrebbero assumere la forma:

$A(a)$,

dove A è un termine *atomico* ed a è un nominale; in questo modo non si perderebbe generalità, pur di aggiungere alla TBox opportune definizioni terminologiche del tipo:

$A \equiv C$.

Nel caso dell'esempio potremmo avere nella TBox la definizione

UOMO ≡ PERSONA ⊓ ¬FEMMINA,

o in alternativa le due definizioni

UOMO ≡ PERSONA ⊓ MASCHIO,

MASCHIO ≡ ¬FEMMINA,

che consentirebbero di riscrivere la 8.1 come

UOMO(marco).

Un approccio di questo genere, tuttavia, tende a riempire la TBox di definizioni poco significative; è quindi preferibile ammettere nell'ABox asserzioni basate su termini arbitrari. Un caso particolare è costituito dalle "asserzioni negative", come ad esempio

¬FEMMINA(marco),

che, come abbiamo già notato (par. 6), non va vista come la negazione dell'"asserzione positiva" FEMMINA(bob), quanto piuttosto come l'asserzione che il termine complesso ¬FEMMINA si applica al nominale "marco".

Ci occuperemo ora di alcuni *compiti di ragionamento relativi alle ABox*, che riguardano le asserzioni e utilizzano le conoscenze contenute in tutta la base delle conoscenze, sia nella TBox sia nell'ABox. Consideriamo una base di conoscenze $\langle \mathbf{T},\mathbf{A} \rangle$, formata dalla TBox **T** e dall'ABox **A**. I più significativi compiti di ragionamento relativi all'ABox sono i seguenti:

- *instance check*: dati una TBox **T**, un'ABox **A**, un termine arbitrario C e un nominale a, stabilire se si ha

  $\mathbf{T},\mathbf{A} \models C(a)$;

- *retrieval*: dati una TBox **T**, un'ABox **A** e un termine arbitrario *C*, fra tutti i nominali presenti nella base di conoscenze trovare tutti i nominali a_1, ..., a_n tali che

  $\mathbf{T},\mathbf{A} \models C(a_k)$;
- *realizzazione*: dati una TBox **T**, un'ABox **A**, un insieme di termini arbitrari $\{C_1, ..., C_n\}$ e un nominale *a*, determinare gli *m* termini $\{C_{i_1}, ...,C_{i_m}\}$ *più specifici* in $\{C_1, ..., C_n\}$ per cui si ha

  $\mathbf{T},\mathbf{A} \models C_k(a)$.

A chiarimento del compito di realizzazione, precisiamo che un termine *C* si dice *più specifico* di un termine *D* quando vale $C \sqsubseteq D$. Si noti inoltre che la sussunzione è una relazione d'ordine parziale fra i termini; i termini più specifici in $\{C_1, ..., C_n\}$ per cui vale $\mathbf{T},\mathbf{A} \models C_k(a)$ sono quindi gli elementi di $\{C_1, ..., C_n\}$, minimali rispetto alla relazione di sussunzione, per cui vale $\mathbf{T},\mathbf{A} \models C_k(a)$.

Cosideriamo ad esempio la TBox **T** seguente (estensione della TBox del par. 9.1):

T1. GENITORE ≡ PERSONA ⊓ ∃Figlio,

T2. $\top \sqsubseteq \forall \text{Figlio}^{-}.\text{PERSONA}$,

T3. $\top \sqsubseteq \forall \text{Figlio}.\text{PERSONA}$,

T4. DONNA ≡ PERSONA ⊓ FEMMINA,

T5. UOMO ≡ PERSONA ⊓ ¬FEMMINA,

T6. MADRE ≡ GENITORE ⊓ FEMMINA,

T7. PADRE ≡ GENITORE ⊓ ¬FEMMINA,

T8. CITTADINANZA ≡ {au,ch,de,es,fr,it,uk},

T9. $\top \sqsubseteq \forall \text{Cittadinanza}^{-}.\text{PERSONA}$,

T10. $\top \sqsubseteq \forall \text{Cittadinanza}.\text{CITTADINANZA}$,

T11. ITALIANO ≡ PERSONA ⊓ ∃Cittadinanza.{it},

T12. BRITANNICO ≡ PERSONA ⊓ ∃Cittadinanza.{uk}.

Definiamo l'ABox **A** seguente:

A1. DONNA(anna),

A2. DONNA(cecilia),

A3. UOMO(bob),

A4. Figlio(anna,cecilia),

A5. Figlio(bob,cecilia),

A6. Cittadinanza(anna,it),

A7. Cittadinanza(bob,uk),

A8. Cittadinanza(cecilia,it),

A9. Cittadinanza(cecilia,uk).

Vediamo ora i risultati di alcuni ragionamenti relativi all'ABox.

Instance check

Dati il termine FEMMINA ⊓ ∃Figlio e il nominale "anna" si ha:

$\mathbf{T},\mathbf{A} \models$ FEMMINA ⊓ ∃Figlio(anna),

ovvero l'instance check *ha successo*.

Dati il termine FEMMINA ⊓ ∃Figlio e il nominale "cecilia" si ha:

$\mathbf{T},\mathbf{A} \not\models$ FEMMINA ⊓ ∃Figlio(cecilia),

ovvero l'instance check *fallisce*.

Retrieval

Dato il termine GENITORE si ha:

$\mathbf{T},\mathbf{A} \models$ GENITORE(anna),

$\mathbf{T},\mathbf{A} \models$ GENITORE(bob).

Realizzazione

Dati l'insieme di tutti i termini atomici della TBox e il nominale "anna" si ha:

$\mathbf{T},\mathbf{A} \models$ MADRE(anna),

$\mathbf{T},\mathbf{A} \models$ ITALIANO(anna).

Si noti che tutte le asserzioni seguenti sono implicati logicamente dalla base di conoscenze assegnata:

$\mathbf{T},\mathbf{A} \models$ PERSONA(anna),

$\mathbf{T},\mathbf{A} \models$ FEMMINA(anna),

$\mathbf{T},\mathbf{A} \models$ DONNA(anna),

$\mathbf{T},\mathbf{A} \models$ GENITORE(anna),

$\mathbf{T},\mathbf{A} \models$ MADRE(anna),

$\mathbf{T},\mathbf{A} \models$ ITALIANO(anna).

Tuttavia, soltanto i termini MADRE e ITALIANO risolvono il problema di realizzazione assegnato: infatti sono gli unici termini minimali nell'insieme

{PERSONA,FEMMINA,DONNA,GENITORE,MADRE,ITALIANO}

in quanto:

$\mathbf{T} \models$ MADRE $\sqsubseteq$ PERSONA,

$\mathbf{T} \models$ MADRE $\sqsubseteq$ FEMMINA,

$\mathbf{T} \models$ MADRE $\sqsubseteq$ DONNA,

$\mathbf{T} \models$ MADRE $\sqsubseteq$ GENITORE.

9.5 Riduzione alla soddisfacibilità dei compiti di ragionamento relativi all'ABox

I compiti di ragionamento relativi all'ABox si possono ridurre al problema di stabilire la *soddisfacibilità di un insieme di asserzioni*. Notiamo innanzi tutto che

$\mathbf{T},\mathbf{A} \models C(a)$ se e solo se $\mathbf{T} \models$ "l'unione di $\mathbf{A}$ e $\{\neg C(a)\}$ è insoddisfacibile".

Quindi un compito di instance check può essere ridotto a un problema di soddisfacibilità. Un compito di retrieval, poi, almeno in linea di principio può essere ridotto a un compito di instance check per ciascun nominale presente nella base di conoscenze. Infine, un compito di realizzazione può essere ridotto a una serie di compiti di instance check e a una serie di compiti di sussunzione. Ciò significa che, almeno in linea di principio, tutti i compiti di ragionamento che abbiamo esaminato possono essere ridotti a problemi di soddisfacibilità.

9.6 Invocazione dei servizi di ragionamento

Le specifiche di OWL definiscono in modo rigoroso la sintassi dei termini e degli enunciati logici ammissibili, ma curiosamente non stabiliscono una modalità standard per l'invocazione dei servizi di ragionamento fondamentali. In questi appunti per rappresentare le invocazioni di servizi di ragionamento e le relative risposte utilizzaremo le convenzioni seguenti:

- sussunzione: ?– $C \sqsubseteq D$ $\longrightarrow$ *yes*/*no*,
- equivalenza: ?– $C \equiv D$ $\longrightarrow$ *yes*/*no*,
- soddisfacibilità ?– C $\longrightarrow$ *yes*/*no*,

- disgiunzione ?– $C \sqcap D \sqsubseteq \bot$ $\longrightarrow$ *yes*/*no*,
- instance check ?– $C(a)$ $\longrightarrow$ *yes*/*no*,
- retrieval ?– $C(*)$ $\longrightarrow$ $\{a_1, ..., a_n\}$,
- realizzazione ?– $a : C_1, ..., C_n$ $\longrightarrow$ $\{C_{i_1}, ..., C_{i_m}\}$.

Bibliografia

Testi a stampa

Antoniou, G., F. van Harmelen, 2004. *A Semantic Seb Primer*, MIT Press, Cambridge, MA.

Daconta, M. C., L. J. Obrst, K. T. Smith, 2003. *The Semantic Web: A guide to the future of XML, Web Services, and Knowledge Management*, Wiley.

Dretske, F. I., 1981. *Knowledge and the flow of information*, Basil Blackwell.

Feigenbaum, E. A., 1977. The art of artificial intelligence: Themes and case studies of knowledge engineering. *Proceedings of the 5th International Conference on Artificial Intelligence* (*JCAI77*), 1014–1029.

Ryle, G., 1940. *The concept of mind*, Hutchinson.

Stefik, M., 1995. *Introduction to knowledge systems*, Morgan Kaufmann.

Documenti in rete (ultima visita 14.02.2005)

Berners-Lee, T., J. Hendler and O. Lassila, 2001. The Semantic Web, *Scientific American*, May 2001, http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

Palmer, S. B., 2001. The Semantic Web, http://infomesh.net/2001/swintro/.

Swartz, A., 2002. The Semantic Web in Breadth, http://logicerror.com/semanticWeb-long.

Appendice I

Grammatica degli enunciati terminologici (logica $\mathcal{SHOIQ}$ senza unicità dei nomi)

A rappresenta un *termine atomico*
C, *D* rappresentano *termini arbitrari*
R, *S* rappresentano *ruoli* (atomici)
a, *b*, a_1, a_2, ... rappresentano *nominali*
n rappesenta un *numerale naturale* (0, 1, ...)
φ rappresenta un *enunciato*
α rappresenta un'*asserzione*
[...] indica una componente opzionale

$$\varphi \longrightarrow C \sqsubseteq D \mid C \equiv D \mid R \sqsubseteq S \mid R \equiv S \mid Tr(R)$$

$$\alpha \longrightarrow C(a) \mid R(a,b) \mid a = b \mid a \neq b$$

$$\begin{aligned} C \longrightarrow\ & \top \mid \bot \mid A \mid \\ & \neg C \mid (C \sqcap D) \mid (C \sqcup D) \mid \\ & \forall R.C \mid \exists R[.C] \mid \\ & \leq nR[.C] \mid \geq nR[.C] \mid =nR[.C] \mid \\ & \{a_1, ..., a_n\} \end{aligned}$$

Appendice II

Semantica formale degli enunciati terminologici (logica $\mathcal{SHOIQ}$ senza unicità dei nomi)

In quest'appendice definiamo in modo insiemistico la semantica formale delle espressioni costruibili con la grammatica specificata nell'appendice I. Le espressioni del tipo $\exists R$, $\leq nR$, $\geq nR$ e $=nR$ vanno interpretate come rispettivamente equivalenti a $\exists R.\top$, $\leq nR.\top$, $\geq nR.\top$ e $=nR.\top$.

Modelli

Un modello è una coppia ordinata $M = \langle \mathbf{U}, ^* \rangle$, dove $\mathbf{U} = \{u, v, ...\}$ (detto l'*universo*) è un insieme non vuoto di *individui* e $-^*$ è una *funzione d'interpretazione* che associa:

- un individuo $a^* \in \mathbf{U}$ a ogni nominale a;
- un insieme $A^* \subseteq \mathbf{U}$ a ogni termine atomico A, con $\top^* = \mathbf{U}$ e $\bot^* = \varnothing$;
- una relazione binaria $R^* \subseteq \mathbf{U} \times \mathbf{U}$ a ogni ruolo R.

La funzione * può essere estesa ai termini arbitrari definendo:

- $(\neg C)^* = \mathbf{U} \setminus C^*$;
- $(C \sqcap D)^* = C^* \cap D^*$;
- $(C \sqcup D)^* = C^* \cup D^*$;
- $(\forall R.C)^* = \{u \in \mathbf{U} :$ se $\langle u,v \rangle \in R^*$, allora $v \in C^*\}$;
- $(\exists R.C)^* = \{u \in \mathbf{U} : \langle u,v \rangle \in R^*$ per qualche $v \in C^*\}$;
- $(\leq nR.C)^* = \{u \in \mathbf{U} : \langle u,v \rangle \in R^*$ per al più n individui $v \in C^*\}$;
- $(\geq nR.C)^* = \{u \in \mathbf{U} : \langle u,v \rangle \in R^*$ per almeno n individui $v \in C^*\}$;
- $(=nR.C)^* = \{u \in \mathbf{U} : \langle u,v \rangle \in R^*$ per esattamente n individui $v \in C^*\}$;
- $(\{a_1, ..., a_n\})^* = \{a_1^*,...,a_n^*\}$.

Verità e conseguenza logica

Un enunciato terminologico φ è *vero in un modello M*,

$M \models \varphi$,

sotto le seguenti condizioni:

$M \models C \sqsubseteq D$ se e solo se $C^* \subseteq D^*$,

$M \models C \equiv D$ se e solo se $C^* = D^*$,

$M \models R \sqsubseteq S$ se e solo se $R^* \subseteq S^*$,

$M \models R \equiv S$ se e solo se $R^* = S^*$,

$M \models Tr(R)$ se e solo se R^* è transitiva.

Un'asserzione α è *vera in un modello M*,

$M \models \alpha$,

sotto le seguenti condizioni:

$M \models C(a)$ se e solo se $a^* \in C^*$,

$M \models R(a,b)$ se e solo se $\langle a^*,b^* \rangle \in R^*$,

$M \models a = b$ se e solo se $a^* = b^*$,

$M \models a \neq b$ se e solo se $a^* \neq b^*$.

Sia ora **KB** = ⟨**T**,**A**⟩, dove **T** è una TBox e **A** è un'Abox. Un modello *M soddisfa* **KB** (o *è un modello di* **KB**) se e solo se:

$M \models \varphi$ per ogni $\varphi \in \mathbf{T}$,

$M \models \alpha$ per ogni $\alpha \in \mathbf{A}$.

Un enunciato terminologico φ (un'asserzione α) è *conseguenza logica* di **KB**,

$\mathbf{KB} \models \varphi$ ($\mathbf{KB} \models \alpha$),

se e solo se

$M \models \varphi$ ($M \models \alpha$),

per ogni modello M che soddisfi **KB**.

Un enunciato terminologico φ è *valido*,

$\models \varphi$,

se e solo se

$\varnothing \models \varphi$,

dove $\varnothing$ indica la **KB** vuota.

Definizione semantica dei compiti di ragionamento relativi alla TBox

Data una TBox **T**:

- C è sussunto da D (D sussume C) se e solo se

  $\langle \mathbf{T},\varnothing \rangle \models C \sqsubseteq D$;
- C è equivalente a D se e solo se

  $\langle \mathbf{T},\varnothing \rangle \models C \equiv D$;
- *C è soddisfacibile* se e solo se esiste un modello $M = \langle \mathbf{U},* \rangle$ di $\langle \mathbf{T},\varnothing \rangle$ tale che

  $u \in C^*$ per qualche $u \in \mathbf{U}$;
- *C* e *D sono disgiunti* se e solo se

  $\langle \mathbf{T},\varnothing \rangle \models C \sqcap D \sqsubseteq \bot$.

Definizione semantica dei compiti di ragionamento relativi all'ABox

Date una TBox **T** e un'ABox **A**:

- l'*instance check* di *C* ed *a* restituisce *yes* se e solo se

  $\langle \mathbf{T},\mathbf{A} \rangle \models C(a)$;
- il *retrieval* relativo a *C* restituisce $\{a_1, ..., a_n\}$, con $n \geq 0$, se e solo se $a_1, ..., a_n$ sono tutti e soli i nominali occorrenti in $\langle \mathbf{T}, \mathbf{A} \rangle$ tali che

  $\langle \mathbf{T},\mathbf{A} \rangle \models C(a_k)$;
- la *realizzazione di a rispetto a* $\{C_1, ..., C_n\}$ restituisce l'insieme $\{D_1, ..., D_m\}$ di tutti e soli i $D_k \in \{C_1, ..., C_n\}$ tali che: $\langle \mathbf{T},\mathbf{A} \rangle \models D_k(a)$, e inoltre non esiste alcun $C_i \in \{D_1, ..., D_m\}$ tale che

  $\langle \mathbf{T},\mathbf{A} \rangle \models C_i(a)$,

  $\langle \mathbf{T},\mathbf{A} \rangle \models C_i \sqsubseteq D_k$,

  $\langle \mathbf{T},\mathbf{A} \rangle \not\models D_k \sqsubseteq C_i$.