Dipartimento di elettronica e informazione
Politecnico di Milano

Ingegneria della conoscenza: modelli semantici

Edizione 2010-11
Parti I e II

Marco Colombetti[1], 6 marzo 2011

Indice

Parte I: Introduzione

1. L'ingegneria della conoscenza

Il termine *ingegneria della conoscenza* (*knowledge engineering*) è stato coniato da Edward Feigenbaum (1977). Nata nell'ambito dell'intelligenza artificiale, questa disciplina si occupa del progetto, della realizzazione e della gestione di *sistemi basati su conoscenze* (*knowledge based system*, KBS), ovvero di sistemi informatici in grado di sfruttare le informazioni contenute in una *base di conoscenze* (*knowledge base*, KB) mediante procedure automatiche di ragionamento,.

Con i metodi e le tecnologie dell'ingegneria della conoscenza sono stati sviluppati diversi tipi di applicazioni d'interesse industriale. A partire dalla metà degli anni Settanta e per tutti gli anni Ottanta del Novecento le applicazioni più diffuse sono stati i *sistemi esperti* (vedi ad es. Jackson, 1998), intesi come sistemi software in grado di risolvere problemi utilizzando rappresentazioni computabili di conoscenze specialistiche; applicazioni tipiche dei sistemi esperti sono l'analisi di grandi volumi di dati, la diagnosi dei malfunzionamenti di impianti industriali, la diagnosi medica, la configurazione di sistemi informatici complessi, il progetto industriale, la pianificazione della produzione e così via. In molti casi l'adozione di sistemi esperti ha permesso di automatizzare attività complesse là dove tecniche informatiche più tradizionali avevano fallito. Tuttavia i sistemi esperti sono rimasti prodotti di nicchia perché il loro sviluppo è un'impresa lunga e costosa, che può essere affrontata soltanto per applicazioni particolari, in cui sia possibile investire una notevole quantità di risorse; inoltre, proprio perché si basano su conoscenze specialistiche, i sistemi esperti non possono essere facilmente trasferiti da un'area applicativa a un'altra.

La tecnologia dei sistemi esperti continua a rivestire interesse in alcuni settori applicativi, ma è improbabile che possa dar luogo a prodotti di larga diffusione. Oggi, tuttavia, l'ingegneria della conoscenza è diventata importante per altri ambiti di grande interesse potenziale, e in particolare per il *web semantico* (noto anche come *web 3.0*). Ma prima di entrare in argomento è opportuno premettere alcune considerazioni generali sulle conoscenze e sulla loro rappresentazione nei sistemi informatici.

1.1 Le conoscenze

In prima approssimazione la conoscenza può essere definita come *informazione disponibile per l'azione*: è grazie alle loro conoscenze che gli esseri umani possono agire in modo razionale, ovvero basando le loro azioni su ragioni. Più precisamente le conoscenze servono per *interpretare la realtà*, ad esempio per comprendere le situazioni in cui ci troviamo e per capire che cosa è successo (ad esempio, per ipotizzare le cause di un evento che si è verificato); per *prevedere l'evoluzione della realtà*, ovvero per prevedere con sufficiente approssimazione come evolverà una situazione e quali eventi si verificheranno nel futuro; e per *agire in modo razionale modificando la realtà*, ovvero per costruire ed eseguire piani d'azione che mirano a raggiungere determinati obiettivi o per progettare artefatti che rispondano a certe specifiche. Anche nei computer risiedono grandi quantità di informazioni, ma solo una piccola parte di queste può essere sfruttata dagli stessi computer per agire: basti pensare che la stragrande maggioranza dei file memorizzati nei computer è costituita da documenti in linguaggio naturale, destinati agli esseri umani, il cui significato è inaccessibile alle applicazioni software. In questo corso esploreremo la possibilità di automatizzare certe attività che oggi richiedono l'intervento umano perché i computer non riescono a sfruttare appieno le informazioni disponibili.

Conoscenze dichiarative e conoscenze procedurali

Una prima distinzione va tracciata fra *conoscenze dichiarative* e *conoscenze procedurali*, ovvero fra *conoscere* (*knowing that*) e *saper fare* (*knowing how*)[2]. Si *conosce* la storia contemporanea o la teoria della relatività, mentre si *sa* nuotare, andare in bicicletta e così via ("so nuotare" in inglese si dice "I can swim", ovvero "sono capace di nuotare"). In generale le conoscenze dichiarative sono esprimibili nel linguaggio ordinario (italiano, inglese, ...), eventualmente integrato con termini e simboli scientifici o tecnici; le conoscenze procedurali, al contrario, consistono essenzialmente in abilità percetto-motorie e spesso sono difficili o impossibili da verbalizzare (provate a spiegare a parole come si nuota!).

In questo corso ci concentriamo sulle conoscenze dichiarative, ovvero sul *knowing that*. In quest'ambito possiamo riconoscere tre tipi principali di conoscenze:

- *conoscenze concettuali*: è la padronanza dei concetti che utilizziamo per interpretare e descrivere la realtà; ad esempio, so che cos'è una madre (una donna con almeno un figlio), che cos'è un'automobile (un veicolo a motore per il trasporto su strada di persone e cose) e così via;
- *conoscenze nomologiche*: è la conoscenza di regolarità, di leggi generali che regolano il mondo (dal greco *nòmos*, regola); ad esempio so che le madri (in genere) amano i loro figli, che le automobili si sfasciano se si scontrano in velocità con un muro e così via;
- *conoscenze fattuali*: è la conoscenza di fatti particolari; ad esempio, so che Anna è la madre di Bruno, che la mia automobile è blu e così via.

Le conoscenze dichiarative di un soggetto provengono da varie fonti, fra cui le più ovvie sono: *l'esperienza diretta*, ovvero l'interazione del soggetto con il mondo circostante, in particolare tramite la percezione; *il ragionamento*, che può essere *deduttivo* (dalle premesse alle conclusioni), *abduttivo* (dagli effetti osservati alle possibili cause) o *induttivo* (da fatti particolari a leggi generali); *la comunicazione*, ovvero l'uso di sistemi di segni e in particolare del linguaggio per trasferire informazioni da un soggetto a un altro. Qualunque sia la fonte, poi, è essenziale la funzione della *memoria*, intesa come la capacità di conservare nel tempo elementi di conoscenza e soprattutto di recuperarli con efficienza quando occorre farne uso.

1.2 Schemi concettuali e modelli semantici

Ritorniamo alla distinzione fra conoscenze concettuali, nomologiche e fattuali. Fra queste, le conoscenze concettuali hanno una sorta di precedenza logica, perché senza la padronanza dei concetti non è possibile rappresentare conoscenze nomologiche o fattuali. Ad esempio, un abitante della Grecia classica non potrebbe formulare il pensiero "le società per azioni vanno costituite davanti a un notaio" semplicemente perché nella cultura della Grecia classica non esistevano né il concetto di società per azioni né il concetto di notaio. Parlando metaforicamente, le conoscenze concettuali costituiscono una lente attraverso cui guardiamo il mondo: ciò che vediamo dipende da come è fatto il mondo, ma anche da come è fatta la lente.

Le conoscenze concettuali a disposizione di un soggetto costituiscono una grande rete di elementi, connessi fra loro da relazioni logiche; ad esempio, il concetto di gatto e il concetto di felino sono collegati da una relazione ben precisa, per lo meno in ogni soggetto che sappia che i gatti sono felini. Possiamo vedere questa rete come uno *schema concettuale*, o meglio come un sistema di schemi concettuali interconnessi. In linea di principio gli schemi concettuali di un soggetto possono differire da quelli di altri soggetti; i soggetti appartenenti alla stessa collettività tendono però a condividere gli stessi schemi concettuali, e in questo senso non è scorretto considerare uno schema concettuale come tipico di una collettività, piuttosto che di un singolo individuo.

La nozione di collettività, tuttavia, va precisata. L'autore di queste note, ad esempio, appartiene allo stesso tempo a molte collettività, in parte sovrapposte l'una all'altra; in questo senso condivide parte dei suoi schemi concettuali con tutte le persone di cultura occidentale che vivono fra la fine del XX e

[2] Questa distinzione è stata introdotta da Gilbert Ryle (1940).

l'inizio del XXI secolo; altri schemi concettuali, invece, li condivide solo con i suoi connazionali, altri con i professori universitari (connazionali o non connazionali), altri con gli informatici, e altri ancora con i docenti del Politecnico di Milano. Ad esempio, è probabile che il concetto di esame dell'autore sia praticamente indistinguibile da quello di un suo collega del Politecnico di Milano, ma differisca almeno in parte dal concetto di esame universitario di un docente, poniamo, della University of Otago in Nuova Zelanda.

Gli schemi concettuali giocano un ruolo importante in molte aree dell'informatica; ad esempio, il modello concettuale E-R (*entity-relationship*) di una base di dati può essere visto come la rappresentazione formale di uno schema concettuale. Una base di dati sviluppata secondo un dato modello concettuale potrà rappresentare solo certi aspetti della realtà: precisamente quegli aspetti che sono presi in considerazione nel modello concettuale. Chiameremo in generale *modello semantico* uno schema concettuale definito tramite un liguaggio formale dotato di una semantica non ambigua. Il linguaggio E-R è solo uno dei tanti linguaggi formali a disposizione degli informatici per la definizione di modelli semantici. In questo corso ci occuperemo di un'importante famiglia di questi linguaggi, studiata nell'ambito della *logica descrittiva* (*description logic*, DL).

Un modello semantico ha lo scopo di definire un sistema di simboli dotati di significato. Il concetto di significato, a sua volta, chiama in causa il rapporto fra linguaggio e realtà; ad esempio il termine "seggiola" (un sostantivo dell'italiano) ha una relazione ben precisa con le seggiole intese come oggetti concreti. La relazione fra linguaggio e realtà (e quindi fra il termine "seggiola" e le seggiole concrete) è detta *relazione semantica*. Secondo una linea di pensiero che possiamo 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 concrete proprio perché possiede un concetto di seggiola. Più precisamente, dobbiamo distinguere fra:

- il *termine* "seggiola" dell'italiano ("chair" in inglese, "Stuhl" in tedesco e così via);
- il *concetto* di seggiola (indipendente dalla lingua);
- gli *oggetti* della realtà che esemplificano il concetto di seggiola (le seggiole concrete).

La relazione fra queste entità è schematizzata nel *triangolo semiotico* riportato nella figura 1.1.

Se per rappresentare la realtà si utilizzano espressioni di un linguaggio formale, a rigore gli elementi della rappresentazione andrebbero chiamati "termini", dato che si tratta comunque di entità linguistiche; nell'ambito dell'ingegneria della conoscenza, tuttavia, questi elementi sono più spesso chiamati "concetti" o "classi". Più precisamente in queste note:

- chiameremo *classe* un'espressione formale che rappresenta un insieme di oggetti;
- chiameremo *estensione* della classe l'insieme di oggetti rappresentato dalla classe;
- diremo che una classe *denota* la propria estensione.

2. Rappresentazione delle conoscenze

Come abbiamo già detto, il nostro obiettivo 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 accedere alle conoscenze e utilizzarle per eseguire elaborazioni appropriate).

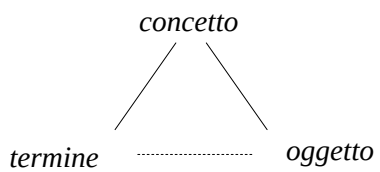

Figura 1.1. Un triangolo semiotico.

In questo corso ci concentreremo sulle rappresentazioni dichiarative e sulle loro applicazioni nell'ambito del web semantico. Il paradigma di riferimento per le rappresentazioni dichiarative è la *logica simbolica* (o *logica formale*) e in particolare la *logica dei predicati del primo ordine*[3] (FOL, da *First Order Logic*). In FOL si assume che tutte le rappresentazioni riguardino un insieme arbitrario, purché non vuoto, di oggetti, detto *universo*[4]. Di questi oggetti possiamo rappresentare *qualità*, oppure *relazioni* che li leghino fra loro; un *fatto* è poi il sussistere di una qualità di un determinato oggetto ("Anna è bionda", "Milano è una città" ecc.) o il sussistere di una relazione fra più oggetti ("Bruno è più alto di Anna", "Anna è nata a Milano", ecc.). Nei sistemi di rappresentazione che esamineremo si fanno assunzioni analoghe, con la limitazione che le relazioni prese in considerazione sono esclusivamente binarie[5].

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*. Nel seguito cercheremo di chiarire questi due concetti.

2.1 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 in FOL il concetto di "madre" come "donna con almeno un figlio". Per cominciare parafrasiamo la definizione di "madre" come:

per ogni *x*, *x* è una madre se e solo se: *x* è una donna ed esiste almeno un *y* tale che *x* è genitore di *y*.

Possiamo rappresentare questa definizione usando un enunciato FOL; adottando la notazione di FOL solitamente utilizzata nei testi di logica[6] abbiamo:

(1.1) $\forall x\,(\mathsf{Madre}(x) \leftrightarrow \mathsf{Donna}(x) \wedge \exists y\,\mathsf{genDi}(x,y))$

dove:

- $\forall x$ ("per ogni *x*") è il *quantificatore universale*;
- $\exists y$ ("esiste almeno un *y* tale che" oppure "per qualche *y*") è il *quantificatore esistenziale*;
- $\leftrightarrow$ ("se e solo se") è il *connettivo bicondizionale*;
- $\wedge$ ("e") è il *connettivo di congiunzione*;
- Madre(–) e Donna(–) sono due *simboli predicativi monoargomentali*, che esprimono delle qualità del loro argomento;
- genDi(–,–) è un *simbolo predicativo biargomentale*, che esprime una relazione binaria fra i suoi due argomenti.

Spesso nei testi d'informatica le formule logiche sono rappresentate con notazioni testuali ASCII, che non utilizzano simboli matematici come $\forall$ e $\exists$. Ad esempio in CL (*Common Logic Standard*, una raccomandazione ISO per la logica predicativa, `http://cl.tamu.edu/`) la 1.1 sarebbe resa come:

```
(forall (?x) (iff (Madre ?x)
                  (and (Donna ?x)
                       (exists (?y) (genDi ?x ?y)))))
```

---

[3] Si assume che lo studente abbia già conoscenze di logica predicativa del primo ordine. In caso contrario si consiglia di scaricare e leggere la dispensa *Introduzione alla logica predicativa del primo ordine*, `http://home.dei.polimi.it//colombet/IC/`, Materiale didattico.

[4] 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 una proprietà* (vedi il par. 5.2).

[5] In FOL si considerano le relazioni di ordine finito qualsiasi (ovvero relazioni binarie, ternarie e così via). Si può dimostrare che utilizzare solo relazioni binarie non è limitativo, purché si abbia a disposizione l'intera espressività di FOL. Nei sistemi che prenderemo in esame, invece, si ha a disposizione soltanto un sottoinsieme proprio di FOL: in un contesto del genere, come vedremo, considerare soltanto relazioni binarie limita effettivamente l'espressività del linguaggio.

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

Le espressioni possono anche essere serializzate utilizzando un formato adatto allo scambio dei dati fra applicazioni software: tipicamente si tratta di file di testo strutturati secondo un opportuno schema XML. Indipendentemente dalla notazione prescelta (classica, ASCII, XML ecc.) occorrerà comunque che le espressioni utilizzate rispettino una grammatica non ambigua (la *sintassi* dell'espressione) e abbiamo un significato rigorosamente definito (la *semantica* dell'espressione).

Nei testi di logica la semantica delle formule è solitamente definita utilizzando modelli insiemistici e anche in queste note seguiremo questo procedimento. In alternativa è possibile definire una *traduzione* univoca da una notazione nuova a una notazione preesistente e contare poi sul fatto che la notazione preesistente ha già una semantica rigorosamente definita (un esempio di questo procedimento è mostrato nell'appendice IV).

2.2 Il 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'ulteriore espressione (detta *conclusione* o *tesi*), in modo tale da *conservare l'eventuale verità delle premesse*: in altre parole, in ogni situazione in cui le premesse siano vere, sarà vera anche la conclusione[7]; ad esempio, dati come premesse la definizione di "madre", il fatto che Anna è una donna e il fatto che Anna è genitore di Bruno, si può dedurre come conclusione che Anna è una madre. In FOL questa deduzione può essere rappresentata in vari modi, ad esempio tramite la prova seguente, in cui le costanti `a` e `b` stanno rispettivamente per Anna e Bruno (il metodo di prova utilizzato è noto come *calcolo della deduzione naturale*; l'ultima riga della prova rappresenta la conclusione):

1. $\forall x\,(\text{Madre}(x) \leftrightarrow \text{Donna}(x) \wedge \exists y\,\text{genDi}(x,y))$ premessa
2. $\text{Donna}(a)$ premessa
3. $\text{genDi}(a,b)$ premessa
4. $\exists y\,\text{genDi}(a,y)$ da 3, per introduzione di $\exists$
5. $\text{Donna}(a) \wedge \exists y\,\text{genDi}(a,y)$ da 2 e 4, per introduzione di $\wedge$
6. $\text{Madre}(a) \leftrightarrow \text{Donna}(a) \wedge \exists y\,\text{genDi}(a,y)$ da 1, per eliminazione di $\forall$
7. $\text{Madre}(a)$ da 6 e 5, per eliminazione di $\leftrightarrow$

Utilizzando FOL si possono esprimere conoscenze molto articolate e, almeno in linea di principio, eseguire ragionamenti complessi in modo automatico. 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 deriva (ovvero è deducibile) dalle premesse, la procedura termina sempre dopo un numero finito di passi producendo una prova;
- quando invece la conclusione non deriva delle premesse, la procedura può terminare dopo un numero finito di passi con un insuccesso, ma può anche non terminare.

In altre parole una procedura di prova *può non terminare se si tenta di dedurre una conclusione che non deriva delle premesse*. È noto inoltre che non si tratta di una limitazione transitoria, dovuta al fatto che una procedura di decisione per la deducibilità in FOL non è ancora stata scoperta: come hanno indipendetemente dimostrato Turing (1936) e Church (1936), una procedura del genere può esistere.

La semidecidibilità di FOL è conseguenza della notevole espressività del linguaggio: più un linguaggio di rappresentazione è espressivo, infatti, più risultano problematiche le procedure di ragionamento. Molte ricerche nel campo dei linguaggi di rappresentazione delle conoscenze hanno l'obiettivo di identificare un sottolinguaggio di FOL tale che:

[7] È un errore comune pensare che una deduzione dimostri la verità della conclusione, ma non è così: una deduzione dimostra che una conclusione deriva delle premesse e quindi, se il calcolo logico utilizzato è corretto, che la conclusione è conseguenza logica delle premesse, ovvero che la conclusione *è vera nell'ipotesi che siano vere le premesse* (vedi il par. 7.2 e la già citata dispensa *Introduzione alla logica predicativa del primo ordine*, `http://home.dei.polimi.it//colombet/IC/`, Materiale didattico).

- il linguaggio sia comunque abbastanza espressivo per le applicazioni d'interesse;
- la deduzione si basi su una procedura di decisione e 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 compatibile con le esigenze applicative (ovvero richieda una quantità accettabile di risorse di calcolo).

La già menzionata *logica descrittiva* studia una famiglia di linguaggi di questo genere. Esistono svariate logiche descrittive[8], studiate fino dall'inizio degli anni Ottanta del Novecento e contraddistinte da acronimi (come $\mathcal{AL}$, $\mathcal{ALC}$, $\mathcal{SHIF}$, $\mathcal{SHIQ}$, $\mathcal{SHOIN}$, $\mathcal{SROIQ}$ e così via) il cui senso sarà chiarito più avanti (par. 7). In queste note presenteremo la DL nota come $\mathcal{SROIQ}$, che sta alla base del linguaggio OWL 2 DL (`http://www.w3.org/2007/OWL/wiki/OWL_Working_Group`), una raccomandazione emanata dal W3C il 27 ottobre 2009.

2.3 Livelli delle rappresentazioni

Nell'informatica si utilizzano sistemi di rappresentazione per scopi e usi differenti. Sono ad esempio sistemi di rappresentazione le basi di dati, i diagrammi UML, i documenti XML. Al di là delle differenze, tutti i sistemi di rappresentazione condividono alcune caratteristiche comuni.

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

- Livello 1: *modello concreto*. Il modello concreto di un frammento di realtà è una *rappresentazione dei fatti* che sussistono in tale frammento; i fatti sono rappresentati mediante *asserzioni* (ad esempio, "Marco Colombetti è il docente titolare del corso di Ingegneria della conoscenza") che riguardano determinati *oggetti* dell'universo (Marco Colombetti, il corso di Ingegneria della conoscenza).
- Livello 2: *modello semantico* (o *modello concettuale*). Il modello semantico di un frammento di realtà è una *rappresentazione dei concetti e delle relazioni* utilizzati per formulare il modello concreto; ad esempio, per esprimere il fatto dell'esempio precedente è necessario definire i concetti di persona (dotata di nome e cognome) e di corso (dotato di nome), nonché la relazione "docente titolare di" che può mettere in corrispondenza una persona con uno o più corsi.
- Livello 3: *linguaggio formale*. Il linguaggio formale di un sistema di rappresentazione è *l'insieme degli strumenti formali* utilizzabili per definire il modello semantico e il modello concreto; ad esempio il linguaggio di FOL, oppure il linguaggio di una DL.

A volte (ad esempio nel caso di UML) il livello 3 viene denominato *metamodello*. Vale la pena di chiarire, quindi, la differenza fra un linguaggio formale e un metamodello.

Linguaggio formale e metamodello

Nell'informatica si usano diversi tipi di linguaggi formali. Un *linguaggio di programmazione*, ad esempio, è un linguaggio formale utilizzabile per descrivere programmi, ovvero strutture di dati e algoritmi; in queste note consideriamo invece linguaggi formali per la rappresentazione di conoscenze dichiarative, che più semplicemente possiamo chiamare *linguaggi logici*.

I linguaggi logici consentono di scrivere espressioni costituite da simboli disposti secondo una specifica sintassi (testuale o grafica). In generale i simboli di un linguaggio logico possono essere distinti in due categorie: *simboli logici* e *simboli denotanti*. Consideriamo di nuovo l'espressione FOL

$$\forall x\,(\mathsf{Madre}(x) \leftrightarrow \mathsf{Donna}(x) \wedge \exists y\,\mathsf{genDi}(x,y))$$

[8] Il lettore avrà notato che a volte si parla di "logica", al singolare, e a volte di "logiche", al plurale. Quando si usa il singolare, come ad esempio in "logica descrittiva", si intende un'area disciplinare (come fisica, matematica e così via); quando invece si usa il plurale, come in "logiche descrittive", si intendono gli specifici linguaggi formali proposti e analizzati nell'area disciplinare.

I simboli Madre, Donna e genDi sono rispettivamente utilizzati per denotare due insiemi di oggetti e una relazione binaria fra oggetti nell'universo di un'applicazione. Si tratta quindi di simboli denotanti, nel senso che si assume che tali simboli abbiano, nell'ambito dell'applicazione di riferimento, una denotazione ben precisa; si noti tale denotazione non è fissata nella definizione del linguaggio formale, ma si basa, per così dire, sulle intenzioni di chi utilizza il linguaggio per rappresentare qualcosa. I simboli $\forall$, $\leftrightarrow$ e $\wedge$ sono invece simboli logici. Il significato di questi simboli è fissato una volta per tutte nella definizione di FOL; inoltre tali simboli non denotano entità dell'universo (come oggetti, insiemi di oggetti e relazioni fra oggetti), ma operano sul significato delle sottoespressioni cui sono applicati. Ritroveremo la distinzione fra simboli logici e simboli denotanti nelle DL.

Consideriamo ora un linguaggio formale come UML. Anche in questo caso abbiamo simboli denotanti (come ad es. le classi e le istanze) e simboli logici (come ad esempio le frecce di sottoclasse e di aggregato). Rispetto a FOL, tuttavia, c'è una differenza: l'interpretazione dei simboli denotanti non è del tutto libera, come nel caso di FOL, ma è invece parzialmente vincolata. Un simbolo di classe, ad esempio, non può rappresentare un'entità di tipo qualsiasi, ma deve rappresentare appunto una classe (nel senso della programmazione orientata agli oggetti); un simbolo d'istanza rappresenta l'istanza di una classe; e così via. In altre parole, il linguaggio UML contiene già dall'inizio un vincolo sull'interpretazione dei simboli denotanti, e questo vincolo deve essere rispettato quando il linguaggio viene utilizzato per rappresentare modelli semantici. Rispetto a FOL, quindi, UML presuppone alcune assunzioni sul tipo di realtà che viene modellato: per questo motivo, il linguaggio di UML è considerato non come un puro linguaggio formale, ma come un *metamodello* (dei sistemi software orientati agli oggetti).

2.4 I sistemi basati su conoscenze (KBS)

In che cosa un KBS si distingue, ad esempio, da una rappresentazione nel campo delle basi dei dati? La differenza principale sta in questo: nel campo delle basi di dati i modelli semantici (modelli concettuali E-R) rappresentano un passo importante nello sviluppo di un sistema, ma non fanno parte del prodotto finale; nei KBS, invece, i modelli semantici sono parte integrante del prodotto finale e vengono utilizzati a runtime, mediante processi di deduzione, per il normale funzionamento del sistema. Un KBS basato su una DL, ad esempio, è costituito dalle seguenti componenti (fig. 2.1):

- una *base di conoscenze* (KB), a sua volta costituita da:
  - una *terminological box* (*TBox*), in cui vengono specificate le relazioni logiche fra *classi* (o *concetti*);
  - una *role box* (*RBox*), in cui vengono specificate le relazioni logiche fra *proprietà* (dette anche *ruoli*);
  - un'*assertion box* (*ABox*), in cui vengono specificate le *asserzioni* (o *fatti*) concernenti specifici individui;
- un'*interfaccia di accesso*, costituita da oppurtune API che consentono di costruire, modificare e interrogare la KB;
- un repertorio di *strumenti software*, quali:
  - un *editor della base di conoscenze*, ovvero un'applicazione che consente la gestione dei contenuti della KB da parte di un operatore umano;
  - un *reasoner*, ovvero un'applicazione in grado di operare deduzioni a partire dalle conoscenze contenute nella KB;
- determinate *applicazioni software*, che utilizzano la KB e il reasoner in funzione di determinati obiettivi applicativi.

Nei KBS odierni l'interoperabilità fra una KB, gli strumenti e le applicazioni è assicurata dall'adozione sistematica di standard basati su XML; ciò consente, in particolare, di utilizzare strumenti e applicazioni sviluppati da produttori indipendenti.

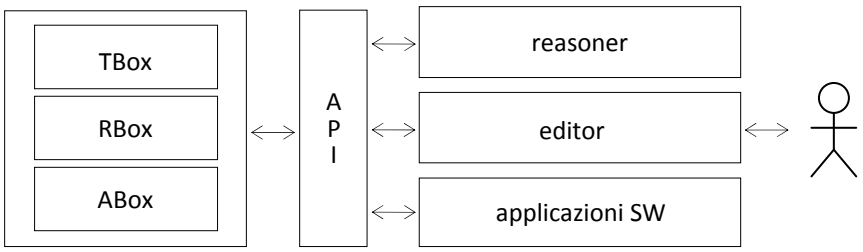

Figura 2.1. L'architettura di un KBS basato su una logica descrittiva.

Come si è già detto, in questo corso ci interessiamo ai linguaggi per la definizione di KB basati su logiche descrittive; alcuni esempi di linguaggi del genere sono il linguaggio DAML+OIL, basato su una DL denominata $\mathcal{SHIQ}$; il linguaggio OWL DL, (una raccomandazione W3C approvata nel 2004), basato su una DL denominata $\mathcal{SHOIN}(\mathrm{D}_n)$; e il linguaggio OWL 2 DL (la più recente raccomandazione W3C, approvata nel 2009), basato su una DL denominata $\mathcal{SROIQ}(\mathrm{D}_n)$, che estende sia $\mathcal{SHIQ}$, sia $\mathcal{SHOIN}(\mathrm{D}_n)$. A sua volta ogni DL si caratterizza per l'utilizzo di espressioni logiche costruite a partire da un certo numero di simboli logici, detti *costruttori*, scelti da un repertorio di opzioni possibili.

Nella parte II introdurremo le espressioni della logica $\mathcal{SROIQ}(\mathrm{D}_n)$ utilizzando la sintassi logica definita in modo completo nell'appendice I (per un confronto con altre notazioni convenzionali, come la sintassi funzionale di OWL 2 o la sintassi di Manchester, si veda l'appendice II). Nel paragrafo 4 ci occuperemo delle espressioni utilizzate per caratterizzare *classi*; tali espressioni formano la TBox di una KB. Nel paragrafo 5 introdurremo le espressioni utilizzate per caratterizzare *proprietà*; come vedremo, alcune di queste espressioni appartengono alla TBox, altre alla RBox della KB. Nel paragrafo 6, poi, vedremo le espressioni utilizzate per caratterizzare individui, e che appartengono all'ABox. Va tenuto presente, comunque, che la distinzione fra TBox e ABox, importante dal punto di vista concettuale, non è sempre netta: in certi casi, come vedremo, la stessa cosa può essere detta con un'enunciato di TBox o con un'enunciato di ABox.

3. L'ingegneria della conoscenza e il web semantico

A partire dagli anni novanta del Novecento ha cominciato a profilarsi un nuovo settore di grande interesse per l'ingegneria della conoscenza: il *web semantico*. Prima di occuparci di questo descriveremo brevemente alcuni fattori che stanno influenzando sensibilmente il progresso dell'informatica, e più precisamente: la centralità dei dati, l'interoperabilità delle applicazioni e l'utilizzo delle tecnologie del web in reti di diversi tipi.

La centralità dei dati

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

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 dei casi si realizzerà 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: non è quindi scorretto affermare che nel corso degli anni l'interesse principale 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 in formato proprietario. Questo tipo di applicazioni crea problemi già all'interno della singola azienda, perché in una situazione del genere è difficile interfacciare applicazioni diverse; a maggior ragione è arduo far interagire 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, a causa del fatto che è difficile garantire che agli stessi concetti corrispondano semopre gli stessi tag, e viceversa che a un certo tag venga assegnato sempre lo stesso significato. In altre parole, l'adozione di XML risolve almeno in parte il problema dell'interoperabilità sul piano della sintassi (ovvero della struttura dei dati), ma non sul piano della semantica (ovvero del significato dei dati).

L'interoperabilità esterna e i sistemi aperti

Da qualche anno è nato un forte interesse per le applicazioni aperte, cioè in grado di interoperare con applicazioni appartenenti ad aziende federate (all'interno di un'extranet) 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 le loro maggiori potenzialità si rivelano in 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 individuato 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, ma non la loro semantica. 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. L'esempio suggerisce che l'interoperabilità in un ambiente aperto richieda 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.

Il web semantico

Ufficialmente, il web è nato il 6 agosto del 1991, il giorno in cui Tim Berners-Lee ha messo il primo sito web in linea su internet. 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 nel web abbiano una struttura chiaramente definita (come è possibile fare utilizzando HTML o,

più in generale, un linguaggio applicativo XML), ma è anche necessario che i dati abbiano un significato condiviso dalla comunità che li utilizza (vedi Berners-Lee *et al*., 2001; Shadbolt *et al*., 2006).

Per consentire lo sviluppo del web in questa direzione sono state proposte e si stanno tuttora elaborando diverse tecnologie (come i linguaggi RDF, RDFS e OWL, tutti standard raccomandati dal W3C), che analizzeremo in dettaglio in questo corso.

Testi disponibili

Sul tema del web semantico e dei relativi linguaggi (RDFS ed OWL) sono stati pubblicati numerosi testi. Ecco un elenco parziale:

1. Alesso, H.P., and Smith, C.F. (2008). *Thinking on the Web: Berners Lee, Gödel and Turing*, Wiley Blackwell.
2. Allemang, D,. and Hendler, J. (2008). *Semantic Web for the working ontologist: Effective modeling in RDFS and OWL*, Morgan Kaufmann.
3. Antoniou, G., van Harmelen, F. (2008). *Semantic Web primer (2nd edition)*, MIT Press.
4. Breitman, K.K., Casanova, M.A., and Truszkowski, W. (2007). *Semantic Web: Concepts, technologies and applications*, Springer.
5. Davies, J., Studer, R., and Warren, P. (2006). *Semantic Web technologies: Trends and research in ontology-based systems*, Wiley Blackwell.
6. Fensel, D. (2005). *Spinning the Semantic Web: Bringing the World Wide Web to its full potential*, MIT Press.
7. Hitzler, P., Krötzsch, M., and Rudolph, S. (2009). *Foundations of Semantic Web Technologies*, Chapman & Hall/CRC.
8. Lacy, L.W. (2005). *QWL: Representing information using the Web Ontology Language*, Trafford Publishing.
9. Passin, T.B. (2004). *Explorer's guide to the Semantic Web*, Manning Publications.
10. Pollock, J.T. (2009). *Semantic Web for dummies*, John Wiley & Sons.

Per chi volesse integrare lo studio di queste note consigliamo in particolare il testo 7.

Parte II: La logica descrittiva e il linguaggio OWL

4. Le classi

4.1 Descrizioni e definizioni di classi

Il linguaggio formale di una DL consente di definire *classi* (dette anche *concetti* o *termini*). Ad esempio l'espressione

Donna

rappresenta una *classe atomica*, ovvero una classe identificata da un nome. L'espressione[9]

(4.1) Persona ⊓ Femmina

che si legge "persona e femmina" o "persona intersezione femmina", rappresenta invece una *classe complessa*, ovvero una classe descritta da un'espressione contenente un *costruttore di classi*, e più precisamente il costruttore d'*intersezione* (o *congiunzione*) ⊓.

L'espressione

(4.2) Donna ≡ Persona ⊓ Femmina

che si legge "donna equivale a persona e femmina", rappresenta un'*equivalenza fra classi*. In queste note seguiremo l'uso diffuso di indicare una generica classe atomica con la lettera A o B e una generica classe arbitraria (ovvero atomica o complessa) con la lettera C o D. In generale un'espressione della forma

$C \equiv D$

si legge "C equivale a D" e rappresenta l'equivalenza fra le due classi C e D, ovvero il fatto che le estensioni delle due classi siano uguali in qualunque universo. Nel caso particolare in cui si abbia

$A \equiv C$

(dove A è una classe atomica) l'espressione è detta *definizione di classe*. Dunque la 4.2 è una definizione di classe: più precisamente si tratta della definizione della classe Donna a partire dalle classi Persona e Femmina. L'espressione

(4.3) Ragazza ⊑ Donna

che si legge "ragazza è sottoclasse di donna" e intuitivamente significa che ogni ragazza è una donna, rappresenta una relazione di *sottoclasse*, detta anche *sussunzione* nel gergo delle DL. In generale una relazione di sottoclasse,

$C \sqsubseteq D$

esprime il fatto che in qualsiasi universo l'estensione della classe C (ovvero, l'insieme degli oggetti che soddisfano la definizione della classe C) è un sottoinsieme, proprio o improprio, dell'estensione della classe D. L'equivalenza $C \equiv D$ coincide con la doppia relazione di sottoclasse $C \sqsubseteq D$ e $D \sqsubseteq C$. Nel caso particolare in cui si abbia

$A \sqsubseteq C$

(dove A è una classe atomica) l'espressione è detta *descrizione di classe*. Si noti che le espressioni del tipo $C \sqsubseteq D$ e $C \equiv D$ si possono sempre ridurre, introducendo nuove classi atomiche, a espressioni del tipo $A \sqsubseteq C$ e $A \equiv C$. Ad esempio, la relazione di sottoclasse

[9] Per i simboli delle DL si è utilizzato il subset Mathematical Operators del font Lucida Sans Unicode. I nomi di classi ecc. sono invece in font Calibri.

$A1 \sqcap A2 \sqsubseteq B1 \sqcap B2$

si può ridurre alle due espressioni:

$A3 \equiv A1 \sqcap A2$

$A3 \sqsubseteq B1 \sqcap B2$

4.2 Semantica delle classi e degli enunciati

Chiameremo *enunciati* le espressioni che esprimono relazioni di sottoclasse o di equivalenza fra classi. Gli enunciati possono essere assunti come *assiomi*, in modo analogo a quanto si fa nelle teorie del primo ordine: un assioma è semplicemente un enunciato che si assume essere vero in tutti gli universi d'interesse. Naturalmente, dagli assiomi sarà importante dedurre opportuni *teoremi* con l'uso di un reasoner[10].

Chiamiamo poi *ontologia* un insieme finito di assiomi; un'ontologia non è altro, quindi, che una teoria assiomatica del primo ordine esprimibile in una DL. Come vedremo, la funzione di un'ontologia è di specificare un modello concettuale del frammento di realtà che interessa ai fini di qualche applicazione. Come abbiamo già detto (par. 2.4), in un'ontologia l'insieme degli assiomi che stabiliscono relazioni logiche fra classi è detto *TBox* (dove "*T*" sta per *terminological*).

Classi ed enunciati sono espressioni di un linguaggio formale (per una grammatica completa del linguaggio si veda l'appendice I). Per queste espressioni è possibile specificare una semantica formale, che associa a ogni classe e a ogni enunciato un'interpretazione definita in modo insiemistico (per una definizione sistematica della semantica formale secondo queste linee si veda l'appendice III). È anche possibile seguire una strada diversa, ovvero associare una semantica alle espressioni di una DL in modo indiretto, fornendo una traduzione di classi ed enunciati nel linguaggio di FOL (le regole di traduzione da $\mathcal{SROIQ}(D_n)$ a FOL sono riportate nell'appendice IV). Non si deve però dimenticare una cosa: mentre a una classe o a un enunciato DL è sempre possibile associare una formula di FOL, non è vero l'inverso: le DL sono sottoinsiemi propri di FOL, e quindi esistono formule di FOL che non corrispondono ad alcuna classe o enunciato DL (fig. 4.1).

Semantica delle classi

Come in FOL, la semantica di un'espressione simbolica DL è definita nell'ambito di una struttura insiemistica detta *modello*. Un modello è costituito da due componenti; la prima, detta *universo*, è un insieme non vuoto i cui elementi, chiamati *oggetti*, vanno intuitivamente interpretati come 'oggetti del discorso' (ovvero gli oggetti di cui parlano le espressioni simboliche); la seconda componente è una funzione, detta *funzione d'interpretazione* o semplicemente *interpretazione*, che associa a certi simboli del linguaggio determinate entità insiemistiche costruite nell'universo. Per il momento ci limitiamo a considerare l'interpretazione delle classi, rimandando ai paragrafi successivi l'interpretazione di altri tipi di espressioni (proprietà e individui).

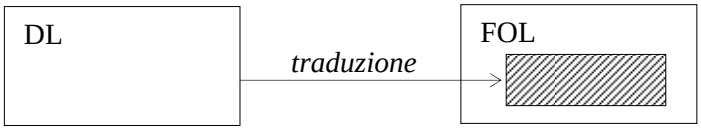

Figura 4.1. Una DL come frammento di FOL.

[10] Sfortunatamente, la comunità internazionale degli utenti di OWL ha adottato una terminologia che contrasta con la terminologia logica tradizionalmente accettata: nel mondo di OWL "assioma" significa semplicemente enunciato, e i teoremi vengono denominati "assiomi dedotti" (*inferred axioms*). Per evitare malintesi, in queste disponse ci atteniamo invece alla terminologia logica tradizionale.

Dato un modello $M = \langle \Delta, -^I \rangle$, dove Δ è un universo non vuoto di oggetti e $-^I$ è una funzione d'interpretazione, a ogni classe atomica A viene associato un insieme di oggetti $A^I \subseteq \Delta$. L'insieme A^I è detto *estensione* di A in M; si dice anche che la classe A *denota* la propria estensione A^I. Intuitivamente, A^I rappresenta l'insieme di tutti gli oggetti dell'universo che esemplificano la classe A: ad esempio, se la classe atomica è Donna, l'estensione Donna^I sarà costituita da tutti gli oggetti dell'universo che sono donne.

Nelle DL sono previste due classi atomiche predefinite, $\top$ (*top*) e $\bot$ (*bottom*), che corrispondono rispettivamente alla *classe universale* (che denota la totalità degli oggetti dell'universo) e alla *classe vuota* (che denota l'insieme vuoto di oggetti dell'universo). L'interpretazione di queste classi è quindi definita come

$\top^I = \Delta$

$\bot^I = \varnothing$

Una volta definita l'interpretazione delle classi atomiche, si associa ricorsivamente un'estensione anche alle classi complesse estendendo la funzione d'interpretazione tramite alcune *regole semantiche*. Il costruttore d'intersezione $\sqcap$ corrisponde al connettivo booleano di congiunzione e all'operazione insiemistica di intersezione; la regola semantica corrispondente definisce quindi l'estensione di $C \sqcap D$ come:

$(C \sqcap D)^I = C^I \cap D^I$

Ad esempio, la classe complessa 4.1 si interpreta come:

$(\text{Persona} \sqcap \text{Femmina})^I = \text{Persona}^I \cap \text{Femmina}^I$

Semantica degli enunciati

Mentre le classi sono espressioni che denotano insiemi di oggetti dell'universo, gli enunciati sono espressioni che in un modello assumono un *valore di verità* (*vero* o *falso*). Di ogni tipo di enunciato è quindi necessario definire le *condizioni di verità*, ovvero le condizioni in cui l'enunciato assume valore vero (dato che ci muoviamo nell'ambito della logica classica, quando un enunciato non è vero è necessariamente falso: non sussiste una terza possibilità). Per quanto riguarda gli enunciati di sottoclasse, diciamo che l'enunciato $C \sqsubseteq D$ è vero in M (e scriviamo $M \vDash C \sqsubseteq D$) se, e solo se, nel modello l'estensione di C è contenuta nell'estensione di D:

$M \vDash C \sqsubseteq D$ sse $C^I \subseteq D^I$

Per quanto riguarda invece gli enunciati di equivalenza di classi, diciamo che l'enunciato $C \equiv D$ è vero in M se, e solo se, l'estensione di C è uguale all'estensione di D:

$M \vDash C \equiv D$ sse $C^I = D^I$

Ad esempio, per gli enunciati 4.2 e 4.3 vale:

$M \vDash \text{Donna} \equiv \text{Persona} \sqcap \text{Femmina}$ sse $\text{Donna}^I = \text{Persona}^I \cap \text{Femmina}^I$

$M \vDash \text{Ragazza} \sqsubseteq \text{Donna}$ sse $\text{Ragazza}^I \subseteq \text{Donna}^I$

Enunciati contenenti $\top$ e $\bot$

Gli enunciati della forma $C \sqsubseteq D$ oppure $C \equiv D$ possono contenere le classi predefinite $\top$ e $\bot$. Ad esempio, l'enunciato

$\text{Persona} \sqsubseteq \bot$

significa che l'insieme delle persone è vuoto, ovvero che nell'universo di riferimento non ci sono persone. Ciò può essere verificato sulla base dell'interpretazione:

$M \vDash \text{Persona} \sqsubseteq \bot$ sse $\text{Persona}^I \subseteq \bot^I = \varnothing$

L'enunciato

$\top \sqsubseteq \text{Persona}$

significa invece che nell'universo di riferimento ci sono soltanto persone, ovvero che l'insieme delle persone coincide con l'intero universo. Ciò può essere verificato sulla base dell'interpretazione:

$M \models \top \sqsubseteq \mathsf{Persona}$ sse $\top^I = \Delta \subseteq \mathsf{Persona}^I$

Ogni enunciato della forma

$C \sqsubseteq \top$

è *logicamente vero* (o *tautologico*), cioè vero in ogni possibile modello, in quanto afferma l'ovvia verità che l'insieme degli oggetti dell'universo che appartengono a C^I è contenuto nell'universo. Analogamente è logicamente vero ogni enunciato della forma

$\bot \sqsubseteq C$

in quanto afferma che l'insieme vuoto è contenuto nell'insieme degli oggetti dell'universo che appartengono a C^I. In particolare è logicamente vero l'enunciato

$\bot \sqsubseteq \top$

Al contrario l'enunciato

$\top \sqsubseteq \bot$

risulta *logicamente falso* (o *insoddisfacibile*), cioè falso in ogni possibile modello, in quanto afferma che l'universo è vuoto, mentre per ipotesi l'universo contiene sempre almeno un oggetto (questa assunzione vale sia per i modelli di FOL, sia per i modelli delle DL).

Negazione e disgiunzione

Gli uomini possono essere definiti come quelle persone che non sono femmine. Utilizzando il costruttore $\neg$ di *complemento* (o *negazione*) è quindi possibile definire:

$\mathsf{Uomo} \equiv \mathsf{Persona} \sqcap \neg\mathsf{Femmina}.$

La regola semantica associata al complemento è

$(\neg C)^I = \Delta \setminus C^I$

(dove l'operatore \ indica la differenza fra insiemi). Ad esempio,

$(\mathsf{Persona} \sqcap \neg\mathsf{Femmina})^I = \mathsf{Persona}^I \cap (\Delta \setminus \mathsf{Femmina})^I = \mathsf{Persona}^I \setminus \mathsf{Femmina}^I$

Un altro costruttore utile è l'*unione* (o *disgiunzione*) $\sqcup$, che permette ad esempio di definire gli esseri viventi come unione di vegetali e animali:

$\mathsf{Vivente} \equiv \mathsf{Vegetale} \sqcup \mathsf{Animale}.$

La regola semantica associata al costruttore d'unione è

$(C \sqcup D)^I = C^I \cup D^I$

Dalle regole semantiche si ottiene in particolare che[11]:

$\neg\top$	equivale a $\bot$	$\neg(C \sqcap D)$	equivale a $\neg C \sqcup \neg D$
$\neg\bot$	equivale a $\top$	$\neg(C \sqcup D)$	equivale a $\neg C \sqcap \neg D$
$\neg\neg C$	equivale a C		

4.3 Classi definite tramite restrizioni di proprietà

Oltre che utilizzando i costruttori booleani, le classi possono essere definite utilizzando *proprietà* (o *ruoli*), che rappresentano relazioni binarie fra oggetti dell'universo. Così come una classe denota un insieme di oggetti dell'universo (chiamato *estensione della classe*), una proprietà denota una relazione binaria, ovvero un insieme di coppie ordinate di oggetti dell'universo (chiamato *estensione della proprietà*).

[11] Più in generale le classi formano, rispetto ai costruttori di complemento, intersezione e unione, un'*algebra booleana*.

La restrizione esistenziale

Un esempio di enunciato che utilizza una proprietà è

(4.4) $\mathsf{Madre} \sqsubseteq \exists \mathsf{genDi}$

che intuitivamente significa “ogni madre è genitore di almeno un oggetto”. L’espressione

(4.5) $\exists \mathsf{genDi}$

è una classe complessa, formata dal costruttore di *restrizione esistenziale* $\exists$ e dalla *proprietà* genDi. In un modello $M = \langle \Delta, -^I \rangle$ la funzione d’interpretazione assegna a ogni proprietà una relazione binaria in Δ, ovvero un insieme di coppie ordinate di oggetti dell’universo. Data una proprietà R abbiamo quindi

$R^I \subseteq \Delta \times \Delta$

Come ogni classe, una classe complessa $\exists R$ denota un insieme di oggetti dell’universo. Tale insieme è definito dalla regola semantica seguente:

$(\exists R)^I = \{x \in \Delta \mid \text{per qualche } y \in \Delta, \langle x,y \rangle \in R^I\}$

Ora possiamo dare le condizioni di verità dell’enunciato 4.4:

$M \vDash \mathsf{Madre} \sqsubseteq \exists \mathsf{genDi}$ sse $\mathsf{Madre}^I \subseteq \{x \in \Delta \mid \text{per qualche } y \in \Delta, \langle x,y \rangle \in \mathsf{genDi}^I\}$

Si noti che mentre una proprietà R denota *una relazione binaria* (ovvero un insieme di coppie ordinate di oggetti dell’universo), la classe complessa $\exists R$ denota *un insieme di oggetti*, e precisamente l’insieme di oggetti dell’universo che sono in relazione R con qualche oggetto (vedi la fig. 4.2, in cui le frecce rappresentano le coppie ordinate appartenenti alla relazione R).

Una restrizione esistenziale può essere *qualificata*; ad esempio, la classe complessa

(4.6) $\exists \mathsf{genDi.Femmina}$

denota l’insieme di tutti gli oggetti dell’universo che sono genitori di almeno una femmina. La regola semantica corrispondente è:

$(\exists R.C)^I = \{x \in \Delta \mid \text{per qualche } y \in C^I, \langle x,y \rangle \in R^I\}$

Ad esempio la classe 4.6 si interpreta come:

$(\exists \mathsf{genDi.Femmina})^I = \{x \in \Delta \mid \text{per qualche } y \in \mathsf{Femmina}^I, \langle x,y \rangle \in \mathsf{genDi}^I\}$

La classe complessa $\exists R.C$ denota quindi l’insieme degli oggetti dell’universo che sono in relazione R con qualche oggetto appartenente all’estensione della classe C (vedi la fig. 4.2). È facile verificare che l’espressione $\exists R$ equivale all’espressione $\exists R.\top$; infatti:

$(\exists R.\top)^I = \{x \in \Delta \mid \text{per qualche } y \in \top^I, \langle x,y \rangle \in R^I\} = \{x \in \Delta \mid \text{per qualche } y \in \Delta, \langle x,y \rangle \in R^I\} = (\exists R)^I$

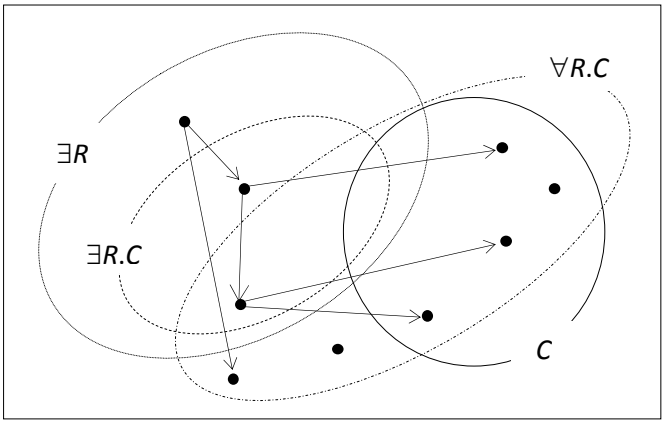

Figura 4.2. Estensioni delle classi $\exists R$, $\exists R.C$ e $\forall R.C$.

Al di là della semantica formale, che definisce il significato delle espressioni DL in modo rigoroso e privo di ambiguità, è importante imparare a 'leggere' le classi in modo intuitivo ma corretto. Un errore comune consiste nel leggere le espressioni 4.5 e 4.6 rispettivamente come "esiste un genitore" ed "esiste un genitore di una femmina". Queste due letture sono errate perché trattano una classe come se esprimesse una *proposizione*; bisogna invece ricordare che una classe rappresenta l'*insieme degli oggetti dell'universo che esemplificano la classe*. Pertanto la classe 4.5 rappresenta l'insieme degli oggetti dell'universo che sono genitori di qualcuno, e la classe 4.6 rappresenta l'insieme degli oggetti dell'universo che sono genitori di almeno una femmina.

La restrizione universale

Le DL prevedono anche l'uso di un costruttore di *restrizione universale*, $\forall$. Ad esempio, la classe

(4.7) $\forall \mathsf{genDi}.\mathsf{Femmina}$

denota l'insieme di tutti gli oggetti dell'universo che sono genitori *soltanto* di femmine (ovvero gli oggetti tali che tutti i loro figli, *se ne hanno*, sono femmine). L'interpretazione di una classe della forma $\forall R.C$, dove R è una proprietà e C è una classe arbitraria, è:

$$(\forall R.C)^I = \{x \in \Delta \mid \text{se } \langle x,y\rangle \in R^I, \text{ allora } y \in C^I\}$$

Ad esempio la classe 4.7 si interpreta come:

$$(\forall \mathsf{genDi}.\mathsf{Femmina})^I = \{x \in \Delta \mid \text{se } \langle x,y\rangle \in \mathsf{genDi}^I, \text{ allora } y \in \mathsf{Femmina}^I\}$$

Anche in questo caso valgono le considerazioni già fatte per la restrizione esistenziale: la classe 4.7 non rappresenta la proposizione "tutti sono genitori di femmine", bensì l'insieme degli oggetti dell'universo che sono genitori di sole femmine. Un'osservazione importante è che nelle DL, come del resto anche in FOL, la restrizione universale *non* presuppone l'esistenza di almeno un oggetto con la proprietà specificata. Ad esempio la classe 4.7 denota sia gli oggetti dell'universo che sono genitori e i cui figli sono tutti femmine, sia gli oggetti dell'universo che non sono genitori di alcun oggetto. In altre parole la lettura più precisa della classe 4.7 è: "l'insieme degli oggetti dell'universo che o non sono genitori di nessun oggetto, oppure sono genitori di sole femmine" oppure, in modo forse più chiaro nonostante l'uso della doppia negazione, "l'insieme degli oggetti dell'universo che non sono genitori oggetti che non siano femmine". In generale, la classe complessa $\forall R.C$ denota l'insieme degli oggetti dell'universo che o non sono in relazione R con altri oggetti, o sono in relazione R soltanto con oggetti appartenenti all'estensione della classe C (fig. 4.2); ovvero, $\forall R.C$ denota l'insieme degli oggetti dell'universo che *non* sono in relazione R con oggetti che *non* appartengano alla classe C.

Mentre la restrizione esistenziale si usa sia nella *forma semplice* $\exists R$ (vedi la 4.5), sia nella *forma qualificata* $\exists R.C$ (vedi la 4.6), la restrizione universale si usa solo nella forma qualificata $\forall R.C$ perché la classe $\forall R$, per definizione equivalente a $\forall R.\top$, coincide semplicemente con $\top$. Infatti:

$$(\forall R.\top)^I = \{x \in \Delta \mid \text{se } \langle x,y\rangle \in R^I, \text{ allora } y \in \top^I\} = \Delta$$

Si noti infine che, grazie alle regole semantiche introdotte, si ha 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	$\exists R.\neg C$
$\neg\forall R$	equivale a	$\exists R.\bot$ (e quindi ha estensione vuota)
$\forall R.C \sqcap \forall R.D$	equivale a	$\forall R.(C \sqcap D)$
$\exists R.C \sqcup \exists R.D$	equivale a	$\exists R.(C \sqcup D)$

4.4 Restrizioni di cardinalità

Se R è una proprietà, in molte DL è possibile esprimere sui ruoli *restrizioni di cardinalità semplici*, come $\leq nR$ e $\geq nR$, oppure *restrizioni di cardinalità qualificate*, come $\leq nR.C$ e $\geq nR.C$, dove n è un numero naturale arbitrario (un intero senza segno). Ad esempio, gli oggetti con almeno tre figlie si definiscono come:

(4.8) $\mathsf{GenDi3F} \equiv \geq 3\ \mathsf{genDi.Femmina}$

Ancora una volta richiamiamo l'attenzione sul fatto che la classe complessa

$\geq 3\ \mathsf{genDi.Femmina}$

non rappresenta la proposizione "esistono almeno tre genitori di femmine" oppure "esistono genitori di almeno tre femmine", bensì l'insieme degli oggetti dell'universo che sono genitori di almeno tre femmine. Le restrizioni di cardinalità si interpretano nel modo seguente (dove $\#X$ indica la cardinalità dell'insieme X):

$(\leq nR)^I = \{x \in \Delta \mid \#\{y \in \Delta \mid \langle x,y\rangle \in R^I\} \leq n\}$

$(\geq nR)^I = \{x \in \Delta \mid \#\{y \in \Delta \mid \langle x,y\rangle \in R^I\} \geq n\}$

$(\leq nR.C)^I = \{x \in \Delta \mid \#\{y \in C^I \mid \langle x,y\rangle \in R^I\} \leq n\}$

$(\geq nR.C)^I = \{x \in \Delta \mid \#\{y \in C^I \mid \langle x,y\rangle \in R^I\} \geq n\}$

Pertanto le condizioni di verità dell'enunciato 4.8 sono:

$M \vDash \mathsf{GenDi3F} \equiv \geq 3\ \mathsf{genDi.Femmina}$ sse $\mathsf{GenDi3F}^I = \{x \in \Delta \mid \#\{y \in \mathsf{Femmina}^I \mid \langle x,y\rangle \in \mathsf{genDi}^I\} \geq 3\}$

Le restrizioni di cardinalità esatte, semplici o qualificate, si possono ora esprimere nel modo seguente:

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

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

Si noti poi che

$\geq 1R.C$ equivale a $\exists R.C$

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

Inoltre la classe

$\geq 0R.C$

è del tutto inutile perché equivale semplicemente a $\top$. Infatti questa denota l'insieme di tutti gli oggetti dell'universo che hanno una relazione di tipo R con zero o più oggetti: indipendentemente da R, tutti gli oggetti del dominio soddisfano questa condizione.

Esistono poi altri due tipi di restrizioni, che saranno introdotti in seguito (*restrizione di riflessività locale*, par. 5.7, e *restrizione di valore*, par. 5.1).

4.5 Classi disgiunte

Con gli strumenti a disposizione è facile specificare che due classi sono disgiunte, nel senso che in ogni universo è vuota l'intersezione delle loro estensioni. Ad esempio, l'enunciato

$\mathsf{Donna} \sqcap \mathsf{Uomo} \sqsubseteq \bot$

afferma che in ogni universo l'insieme delle donne è disgiunto dall'insieme degli uomini. È anche possibile specificare in modo conciso che un certo numero di classi arbitrarie sono a due a due disgiunte:

$\mathit{DisCla}(C_1,\ldots,C_n)$

Questo assioma equivale a tutti gli effetti all'inserimento nella TBox degli $n\cdot(n-1)/2$ assiomi della forma $C_i \sqcap C_j \sqsubseteq \bot$, con $1 \leq i < j \leq n$. Un altro tipo di assioma,

$\mathit{DisUni}(A,C_1,\ldots,C_n)$

consente di definire la classe atomica A come unione disgiunta delle classi arbitrarie C_1, ..., C_n. Tale assioma equivale infatti a:

$A \equiv C_1 \sqcup ... \sqcup C_n$
$DisCla(C_1,...,C_n)$

5. Le proprietà

Nel paragrafo precedente abbiamo visto come utilizzare proprietà per specificare classi complesse. In questo paragrafo vedremo che è possibile anche definire relazioni logiche che sussistono fra proprietà. Gli assiomi utilizzati a questo fine appartengono a volte alla TBox, a volte alla RBox (dove "R" sta per *role*, sinonimo di "proprietà").

5.1 Proprietà top e proprietà bottom

Nella logica $\mathcal{SROIQ}(D_n)$ esistono due proprietà predefinite, $\dot{\perp}$ (proprietà *bottom*) e $\dot{\top}$ (proprietà *top*), la cui estensioni sono rispettivamente la relazione vuota e la relazione universale:

$\dot{\perp}^I = \varnothing$,
$\dot{\top}^I = \{\langle x,y\rangle \in \Delta^2 \mid x \in \Delta \text{ e } y \in \Delta\}$

La proprietà top, in particolare, consente di specificare varie forme di quantificazione esistenziale che non sarebbero esprimibili altrimenti. Consideriamo infatti il problema di rappresentare in $\mathcal{SROIQ}(D_n)$ i due casi più semplici di quantificazione FOL, ovvero $\forall x A(x)$ e $\exists x A(x)$. Il primo enunciato corrisponde semplicemente a

$\top \sqsubseteq A$

che impone a ogni oggetto dell'universo di appartenere all'estensione di A. Il secondo enunciato, invece, non si può rappresentare così facilmente. Si consideri però che esiste un oggetto nell'estensione di A se, e solo se, ogni oggetto dell'universo è messo in relazione con tale oggetto dalla proprietà universale. Dunque l'enunciato FOL $\exists x A(x)$ è equivalente a[12]:

$\top \sqsubseteq \exists \dot{\top}.A$

Non è invece possibile utilizzare la proprietà top per specificare la cardinalità di una classe, utilizzandola all'interno di una restrizione di cardinalità. Il motivo è che la proprietà top in $\mathcal{SROIQ}(D_n)$ è considerata come una *proprietà composita* (vedi il par. 5.8), e pertanto non può essere utilizzata nelle restrizioni di cardinalità. Un metodo indiretto per specificare la cardinalità di una classe verrà presentato nella parte III (par. 8.3).

5.2 Dominio e codominio

Come già sappiamo le proprietà rappresentano relazioni binarie nell'universo, ovvero insiemi di coppie ordinate di oggetti dell'universo. In generale tali relazioni hanno senso solo limitatamente a certi sottoinsiemi dell'universo: ad esempio, la proprietà `genDi` mette in relazione fra loro due persone o comunque due esseri viventi e non è appropriato applicarla, poniamo, alle pietre o alle nuvole. A ogni proprietà R si associano quindi due insiemi di oggetti, detti il *dominio* e il *codominio* (in inglese *domain* e *range*) di R, che rappresentano gli insiemi di oggetti che possono essere legati da R.

Utilizzando gli strumenti logici che abbiamo già a disposizione, il dominio C e il codominio D di una proprietà R si definiscono nel modo seguente:

(5.1) $\exists R \sqsubseteq C$
(5.2) $\top \sqsubseteq \forall R.D$

[12] Per la verità, la formulazione originale di $\mathcal{SROIQ}(D_n)$ (Horrocks *et al.*, 2006) non consente un uso esplicito della proprietà top nelle espressioni del linguaggio; questa possibilità è però prevista nelle specifiche del linguaggio OWL 2 DL (`http://www.w3.org/2007/OWL/wiki/OWL_Working_Group`).

L'enunciato 5.1 dice che l'insieme di tutti gli oggetti che sono in relazione R con qualche oggetto dell'universo è contenuto nell'estensione di C, e ciò significa appunto che la classe C è il dominio della proprietà R. Per quanto riguarda l'enunciato 5.2, ricordiamo che la classe $\forall R.D$ denota l'insieme di tutti gli oggetti che o non sono in relazione R con alcun oggetto dell'universo, oppure sono in relazione R soltanto con oggetti appartenenti all'estensione di D. L'enunciato 5.2 dice poi che l'estensione della classe $\forall R.D$ coincide con l'intero universo; tale enunciato afferma quindi che se un oggetto dell'universo è in relazione R con qualche oggetto, quest'ultimo appartiene all'estensione di D: in altre parole, D è il codominio di R.

Come esempio della definizione di dominio e codominio possiamo specificare nel modo seguente il fatto che la proprietà "proprietario di" metta in relazione una persona con un bene:

$\exists \mathsf{propDi} \sqsubseteq \mathsf{Persona}$

$\top \sqsubseteq \forall \mathsf{propDi}.\mathsf{Bene}$

Va notato che gli assiomi di dominio e di codominio, pur contribuendo a definire una proprietà, appartengono alla TBox, in quanto specificano relazioni logiche fra classi, e più precisamente fra le classi $\top$, C, $\exists R$ e $\forall R.D$. In genere gli enunciati 5.1 e 5.2 si abbreviano come

(5.3) $\mathit{Dom}(R,C)$

(5.4) $\mathit{Rng}(R,D)$

Relativamente all'esempio scriveremo quindi

$\mathit{Dom}(\mathsf{propDi},\mathsf{Persona})$

$\mathit{Rng}(\mathsf{propDi},\mathsf{Bene})$

Per ragioni di leggibilità e concisione, infine, nel seguito di queste note utilizzeremo spesso una notazione ancora più abbreviata, rappresentando la coppia di enunciati 5.3 e 5.4 con l'unico enunciato:

$R\colon C \longrightarrow D$

Domini e di codomini multipli

Occorre fare molta attenzione all'eventuale presenza di più assiomi che definiscono congiuntamente il dominio e il codominio di una proprietà. Supponiamo ad esempio di scrivere

$R\colon C_1 \longrightarrow D_1$

$R\colon C_2 \longrightarrow D_2$

Il significato complessivo dei due enunciati è il seguente:

$R\colon C_1 \sqcap C_2 \longrightarrow D_1 \sqcap D_2$

Infatti i due enunciati di partenza equivalgono ai quattro enunciati

$\exists R \sqsubseteq C_1$ $\qquad$ $\exists R \sqsubseteq C_2$

$\top \sqsubseteq \forall R.D_1$ $\qquad$ $\top \sqsubseteq \forall R.D_2$

chea loro volta equivalgono ai due enunciati

$\exists R \sqsubseteq C_1 \sqcap C_2$

$\top \sqsubseteq \forall R.(D_1 \sqcap D_2)$

Quindi aggiungendo a una specifica di domininio e di codominio di una proprietà R una nuova specifica di dominio e di codominio per la stessa proprietà, il dominio e il codominio di R non si 'allargano', bensì si 'restringono', riducendosi rispettivamente alle intersezioni dei due domini e dei due codomini. Ad esempio, affermare che

$\mathsf{propDi}\colon \mathsf{Persona} \longrightarrow \mathsf{BeneMobile}$

$\mathsf{propDi}\colon \mathsf{Persona} \longrightarrow \mathsf{BeneImmobile}$

significa dire che una persona può essere proprietaria soltanto di oggetti dell'universo che siano contemporaneamente beni mobili e beni immobili. Se invece, come è ragionevole pensare, si vuole

affermare che una persona può essere proprietaria sia di beni mobili, sia di beni immobili, occorrerà scrivere:

propDi: Persona $\longrightarrow$ (BeneMobile $\sqcup$ BeneImmobile)

5.3 La proprietà inversa

Data una proprietà R, che rappresenta una relazione binaria $R(x,y)$, è spesso utile esprimere la *proprietà inversa* R^-, che rappresenta la relazione $R(y,x)$. Ad esempio, per esprimere la relazione "figlio di" partendo dalla proprietà genDi utilizzeremo la proprietà inversa genDi$^-$. La semantica della proprietà inversa si specifica come segue:

$(R^-)^I = (R^I)^{-1} = \{\langle y,x\rangle \in \Delta^2 \mid \langle x,y\rangle \in R^I\}$

Per quanto riguarda il dominio e il codominio, nella proprietà inversa essi si ritrovano scambiati; in altre parole, se R: $C \longrightarrow D$ avremo di conseguenza R^-: $D \longrightarrow C$.

L'inversione di una proprietà è un'operazione involutoria, nel senso che l'inversa R^{--} dell'inversa di R coincide con R. Proseguendo, R^{---} coincide con R^- e così via. Per evitare scritture inutilmente complesse stabiliamo una volta per tutte che una proprietà può essere soltanto *diretta* (come propDi) o inversa (come propDi$^-$); al posto di R^{--} scriveremo quindi R, e così via. Si noti infine che è possibile specificare che due proprietà R e S sono l'una l'inversa dell'altra:

Inv(R,S)

Tale assioma equivale all'assioma di RBox (vedi il par. 5.5):

$R \equiv S^-$

5.4 Proprietà funzionali e funzioni

Proprietà funzionali

Com'è noto, una relazione binaria è *funzionale* se ogni oggetto del dominio è in relazione con *al più un oggetto* del codominio. Nelle DL, una proprietà che rappresenti una relazione funzionale è detta *proprietà funzionale*. Consideriamo ad esempio la proprietà "coniuge di", i cui dominio e codominio si possono definire come segue:

coniugeDi: Persona $\longrightarrow$ Persona

Nell'ordinamento legale italiano la proprietà coniugeDi è funzionale perché ciascuna persona può avere al massimo un coniuge (uno per volta, s'intende). Per esprimere questo fatto è sufficiente affermare che ciascun oggetto dell'universo è coniuge al massimo di un oggetto:

(5.5) $\top \sqsubseteq \leq 1$ coniugeDi

In alternativa è possibile specificare che la proprietà coniugeDi è funzionale:

(5.6) *Fun*(coniugeDi)

L'assioma 5.6 è del tutto equivalente all'assioma 5.5 e va quindi considerato come parte della TBox.

In certi casi risulta essere funzionale l'inversa di una proprietà. Consideriamo nuovamente la proprietà propDi del paragrafo 5.3; se ammettiamo che un bene sia o privo di proprietario o posseduto da un unico proprietario, la proprietà propDi risulta *inversamente funzionale*, ovvero

$\top \sqsubseteq \leq 1$ propDi$^-$

In alternativa, e con effetti equivalenti, è possibile specificare direttamente che la proprietà propDi è inversamente funzionale:

InvFun(propDi)

Funzioni

Una relazione binaria è una *funzione* se è funzionale e se ogni oggetto del dominio è in relazione con *almeno un oggetto* del codominio; ovvero, se la relazione è *funzionale* nonché *totale sul dominio*. Ne

segue che ogni funzione mette in relazione ciascun oggetto del dominio (l'*argomento* della funzione) con *esattamente un oggetto* del codominio (il *valore* della funzione corrispondente all'argomento). Nelle DL, una proprietà che sia una funzione è detta *proprietà funzione* o semplicemente *funzione*.

Ad esempio, è una funzione la proprietà

$\mathsf{natoA}: \mathsf{Persona} \longrightarrow \mathsf{Comune}$

che associa a ogni persona il suo comune di nascita. Per specificare che questa proprietà è una funzione possiamo specificare che natoA è funzionale,

(5.7) $\mathit{Fun}(\mathsf{natoA})$

e poi imporre che la proprietà natoA sia totale sul suo dominio, ovvero che ogni oggetto del dominio di natoA sia nato in almeno un luogo:

(5.8) $\mathsf{Persona} \sqsubseteq \exists \mathsf{natoA}$

In alternativa possiamo rimpiazzare 5.7 e 5.8 con l'unico assioma di TBox

$\mathsf{Persona} \sqsubseteq {=}1\ \mathsf{natoA}$

Quest'ultima soluzione non può però essere adottata in certe DL espressivamente deboli (come la logica $\mathcal{SHIF}$, che sta alla base del linguaggio OWL 1 Lite) in cui l'unica restrizione di cardinalità ammessa è $\leq 1R$.

Una proprietà *inversa di funzione* è invece la proprietà

$\mathsf{madreDi}: \mathsf{Donna} \longrightarrow \mathsf{Persona}$

che associa a ogni donna i suoi eventuali figli. Dato che ogni persona ha esattamente una madre[13] avremo

$\mathit{InvFun}(\mathsf{madreDi})$

$\mathsf{Persona} \sqsubseteq \exists \mathsf{madreDi}^{-}$

o in alternativa

$\mathsf{Persona} \sqsubseteq {=}1\ \mathsf{madreDi}^{-}$

Suriezioni, iniezioni e biiezioni

Come è noto, una *suriezione* è una funzione i cui valori esauriscono completamente il codominio; un'iniezione è una funzione che a due argomenti diversi associa sempre valori diversi; e una biiezione è una corrispondenza biunivoca (quindi suriettiva e iniettiva). Da quanto detto in precedenza è chiaro che queste caratteristiche di una funzione sono tutte specificabili in OWL. Più precisamente, se

$R: C \longrightarrow D$

$\mathit{Fun}(R)$

$C \sqsubseteq \exists R$

è una funzione, allora per specificare che R è suriettiva aggiungeremo:

$D \sqsubseteq \exists R^{-}$

Per specificare che R è iniettiva aggiungeremo invece:

$\mathit{InvFun}(R)$

Infine, aggiungeremo entrambe le condizioni per specificare che R è biiettiva.

[13] Affermare che ogni persona ha esattamente una madre ha conseguenze tutt'altro che banali. Se ammettiamo che una catena di relazioni $\mathsf{madreDi}^{-}$ non ammetta cicli (ovvero, che nessuna donna sia antenata di se stessa), l'universo deve contenere infinite persone; d'altra parte, il numero di persone che sono esistite o esistono fino a un particolare istante è finito. Il lettore è invitato a trovare una soluzione a questo apparente paradosso, basandosi sulla teoria dell'evoluzione o su credenze religiose di sua scelta.

5.5 Sottoproprietà e proprietà disgiunte

In molte DL è consentito esprimere relazioni di sottoproprietà o di equivalenza fra proprietà con enunciati della forma[14]:

$R \sqsubseteq S$

$R \equiv S$

Ad esempio, è possibile dire che la proprietà "genitore di" *è una sottoproprietà* della (o *è inclusa* nella) proprietà "antenato di",

$\mathsf{genDi} \sqsubseteq \mathsf{antDi}$

o che "figlio di" *è equivalente* all'inverso di "genitore di",

$\mathsf{figlioDi} \equiv \mathsf{genDi}^-$

Le condizioni di verità di questi tipi di enunciati sono:

$M \vDash R \sqsubseteq S$ sse $R^I \subseteq S^I$

$M \vDash R \equiv S$ sse $R^I = S^I$

Si noti che, contrariamente alla maggior parte dei casi discussi nei sottoparagrafi precedenti, le inclusioni e le equivalenze di proprietà appartengono alla RBox, in quanto non equivalgono ad alcun enunciato concernente classi.

Un'altra opzione utile è la possibilità di specificare che due proprietà sono disgiunte. Ad esempio diremo nel modo seguente che le proprietà "madre di" e "padre di" sono due sottoproprietà mutuamente esclusive di "genitore di":

$\mathsf{madreDi} \sqsubseteq \mathsf{genDi}$

$\mathsf{padreDi} \sqsubseteq \mathsf{genDi}$

$\mathit{DisPro}(\mathsf{madreDi},\mathsf{padreDi})$

L'assioma di RBox $\mathit{DisPro}(R_1,...,R_n)$ è analogo all'assioma di disgiunzione fra classi (par. 4.5). C'è però una differenza: nel caso delle classi l'assioma $\mathit{DisCla}(C_1,...,C_n)$ è solo una comoda abbreviazione, perché la disgiunzione può comunque essere imposta specificando che l'intersezione di ciascuna coppia di classi è sottoclasse della classe vuota; nel caso delle proprietà, invece, gli assiomi della forma $\mathit{DisPro}(R_1,...,R_n)$ non sono eliminabili: la disgiunzione non può essere imposta mediante assiomi di sottoproprietà, perché la logica $\mathcal{SROIQ}(\mathrm{D}_n)$ non possiede un costruttore d'intersezione per le proprietà.

5.6 Catene di proprietà

In alcune DL è possibile costruire proprietà complesse utilizzando *catene di proprietà*. Date due proprietà R e S, catena di proprietà $R \circ S$ denota la relazione binaria composta:

$(R \circ S)^I = R^I \circ S^I = \{\langle x,y\rangle \in \Delta \mid \text{per qualche } z \in \Delta, \langle x,z\rangle \in R^I \text{ e } \langle z,y\rangle \in S^I\}$

Le catene di proprietà sono molto utili nelle applicazioni, ma il loro uso deve essere sottoposto a particolari vincoli per evitare che la logica diventi indecidibile. Nella logica $\mathcal{SROIQ}(\mathrm{D}_n)$ sono consentiti gli assiomi di inclusione di proprietà contenenti catene nelle forme seguenti:

$R \circ R \sqsubseteq R$

$S_1 \circ ... \circ S_n \sqsubseteq R$

$S_1 \circ ... \circ S_n \circ R \sqsubseteq R$

$R \circ S_1 \circ ... \circ S_n \sqsubseteq R$

[14] In alcuni testi questi enunciati sono rappresentati con la notazione $R \dot{\sqsubseteq} S$ e $R \dot{\equiv} S$, per distinguerli dagli analoghi enunciati concernenti le classi; non seguiremo questa convenzione perché il contesto d'uso dei simboli $\sqsubseteq$ e $\equiv$ è sufficiente per risolvere l'ambiguità.

con il vincolo aggiuntivo che fra le proprietà S_i e la proprietà R non sussistano *dipendenze cicliche*, nel senso sarà chiarito nel paragrafo 5.8 (vedi Horrocks *et al*., 2006). Per esemplificare l'uso di questo tipo di assiomi di sottoproprietà, supponiamo di voler affermare che un certo numero di aziende possono essere federate fra loro, e che un'azienda che sia fornitrice di un'altra azienda è fornitrice anche delle aziende federate con quest'ultima. Definiamo innanzitutto due proprietà,

federataCon: Azienda $\longrightarrow$ Azienda

fornitriceDi: Azienda $\longrightarrow$ Azienda

La proprietà federataCon sarà simmetrica e transitiva (vedi il par. 5.8). Per specificare poi che un'azienda che sia fornitrice di un'altra azienda è fornitrice anche delle aziende federate di quest'ultima utilizziamo l'assioma

fornitriceDi $\circ$ federataCon $\sqsubseteq$ fornitriceDi

Va infine osservato che anche in assenza di un costruttore di concatenazione di proprietà è comunque possibile definire classi complesse mediante *concatenazioni ristrette*, in quanto:

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

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

5.7 Qualità formali delle proprietà

Spesso è utile specificare che una proprietà ha una determinata *qualità formale*[15], ad esempio che è riflessiva, simmetrica, transitiva e così via. Alcune di queste qualità formali possono essere specificate utilizzando gli strumenti già acquisiti; ad esempio, per specificare che una proprietà R è *simmetrica* è sufficiente l'assioma di RBox

$R \sqsubseteq R^{-}$

Per specificare che R è *totale sul dominio* è sufficiente l'assioma di TBox

$C \sqsubseteq \exists R$

Per specificare che R è *transitiva* è sufficiente l'assioma di RBox

$R \circ R \sqsubseteq R$

Si noti che alcune DL, tra cui la logica $\mathcal{SHOIN}(\mathrm{D}_n)$ su cui si basa il linguaggio OWL 1 DL, pur non consentendo la concatenazione di proprietà prevedono comunque la possibilità di specificare direttamente che una proprietà R è transitiva:

$Tra(R)$ R è transitiva

Con gli strumenti introdotti finora non è possibile specificare la riflessività, l'irriflessività o l'asimmetria di una proprietà. La logica $\mathcal{SROIQ}(\mathrm{D}_n)$ prevede comunque la possibilità di specificare direttamente che una proprietà R è riflessiva, irriflessiva o asimmetrica con opportuni assiomi di RBox:

$Ref(R)$ R è riflessiva

$Irr(R)$ R è irriflessiva

$Asy(R)$ R è asimmerica

I concetti di riflessività e di asimmetria richiedono alcune precisazioni.

Riflessività globale e riflessività locale

L'assioma $Rif(R)$ specifica che la proprietà R è *globalmente riflessiva*, ovvero che R^I include la *relazione d'identità* o *relazione diagonale* $\{\langle x,x\rangle \mid \text{per ogni } x \in \Delta\}$. Ciò implica che sia il dominio, sia il codominio di R coincidano necessariamente con $\top$:

R: $\top \longrightarrow \top$

[15] Nella terminologia matematica ordinaria ciò che qui chiamiamo "qualità formale" è denominata "proprietà formale." Preferiamo il termine "qualità" per evitare l'espressione "proprietà formale di una proprietà".

Nelle applicazioni capita molto di rado che una proprietà interessante abbia dominio e codominio coincidente con l'intero universo; per questo motivo la riflessività globale è una qualità di limitato interesse applicativo. In genere ciò che si può voler specificare è un'altra qualità, detta *riflessività locale*. Una proprietà R si dice *localmente riflessiva* se ogni oggetto del dominio C di R è in relazione R con se stesso, ovvero se vale

$\{\langle x,x\rangle \mid \text{per ogni } x \in C^I\} \subseteq R^I$

Per consentire la specifica della riflessività locale è stata introdotta nella logica $\mathcal{SROIQ}(\mathrm{D}_n)$ la *restrizione di riflessività locale*,

$\exists R.\mathsf{Self}$

la cui interpretazione è data da

$(\exists R.\mathsf{Self})^I = \{x \in \Delta \mid \langle x,x\rangle \in R^I\}$

La classe complessa $\exists R.\mathsf{Self}$ denota quindi l'insieme di tutti gli oggetti dell'universo che sono in relazione R con se stessi. Se C è il dominio di R, la riflessività locale si può ora imporre a R con l'assioma di TBox

$C \sqsubseteq \exists R.\mathsf{Self}$

Ad esempio, possiamo specificare nel modo seguente che la proprietà ama è localmente riflessiva, in quanto ogni persona ama se stessa (oltre eventualmente ad altre persone):

$\mathsf{ama}: \mathsf{Persona} \longrightarrow \mathsf{Persona}$

$\mathsf{Persona} \sqsubseteq \exists \mathsf{ama}.\mathsf{Self}$

Aggiungendo altri assiomi di TBox è poi facile definire i narcisisti, intesi come quelle persone che amano *soltanto* se stesse:

(5.9) $\mathsf{Narcisista} \sqsubseteq \mathsf{Persona}$

(5.10) $\mathsf{Narcisista} \equiv \leq 1\ \mathsf{ama}$

Si noti che l'assioma 5.9 è necessario, perché utilizzando solo 5.10 ogni oggetto dell'universo che non sia una persona (e quindi non appartenga al dominio di ama) sarebbe automaticamente considerato narcisista! In alternativa a 5.9 e 5.10 è possibile utilizzare il solo assioma

$\mathsf{Narcisista} \equiv =1\ \mathsf{ama}$

Si noti che il simbolo Self, che dal punto di vista sintattico si comporta in modo analogo a una classe atomica, può essere utilizzato soltanto nel contesto della restrizione $\exists R.\mathsf{Self}$, dove R è una proprietà diretta o inversa. Non è quindi ammesso l'uso di Self in una restrizione universale del tipo $\forall R.\mathsf{Self}$, né in restrizioni di cardinalità, mentre è ammessa la classe $\neg\exists R.\mathsf{Self}$.

Sfruttando la riflessività è possibile definire la funzione identità. L'*identità globale* è infatti l'unica proprietà che sia contemporaneamente funzionale e globalmente riflessiva:

(5.11) $\mathsf{id}: \top \longrightarrow \top$

$\mathit{Fun}(\mathsf{id})$

$\mathit{Ref}(\mathsf{id})$

Più in generale si può definire l'*identità locale* su una classe A come l'unica proprietà da A ad A che sia funzionale e localmente riflessiva:

$\mathsf{id}A: A \longrightarrow A$

$\mathit{Fun}(\mathsf{id}A)$

$A \sqsubseteq \exists \mathsf{id}A.\mathsf{Self}$

Tuttavia, quando interessa definire una classe A insieme alla sua identità locale $\mathsf{id}A$ è preferibile procedere in un altro modo. Innanzi tutto notiamo che:

– se è vero che un insieme A determina univocamente la sua identità locale, in quanto l'identità locale sull'insieme A non è altro che l'insieme di coppie ordinate $\{\langle x,x\rangle \mid x \in A\}$),

– è anche vero che da un insieme X di coppie ordinate della forma $\langle x,x\rangle$ possiamo definire l'insieme A di cui X costituisce l'identità locale: si tratta dell'insieme $\{x \mid \langle x,x\rangle \in X\}$.

In altre parole: se partiamo da una relazione binaria che può essere formalmente considerata come un'identità locale (ovvero, una relazione binaria in cui tutte le coppie ordinate hanno la forma $\langle x,x\rangle$), possiamo facilmente ricavare l'insieme di cui la relazione binaria è l'identità locale. Supponiamo ora di voler introdurre in un'ontologia una classe A insieme alla sua identità locale $\mathsf{id}A$; procediamo come segue:

– specifichiamo che $\mathsf{id}A$ è una possibile identità locale; a questo scopo è sufficiente specificare

$$\mathsf{id}A \sqsubseteq \mathsf{id}$$

dove id è specificata da 5.11;

– definiamo A come la classe degli oggetti dell'universo che sono messi in relazione con qualche oggetto dalla proprietà $\mathsf{id}A$:

$$A \equiv \exists \mathsf{id}A$$

Vedremo in seguito (par. 8.4) un uso interessante dell'identità.

Asimmetria e antisimmetria

Una relazione binaria è *asimmetrica* se l'appartenenza alla relazione della coppia $\langle x,y\rangle$ esclude l'appartenenza alla stessa relazione della coppia $\langle y,x\rangle$. Questa qualità non va confusa con l'*antisimmetria*: una relazione binaria è infatti *antisimmetrica* se l'appartenenza alla relazione della coppia $\langle x,y\rangle$ assieme alla coppia $\langle y,x\rangle$ comporta che $x = y$.

L'asimmetria e l'antisimmetria sono qualità distinte[16]: ad esempio, la relazione $<$ fra numeri è asimmetrica, mentre la relazione $\leq$ è antisimmetrica. Come abbiamo già visto, in $\mathcal{SROIQ}(\mathrm{D}_n)$ l'asimmetria non può essere definita con assiomi di sottoclasse o di sottoproprietà, ma può essere direttamente specificata nella RBox; l'antisimmetria, invece, non può essere specificata in $\mathcal{SROIQ}(\mathrm{D}_n)$.

5.8 Vincoli non strutturali

La sintassi di $\mathcal{SROIQ}(\mathrm{D}_n)$ può essere vista come un insieme di vincoli sulla struttura delle espressioni ammesse in un'ontologia. Tuttavia, non ogni insieme di espressioni strutturalmente corrette costituisce un'ontologia valida: esistono infatti vincoli ulteriori, detti *vincoli non strutturali*, che vanno rispettati. Si noti che tali vincoli, presi nel loro insieme, esprimono una *condizione sufficiente* affinché il linguaggio sia decidibile; non si tratta però di *condizioni necessarie*, per cui il ragionamento potrebbe non dare problemi anche nel caso che un'ontologia violi qualche vincolo non strutturale. Il motivo per cui si preferisce accontentarsi di vincoli che sono sufficienti ma non necessari (e sono quindi *più restrittivi* di quanto sarebbe strettamente richiesto) è che le *condizioni necessarie e sufficienti* per la decidibilità di $\mathcal{SROIQ}(\mathrm{D}_n)$ sono molto difficili da formulare.

Abbiamo già menzionato un tipo di vincoli non strutturali nel paragrafo 5.5, dove abbiamo precisato che un insieme di assiomi di sottoproprietà non deve introdurre dipendenze cicliche fra proprietà. Intuitivamente l'idea è la seguente: un assioma della forma $S \sqsubseteq R$ fa sì che R *dipenda da* S; se poi S a sua volta dipende da S', R dipende (seppure indirettamente) anche da S'. Ciò che va evitato è che una proprietà, grazie a un insieme di assiomi, finisca per dipendere da se stessa, in modo diretto o indiretto. Più precisamente, diciamo che una proprietà *dipende direttamente* da un'altra proprietà se, e solo se, si verifica almeno uno dei casi seguenti:

– se la RBox contiene un assioma della forma $S \sqsubseteq R$, allora R dipende direttamente da S;

– se la RBox contiene un assioma della forma $S \equiv R$, allora R dipende direttamente da S, ed S dipende direttamente da R;

– se la RBox contiene un assioma della forma $S_1 \circ \ldots \circ S_n \sqsubseteq R$ (con $n > 1$), allora R dipende direttamente da S_i, per $1 \leq i \leq n$;

[16] Nei testi sul web semantico l'asimmetria è a volte impropriamente chiamata antisimmetria.

- se la RBox contiene un assioma della forma $InvPro(S,R)$, allora R dipende direttamente da S^-, ed S^- dipende direttamente da R;
- se la RBox contiene un assioma della forma $Sym(R)$, allora R dipende direttamente da R^-;
- se R dipende direttamente da S, allora R^- dipende direttamente da S^-.

A questo punto prendiamo la chiusura riflessiva e transitiva della relazione di dipendenza diretta e diciamo che R *dipende* da S se, e solo se, R è in relazione con S grazie a tale chiusura. In altre parole, R dipende da S se R coincide con S o se c'è una sequenza finita di dipendenze dirette che porti da R a S.

Una RBox si dice *regolare* se, e solo se, negli assiomi di sottoproprietà della forma

$S_1 \circ ... \circ S_n \sqsubseteq R$

$S_1 \circ ... \circ S_n \circ R \sqsubseteq R$

$R \circ S_1 \circ ... \circ S_n \sqsubseteq R$

(con $n \geq 1$) non si verifica mai che uno degli S_i dipenda da R.

Il primo vincolo non strutturale si enuncia ora così: *ogni RBox deve essere regolare*. Va notato che, in linea di principio, ogni assioma di equivalenza di proprietà (della forma $S \equiv R$) rende una RBox irregolare, in quanto equivale ai due assiomi $S \sqsubseteq R$ and $R \sqsubseteq S$, che inducono una dipendenza ciclica di S (ed R) da se stessa. In pratica, tuttavia, le irregolarità di questo genere si possono rimuovere facilmente, rimpiazzando nell'intera ontologia ogni occorrenza di S con un'occorrenza di R. Ciò comporta che gli assiomi di equivalenza di proprietà non rendano irregolare un'RBox.

Esiste un ulteriore tipo di vincoli non strutturali. Prima di introdurlo dobbiamo spiegare che cosa si intenda per *proprietà semplice* e per *proprietà composita*. Intuitivamente, una proprietà è *composita* se contiene una catena di proprietà; più precisamente, una proprietà è composita se, e solo se, si verifica almeno uno dei casi seguenti:

- la RBox contiene un assioma della forma $Tra(R)$;
- la RBox contiene un assioma della forma $S_1 \circ ... \circ S_n \sqsubseteq R$ (dove o S_1 o S_n possono eventualmente coincidere con R), con $n > 1$;
- la RBox contiene un assioma della forma $S \sqsubseteq R$, dove S è a sua volta composita;
- R^- è composita.

Una proprietà si dice *semplice* se non è composita. A questo punto possiamo formulare i seguenti vincoli non strutturali: la proprietà R deve essere semplice (quindi non composita)

- nelle restrizioni di cardinalità semplici ($\leq nR$, $\geq nR$, $=nR$) o qualificate ($\leq nR.C$, $\geq nR.C$, $=nR.C$) e nelle restrizioni di riflessività locale ($\exists R.\mathsf{Self}$);
- negli assiomi di disgiunzione di proprietà ($DisPro(R_1,...,R_n)$), di funzionalità diretta o inversa ($Fun(R)$, $InvFun(R)$), di irriflessività ($Irr(R)$) e di asimmetria ($Asy(R)$).

5.9 Dati e attributi

Le proprietà di cui ci siamo occupati finora sono relazioni binarie *fra oggetti dell'universo*; per questo motivo una proprietà del genere è spesso denominata *object property*. In molti casi è utile definire relazioni anche *fra oggetti dell'universo e costanti* (o *dati*), come numeri o stringhe. Una proprietà di questo genere è denominata *data property*. In queste note useremo il termine "attributo" per le data property e riserveremo il termine "proprietà" per le object property; per indicare generici attributi (data property) useremo poi le lettere P e Q.

Nel linguaggio OWL si possono definire attributi appartenenti a (quasi) tutti i tipi di dati predefiniti di RDF, che a loro volta sono mutuati da XML Schema; tali tipi sono denominati *domini di dati* o *domini concreti*. In queste note utilizzeremo soltanto i domini di dati integer, nonNegativeInteger, positiveInteger, float e string. Si noti che gli elementi dei domini di dati *non* sono considerati oggetti dell'universo.

Come esempio consideriamo l'attributo haEtà, che associa alle persone un intero non negativo (l'età espressa in anni):

haEtà: Persona $\longrightarrow_D$ nonNegativeInteger

dove il simbolo $\longrightarrow_D$ indica che stiamo specificando dominio e codominio di un attributo (*d*ata property). Gli attributi, al contrario delle proprietà, non possono essere invertiti. Infatti gli attributi rappresentano sempre relazioni *da classi a insiemi di dati*; l'inverso di un attributo rappresenterebbe quindi una relazione *da insiemi di dati a classi*, e ciò non è consentito. Per un motivo analogo gli attributi non possono essere concatenati.

Nella logica $\mathcal{SROIQ}(\mathrm{D}_n)$ è possibile:

- definire relazioni di sottoattributo ed equivalenze di attributi;
- specificare che due attributi sono disgiunti;
- utilizzare restrizioni esistenziali, universali, di cardinalità e di valore relativi ad attributi;
- specificare che un attributo è funzionale.

Ad esempio, per specificare che l'attributo haEtà è una funzione possiamo utilizzare i due assiomi:

Persona $\sqsubseteq \exists$haEtà

Fun(Persona)

Per trattare i domini di dati dobbiamo estendere leggermente il concetto di modello. D'ora in avanti concepiremo un modello come una terna ordinata $M = \langle \Delta, \{\Delta_n\}, \cdot^I \rangle$, dove Δ è il solito universo non vuoto di oggetti, $\{\Delta_n\}$ è una famiglia di domini di dati, e la funzione d'interpretazione $\cdot^I$, oltre ad associare un insieme di oggetti a ogni classe atomica e una relazione binaria fra oggetti a ogni proprietà, associa una relazione binaria fra oggetti e dati a ogni attributo. Ad esempio, per l'attributo haEtà avremo:

$\mathsf{haEtà}^I \subseteq \mathsf{Persona} \times \Delta_{\mathsf{nonNegativeInteger}}$

6. Gli individui

6.1 Individui e nominali

Nelle DL è possibile fare riferimento a oggetti specifici dell'universo utilizzando simboli a, b e così via, detti *individui* e analoghi alle costanti individuali di FOL. Dal punto di vista semantico, ogni individuo *denota* (ovvero *fa riferimento a*) uno specifico oggetto dell'universo, detto *referente* dell'individuo. In un modello, l'associazione fra un individuo e il suo referente è stabilito dalla funzione d'interpretazione; pertanto, se a è un generico individuo avremo

$a^I \in \Delta$

A ogni individuo corrisponde uno e un solo oggetto dell'universo. Non vale invece l'inverso: infatti è possibile che a un oggetto dell'universo (fig. 6.1):

- non corrisponda alcun individuo;
- corrisponda esattamente un individuo, ad esempio a;
- corrispondano più individui, ad esempio b, c e così via.

Nel mondo delle KB un oggetto denotato da un individuo è considerato un oggetto "noto", mentre un oggetto che non è denotato da nessun individuo è considerato "ignoto". Come vedremo più avanti, la distinzione fra oggetti noti e oggetti ignoti dell'universo è spesso importante.

A questo punto è possibile specificare quale sia il 'modello minimo' $M_0 = \langle \Delta_0, \cdot^{I_0} \rangle$ di $\mathcal{SROIQ}(\mathrm{D}_n)$. Ricordiamo che, per definizione, ogni universo contiene almeno un oggetto, che possiamo indicare con

$\mathbf{0} \in \Delta_0$

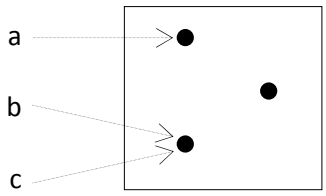

Figura 6.1. Oggetti denotati da un individuo, da due individui, da nessun individuo.

Per le definizioni della classe top, $\top$, della classe bottom, $\bot$, della proprietà top, $\dot{\top}$, e della proprietà bottom, $\dot{\bot}$, avremo allora:

$\top^{\mathcal{I}_0} = \{\textbf{0}\}$ $\quad\quad$ $\bot^{\mathcal{I}_0} = \{\ \}$

$\dot{\top}^{\mathcal{I}_0} = \{\textbf{0},\textbf{0}\}$ $\quad\quad$ $\dot{\bot}^{\mathcal{I}_0} = \{\ \}$

Il modello M_0 è rappresentato graficamente nella figura 6.2.

I nominali

Dato un individuo qualsiasi a, è possibile definire la classe che contiene esattamente a. Una classe del genere è detta *nominale* ed è rappresentata dall'espressione

$\{a\}$

Le *classi enumerative* (anch'esse spesso chiamate semplicemente "nominali") sono poi l'unione di più nominali: per definizione, infatti, la classe enumerativa $\{a_1, ..., a_n\}$ coincide con la classe

$\{a_1\} \sqcup \ ... \sqcup \{a_n\}$

Ad esempio:

(6.1) $\mathsf{ColoreRGB} \equiv \{\mathsf{red}, \mathsf{green}, \mathsf{blue}\}$

L'interpretazione di un nominale è la seguente:

$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$

$\{a_1, ..., a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, ..., a_n^{\mathcal{I}}\}$.

Il costruttore di classi {...} è spesso chiamato *one-of*, perché ciascuno dei suoi elementi è uno degli individui elencati. Come ogni altra classe, un nominale può essere utilizzato per qualificare una restrizione di proprietà; in particolare la restrizione $\exists R.\{a\}$ denota l'insieme degli oggetti dell'universo che sono in relazione R con l'oggetto denotato da a. Per rappresentare questa restrizione, detta *restrizione di valore*, è spesso utilizzata la notazione alternativa, ma equivalente,

$R \ni a$

che rappresenta l'insieme degli oggetti dell'universo che sono in relazione R con il referente di a:

$(R \ni a)^{\mathcal{I}} = \{x \in \Delta \mid \langle x, a^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\}$

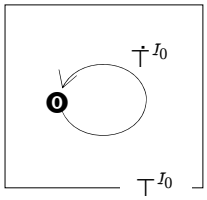

Figura 6.2. Il modello minimale M_0 di $\mathcal{SROIQ}(\mathrm{D}_n)$.

6.2 Unicità dei nomi ed altre assunzioni

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* (UNA, *unique name assumption*): ciò significa che due individui distinti non fanno mai riferimento allo stesso oggetto dell'universo. In termini logici, se si utilizzano n individui a_1, ..., a_n l'assunzione di unicità del nome equivale alle $n\cdot(n-1)/2$ asserzioni

$$\begin{array}{llll} a_1 \neq a_2 & a_1 \neq a_3 & \ldots & a_1 \neq a_n \\ & a_2 \neq a_3 & \ldots & a_2 \neq a_n \\ & & & \ldots \\ & & & a_{n-1} \neq a_n \end{array}$$

o, più concisamente,

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

La UNA è considerata irrealistica nell'ambito del web semantico, in quanto a una risorsa disponibile nel web (vista come oggetto dell'universo) possono corrispondere più URI (viste come individui). Nel seguito, quindi, *non assumeremo l'unicità dei nomi*[17]: quando occorra, la diversità degli individui andrà specificata esplicitamente con asserzioni della forma 6.2.

Altre due assunzioni analoghe, ancora una volta tipiche delle basi di dati ma non della logica, sono l'*assunzione di chiusura dell'universo* (UCA, *universe closure assumption*) e l'*assunzione del mondo chiuso* (CWA, *closed world assumption*). L'assunzione di chiusura dell'universo consiste nell'ipotesi che l'universo contenga soltanto gli oggetti noti, ovvero gli oggetti cui si fa riferimento con un individuo presente nel sistema. Questa assunzione *non viene adottata* nelle DL. Dell'assunzione del mondo chiuso, anch'essa estranea alle DL, parleremo nel prossimo paragrafo.

6.3 Le asserzioni

Come si è detto nel paragrafo 2, un sistema di rappresentazione delle conoscenze consta di una TBox, una RBox e un'ABox. La TBox e la RBox contengono assiomi che definiscono caratteristiche logiche generali delle classi e delle proprietà utilizzate per modellare il mondo dell'applicazione; l'ABox contiene invece conoscenze fattuali espresse sotto forma di *asserzioni* che concernono specifici individui.

Nella logica $\mathcal{SROIQ}(\mathrm{D}_n)$ si possono esprimere diversi tipi di conoscenze fattuali. Se C è una classe arbitraria (quindi atomica o complessa) e a è un individuo, si può asserire che a appartiene alla classe C utilizzando la notazione

$C(a)$

Se poi R è una proprietà e a, b sono individui (non necessariamente distinti), si può asserire fra a e b sussiste oppore non sussiste la proprietà R:

$R(a,b)$

$\neg R(a,b)$

Esempi:

Madre(anna)

Donna ⊓ ∃genDi(anna) (dove C è la classe complessa Donna ⊓ ∃genDi)

genDi(anna,bruno)

¬genDi(anna,cecilia)

[17] Gli strumenti di ragionamento più diffusi consentono comunque all'utente di adottare la UNA quando ciò sia opportuno.

La asserzioni sono un tipo di enunciati, nel senso che possono essere vere o false in un modello. Le relative condizioni di verità sono le seguenti:

$M \vDash C(a)$ sse $a^I \in C^I$

$M \vDash R(a,b)$ sse $\langle a^I, b^I \rangle \in R^I$

$M \vDash \neg R(a,b)$ sse $\langle a^I, b^I \rangle \notin R^I$

Infine l'ABox può contenere asserzioni del tipo

$a = b$

$a \neq b$

ed eventualmente abbreviazioni del tipo $=(a_1, ..., a_n)$ o $\neq(a_1, ..., a_n)$. Anche in questi casi, come nel precedente, le condizioni di verità sono ovvie:

$M \vDash a = b$ sse $a^I = b^I$

$M \vDash a \neq b$ sse $a^I \neq b^I$

Va sottolineato che le asserzioni di appartenenza di un individuo a una classe si possono limitare alle sole classi atomiche, ovvero alla forma $A(a)$, senza perdere generalità pur di aggiungere alla TBox opportune definizioni di classe del tipo

$A \equiv C$

Ad esempio, invece di asserire nell'ABox

Persona ⊓ Maschio(bruno)

possiamo definire nella TBox

Uomo ≡ Persona ⊓ Maschio

e riscrivere l'asserzione come

Uomo(bruno)

Infine va notata una differenza fra i concetti di dominio e codominio nelle DL e gli analoghi concetti di dominio e di codominio di una relazione binaria o di una funzione, così come sono tradizionalmente definiti in altri settori della matematica. Nella teoria degli insiemi, ad esempio, se R è una relazione con dominio C e codominio D, una formula del tipo $R(a,b)$ con $a \notin C$ o $b \notin D$ sarebbe considerata scorretta e priva di senso, in quanto gli argomenti sono di tipo inappropriato; nelle DL, invece, una formula del genere *è corretta, ma è falsa*. In altre parole, l'appartenenza dei due argomenti rispettivamente al dominio e al codominio di una relazione sono condizioni necessarie (ancorché non sufficienti) perché la formula $R(a,b)$ sia vera; se tali condizioni non sono verificate, la formula è semplicemente falsa. Ad esempio, in presenza degli assiomi

DisCla(Donna,Uomo)

Dom(madreDi,Donna)

Uomo(antonio)

Uomo(bruno)

l'espressione

(6.1) madreDi(antonio,bruno)

non è scorretta, ma risulta falsa in ogni modello che soddisfi gli assiomi. Attenzione, però: asserire esplicitamente 6.1 in presenza degli assiomi precedenti introdurrebbe una contraddizione nell'ontologia, rendendola inconsistente (per il concetto di consistenza si veda il par. 7.2).

L'assunzione del mondo chiuso

Abbiamo ora occasione di discutere l'assunzione del mondo chiuso (CWA, già menzionata nel par. 6.2), tipica delle basi di dati e di molti sistemi d'intelligenza artificiale. Tradotta nei nostri termini, la CWA suona 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 la CWA 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 di incertezza (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, su altre ancora si è incerti. Supponiamo ad esempio che nell'universo del discorso siano presenti tre persone, di nome Anna, Bruno e Tsukiko. Potremmo sapere che Anna è una donna e che Bruno è un uomo, ma non sapere se Tsukiko sia una donna un uomo. In tal caso possiamo asserire nell'ABox

Donna(anna)
Uomo(bruno)

e non dire nulla su Tsukiko, salvo magari

Persona(tsukiko)

L'operatore di chiusura **K**

In molte applicazioni si possiede effettivamente conoscenza completa su qualche aspetto dell'universo: ad esempio, si potrebbe essere sicuri che Tsukiko è una donna, e che le uniche donne presenti nell'universo sono proprio Anna e Tsukiko. Nella letteratura specializzata è stato proposto l'uso di uno speciale operatore, solitamente denominato **K** (da *know*) per denotare la totalità degli individui che sono noti possedere una determinata qualità. Supponiamo ad esempio di sapere che sia Anna, sia Tsukiko siano ingegneri:

Donna(anna) Donna(tsukiko)
Ing(anna) Ing(tsukiko)

In base a queste asserzioni non è ovviamente possibile dimostrare che tutte le donne sono ingegneri, perché nulla ci garantisce che Anna e Tsukiko siano tutte le donne presenti nell'universo. Tuttavia, se nell'ABox non ci sono altre asserzioni del tipo Donna(*a*), sappiamo almeno che *tutte le donne note* sono ingegneri. Utilizzando l'operatore **K** la classe delle *donne note* si esprime come

K Donna

Uno strumento di ragionamento in grado di trattare correttamente l'operatore **K** potrà quindi dedurre che vale la relazione:

K Donna $\sqsubseteq$ Ing

L'uso dell'operatore **K** pone comunque un certo numero di problemi logici, che non abbiamo modo di approfondire in questa sede (per un approfondimento si veda Donini et al., 1998).

6.4 I dati

Come abbiamo visto nel paragrafo 6.9, gli attributi (ovvero le data property) connettono oggetti dell'universo con dati. Dobbiamo ora precisare come si possano rappresentare i valori degli attributi, ad esempio per asserire che l'età di Anna, espressa in anni, è l'intero non negativo 25.

In OWL i valori dei dati sono rappresentati come in RDF, che a sua volta eredita la rappresentazione di XML Schema. Ad esempio, l'intero non negativo 25 è rappresentato come:

"25"^^nonNegativeInteger

Più in generale, un dato è rappresentato da un *letterale* (ovvero una stringa di caratteri, "25" nell'esempio) e da un *tipo di dati* (nonNegativeInteger nell'esempio) separati dal simbolo ^^. Si noti che il letterale, da solo, non è sufficiente a identificare un dato: ad esempio il letterale "25" può essere utilizzato per rappresentare una stringa, un intero, un intero non negativo, un intero positivo, un float, e così via:

"25"^^string
"25"^^integer
"25"^^nonNegativeInteger
"25"^^positiveInteger
"25"^^float

I dati, così rappresentati, possono essere utilizzati in asserzioni, in restrizioni e così via. Ad esempio:

haEtà(anna,"25"^^nonNegativeInteger)	l'età di Anna è 25
haEtà ∋ "25"^^nonNegativeInteger	gli oggetti dell'universo la cui età è 25

7. Dalle DL alle ontologie

Come abbiamo già detto, le DL sono linguaggi di rappresentazione adatti alla definizione di ontologie. Fino ad ora, le DL più utilizzate a questo scopo sono note come: $\mathcal{SHIQ}$, che sta alla base del linguaggio DAML+OIL; $\mathcal{SHOIN}(D_n)$, che sta alla base di OWL DL; e $\mathcal{SROIQ}(D_n)$, che sta alla base di OWL 2 DL. Il significato di questi acronimi può essere ricostruito a partire dal codice riportato nella tabella 7.1.

Tabella 7.1. Codice degli acronimi delle DL (A è una classe atomica, C e D sono classi arbitrarie, R e S sono proprietà, a e b sono individui)

codice	*espressività della DL corrispondente*
$\mathcal{AL}$	(i) assiomi d'inclusione ($C \sqsubseteq D$) e di equivalenza ($C \equiv D$) di classi; (ii) classi atomiche A, classe universale $\top$ e classi complesse formate con i costruttori $C \sqcap D$, $\forall R.C$, $\exists R$; asserzioni $C(a)$, $R(a,b)$, $a = b$, $a \neq b$
$\mathcal{ALC}$	come $\mathcal{AL}$, più la classe vuota $\bot$ e le classi complesse formate con i costruttori $\neg C$, $C \sqcup D$, $\exists R.C$
$\mathcal{S}$	come $\mathcal{ALC}$, più la specifica di transitività delle proprietà, $Tra(R)$
$\mathcal{H}$	assiomi d'inclusione ($R \sqsubseteq S$) e di equivalenza ($R \equiv S$) di proprietà
$\mathcal{R}$	come $\mathcal{H}$ più: (i) disgiunzione di proprietà, riflessività globale, irriflessività e asimmetria di proprietà; (ii) assiomi di sottoproprietà basati su catene (par. 6.5); (iii) restrizione di riflessività locale $\exists R.\mathsf{Self}$; (iv) asserzioni negative $\neg R(a,b)$
$\mathcal{O}$	nominali $\{a\}$ (*one-of*), restrizioni di valore $R \ni a$
$\mathcal{I}$	proprietà inversa, R^-
$\mathcal{F}$	funzionalità di una proprietà: $\leq 1R$ o $Fun(R)$
$\mathcal{N}$	restrizioni di cardinalità non qualificate: $\leq nR$, $\geq nR$, $=nR$
$\mathcal{Q}$	restrizioni di cardinalità qualificate: $\leq nR.C$, $\geq nR.C$ e $=nR.C$
D_n o D	attributi con valori in domini di dati

OWL

Il linguaggio OWL (*Web Ontology Language*, `http://www.w3.org/TR/owl-ref/`) è lo standard raccomandato nel 2004 dal W3C (`http://www.w3.org`) per la definizione di ontologie per il web semantico (`http://www.w3.org/2001/sw/`). Sviluppato a partire da DAML+OIL (un linguaggio con logica $\mathcal{SHIQ}$, a sua volta basato sui precedenti linguaggi OIL e DAML-ONT), OWL prevede tre livelli di complessità crescente:

- *OWL Lite*, che realizza la logica descrittiva $\mathcal{SHIF}(\mathrm{D}_n)$, semplice da implementare ma scarsamente espressiva;
- *OWL DL*, che realizza la logica descrittiva $\mathcal{SHOIN}(\mathrm{D}_n)$, abbastanza espressiva ma comunque decidibile e dotata di procedure di ragionamento di complessità nota, molto 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, ed è quindi problematico dal punto di vista della meccanizzazione del ragionamento).

OWL 2

Il linguaggio OWL 2 (`http://www.w3.org/Submission/owl11-overview/`) è un'estensione di OWL (oggi spesso chiamato OWL 1 per distinguerlo da OWL 2) basato sulla logica descrittiva $\mathcal{SROIQ}(\mathrm{D}_n)$. È raccomandato dal W3C a partire dal 2009, ma alcuni strumenti per il web semantico lo implementano per ora in forma incompleta.

Anche per il linguaggio OWL 2 è prevista una versione DL, che realizza la logica descrittiva $\mathcal{SROIQ}(\mathrm{D}_n)$, e una verione più potente, OWL 2 Full. Non è prevista una versione Lite, ma sono definiti numerosi frammenti di OWL 2 DL (chiamati "profili") pensati per applicazioni particolari, come l'integrazione di basi di dati distribuite. Nel seguito, comunque, ci occuperemo soltanto del linguaggio OWL 2 DL, che per brevità chiameremo semplicemente OWL.

7.1 Rappresentazione di ontologie OWL

Un'ontologia OWL può essere rappresentata in molti modi diversi e in particolare con: la notazione DL, la sintassi funzionale, la sintassi di Manchester, il formato OWL/XML e il formato OWL/RDF. Mentre la notazione DL, utilizzata in queste note, è molto diffusa nelle pubblicazioni scientifiche grazie alla sua chiarezza e concisione, la sintassi funzionale è sistematicamente utilizzata nei documenti ufficiali del W3C. La sintassi di Manchester, poi, è utilizzata da Protégé, il più diffuso editor di ontologie OWL. Tutte queste notazioni sono comunque orientate alla comunicazione fra esseri umani o fra esseri umani e computer, più che alla rappresentazione interna nei computer e allo scambio di dati fra applicazioni.

La rappresentazione OWL/XML è un linguaggio applicativo XML specificamente definito per OWL 2 DL, orientato allo scambio di dati fra applicazioni. A questo fine, tuttavia, è prevedibile che la rappresentazione più utilizzata (come è già avvenuto per le ontologie OWL 1) sarà OWL/RDF; in tale rappresentazione, ogni ontologia è rappresentata come un documento RDF (il quale, a sua volta, è usualmente serializzato nel formato RDF/XML).

La rappresentazione RDF delle ontologie presenta grandi vantaggi per l'interoperabilità delle applicazioni, ma è difficilmente leggibile da parte degli esseri umani. D'altra parte gli strumenti per la definizione di ontologie utilizzano in genere interfacce testuali o grafiche che nascondono all'utente la rappresentazione RDF. Per questi motivi non adotteremo la rappresentazione RDF in queste note, ma continueremo a utilizzare la sintassi DL. Tuttavia, va tenuto presente che la rappresentazione concreta di un'ontologia OWL è tipicamente costituita da un grafo RDF: questa caratteristica consente l'accesso a un'ontologia OWL anche con gli strumenti per l'elaborazione di grafi RDF e rende possibile l'adozione di espedienti tecnici che, purché ben utilizzati, permettono di superare alcune limitazioni espressive di OWL.

7.2 I servizi di ragionamento

Un aspetto che distingue le basi di conoscenze dalle base di dati è la possibilità di condurre *ragionamenti* in modo automatico. Nel contesto della logica quando si parla di "ragionamento" ci si riferisce a ragionamenti di tipo deduttivo (o più semplicemente *deduzioni*): non prenderemo quindi in considerazione i ragionamenti di tipo induttivo o abduttivo, menzionati nel paragrafo 1.1. Un ragionamento è dunque un procedimento che porta a verificare se un enunciato o asserzione X (ad esempio una relazione di sottoclasse) è *conseguenza logica* di una base di conoscenze.

Conseguenza logica

In generale una base di conoscenze, K, è costituita da una TBox, T, da una RBox, R, e da un'ABox, A: scriveremo quindi $K = T \cup R \cup A$; in certi casi ci interesserà considerare soltanto il contributo della TBox e dell'RBox; in tal caso avremo $K = T \cup R$, o della sola TBox, e in tal caso avremo $K = T$.

Intuitivamente un enunciato o asserzione X è conseguenza logica di una base di conoscenze K quando la verità degli assiomi e delle asserzioni contenuti in K è sufficiente a garantire la verità di X. Più precisamente, diremo che *X segue logicamente da K*, o che *X è conseguenza logica di K*, quando un enunciato o asserzione X è vero in ogni modello che soddisfi la base di conoscenze K (ovvero, tale che ogni enunciato e ogni asserzione di K sia vero nel modello). In tal caso scriveremo[18]

$K \models X$

Consideriamo ad esempio $K = T$ contenente gli assiomi seguenti:

T:
- T1. Persona ≡ *DisUni*(Uomo,Donna)
- T2. Genitore ≡ ∃genDi
- T3. Madre ≡ Genitore ⊓ Donna
- T4. Padre ≡ Genitore ⊓ Uomo
- T5. *Dom*(genDi,Persona)
- T6. *Rng*(genDi,Persona)

Il contenuto di K implica logicamente certi enunciati che, pur non essendo contenuti esplicitamente in K, sono necessariamente veri sotto l'ipotesi che sia vero il contenuto di K. Ad esempio:

Madre ⊑ Persona	Madre ⊑ ¬Uomo	Madre ⊓ Padre ⊑ ⊥
Padre ⊑ Persona	Padre ⊑ ¬Donna	

Per indicare che questi enunciati sono conseguenze logiche di K scriviamo

$K \models$ Madre ⊑ Persona

e così via per gli altri enunciati. Ci sono anche enunciati che *non sono* conseguenza logica di K: ad esempio, da K non segue logicamente che ogni persona abbia dei genitori; per esprimere questo fatto scriviamo:

$K \not\models$ Persona ⊑ ∃genDi$^-$

Servizi di ragionamento

In logica si definisce *calcolo* o *procedura di prova* una procedura in grado di verificare se X segue logicamente da un insieme assegnato di premesse. Il calcolo esegue il compito *deducendo X dalle premesse*, ovvero costruendo una *prova* (o *dimostrazione*) che, utilizzando determinate *regole d'inferenza*, porta dalle premesse a X.

Nel campo delle DL si preferisce parlare di *ragionamento* anziché di deduzione; un sistema software che implementa un calcolo viene quindi denominato *ragionatore* (*reasoner*). Un ragionatore implementa un insieme di *servizi di ragionamento*: a sua volta, ogni servizio di ragionamento è caratterizzato dal tipo di deduzione che il servizio è in grado di eseguire. Quando poi una DL è

[18] È importante non confondere la scrittura $M \models X$ (che significa "l'enunciato X è vero nel modello M") con la scrittura $K \models X$ (che significa "l'enunciato X è conseguenza logica dell'insieme di enunciati K").

decidibile, come nel caso di $\mathcal{SROIQ}(D_n)$, si ha la garanzia che ogni servizio di ragionamento terminerà l'esecuzione dopo un numero finito di passi.

Non è facile descrivere in generale i servizi di ragionamento offerti dai ragionatori oggi disponibili, perché ragionatori diversi implementano insiemi di servizi almeno in parte differenti. Possiamo però offrire una classificazione e una descrizione di massima dei servizi di ragionamento offerti dalla maggior parte dei ragionatori esistenti. A questo scopo classifichiamo i servizi di ragionamento in due categorie:

- servizi *globali*: sono servizi che deducono proprietà globali della KB, come ad esempio la consistenza[19];
- servizi *locali*: sono di servizi che dimostrano specifici enunciati o asserzioni, deducendoli dagli assiomi della TBox e della RBox, nonché dalle asserzioni dell'ABox.

Servizi di ragionamento globali

Data una base di conoscenze *K*, fra i più comuni servizi di ragionamento globali citiamo:

- il *controllo di consistenza*: il servizio stabilisce se *K* è o non è consistente;
- la *classificazione delle classi atomiche*: il servizio costruisce la gerarchia completa delle relazioni di sottoclasse che sussistono fra le classi atomiche di *K*;
- la *classificazione degli individui* (detta anche *realizzazione*): il servizio assegna a ogni individuo le n classi atomiche $\{A_1,...,A_n\}$ *più specifiche* cui l'individuo appartiene.

A chiarimento del compito di classificazione degli individui, precisiamo che una classe A si dice *più specifica* di una classe B quando vale $A \sqsubseteq B$ e non vale $B \sqsubseteq A$. Si noti inoltre che la relazione di sottoclasse induce un ordinamento parziale fra le classi: le classi atomiche più specifiche cui l'individuo a appartiene sono le classi A_i per cui vale $K \models A_i(a)$ e minimali rispetto alla relazione di sottoclasse.

Servizi di ragionamento locali

Data una base di conoscenze *K*, fra i più comuni servizi di questo tipo citiamo:

- la verifica della *relazione di sottoclasse* fra due classi arbitrarie: il servizio stabilisce se vale $K \models C \sqsubseteq D$ oppure $K \not\models C \sqsubseteq D$;
- la verifica dell'*equivalenza* di due classi arbitrarie: il servizio stabilisce se vale $K \models C \equiv D$ oppure $K \not\models C \equiv D$;
- la verifica della *soddisfacibilità* di una classe: il servizio verifica se una classe C è *soddisfacibile*, cioè se esiste almeno un modello di *K* in cui non è vuoto l'insieme degli oggetti che soddisfano C; una classe C è quindi soddisfacibile se vale $K \not\models C \sqsubseteq \bot$ e non lo è se vale $K \models C \sqsubseteq \bot$;
- la verifica della *disgiunzione* di due classi: il servizio stabilisce se due classi C e D sono *disgiunte*, nel senso che in ogni modello di *K* è vuoto l'insieme degli oggetti che soddisfano entrambe le classi C e D; le classi C e D sono quindi disgiunte se vale $K \models C \sqcap D \sqsubseteq \bot$ e non lo sono se vale $K \not\models C \sqcap D \sqsubseteq \bot$;
- l'*instance check*: data una classe arbitraria C e un individuo a, il servizio stabilisce se l'individuo a soddisfa la classe C, ovvero se vale $K \models C(a)$ oppure $K \not\models C(a)$;
- il *retrieval*: data una classe arbitraria C, il servizio trova tutti gli individi a_1, ..., a_n noti alla base di conoscenze tali che $K \models C(a_k)$; il servizio restituisce un insieme $\{a_1, ..., a_m\}$, eventualmente vuoto, di individui.

[19] Una KB si dice consistente se ammette almeno un modello. Il termine italiano più appropriato sarebbe in realtà *coerenza*, ma questo termine è stato ormai sostituito nell'uso dal termine *consistenza*, traduzione 'a orecchio' dell'inglese *consistency*. Con una KB inconsistente non ha senso eseguire ragionamenti, perché da essa discende logicamente *qualsiasi enunciato o asserzione* (nella logica antica questo principio era noto come *de falso quodlibet*).

Riducibilità di tutti i servizi al servizio di soddisfacibilità di una classe

Tutti i servizi di ragionamento presentati nel sottoparagrafo precedente possono essere ridotti ad un unico servizio, e in particolare alla *verifica di soddisfacibilità di una classe*. Come abbiamo già detto, la classe C è soddisfacibile, relativamente a una base di conoscenze K, se K ammette almeno un modello in cui l'estensione di C non sia vuota. Supponiamo ora di disporre di un servizio di ragionamento che, date una classe arbitraria C e una base di conoscenze K, è in grado di verificare se C è o non è soddisfacibile relativamente a K, in simboli:

$K \not\models C \sqsubseteq \bot$ (*C è soddisfacibile relativamente a K*),

$K \models C \sqsubseteq \bot$ (*C è insoddisfacibile relativamente a K*).

È facile dimostrare che la verifica di sottoclasse si riduce alla verifica di soddisfacibilità nel modo seguente:

$K \models C \sqsubseteq D$ se, e solo se, $K \models C \sqcap \neg D \sqsubseteq \bot$

ovvero: C è sottoclasse di D se, e solo se, l'intersezione fra C e il complemento di D è vuota in ogni modello di K, ovvero se la classe $C \sqcap \neg D$ è insoddisfacibile.

La verifica di equivalenza, di soddisfacibilità e di disgiunzione, nonché la classificazione delle classi atomiche, si riportano alla verifica di soddisfacibilità di una classe partendo dalle rispettive definizioni. Anche tutti gli altri servizi di ragionamento si possono ridurre alla verifica di soddisfacibilità di una classe in modo analogo. Dunque un unico servizio di ragionamento, in grado di stabilire se una classe è o non è soddisfacibile, consente di realizzare tutti i servizi di ragionamento precedentemente descritti.

A questo punto, è chiaro che per implementare concretamente tutti i servizi di ragionamento fin qui descritti è sufficiente implementare il servizio di verifica di soddisfacibilità di una classe. Questa è appunto la strada che si segue per implementare i servizi di ragionamento per le DL molto espressive, come $\mathcal{SHOIN}(\mathrm{D}_n)$ e $\mathcal{SROIQ}(\mathrm{D}_n)$. Di questo ci occupiamo nel prossimo paragrafo.

7.3 La procedura SAT

Come abbiamo già detto, la classe C è soddisfacibile, relativamente a una base di conoscenze K, se K ammette almeno un modello in cui l'estensione di C non sia vuota. Per le DL decidibili si può formulare una procedura effettiva $Sat(K,C)$ che, prese in ingresso K e una classe arbitraria C, stabilisce in un numero finito di passi se C è o non è soddisfacibile relativamente a K:

$Sat(K,C) = 1$ sse $K \not\models C \sqsubseteq \bot$, ovvero sse *$C$ è soddisfacibile relativamente a K*,

$Sat(K,C) = 0$ sse $K \models C \sqsubseteq \bot$, ovvero sse *$C$ è insoddisfacibile relativamente a K*.

Nelle sue versioni più diffuse la procedura *Sat* è basata sul cosiddetto *metodo dei tableaux*, da tempo studiato e applicato a diversi tipi di logiche. Qui ci limitiamo a dare un'idea intuitiva del suo funzionamento basandoci su un semplice esempio. Supponiamo di voler mostrare che, assumendo la K del paragrafo 7.2, vale la relazione di sottoclasse

$K \models \mathsf{Madre} \sqsubseteq \exists \mathsf{genDi}$.

Innanzitutto la verifica di sottoclasse viene ridotta alla verifica di soddisfacibilità di $\mathsf{Madre} \sqcap \neg\exists \mathsf{genDi}$ relativamente a K:

$Sat(K, \mathsf{Madre} \sqcap \neg\exists \mathsf{genDi})$

La procedura *Sat* parte dall'ipotesi che la classe sia soddisfacibile e sviluppa un ragionamento sulla base di questa ipotesi iniziale. A grandi linee i passi del ragionamento sono i seguenti:

– assumendo che $\mathsf{Madre} \sqcap \neg\exists \mathsf{genDi}$ sia soddisfacibile, allora in qualche modello M di K esiste un oggetto che appartiene all'estensione della classe; introducendo un individuo, a, per identificare questo oggetto possiamo scrivere

1. $\mathsf{Madre} \sqcap \neg\exists \mathsf{genDi}(\mathsf{a})$

– da 1, poiché l'intersezione di due classi si distribuisce rispetto all'argomento deduciamo che:
  2.1 Madre(a)
  2.2 $\neg\exists$genDi(a)
– da 2.1, tenendo conto dell'assioma T3, deduciamo che:
  3. Genitore $\sqcap$ Donna(a).
– da 3, poiché l'intersezione di due classi si distribuisce rispetto all'argomento deduciamo che:
  4.1 Genitore(a)
  4.2 Donna(a)
– da 4.1, tenendo conto dell'assioma T2, deduciamo che:
  5. $\exists$genDi(a)
– l'asserzione 5 è in contraddizione con la 2.2: l'ipotesi iniziale che la classe Madre $\sqcap \neg\exists$genDi sia soddisfacibile va quindi rigettata.

Questo ragionamento mostra che la classe Madre $\sqcap \neg\exists$genDi è insoddisfacibile. Dato il rapporto che lega l'insoddisfacibilità alla relazione di sottoclasse, risulta stabilito che:

$K \models$ Madre $\sqsubseteq \exists$genDi.

Naturalmente l'esempio precedente è molto schematico: i reasoner reali sono notevolmente complessi e nello stabilire se una classe sia o non sia soddisfacibile utilizzano tutta una serie di euristiche di ottimizzazione, senza le quali i tempi di risposta sarebbero inaccettabili.

Ci si può chiedere come sia possibile utilizzare un reasoner in un'applicazione senza avere un'idea precisa e dettagliata di come funzionino. Ma questo è il lato positivo dell'uso di una logica dotata di una semantica rigorosa: per sapere come un reasoner risponderà a un'interrogazione non è necessario sapere come funziona! Infatti ci basta sapere che ogni reasoner implementa i servizi di ragionamento rispettando rigorosamente la loro definizione semantica. Per chi specifica e utilizza un'ontologia non è necessario sapere altro.

7.4 Invocazione dei servizi di ragionamento

Le specifiche di OWL definiscono in modo rigoroso la sintassi delle classi e degli enunciati logici ammissibili, ma non definiscono una sintassi standard per l'invocazione dei servizi di ragionamento. In questi appunti per rappresentare le invocazioni di servizi di ragionamento e le relative risposte utilizzeremo la seguente scrittura convenzionale:

?– *interrogazione* $\Rightarrow$ *risposta*.

Nel seguito faremo riferimento a un numero limitato di servizi di ragionamento: le verifiche di sussunzione, equivalenza e soddisfacibilità, l'instance check e il retrieval. In particolare esprimeremo le invocazioni di questi servizi nel modo seguente:

– verifica di sussunzione fra classi: ?– $C \sqsubseteq D$ $\Rightarrow$ *true*/*false*
– verifica di equivalenza fra classi: ?– $C \equiv D$ $\Rightarrow$ *true*/*false*
– verifica di soddisfacibilità di una classe: ?– C $\Rightarrow$ *true*/*false*
– instance check: ?– $C(a)$ $\Rightarrow$ *true*/*false*
– retrieval: ?– $C(*)$ $\Rightarrow$ $\{a_1, ..., a_n\}$

Ad esempio, aggiungiamo alla *K* del paragrafo 7.2 la seguente ABox:

A: A1. Donna(anna)
  A2. Donna(cecilia)
  A3. Uomo(bob)
  A4. genDi(anna,cecilia)
  A5. genDi(bob,cecilia)

Vediamo ora i risultati di alcuni ragionamenti.

Instance check

Dati la classe complessa Femmina ⊓ ∃genDi e l'ndividuo anna si ha

?– Femmina ⊓ ∃genDi(anna) ⇒ *true*

in quanto $K \models$ Femmina ⊓ ∃genDi(anna).

Dati la classe complessa Femmina ⊓ ∃genDi e l'individuo cecilia si ha

?– Femmina ⊓ ∃genDi(cecilia) ⇒ *false*

in quanto $K \not\models$ Femmina ⊓ ∃genDi(cecilia). Attenzione: è importante interpretare correttamente il significato della risposta *false*: ad esempio

?– ∃genDi(cecilia) ⇒ *false*

non significa, come si potrebbe pensare, che *da K segue logicamente che l'individuo* cecilia *non è genitore di nessuno*, bensì che *da K non segue logicamente che l'individuo* cecilia *sia genitore di qualcuno*. In generale, la risposta *false* non ci dice che da *K* è dimostrabile *non-X*, bensì che non è dimostrabile *X*.

Retrieval

Data la classe Genitore si ha

?– Genitore(∗) ⇒ {anna,bob}

in quanto $K \models$ Genitore(anna) e $K \models$ Genitore(bob).

7.5 Gli editor di ontologie

Oggi sono reperibili in rete diversi editor di ontologie OWL. Il più utilizzato è Protégé (`http://protege.stanford.edu/`), che nella versione 4.1 (attualmente distribuito in versione beta, `http://protege.stanford.edu/download/protege/4.1/installanywhere/`) copre completamente OWL 2 DL.

Una volta compresi i principi della logica $\mathcal{SROIQ}(D_n)$ non è difficile specificare un'ontologia con Protégé utilizzando la sintassi di Manchester (vedi l'appendice II). Occorre però passare da un punto di vista, per così dire, *fact oriented* (tipico della notazione logica adottata in queste note) al punto di vista *entity oriented* adottato da Protégé: con questo vogliamo dire che ogni assioma è inserito nell'ontologia in corrispondenza di una singola *entità* (classe atomica, proprietà, o individuo). Per specificare un'ontologia, infatti, Protégé mette a disposizione dell'utente un certo numero di schede (*tabs*), e in particolare le schede delle classi, delle proprietà (*object properties*), degli attributi (*data properties*), degli individui, delle interrogazioni (*queries*) e così via. Consideriamo un assioma di sottoclasse della forma $C \sqsubseteq D$, ad esempio

A1 ⊓ A2 ⊑ B1 ⊓ B2

Questo assioma di TBox non può essere direttamente definito in Protégé, perché questo strumento associa le relazioni di sottoclasse (e di equivalenza fra classi) soltanto a classi atomiche. In altre parole, è possibile inserire soltanto assiomi di sottoclasse e di equivalenza del tipo

$A \sqsubseteq C$ $A \equiv C$

Per specificare l'assioma dell'esempio sono quindi necessari i passi seguenti:

- utilizzando la scheda delle classi, definire A1, A2, B1 e B2 come sottoclassi della classe Thing (corrispondente a ⊤ nella sintassi di Manchester adottata da Protégé)
- poiché è possibile definire soltanto sovraclassi o classi equivalenti di classi atomiche, specificare come sottoclasse di Thing una nuova classe A3 definendola come equivalente ad A1 **and** A2;
- specificare che A3 è sottoclasse di B1 **and** B2.

Per lo stesso motivo non è possibile specificare direttamente un assioma come

$\{a\} \sqsubseteq \exists R.B$

In questo caso occorrerà:

- nella scheda degli individui, creare un individuo a;
- nella scheda delle classi, creare A e B come sottoclassi di Thing, e specificare che la classe atomica A è equivalente alla classe complessa {a};
- nella scheda delle propertà, creare la proprietà R, eventualmente specificandone il dominio (*domain*) e codominio (*range*);
- di nuovo nella scheda delle classi, specificare che A è sottoclasse della classe complessa R **some** B.

Altri punti da tenere in considerazione sono i seguenti:

- il contenuto della TBox viene specificato in parte nella scheda delle classi, in parte nella scheda delle proprietà (ad esempio per quanto riguarda enunciati quali $Dom(R,C)$, $Rng(R,C)$, $Fun(R)$ e analoghi);
- il contenuto della RBox viene specificato nelle schede delle propertà e degli attributi;
- il contenuto dell'ABox viene specificato nelle schede degli individui: le asserzioni del tipo $C(a)$, $R(a,b)$, $\neg R(a,b)$, $a = b$, $a \neq b$ vanno inserite in corrispondenza all'individuo a, in quanto le asserzioni sono viste come caratteristiche di specifici individui.

Una volta definita un'ontologia con Protégé è possibile utilizzare un reasoner per rispondere alle interrogazioni. Il reasoner HermiT (`http://hermit-reasoner.com/`), un sistema in fase prototipale incluso come plug-in dell'editor Protégé 4.1, è oggi forse il più efficiente[20]. Anche i reasoner Fact++ e Pellet (anch'essi distribuiti come plug-in di Protégé) sono ampiamente utilizzati.

Direttamente da Protégé si possono eseguire molti tipi di ragionamenti (a questo scopo occorre indicare all'interno di Protégé il reasoner che si desidera utilizzare). Alcuni risultati vengono automaticamente presentati nelle diverse schede; inoltre è possibile:

- Classificare un'ontologia, ovvero classificare le sue classi atomiche; questa operazione è comunque necessaria prima di utilizzare un reasoner.
- Verificare se l'ontologia è consistente; in genere questo controllo viene eseguito automaticamente e le eventuali inconsistenze vengono segnalate.;
- Formulare interrogazioni utilizzando la scheda delle interrogazioni, che consente di specificare una classe arbitraria (con la sintassi di Manchester) e ne restituisce le classi atomiche equivalenti, le sovraclassi e sottoclassi atomiche, e gli individui noti.

7.6 Che cos'è un'ontologia?

A questo punto vale la pena di ritornare al concetto di ontologia e di verificare fino a che punto questo risulti chiaro e ben definito. Nella parte I abbiamo parlato di basi di conoscenze (KB) e abbiamo introdotto una distinzione fra conoscenze concettuali, nomologiche e fattuali. Nel suo uso più comune, il termine "ontologia" corrisponde alle conoscenze concettuali e nomologiche, mentre non comprende le conoscenze fattuali; per fare un esempio, faranno parte di una KB geografica i concetti di stato e di città e la relazione "capitale di" fra una città e uno stato, ma non il fatto che Londra sia la capitale del Regno Unito. In altre parole, l'ontologia specifica il *modello semantico* grazie al quale descriviamo il mondo, mentre non comprende il *modello concreto* del mondo (cioè i fatti) che ci formiamo sulla base di tale schema. Un'altra osservazione: le conoscenze nomologiche, al di fuori dai settori strettamente scientifici, sono spesso 'generiche' piuttosto che 'universali': sappiamo che *in genere* (ma non necessariamente) le madri amano i loro figli, che *in genere* (ma non necessariamente) se un bicchiere cade a terra si rompe, e così via. Ora, né FOL né le DL sono adatte a esprimere conoscenze generiche:

[20] In verità il prototipo corrente di HermiT dà dei problemi con i vincoli di cardinalità che superino valori dell'ordine di 7 o 8. Gli altri reasoner citati sono invece immuni da questo difetto.

le conoscenze di questo tipo dovranno quindi essere escluse dalle ontologie, oppure approssimate come conoscenze universali (*tutte* la madri amano i loro figli, e così via).

In linea di massima, in OWL l'ontologia è specificata nella TBox e nella RBox, mentre l'ABox rappresenta le conoscenze fattuali. Ma questo non è sempre vero: in certi casi, infatti, l'ABox contribuisce in modo essenziale alla definizione del modello semantico e quindi dell'ontologia. Consideriamo ad esempio la definizione della classe delle persone, viste come donne (ovvero persone di genere femminile) e uomini (ovvero persone di genere maschile). Se utilizziamo due individui (f e m) per denotare i due generi, e trascuriamo qui il contenuto della TBox, nell'ABox avremo asserzioni del tipo:

A1. $f \neq m$
A2. haGenere(anna,f)
A3. haGenere(bruno,m)

Si vede chiaramente che mentre le asserzioni A2 e A3 rappresentano conoscenze fattuali, l'asserzione A1 fa parte dello schema concettuale che definisce i due generi come *diversi*; quindi A1, pur essendo un'asserzione di ABox, è parte integrante dell'ontologia, mentre A2 e A3 non lo sono.

Occorre tenere ben distinti i concetti di TBox e RBox dalle schede delle classi e delle proprietà in un editor di ontologie come Protégé; infatti, mentre tutti gli assiomi introdotti nella scheda delle classi fanno parte della TBox, gli assiomi introdotti nella scheda delle proprietà sono a volte parte della TBox e a volta parte della RBox. Ad esempio fanno parte della TBox assiomi come $Dom(R,C)$, $Rng(R,C)$ e $Fun(R)$, mentre fanno parte della RBox assiomi come $Sym(R)$, $Tra(R)$ e $R \sqsubseteq S$. Il criterio di classificazione è semplice: appartengono alla TBox tutti gli assiomi esprimibili come relazioni logiche fra classi, e appartengono alla RBox tutti gli altri assiomi (escluse naturalmente le asserzioni, che fanno parte dell'ABox).

Infine un commento su un altro tipo di conoscenze, di cui non ci siamo occupati finora: le *conoscenze terminologiche*. Semplificando al massimo un argomento molto complesso possiamo assumere che lo stesso concetto (ad es. il concetto di Madre) corrisponda a termini diversi in diverse lingue (italiano *madre*, inglese *mother*, tedesco *Mutter* e così via). In prima approssimazione, la parola di una lingua corrispondente a un concetto può essere considerata come il valore di un attributo di tipo string associato al concetto. I concetti, però, corrispondono alle classi, e in una DL non è possibile associare un valore di un attributo a una *classe*, bensì soltanto a un *individuo*[21].

[21] Attenzione: è pratica comune usare l'espressione "attributo di una classe", ad esempio dicendo che haEtà è un attributo della classe Persona. Ma con ciò si intende dire che haEtà è un attributo *degli oggetti appartenenti all'estensione della classe*, non della classe considerata come un'entità unica. I termini linguistici, invece, sono attributi di una classe considerata come un'entità unica.