Dipartimento di elettronica e informazione
Politecnico di Milano

Ingegneria della conoscenza: modelli semantici

Edizione 2010-11
Parte III, Riferimenti bibliografici e Appendici

Marco Colombetti[1], 18 aprile 2011

Indice

Parte III: Rappresentazione di conoscenze in OWL

8. Individui e oggetti

Il linguaggio OWL 2 DL, che d'ora in poi chiameremo semplicemente OWL, è basato sulla logica $\mathcal{SROIQ}(\mathrm{D}_n)$, che rispetto ad altre DL più deboli si caratterizza per la presenza di una complessa RBox ($\mathcal{R}$) e del costruttore *one-of* (O). In questo paragrafo analizzeremo alcuni usi di quest'ultimo.

8.1 Asserzioni nell'ABox e nella TBox

Nelle DL che contengono il costruttore O, ogni asserzione dell'ABox può essere rimpiazzata da un assioma concettuale equivalente nella TBox. In particolare, come si può verificare facilmente,

- asserire nell'ABox $C(a)$ equivale a inserire nella TBox l'assioma $\{a\} \sqsubseteq C$
- asserire nell'ABox $R(a,b)$ equivale a inserire nella TBox l'assioma $\{a\} \sqsubseteq R \ni b$ oppure, in presenza del costruttore di proprietà inversa, l'assioma $\{b\} \sqsubseteq R^- \ni a$
- asserire nell'ABox $\neg R(a,b)$ equivale a inserire nella TBox l'assioma $\{a\} \sqsubseteq \neg(R \ni b)$ oppure, in presenza del costruttore di proprietà inversa, l'assioma $\{b\} \sqsubseteq \neg(R^- \ni a)$
- asserire nell'ABox $a = b$ equivale a inserire nella TBox l'assioma $\{a\} \sqsubseteq \{b\}$
- asserire nell'ABox $a \neq b$ equivale a inserire nella TBox l'assioma $\{a\} \sqcap \{b\} \sqsubseteq \bot$.

È quindi impossibile mantenere una rigida separazione concettuale fra le conoscenze rappresentate nell'ABox e quelle rappresentate nella TBox. In linea di principio le asserzioni nell'ABox e i corrispondenti assiomi nella TBox hanno gli stessi effetti sul ragionamento; tuttavia per ragioni di chiarezza e semplicità è preferibile utilizzare asserzioni nell'ABox ogniqualvolta ciò sia possibile.

8.2 Oggetti noti e oggetti ignoti

Non bisogna confondere gli individui con gli oggetti: un *oggetto*, nella terminologia adottata in queste note, è un elemento dell'universo; un *individuo*, invece, è un simbolo utilizzato per denotare un oggetto. Come abbiamo visto, poi, l'estensione di una classe è l'insieme di tutti gli oggetti dell'universo che soddisfano la definizione di una classe.

Come sappiamo, a ogni individuo corrisponde uno e un solo oggetto dell'universo, che l'individuo denota. Non vale invece l'inverso, nel senso che a un oggetto dell'universo possono corrispondere nessuno, uno, o più individui. Nel gergo 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*.

Un'asserzione di uguaglianza, come

$\mathsf{a} = \mathsf{b}$

impone agli individui a e b di denotare lo stesso oggetto *in ogni possibile modello* della KB. Analogamente, l'asserzione

$\mathsf{a} \neq \mathsf{b}$

impone agli individui a e b di denotare oggetti distinti *in ogni possibile modello* della KB. L'uguaglianza o la diversità di due individui possono essere conseguenza logica del contenuto di una KB anche in assenza di un'asserzione specifica. Ad esempio, se due individui appartengono a classi disgiunte, essi saranno sicuramente diversi. Se però la KB non comporta né l'uguaglianza né la diversità di due individui, esisteranno modelli della KB in cui i due individui denotano oggetti distinti, e modelli della KB in cui i due individui denotano lo stesso oggetto.

8.3 Quanti oggetti?

Mentre il numero degli individui è sempre finito (perché ogni individuo deve essere esplicitamente introdotto in una KB) e può essere nullo (se nessun individuo è stato introdotto nella KB), il numero degli oggetti può essere finito o infinito, ma non è mai nullo perché l'universo, per definizione, è un insieme non vuoto di oggetti. Consideriamo un esempio: in una KB introduciamo una classe A e due individui, a e b:

$A(a)$

$A(b)$

Un errore comune consiste nel pensare che a questo punto la KB ammetta soltanto modelli in cui l'estensione della classe A contiene esattamente due oggetti, corrispondenti agli individui a e b. La KB è invece compatibile con modelli in cui l'estensione di A ha cardinalità positiva qualsiasi; tale estensione potrebbe infatti contenere

- un solo oggetto: in tal caso a e b sono uguali, cioè denotano lo stesso oggetto;
- due oggetti: in tal caso a e b possono denotare due oggetti distinti, oppure lo stesso oggetto; nel secondo caso il modello conterrà un ulteriore oggetto appartenente all'estensione di A, che però è ignoto (nel senso precisato in precedenza);
- qualunque numero positivo, finito o infinito, di oggetti, di cui al massimo due sono noti (in quanto corrispondono agli individui a e b).

Possiamo naturalmente specificare che i due oggetti noti sono distinti:

(8.1) $a \neq b$

A questo punto siamo sicuri che l'estensione di A contenga almeno due oggetti, corrispondenti ai due individui; l'estensione di A potrebbe però contenere anche un numero qualsiasi (finito o infinito) di oggetti ignoti. Se vogliamo imporre all'estensione di A di avere esattamente due elementi possiamo aggiungere a 8.1 l'assioma

(8.2) $A \equiv \{a,b\}$

Presi insieme, gli assiomi 8.1 e 8.2 impongono all'estensione di A di contenere esattamente due oggetti, noti alla KB come gli individui a e b.

Questo metodo può essere utilizzato per imporre qualunque cardinalità finita a una classe, ma naturalmente è poco pratico se vogliamo dire, poniamo, che l'estensione di A contiene esattamente (o almeno, o al massimo) 1000 oggetti distinti in ogni modello della KB. Vediamo ora un metodo generale per specificare la cardinalità di una classe.

Cardinalità di una classe

Supponiamo di voler specificare che l'estensione A^I della classe A deve contenere almeno (o esattamente, o al massimo) 1000 oggetti. Potremmo tentare una soluzione simile a quella adottata nel paragrafo 5.1 per rappresentare la formula esistenziale $\exists x A(x)$. Infatti l'enunciato[2]

(!) $\top \sqsubseteq {=}1000\dot{\top}.A$

sembra dire che ogni oggetto dell'universo è connesso tramite la proprietà top con esattamente 1000 oggetti di A^I, e ciò comporta che A^I contenga esattamente 1000 oggetti. Tuttavia questa soluzione non è ammissibile, perché in OWL a proprietà $\dot{\top}$ è considerata composita, e quindi non ammette restrizioni di cardinalità.

Per risolvere il problema dobbiamo seguire una via indiretta. Introduciamo nell'universo una relazione arbitraria, che chiamiamo conta perché intuitivamente intendiamo usarla per contare oggetti:

$\text{conta}: \top \longrightarrow \top$

[2] Il punto esclamativo che precede una formula, eventualmente insieme a un identificatore numerico, indica che la formula corrispondente è per qualche verso scorretta.

Per contare gli oggetti di A^I scegliamo un oggetto qualsiasi dell'universo (che può essere interno o esterno ad A^I) da utilizzare come 'testimone'. Chiamiamo w (da "witness") l'individuo che denota tale oggetto; dire che w^I conta gli oggetti di A^I equivale a dire che ciascun oggetto di A^I è contato da w^I:

$A \sqsubseteq \text{conta}^- \ni w$

A questo punto, per dire che la cardinalità di A è almeno, al massimo o esattamente *n* diremo rispettivamente (con un'asserzione di ABox) che w conta almeno *n*, al massimo *n* o esattamente *n* elementi di A:

$\geq n$ conta.A (w)

$\leq n$ conta.A (w)

$= n$ conta.A (w)

La figura 8.1 rappresenta due modelli in cui un testimone conta i tre oggetti di A^I: nella 8.1a il testimone è esterno ad A^I, mentre nella 8.1b è interno. Si noti che w^I potrebbe contare anche altri oggetti, non appartenenti ad A^I, ma ciò non avrebbe conseguenze sulla cardinalità di A^I.

Va notato infine che è spesso possibile specificare la cardinalità di una classe utilizzando per il conteggio e come testimone una proprietà e un individuo già esistenti nell'ontologia, senza introdurre conta e w. Supponiamo ad esempio di voler rappresentare l'affermazione che gli apostoli di Gesù sono dodici. Introduciamo una proprietà apostoloDi, un individuo gesù e una classe ApostoloDiGesù che contiene esattamente gli oggetti dell'universo che sono apostoli di Gesù:

apostoloDi: Persona $\longrightarrow$ Persona

ApostoloDiGesù $\equiv$ apostoloDi $\ni$ gesù

A questo punto, per specificare la cardinalità di ApostoloDiGesù è possibile utilizzare la proprietà apostoloDi^- per il conteggio e l'individuo gesù come testimone:

$=12$ apostoloDi^- (gesù)

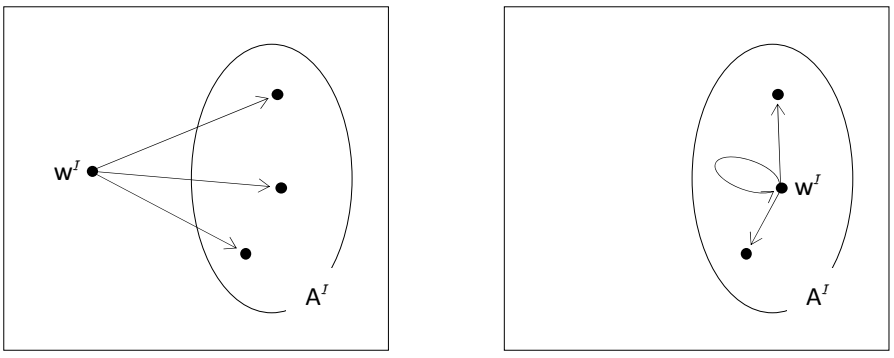
8.4 Il prodotto cartesiano

Com'è noto il prodotto cartesiano di due insiemi *A* e *B* è definito da

$A \times B = \{\langle x,y\rangle \mid x \in A \text{ e } y \in B\}$

Dunque $A \times B$ è una particolare relazione binaria, con dominio *A* e codominio *B*: più precisamente, $A \times B$ è la relazione binaria *completa* da *A* a *B*, nel senso che $A \times B$ mette in relazione ogni elemento di *A* con ogni elemento di *B*. Il prodotto cartesiano, in quanto relazione completa da un insieme a un altro, interviene quando vogliamo rappresentare enunciati FOL della forma

(8.3) $\forall x \, \forall y \, (A(x) \wedge B(y) \rightarrow R(x,y))$



a *b*

Figura 8.1. Contare gli oggetti di un insieme.

che può corrispondere ad esempio a una frase del tipo "i topi temono i gatti", nel senso che ciascun topo teme ciascun gatto. Dato che gli enunciati di questo genere non sono infrequenti, è importante sapere come rappresentarli in OWL. In realtà non esiste una rappresentazione OWL semplice e diretta per gli enunciati del tipo 8.3. Mostreremo ora due rappresentazioni indirette che risolvono il problema.

Prodotto cartesiano con il testimone

Consideriamo la proprietà:

$\mathsf{R}: \mathsf{A} \longrightarrow \mathsf{B}$

Per assicurarci che ciascun oggetto appartenente all'estensione di A sia in relazione R con ciascun oggetto appartenente all'estensione di B, seguendo le linee del precedente sottoparagrafo definiamo (fig. 8.2):

$\mathsf{contaA}: \{\mathsf{w}\} \longrightarrow \mathsf{A}$

$\mathsf{contaB}: \{\mathsf{w}\} \longrightarrow \mathsf{B}$

$\mathsf{A} \sqsubseteq \exists \mathsf{contaA}^{-}$

$\mathsf{B} \sqsubseteq \exists \mathsf{contaB}^{-}$

Notiamo poi che la catena $\mathsf{contaA}^{-} \circ \mathsf{contaB}$ mette in relazione *ciascun oggetto dell'estensione di* A con *ciascun oggetto dell'estensione di* B. L'assioma di RBox

(8.4) $\mathsf{contaA}^{-} \circ \mathsf{contaB} \sqsubseteq \mathsf{R}$

garantisce allora che R sia un'estensione del prodotto cartesiano di A e B. D'altra parte R non può essere più grande del prodotto cartesiano di A per B, dato che il dominio e il codominio di R sono rispettivamente A e B. Ne segue che R rappresenta esattamente il prodotto cartesiano di A e B. Va osservato che a causa della 8.4 la proprietà R risulta composita, con tutte le limitazioni del caso (par. 5.8).

Prodotto cartesiano tramite le identità locali e la proprietà top

Un'altra rappresentazione indiretta di una classe si basa sulla funzione d'identità locale (par. 5.7). Siano A e B due classi con le relative funzioni identità:

$\mathsf{id}: \top \longrightarrow \top$	$\mathsf{idA} \sqsubseteq \mathsf{id}$	$\mathsf{idB} \sqsubseteq \mathsf{id}$
Fun(id)	$\mathsf{A} \equiv \exists \mathsf{idA}$	$\mathsf{B} \equiv \exists \mathsf{idB}$
Ref(id)		

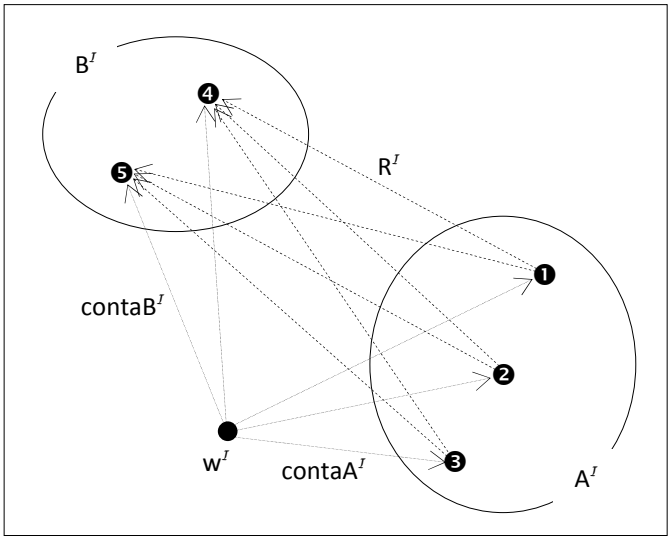


Figura 8.2. Definizione del prodotto cartesiano di due classi tramite testimone.

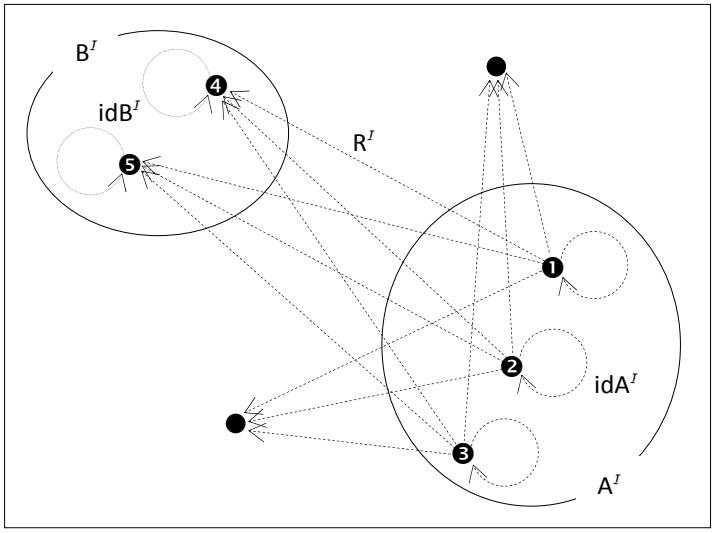


Figura 8.3. Definizione del prodotto cartesiano tramite le identità locali e la proprietà top.

A questo punto consideriamo l'assioma di RBox

(8.5) $\mathsf{idA} \circ \dagger \circ \mathsf{idB} \sqsubseteq \mathsf{R}$

La concatenazione di idA^I e $\dagger^I$ dà luogo alla relazione binaria che manda ciascun oggetto di A^I in ciascun oggetto dell'universo (le frecce tratteggiate nella fig. 8.3); l'ulteriore concatenazione con idB^I dà quindi luogo alla relazione binaria che manda ciascun oggetto di A^I in ciascun oggetto di B^I, come desiderato. Anche questa costruzione, come la precedente, definisce R come proprietà composita.

Può sorgere il sospetto che la 8.5 violi un vincolo non strutturale (par. 4.8), e precisamente il divieto di introdurre nella RBox dipendenze circolari fra proprietà. Infatti, per la 8.5 la proprietà R dipende da $\dagger$; ma si può pensare che, a sua volta, $\dagger$ dipenda da R, perché l'enunciato

(8.6) $\mathsf{R} \sqsubseteq \dagger$

è valido (cioè vero in ogni modello di OWL). Tuttavia, i vincoli non strutturali non sono formulati in base agli *enunciati validi*, bensì agli *enunciati effettivamente presenti nella RBox*. Pertanto, se l'enunciato 8.6 non viene esplicitamente inserito come assioma dell'ontologia, l'assioma 8.5 non viola alcun vincolo non strutturale[3].

8.5 La reificazione

Quando si rappresenta un frammento di realtà utilizzando una KB entrano in gioco entità di tipo diverso. Sul piano del linguaggio formale abbiamo classi, proprietà, attibuti e individui. Sul piano dei modelli, invece, abbiamo l'universo di tutti gli oggetti, i domini dei dati, gli insiemi di oggetti corrispondenti alle classi, le relazioni binarie fra oggetti corrispondenti alle proprietà, le relazioni fra oggetti e dati corrispondenti agli attributi, e gli oggetti (noti, cioè denotati da individui, o ignoti).

Nella maggior parte dei casi una comprensione intuitiva del frammento di realtà da rappresentare è sufficiente a determinare quali entità concrete vadano trattate come classi, quali come proprietà o attributi e quali come individui. In certi casi, tuttavia, risulta utile o addirittura necessario rappresentare entità in un modo, per così dire, indiretto: un'entità che si presenta intuitivamente come

[3] Occorre verificare che un enunciato del tipo di 8.6 non venga inserito automaticamente dall'editor utilizzato per definire l'ontologia. Al momento della pubblicazione di queste note, inoltre, il reasoner HermiT è in grado di trarre le inferenze consentite da 8.5, mentre altri reasoner, come Pellet e Fact++, non lo sono (vedi il par. 5.1, nota 12).

una classe potrà allora essere rappresentata da un individuo o da una proprietà, una proprietà come una classe, e così via. In questo paragrafo vedremo come utilizzare individui per rappresentare classi.

Individui come reificazioni di qualità

Un tipo di rappresentazione indiretta è costituito dalla *reificazione*, che prevede l'uso di individui per rappresentare classi. Come primo esempio di reificazione consideriamo un insieme di oggetti fisici monocolori, che possono essere rossi, verdi o blu. A proposito di un certo oggetto, a, in italiano possiamo dire che "a è rosso" oppure che "il colore di a è il rosso"; nel primo caso utilizziamo l'aggettivo "rosso" per esprimere una qualità dell'oggetto; nel secondo caso diciamo la stessa cosa utilizzando il sostantivo astratto "il rosso", che reifica tale qualità.

Questi differenti modi di esprimersi sono disponibili anche in una DL che preveda il costruttore *one-of*. Come prima opzione possiamo definire gli oggetti monocolori come classi con gli assiomi:

T1. $\mathsf{OggMonocol} \sqsubseteq \mathsf{OggFisico}$

T2. $\mathsf{OggMonocol} \equiv \mathsf{OggRosso} \sqcup \mathsf{OggVerde} \sqcup \mathsf{OggBlu}$

T3. $\mathit{DisCls}(\mathsf{OggRosso},\mathsf{OggVerde},\mathsf{OggBlu})$

In alternativa, possiamo *reificare i colori* trattandoli come degli individui. Avremo allora:

T4. $\mathsf{OggFisico} \sqcap \mathsf{Colore} \sqsubseteq \bot$

T5. $\mathsf{Colore} \equiv \{\mathsf{rosso},\mathsf{verde},\mathsf{blu}\}$

A1. $\neq(\mathsf{rosso},\mathsf{verde},\mathsf{blu})$

T6. $\mathsf{haColore}: \mathsf{OggFisico} \longrightarrow \mathsf{Colore}$

T7. $\mathsf{OggMonocol} \sqsubseteq =1\ \mathsf{haColore}$

A questo punto possiamo definire le classi degli oggetti rossi, verdi e blu come:

T8. $\mathsf{OggRosso} \equiv \mathsf{OggMonocol} \sqcap \exists \mathsf{haColore}.\{\mathsf{rosso}\}$

T9. $\mathsf{OggVerde} \equiv \mathsf{OggMonocol} \sqcap \exists \mathsf{haColore}.\{\mathsf{verde}\}$

T10. $\mathsf{OggBlu} \equiv \mathsf{OggMonocol} \sqcap \exists \mathsf{haColore}.\{\mathsf{blu}\}$

In questo modo T1 e T2 diventano dei teoremi, in quanto sono deducibili dagli altri assiomi. Anche gli assiomi di sottoclasse che congiuntamente equivalgono a T3, ovvero

$\mathsf{OggRosso} \sqcap \mathsf{OggVerde} \sqsubseteq \bot$

$\mathsf{OggRosso} \sqcap \mathsf{OggBlu} \sqsubseteq \bot$

$\mathsf{OggVerde} \sqcap \mathsf{OggBlu} \sqsubseteq \bot$

sono deducibili dagli altri assiomi. Introduciamo ora alcuni individui (diciamo a, b, c) come oggetti monocolori, utilizzando le asserzioni:

$\mathsf{OggMonocol}(\mathsf{a})$

$\mathsf{OggMonocol}(\mathsf{b})$

$\mathsf{OggMonocol}(\mathsf{c})$

Dobbiamo ricordarci che fra gli individui noti alla KB ci sono ora anche i tre individui che reificano i colori (rosso, verde, blu). Ciò si verifica facilmente con una interrogazione di retrieval che ci restituisce tutti gli individui noti:

$?- \top(*) \Rightarrow \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{rosso}, \mathsf{verde}, \mathsf{blu}\}$

Un esempio analogo è costituito dalla reificazione del genere (femminile o maschile) nella definizione di donne e uomini:

$\mathsf{Genere} \equiv \{\mathsf{f},\mathsf{m}\}$

$\mathsf{f} \neq \mathsf{m}$

$\mathsf{haGenere}: \mathsf{Persona} \longrightarrow \mathsf{Genere}$

$\mathsf{Persona} \sqcap \mathsf{Genere} \sqsubseteq \bot$

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

$\mathsf{Donna} \equiv \mathsf{haGenere} \ni \mathsf{f}$

$\mathsf{Uomo} \equiv \mathsf{haGenere} \ni \mathsf{m}$

Da questi assiomi è possibile dedurre, ad esempio, che

$\mathsf{Persona} \equiv \mathsf{Donna} \sqcup \mathsf{Uomo}$

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

A questo punto, asserire nell'ABox

Uomo(alberto)

Donna(barbara)

equivale ad asserire

haGenere(alberto,m)

haGenere(barbara,f)

Ancora una volta è necessario tenere presente che f e m sono due oggetti noti dell'universo, assieme con alberto, barbara e tutti gli altri oggetti denotati da individui. Avremo quindi:

$?- \top(*) \Rightarrow \{\mathsf{f}, \mathsf{m}, \mathsf{alberto}, \mathsf{barbara}, \ldots\}$

Reificazione di specie

Un altro caso di reificazione si ha quando si utilizzano individui per rappresentare delle *specie*. Quando diciamo "mi piace il Barolo" oppure "non porto mai la giacca", i termini "Barolo" e "giacca" non denotano una specifica bottiglia di vino o una specifica giacca, ma piuttosto una specie di vino e una specie d'indumento. Tali specie (Barolo, Sassicaia, ...; giacca, cravatta, ...) possono essere trattate come individui (barolo, sassicaia, ...; giacca, cravatta, ...) che reificano le specie corrispondenti e possono essere raggruppate in *generi*:

$\{\mathsf{barolo},\mathsf{sassicaia}\} \sqsubseteq \mathsf{Vino}$

$\{\mathsf{giacca},\mathsf{cravatta}\} \sqsubseteq \mathsf{Indumento}$

Qui l'uso della relazione di sottoclasse al posto dell'equivalenza suggerisce che l'elenco delle specie non esaurisce il genere corrispondente. Una volta definite le specie è possibile asserire che un certo individuo è di una certa specie; ad esempio, se abbiamo una classe BottigliaDiVino possiamo dire che l'individuo b025 è una bottiglia di Barolo:

$\mathsf{BottigliaDiVino} \sqsubseteq \mathsf{Bottiglia}$

$\mathsf{contiene}: \mathsf{Bottiglia} \longrightarrow \mathsf{Liquido}$

$\mathsf{Bottiglia} \sqsubseteq {\leq}1\ \mathsf{contiene}$

$\mathsf{Vino} \sqsubseteq \mathsf{Liquido}$

$\mathsf{BottigliaDiVino} \sqsubseteq \forall \mathsf{contiene}.\mathsf{Vino}$

contiene(b025,barolo)

Sempre utilizzando le specie reificate possiamo definire nuove classi, ad esempio la classe delle bottiglie di Barolo:

$\mathsf{BottigliaDiBarolo} \equiv \mathsf{contiene} \ni \mathsf{barolo}$

Come ulteriore esempio consideriamo lo stato civile di una persona (ad es. celibe, coniugato, separato). Se a è un individuo che denota la persona chiamata Antonio, scriviamo

(8.7) Celibe(a)

per dire che Antonio è celibe. Supponiamo ora di voler rappresentare il fatto che "celibe" è uno stato civile. È un errore abbastanza comune scrivere una definizione del tipo

(8.8!) $\mathsf{Celibe} \sqsubseteq \mathsf{StatoCivile}$

La 8.8!, tuttavia, è incompatibile con l'uso che abbiamo fatto precedentemente della classe Celibe: dalla 8.7 e dalla 8.8! otterremmo infatti

StatoCivile(a)

che significa che Antonio è uno stato civile! In effetti dire che "celibe" è uno stato civile non significa che l'insieme dei celibi sia *un sottoinsieme* degli stati civili, ma piuttosto che "celibe" è *un elemento* dell'insieme dei possibili stati civili. Poiché "celibe" è stato trattato come una classe, però, è impossibile in una DL asserire che "celibe" è a sua volta un elemento di un'altra classe: si tratterebbe di un'asserzione del secondo ordine, non esprimibile in FOL e a maggior ragione non esprimibile in una DL[4].

Per esprimere il fatto che "celibe" è uno stato civile occorre reificare la classe dei celibi, introducendo un individuo che funga da rappresentante della classe. Ecco una possibile soluzione:

StatoCivile ≡ {celibe,coniugato,vedovo,separato,divorziato}
≠(celibe,coniugato,vedovo,separato,divorziato)
haStatoCivile: Persona ⟶ StatoCivile
Persona ≡ =1 haStatoCivile
Celibe ≡ haStatoCivile ∋ celibe
Coniugato ≡ haStatoCivile ∋ coniugato
...

A questo punto è ancora possibile asserire 8.7, da cui si può ad esempio dedurre:

Persona(a)
haStatoCivile(a,celibe)
¬Coniugato(a)

Esempi più complessi di reificazione saranno introdotti in seguito.

8.6 Nomi propri

L'uso più intuitivo degli individui è per rappresentare oggetti concreti dell'universo. Ad esempio, possiamo utilizzare l'individuo antonio per rappresentare Antonio, una persona; per dire che Antonio è un uomo diremo allora

(8.9) Uomo(antonio).

Tuttavia, è spesso opportuno distinguere fra l'individuo che identifica univocamente una persona e il *nome proprio* della persona: "Antonio", ad esempio, è un nome di persona, ma in generale non è un identificatore univoco, perché ci sono molte persone che si chiamano allo stesso modo. Per distinguere fra individui e nomi propri notiamo che:

- ogni oggetto dell'universo può avere un nome proprio (non necessariamente univoco), perché possiamo assegnare un nome proprio a una persona, a un gatto, a una città, a un fiume, a una montagna, a una società, a un violino, a una spada e, almeno in linea di principio, a qualunque cosa;
- non tutti gli oggetti denotati da individui hanno anche un nome proprio: ad esempio potrei utilizzare tre diversi individui per denotare le tre seggiole del mio ufficio, ma è improbabile che queste seggiole abbiano un nome proprio nello stesso senso in cui hanno un nome una persona, un gatto o una città;
- i nomi propri sono stringhe, ovvero dati del dominio concreto string.

[4] L'asserzione che una classe è un'istanza di un'altra classe non è esprimibile in OWL 2 DL, ma è esprimibile in RDFS e in OWL 2 Full: se si segue questa strada, però, non è più possibile appoggiarsi ai servizi di ragionamento standard.

Per associare un nome proprio a un individuo possiamo allora introdurre un attributo (ovvero una data property), haNome, il cui dominio è $\top$ (in quanto assumiamo che ogni oggetto possa avere un nome proprio) e il cui codominio è string

haNome: $\top \longrightarrow$ string

Se ora a è il simbolo individuale che denota Antonio, l'asserzione 8.9 si rappresenta con le due asserzioni:

Uomo(a)

haNome(a,"Antonio"^^string)

dove "Antonio" è un letterale. Si noti che per trovare i simboli individuali di tutti gli individui che si chiamano Antonio è sufficiente risolvere il compito di retrieval

?– haNome ∋ "Antonio"^^string (∗)

In modo analogo si può specificare che tutte le persone hanno (almeno) un nome, e che tutte e sole le persone hanno esattamente un cognome (i cognomi multipli, come "Arrigoni Neri", possono essere trattati come stringhe uniche contenenti uno spazio):

haCognome: Persona $\longrightarrow$ string

Fun(haCognome)

Persona $\sqsubseteq \exists$haNome $\sqcap \exists$haCognome

8.7 Le descrizioni definite

Nel linguaggio ordinario abbiamo due dispositivi fondamentali per fare riferimento a oggetti: i nomi propri ("Antonio", "Milano", …) e le *descrizioni definite*. Utilizziamo descrizioni definite quando diciamo, ad esempio, "la madre di Antonio", "il capoluogo della Lombardia" e così via.

In OWL, ai nomi propri corrispondono gli individui (antonio, milano, …). Non esiste invece un costruttore corrispondente alle descrizioni definite, anche se i normali costruttori di classi consentono di approssimare questo concetto. Consideriamo ad esempio la descrizione definita "la cattedrale del capoluogo della Lombardia". Assumiamo che a ogni regione sia associata una città come capoluogo,

haCapoluogo: Regione $\longrightarrow$ Città

Regione $\sqsubseteq$ =1 haCapoluogo

e che in ogni città si trovi al massimo una cattedrale:

siTrovaIn: Edificio $\longrightarrow$ Località

Città $\sqsubseteq \leq$1 siTrovaIn$^-$.Cattedrale

(Attenzione: da questa ontologia *non* si deduce che tutte le città siano località né che tutte le cattedrali siano edifici! Se però si asserisce che una particolare cattedrale, ad ed. duomoDiMilano, si trova in una particolare città, ad esempio milano, allora si deduce che duomoDiMilano è un edificio e che milano è una località.) Ora possiamo costruire la classe di tutte le cattedrali che si trovano in almeno un capoluogo della Lombardia:

(8.10) Cattedrale $\sqcap \exists$siTrovaIn.(haCapoluogo$^-$ ∋ lombardia)

Dato che l'ontologia specifica che ogni regione ha al massimo una città come capoluogo e che ogni città ha al massimo una cattedrale, sappiamo che l'estensione di questa classe conterrà al massimo un oggetto. Supponiamo ora che la nostra ontologia contenga anche le asserzioni

haCapoluogo(lombardia,milano)

Cattedrale(duomoDiMilano)

siTrovaIn(duomoDiMilano,milano)

A questo punto, la classe 8.10 conterrà come unico individuo duomoDiMilano, come si può verificare con l'interrogazione:

?– Cattedrale $\sqcap \exists$siTrovaIn.(haCapoluogo$^-$ ∋ lombardia) $\Rightarrow$ {duomoDiMilano}

La classe 8.10 si comporta quindi, nel contesto dell'ontologia che abbiamo specificato, come una descrizione definita.

9. Le classificazioni

Classificare certe entità significa ripartirle in diverse classi. Ad esempio, in una biblioteca un manuale di analisi infinitesimale sarà classificato come testo di matematica, un trattato di storia medievale come testo di storia, e un saggio di storia della matematica sia come testo di matematica, sia come testo di storia.

Le classificazioni costituiscono spesso l'ossatura di un'ontologia. Per questo motivo è importante acquisire la capacità di definire le classificazioni in modo razionale.

9.1 Specializzazione e generalizzazione

Consideriamo due classi C e C' legate dalla relazione di sottoclasse

(9.1) $C \sqsubseteq C'$

Nel contesto dell'enunciato 9.1, la classe C si dice *specie* e la classe C' *genere*; si dice anche che C *specializza* C' e che C' *generalizza* di C. Va sottolineato che una classe non è una specie o un genere in assoluto, bensì soltanto nel contesto di una specifica relazione di sottoclasse. Ad esempio, se consideriamo l'enunciato

(9.2) $C' \sqsubseteq C''$

abbiamo che la classe C', genere nella 9.1, è ora specie nella 9.2.

Eredità

Certe qualità del genere vengono *ereditate* dalla specie per via di deduzione logica. Ad esempio, se vale la 9.1 e C' ha una proprietà funzionale R verso D,

(9.3) $R: C' \longrightarrow D$
$C' \sqsubseteq \leq 1\ R$

allora R è una proprietà funzionale anche per C, perché dalla 9.1 e dalla 9.3 si deduce che $C \sqsubseteq \leq 1\ R$.

Ad esempio, sia

$\mathsf{contiene}: \mathsf{Bottiglia} \longrightarrow \mathsf{Liquido}$
$\mathsf{Bottiglia} \sqsubseteq \leq 1\ \mathsf{contiene}$
$\mathsf{BottigliaDiVetro} \sqsubseteq \mathsf{Bottiglia}$

Da questi assiomi si deduce

$\mathsf{BottigliaDivetro} \sqsubseteq \leq 1\ \mathsf{contiene}$

Considerazioni analoghe valgono per vincoli di cardinalità diversi da ≤ 1. Va sottolineata una differenza fra le DL e altri formalismi che prevedono l'eredità di proprietà: nei diagrammi entità-relazioni delle basi di dati, ad esempio, l'eredità viene introdotta come elemento aggiuntivo nel metamodello degli schemi entità-relazioni. Nelle DL, invece, l'eredità discende direttamente della semantica degli enunciati e non vi sono meccanismi di ragionamento specifici per gestirla: l'eredità si ottiene semplicemente come effetto delle procedure di ragionamento standard.

Relazioni implicite di sottoclasse

Nel processo di sviluppo di un'ontologia è possibile definire una classe come specializzazione di una classe precedente (data C', definisco C con $C \sqsubseteq C'$) oppure come generalizzazione di una classe precedente (data C', definisco C'' con $C' \sqsubseteq C''$). In molti casi queste relazioni di sottoclasse derivano da definizioni che utilizzano l'intersezione o l'unione; ad esempio, se definiamo $\mathsf{Vivente}$ come generalizzazione di $\mathsf{Animale}$ e di $\mathsf{Vegetale}$,

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

stabiliamo implicitamente le due relazioni di sottoclasse

$\mathsf{Animale} \sqsubseteq \mathsf{Vivente}$

$\mathsf{Vegetale} \sqsubseteq \mathsf{Vivente}$

Analogamente, se definiamo Genitore come specializzazione di Persona,

$\mathsf{Genitore} \equiv \mathsf{Persona} \sqcap \exists \mathsf{genDi}$

stabiliamo implicitamente le due relazioni di sottoclasse

$\mathsf{Genitore} \sqsubseteq \mathsf{Persona}$

$\mathsf{Genitore} \sqsubseteq \exists \mathsf{genDi}$

Specificità

Ritorniamo all'esempio delle bottiglie di vino. Dagli assiomi presentati si deduce che una bottiglia di vino, come ogni bottiglia, ha al più un contenuto e che tale contenuto è un liquido; in che cosa consiste allora la *specificità* delle bottiglie di vino, intesa come differenza fra le bottiglie di vino e le bottiglie in generale?

A volte le ontologie non precisano la specificità di una classe rispetto al suo genere; ciò avviene ad esempio nelle pure tassonomie, di cui ci occuperemo più avanti. In altri casi, invece, si desidera rappresentare esplicitamente le specificità. Le differenze fra una specie e il suo genere sono essenzialmente di due tipi:

- *La specie possiede proprietà che il genere non possiede*. Ad esempio sia gli animali, sia i vegetali sono esseri viventi; di un animale si può poi dire che è a sangue freddo oppure a sangue caldo, mentre ciò non si può dire in generale degli esseri viventi, dato che questi comprendono anche i vegetali, che sono privi di sangue.
- *La specie restringe l'insieme dei possibili valori di una proprietà del genere*. Esempi: ogni persona ha un'età, che è un intero non negativo, e quindi ogni minorenne, in quanto persona, ha un'età, che però è compresa fra 0 e 17; ogni persona può avere zero o più fratelli, e un figlio unico è una persona che ha zero fratelli.

A volte poi una specie si definisce a partire da un genere combinando i due tipi di specificità.

9.2 Tassonomie

Una *tassonomia* è una classificazione in cui ogni classe è considerata come un tutto unico oppure è decomposta in un numero finito di classi a due a due disgiunte. Quando una classe A viene decomposta si ha:

$A \equiv B_1 \sqcup \ldots \sqcup B_n$

$\mathit{DisCls}(B_1, \ldots, B_n)$

ovvero, utilizzando la dichiarazione di unione disgiunta,

$\mathit{DisUni}(A, B_1, \ldots, B_n)$

Ad esempio, gli assiomi seguenti definiscono una possibile tassonomia degli strumenti musicali di un'orchestra classica:

$\mathit{DisUni}(\mathsf{Strumento,Arco,Pizzico,Tastiera,Fiato,Percussione})$

$\mathit{DisUni}(\mathsf{Arco,Violino,Viola,Violoncello,Contrabbasso})$

e così via. In realtà non è necessario imporre la disgiunzione delle classi a ogni livello della tassonomia: se le classi $F_1, \ldots, F_n$ sono tutte le foglie (ovvero le classi terminali) di una tassonomia, è sufficiente imporre

$\mathit{DisCls}(F_1, \ldots, F_n)$

La disgiunzione delle classi ai livelli superiori può ora essere dedotta, in quanto unendo collezioni disgiunte di classi a due a due disgiunte si ottengono ancora classi a due a due disgiunte. Non sempre, tuttavia, una classificazione (e in particolare una tassonomia) dà luogo in modo naturale a una struttura ad albero. Consideriamo due diverse classificazioni:

(9.4) Gli esseri viventi si dividono in animali e vegetali; gli animali si dividono in animali a sangue freddo e animali a sangue caldo; i vegetali si dividono in erbe ed alberi; a loro volta gli alberi si dividono in alberi decidui e alberi sempreverdi.

(9.5) Il vino si può classificare come rosso, rosé o bianco; poi come fermo, mosso o spumante; poi come dolce, amabile, aromatico o secco.

La classificazione 9.4 assume in modo naturale una forma ad albero, che si può rappresentare come:

Vivente
- Animale
  - SangueFreddo
  - SangueCaldo
- Vegetale
  - Erba
  - Albero
    - Deciduo
    - Sempreverde

La classificazione 9.5, invece, non dà luogo in modo naturale a un albero: anche se è concettualmente possibile tradurla in un albero (vedi la fig. 9.1), in questa struttura c'è qualcosa di innaturalmente 'verboso'. Perché? Una prima considerazione è che la classificazione 9.3 dà luogo a livelli che non possono essere scambiati fra loro: ad esempio, non possiamo dividere gli esseri viventi *prima* in erbe e alberi e *poi* in animali e vegetali, perché la divisione in erbe e alberi presuppone che ci troviamo già nel mondo dei vegetali. Al contrario, i livelli della classificazione del vino possono essere scambiati a piacimento: possiamo classificare i vini *prima* per colore e *poi* rispetto ad altre caratteristiche, oppure *prima* rispetto alle altre caratteristiche e *poi* per colore: la cosa è del tutto indifferente.

Una seconda considerazione è che la classificazione del vino dà luogo a tutte le 36 combinazioni possibili dei valori di tre caratteristiche. Ciò non avviene invece nella classificazione degli esseri viventi: ad esempio, non esistono animali sempreverdi. La differenza fra le due classificazioni può quindi essere riassunta come segue:

- la classificazione 9.3 divide ogni classe in sottoclassi di oggetti che possiedono *proprietà distinte*: ad esempio, le proprietà degli animali sono diverse dalle proprietà delle piante; in questo caso è naturale descrivere la classificazione come un albero, perché non è possibile scambiare fra loro i livelli della classificazione, né prendere in considerazione tutte le possibili combinazioni di valori delle proprietà;
- al contrario la classificazione 9.4 divide ogni classe in base a proprietà comuni a tutte le sottoclassi e logicamente indipendenti fra loro; in questo caso la struttura ad albero è innaturale, perché l'ordine dei livelli è arbitrario e ogni combinazione di valori delle proprietà ha senso.

Ci resta da stabilire un modo naturale per definire le classificazioni del secondo tipo. In questi casi è preferibile assegnare tutte le proprietà alla classe radice della classificazione; ad esempio:

$\mathsf{Colore} \equiv \{\mathsf{rosso},\mathsf{rosé},\mathsf{bianco}\}$

$\mathsf{haColore}\colon \mathsf{Vino} \longrightarrow \mathsf{Colore}$

$\mathsf{Vino} \sqsubseteq {=}1\ \mathsf{haColore}$

e così via per gli altri attributi. Occorre poi ricordarsi di imporre la diversità fra i valori degli attributi:

$\neq(\mathsf{rosso},\mathsf{rosé},\mathsf{bianco},\mathsf{fermo},\mathsf{mosso},\mathsf{spumante},\mathsf{dolce},\mathsf{amabile},\mathsf{aromatico},\mathsf{secco})$

Ogni tipo terminale della classificazione è ora un punto nello spazio discreto a tre dimensioni determinato dalle tre proprietà.

- Vino
  - Rosso
    - RossoFermo
      - RossoFermoDolce
      - RossoFermoAmabile
      - RossoFermoAromatico
      - RossoFermoSecco
    - RossoMosso
      - RossoMossoDolce
      - RossoMossoAmabile
      - RossoMossoAromatico
      - RossoMossoSecco
    - RossoSpumante
      - RossoSpumanteDolce
      - RossoSpumanteAmabile
      - RossoSpumanteAromatico
      - RossoSpumanteSecco
  - Rosé
    - RoséFermo
      - RoséFermoDolce
      - RoséFermoAmabile
      - RoséFermoAromatico
      - RoséFermoSecco
    - RoséMosso
      - RoséMossoDolce
      - RoséMossoAmabile
      - RoséMossoAromatico
      - RoséMossoSecco
    - RoséSpumante
      - RoséSpumanteDolce
      - RoséSpumanteAmabile
      - RoséSpumanteAromatico
      - RoséSpumanteSecco
  - Bianco
    - BiancoFermo
      - BiancoFermoDolce
      - BiancoFermoAmabile
      - BiancoFermoAromatico
      - BiancoFermoSecco
    - BiancoMosso
      - BiancoMossoDolce
      - BiancoMossoAmabile
      - BiancoMossoAromatico
      - BiancoMossoSecco
    - BiancoSpumante
      - BiancoSpumanteDolce
      - BiancoSpumanteAmabile
      - BiancoSpumanteAromatico
      - BiancoSpumanteSecco

Figura 12.1. Una classificazione ad albero del vino.

10. Uso delle proprietà

10.1 Qualità intrinseche e qualità relazionali

Fra le qualità di un oggetto dobbiamo distinguere nettamente le qualità per così dire *intrinseche* dalle qualità *relazionali*. Consideriamo ad esempio le due classi Donna e Genitore: la prima considera una qualità intrinseca di certi oggetti (essere una persona di genere femminile), la seconda una qualità relazionale (essere genitore di almeno un oggetto).

Nella definizione di un'ontologia è importante distinguere fra qualità intrinseche e qualità relazionali. Supponiamo ad esempio di voler dire che ogni automobile ha esattamente un motore; lo possiamo fare nel modo seguente

(10.1) $\mathsf{haMotore}: \mathsf{Auto} \longrightarrow \mathsf{Motore}$

$\mathsf{Auto} \sqsubseteq {=}1\ \mathsf{haMotore}$

Se ora vogliamo aggiungere che un'automobile ha esattamente un proprietario può sembrarci opportuno replicare la stessa struttura, affermando:

(10.2!) $\mathsf{ha\ Proprietario}: \mathsf{Auto} \longrightarrow \mathsf{Proprietario}$

$\mathsf{Auto} \sqsubseteq {=}1\ \mathsf{haProprietario}$

Mentre la definzione 10.1 è corretta, la 10.2! non lo è. La ragione è semplice:

- la proprietà haMotore mette in relazione due oggetti che, indipendentemente l'uno dall'altro (e quindi 'intrinsecamente'), possono essere considerati un'automobile e un motore;
- la proprietà haProprietario, invece, mette in relazione un oggetto, che è intrinsecamente un'automobile, con un altro oggetto, che è un proprietario non intrinsecamente, ma soltanto in quanto possiede qualcosa.

In altre parole, mentre un motore è tale anche in assenza di un'automobile di cui faccia parte (si pensi ad esempio a un motore prelevato da un'auto ormai rottamata), un proprietario di automobile non resta tale anche in assenza di un'automobile che gli appartenga[5]: essere un motore è una qualità intrinseca, essere un proprietario è invece una qualità relazionale. Ciò che possiamo dire è che un proprietario d'automobile è una persona che intrattiene una particolare relazione (di proprietà) con un'automobile; dobbiamo quindi correggere la 10.2! nel modo seguente:

(10.2) $\mathsf{haProprietario}: \mathsf{Auto} \longrightarrow \mathsf{Persona}$

$\mathsf{Auto} \sqsubseteq {=}1\ \mathsf{haProprietario}$

Se ci torna utile possiamo poi definire la classe dei proprietari di un'automobile come segue:

$\mathsf{Proprietario} \equiv \exists \mathsf{haProprietario}^{-}$

Vediamo ora un altro esempio: supponiamo di voler specificare che un'associazione ha un certo numero di soci come membri. Un errore abbastanza comune consiste nel definire la proprietà

$\mathsf{haMembro}: \mathsf{Associazione} \longrightarrow \mathsf{Socio}$

Anche qui il problema è che il concetto di socio non esprime una qualità intrinseca, bensì una qualità relazionale: una persona non è un socio intrinsecamente, così come può essere un uomo o una donna, ma solo in relazione a un'associazione di cui è membro. La definizione corretta è quindi:

$\mathsf{haMembro}: \mathsf{Associazione} \longrightarrow \mathsf{Persona}$

$\mathsf{Socio} \equiv \exists \mathsf{haMembro}^{-}$

o alternativamente

$\mathsf{membroDi}: \mathsf{Persona} \longrightarrow \mathsf{Associazione}$

$\mathsf{Socio} \equiv \exists \mathsf{membroDi}$

[5] Il lettore può aver incontrato un problema analogo nel campo dei diagrammi di classe di UML, quando si distingue fra *aggregazione debole* e *aggregazione forte*.

Analogamente, sarebbe scorretto definire la proprietà “genitore di” come relazione fra “un genitore” e “un figlio”,

(!) genDi: Genitore $\longrightarrow$ Figlio

perché “genitore” e “figlio” sono qualità non intrinseche, bensì relazionali. La proprietà va quindi definita fra due persone,

genDi: Persona $\longrightarrow$ Persona

dopodiché possiamo definire i concetti di genitore e di figlio nel modo seguente:

Genitore $\equiv \exists$genDi
Figlio $\equiv \exists$genDi$^-$

Naturalmente, se l’ontologia specifica che ogni persona ha esattamente due genitori,

Persona $\sqsubseteq =2$ genDi$^-$

la classe Figlio risulterà equivalente alla classe Persona.

10.2 Sottoproprietà

Supponiamo di voler specificare che un’associazione ha un consiglio direttivo, costituito da cinque soci, uno dei quali è anche il presidente dell’associazione. La prima scelta che facciamo è di rappresentare esplicitamente la relazione binaria “socio di” con la proprietà socioDi; ciò comporta l’introduzione delle classi Persona e Associazione come dominio e codominio della proprietà:

socioDi: Persona $\longrightarrow$ Associazione

Questo assioma, apparentemente innocuo, presuppone in realtà una scelta precisa, cioè la reificazione delle associazioni; in altre parole, ogni associazione sarà considerata come un oggetto dell’universo. Dato che ovviamente persone e associazioni sono oggetti di tipo diverso, specifichiamo che le due classi sono disgiunte:

DisCla(Persona,Associazione)

Ora dobbiamo dire che ogni associazione ha un consiglio direttivo, costituito da cinque soci. Qui abbiamo due scelte: possiamo reificare anche i consigli direttivi (considerandoli come degli ulteriori oggetti dell’universo) oppure no. In generale, è meglio evitare di introdurre reificazioni quando non sia necessario; consideriamo quindi un consiglio direttivo non come un’entità reificata, ma come una relazione binaria “membro del Cd di”, rappresentata dalla proprietà

membroCdDi: Persona $\longrightarrow$ Associazione

Specifichiamo poi che un membro del Cd dev’essere socio dell’associazione con l’assioma di sottoproprietà

membroCdDi $\sqsubseteq$ socioDi

Infine specifichiamo che il Cd è costituito da 5 membri:

Associazione $\sqsubseteq =5$ membroCdDi$^-$

Seguendo lo stesso procedimento possiamo specificare che un membro del Cd è il presidente dell’associazione, senza reificare il concetto di presidente:

presidenteDi: Persona $\longrightarrow$ Associazione
presidenteDi $\sqsubseteq$ membroCdDi
Associazione $\sqsubseteq =1$ presidenteDi$^-$

La figura 10.1 rappresenta graficamente *un possibile modello* dell’ontologia. La figura 10.2, invece, rappresenta sotto forma di diagramma *gli assiomi* dell’ontologia (la freccia tratteggiata con punta triangolare rappresenta una relazione disottoproprietà); diagrammi di questo tipo sono molto utili, in particolare come aiuto alla composizione di interrogazioni complesse (vedi il par. 10.2).

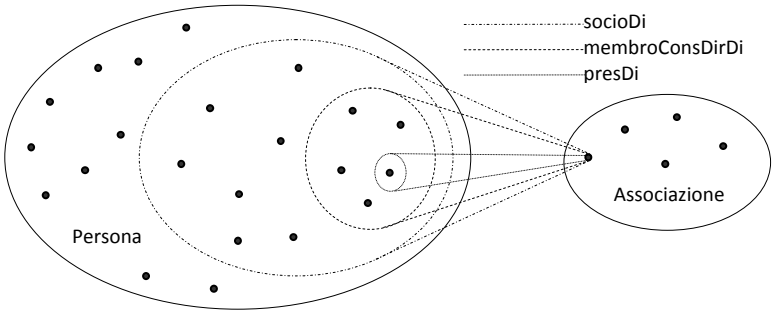


Figura 10.1. Un modello della prima ontologia delle associazioni.

Non è sempre possibile evitare certe reificazioni. Supponiamo ad esempio di voler aggiungere all'ontologia il fatto che il Cd valuta le richieste d'iscrizione dei nuovi soci. Sembra intuitivo introdurre una relazione "valuta richiesta" fra il Cd e una persone; ma questo ci obbliga a reificare il Cd, introducendo una nuova classe, Cd. I primi assiomi della nuova ontologia (che sostituisce la precedente) possono essere (fig. 10.3):

$\mathsf{socioDi}: \mathsf{Persona} \longrightarrow \mathsf{Associazione}$

$\mathsf{cdDi}: \mathsf{Cd} \longrightarrow \mathsf{Associazione}$

$\mathsf{membroCdDi}: \mathsf{Persona} \longrightarrow \mathsf{Cd}$

$\mathsf{Associazione} \sqsubseteq {=}1\ \mathsf{cdDi}^-$

$\mathsf{Cd} \sqsubseteq {=}1\ \mathsf{cdDi} \sqcap {=}5\ \mathsf{membroCdDi}^-$

Ora possiamo aggiungere la proprietà

$\mathsf{valutaRichiesta}: \mathsf{Cd} \longrightarrow \mathsf{Persona}$

Non abbiamo ancora specificato che i membri del Cd di un'associazione devono essere soci *della stessa* associazione. Lo possiamo fare utilizzando un assioma di sottoproprietà con una catena:

$\mathsf{membroCdDi} \circ \mathsf{cdDi} \sqsubseteq \mathsf{socioDi}$

Ora dobbiamo aggiungere il presidente. Per evitare di reificare il concetto di presidente potremmo specificare la proprietà

$\mathsf{presidenteDi}: \mathsf{Persona} \longrightarrow \mathsf{Associazione}$

$\mathsf{Associazione} \sqsubseteq {=}1\ \mathsf{presidenteDi}^-$

In questo modo, però, non abbiamo specificato che il presidente è un membro del Cd (e di conseguenza un socio). Purtroppo non possiamo farlo facilmente; infatti un assioma di sottoproprietà come

$\mathsf{presidenteDi} \circ \mathsf{cdDi}^- \sqsubseteq \mathsf{membroCdDi}$

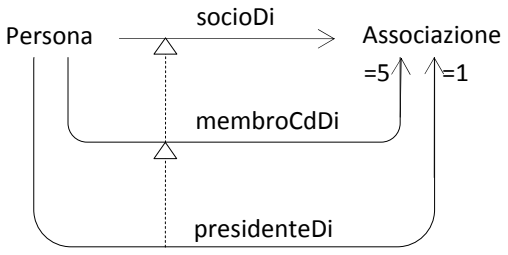


Figura 10.2. Diagramma della prima ontologia delle associazioni.

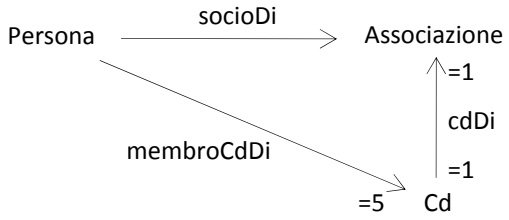


Figura 10.3. Diagramma della seconda ontologia delle associazioni.

renderebbe composità la proprietà membroCdDi, il che sarebbe incompatibile con la restrizione di cardinalità che abbiamo precedentemente imposto a membroCdDi. Se il contesto dell'applicazione lo consente, possiamo aggirare il problema considerando il presidente non come presidente *dell'associazione*, ma come *presidente del Cd*:

presidenteDi: Persona $\longrightarrow$ Cd

presidenteDi $\sqsubseteq$ membroCdDi

Cd $\sqsubseteq$ =1 presidenteDi$^-$

Questo ci consentirebbe comunque di recuperare l'individuo che è presidente dell'associazione a con l'interrogazione di retrieval:

?– $\exists$presidenteDi.(cdDi ∋ a) (∗)

10.3 Concatenazione di proprietà

OWL consente un limitato utilizzo della concatenazione delle proprietà. Come sappiamo, le catene di proprietà si possono utilizzare soltanto all'interno di assiomi di sottoproprietà e rispettando certi vincoli strutturali (par. 5.6) e non strutturali (par 5.8).

In molti casi in cui si vorrebbe poter utilizzare una catena di proprietà in modo più generale; supponiamo ad esempio di disporre delle proprietà che rappresentano le relazioni "genitore di" e "sorella di":

genDi: Persona $\longrightarrow$ Persona

sorDi: Donna $\longrightarrow$ Persona

In OWL non è possibile definire una proprietà ziaDi come concatenazione di sorDi e genDi, ovvero come (fig 10.4)

(10.3!) ziaDi $\equiv$ sorDi $\circ$ genDi

Tuttavia è possibile approssimare la 10.3! specificando la relazione di sottoproprietà

(10.3) sorDi $\circ$ genDi $\sqsubseteq$ ziaDi

che rappresenta il fatto che essere sorella di un genitore di una persona è *condizione sufficiente* per essere zia di quella persona; ciò che non riusciamo a esprimere in OWL è il fatto che tale condizione sia anche *necessaria*.

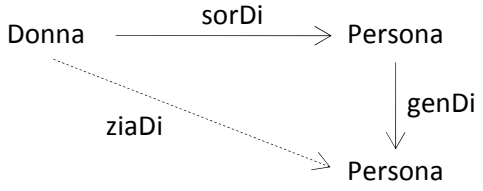


Figura 10.4. Concatenazione di proprietà (non esprimibile completamente in OWL).

È però possibile definire (specificando condizioni sia necessarie, sia sufficienti) la *classe* di tutte le zie,

$\mathsf{Zia} \equiv \exists \mathsf{sorDi}.(\exists \mathsf{genDi})$

nonché la *classe* di tutte le zie degli individui appartenenti a una classe prefissata C,

$\mathsf{ZiaDiC} \equiv \exists \mathsf{sorDi}.(\exists \mathsf{genDi}.\mathsf{C})$

e in particolare la *classe* di tutti le zie di un individuo prefissato *a*,

$\mathsf{ZiaDia} \equiv \exists \mathsf{sorDi}.(\mathsf{genDi} \ni \mathsf{a})$

10.4 Chiavi

Come è noto, nel gergo delle basi di dati una chiave è un insieme minimale di proprietà i cui valori identificano univocamente una tupla. Nel campo delle ontologie si può difinire un concetto analogo. In un'ontologia è detta *chiave* un insieme non vuoto (non necessariamente minimale) di proprietà e/o attributi associati a una classe atomica *A*, tale che l'insieme dei valori associati da tali proprietà e attributi a un individuo *a* della classe *A* identifica univocamente l'individuo. Vediamo alcuni esempi.

Le chiavi costituite da una singola proprietà (*object property*) si possono facilmente esprimere in OWL: è sufficiente specificare che la proprietà è inversamente funzionale. Tuttavia nella maggior parte dei casi le chiavi sono costituite da attributi (ovvero da *data property*), e per gli attributi non è possibile specificare la funzionalità inversa[6]. Consideriamo, ad esempio, un'ontologia che specifica che ogni persona ha esattamente un numero di codice fiscale (stringa), esattamente un nome ed esattamente un cognome:

$\mathsf{haCF}: \mathsf{Persona} \longrightarrow_D \mathsf{string}$
$\mathsf{haNome}: \mathsf{Persona} \longrightarrow_D \mathsf{string}$
$\mathsf{haCognome}: \mathsf{Persona} \longrightarrow_D \mathsf{string}$
$\mathsf{Persona} \sqsubseteq {=}1\ \mathsf{haCF} \sqcap {=}1\ \mathsf{haNome} \sqcap {=}1\ \mathsf{haCognome}$

Per specificare che il codice fiscale costituisce chiave dovremmo asserire che due o più persone con lo stesso codice fiscale sono lo stesso oggetto, ovvero che la proprietà di avere un codice fiscale è inversamente funzionale: ma questo non è possibile.

Uutilizzando le strutture di OWL viste fin qui, inoltre, non è possibile specificare chiavi multiple. Consideriamo ad esempio le classi Esame, Studente e Corso e cerchiamo di rappresentare gli esami superati da ciascuno studente con voto sufficiente e il relativo voto. Possiamo specificare quanto segue (vedi la fig. 10.5):

$\mathsf{superatoDa}: \mathsf{Esame} \longrightarrow \mathsf{Studente}$
$\mathsf{esameDi}: \mathsf{Esame} \longrightarrow \mathsf{Corso}$
$\mathsf{haVoto}: \mathsf{Esame} \longrightarrow \mathsf{Voto}$
$\mathsf{Esame} \sqsubseteq {=}1\ \mathsf{superatoDa} \sqcap {=}1\ \mathsf{esameDi} \sqcap {=}1\ \mathsf{haVoto}$

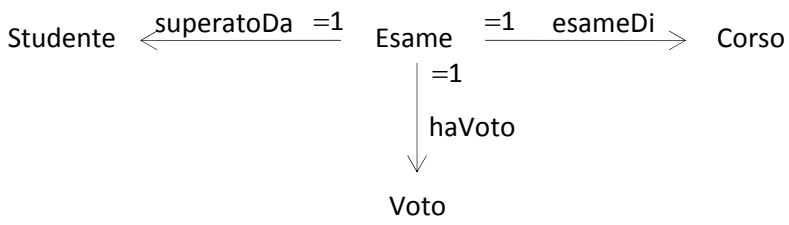


Figura 10.5. Rappresentazione grafica di alcuni assiomi di TBox.

[6] La ragione è che una proprietà è funzionale inversa quando la sua inversa è funzionale: ma un attributo non ammette un inverso.

A ciascuna coppia studente-corso può corrispondere al massimo un esame superato, con il relativo voto; tuttavia, utilizzando le strutture di OWL che conosciamo non è possibile specificare che la coppia di proprietà sostenutoDa ed esameDi costituisce una chiave. Si noti che nessuna delle due proprietà è, presa da sola, inversamente funzionale: infatti ciascuno studente supera molti esami (di corsi diversi) e a ciascun corso corrispondono molti esami superati (da studenti diversi).

Il vincolo in discussione può però essere rappresentato con un assioma speciale di OWL:

(10.4) *HasKey*(Esame; superatoDa,esameDi;)

In generale la sintassi di un enunciato *HasKey* è la seguente:

(10.5) *HasKey*(A; R_1,...,R_n; P_1,...,P_m)

dove A è una classe atomica; R_1, ..., R_n sono proprietà; P_1, ..., P_m sono attributi; e $n + m > 0$ (ovvero, gli insiemi delle proprietà e degli attributi non sono ambedue vuoti). In prima approssimazione la semantica della 10.5 si può esprimere come segue:

(10.6) $M \vDash$ *HasKey*(A; R_1,...,R_n; P_1,...,P_m) sse

per ogni scelta di oggetti u, v in A^I, di oggetti x_1, ..., x_n e di dati d_1, ..., d_m:
se si ha: $\langle u,x_1\rangle \in R_1^I$ e ... e $\langle u,x_n\rangle \in R_n^I$; $\langle u,d_1\rangle \in P_1^I$ e ... e $\langle u,d_m\rangle \in P_m^I$;
$\langle v,x_1\rangle \in R_1^I$ e ... e $\langle v,x_n\rangle \in R_n^I$; $\langle v,d_1\rangle \in P_1^I$ e ... e $\langle v,d_m\rangle \in P_m^I$;
allora: $u = v$.

Nel caso del nostro esempio la stessa condizione può essere espressa, in modo più facilmente comprensibile, con una formula di FOL:

$$\forall u \forall v \forall x_1 \forall x_2\ (\text{Esame}(u) \wedge \text{Esame}(v) \wedge \text{superatoDa}(u,x_1) \wedge \text{superatoDa}(v,x_1) \wedge \text{esameDi}(u,x_2) \wedge \text{esameDi}(v,x_2) \rightarrow u = v)$$

Questa formula FOL non è traducibile in OWL, ma può essere approssimata in un'estensione del linguaggio, costituita dalle *regole SWRL*, di cui però non ci occupiamo in queste note.

Perché abbiamo detto che la 10.6 esprime la semantica della 10.5 "in prima approssimazione"? Il fatto è che l'aggiunta ad OWL di un costrutto con una simile semantica renderebbe il linguaggio indecidibile; per questa ragione si adotta per *HasKey* una semantica più debole, corrispondente a una modalità di ragionamento detta *sicura*, che non comporta indecidibilità. Questa modalità prevede che gli assiomi *HasKey* siano presi in considerazione solo in presenza di asserzioni del tipo $R_1(a,b_1)$, ..., $R_n(a,b_n)$, $P_1(a,d_1)$, ..., $P_m(a,d_m)$ (direttamente inserite in ABox o dedotte tramite ragionamento) che concernono specifici individui a, b_1, ..., b_n e specifici dati d_1, ..., d_m. Ciò comporta che il ragionamento sicuro basato su assiomi *HasKey* possa dedurre soltanto asserzioni della forma $a = b$, dove a e b sono due individui presenti nell'ontologia.

Un ulteriore esempio chiarirà che cosa *non* è possibile dedurre da un assioma *HasKey*. Supponiamo di aggiungere alla precedente ontologia degli esami alcuni assiomi che precisano l'esistenza di esattamente 20 studenti ed esattamente 10 corsi. Dato che la coppia di proprietà superatoDa-esameDi costituisce una chiave per Esame, è chiaro che potranno esistere al massimo 200 esami distinti. Questo fatto, tuttavia, non può essere dedotto da un ragionatore che utilizzi l'assioma 10.4 in modalità sicura, perché il ragionamento coinvolge oggetti dell'universo non rappresentati da specifici individui.

10.5 Algebra delle proprietà

Il linguaggio OWL possiede le risorse necessarie per un'algebra booleana delle classi: in altre parole, sono disponibili la classe top, $\top$, la classe bottom, $\bot$, e i costruttori di complementazione, unione e intersezione. Al contrario, OWL non possiede le risorse per un'algebra booleana delle proprietà: sono sì presenti la proprietà top, $\dot{\top}$, e la proprietà bottom, $\dot{\bot}$, ma non è possibile eseguire complementazioni, unioni o intersezioni di proprietà. Nel seguito vediamo quali elementi di algebra delle proprietà siano praticabili.

Strutture relazionali

Nella teoria degli insiemi e nell'algebra si studiano diversi tipi di strutture relazionali, costituite da un insieme su cui è definita una relazione *R* dotata di certe qualità formali. Fra le più note strutture relazionali ci sono i preordini (*R* riflessiva e transitiva), gli ordini parziali (*R* riflessiva, antisimmetrica e transitiva), gli ordini parziali stretti (*R* irriflessiva, asimmetrica e transitiva) e le equivalenze (*R* riflessiva, simmetrica e transitiva). In generale queste strutture *non* possono essere completamente specificate in OWL perché ciò porterebbe a violare dei vincoli non strutturali.

Iniziamo dai preordini. Se *R* è una proprietà definita sull'intero universo,

$R: \top \longrightarrow \top$

è in effetti possibile specificare che *R* è riflessiva e transitiva:

Ref(*R*)

Tra(*R*)

Se invece *R* è definita su una classe arbitraria *C* diversa dall'intero universo,

$R: C \longrightarrow C$

ciò non è possibile. Dovremmo infatti specificare che

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

Tra(*R*)

ma in questo modo violeremmo un vincolo non strutturale, perché *R* è composita in quanto transitiva e le proprietà composite non possono comparire nelle restrizioni di riflessività.

Per quanto riguarda gli ordini parziali stretti, per motivi analoghi non è possibile specificare l'asimmetria di una proprietà transitiva. Per gli ordini parziali non stretti la situazione non è migliore, dato che OWL non prevede la specifica dell'antisimmetria.

Infine, con le relazioni di equivalenza le cose vanno analogamente ai preordini: mentre è possibile specificare che una proprietà definita su tutto l'universo,

$R: \top \longrightarrow \top$

è riflessiva, simmetrica e transitiva,

Ref(*R*)

Sym(*R*)

Tra(*R*)

la stessa cosa non è possibile se la proprietà *R* è definita su una classe arbitraria *C*, ancora una volta perché *R* è composita in quanto transitiva e le proprietà composite non possono comparire nelle restrizioni di riflessività locale. Spesso, però, è possibile *approssimare* le specifiche non esprimibili in OWL, nel senso che ora preciseremo nel prossimo sottoparagrafo.

Chiusura di proprietà

Se *R* è una relazione binaria la *chiusura transitiva* R^{Tra} *di R* è definita come *la più piccola estensione di R che sia transitiva*. Più precisamente valgono le seguenti condizioni:

- R^{Tra} è un'estensione di *R*: $R \subseteq R^{Tra}$;
- R^{Tra} è transitiva: $R^{Tra} \circ R^{Tra} \subseteq R^{Tra}$;
- R^{Tra} è la più piccola relazione binaria dotata delle due qualità precedenti, nel senso che: per ogni relazione binaria *R'*, se $R \subseteq R'$ e $R' \circ R' \subseteq R'$, allora $R^{Tra} \subseteq R'$.

Un esempio è la proprietà "antenato di", definibile come la chiusura transitiva della relazione "genitore di". Se la proprietà genDi rappresenta la relazione fra un genitore e un figlio (o figlia), allora la proprietà

(!) $\mathsf{antDi} \equiv \mathsf{genDi}^{Tra}$

rappresenta la relazione fra un antenato e un discendente. In OWL non è possibile definire la chiusura transitiva di una proprietà, ma è possibile approssimare questo concetto in modo soddisfacente per molte applicazioni. Supponiamo che un'ontologia contenga la proprietà

genDi: Persona ⟶ Persona

e che l'insieme delle coppie genitore-figlio sia definita da un elenco di asserzioni nell'ABox, ad esempio (fig. 10.6*a*)

genDi(a,b)
genDi(a,c)
genDi(b,d)
genDi(d,e)

Definiamo ora:

(10.7) antDi: Persona ⟶ Persona
genDi ⊑ antDi
Tra(antDi)

In questo modo assicuriamo che antDi sia un'*estensione transitiva* di genDi (ma non necessariamente la più piccola estensione di questo genere!). Se ora utilizziamo questa KB per calcolare gli antenati di uno specifico individuo, troviamo che in effetti viene calcolata la più piccola estensione transitiva di genDi, cioè appunto la chiusura transitiva di genDi; ad esempio:

?– antDi ∋ e (∗) ⇒ {a,b,d}

Infatti gli individui per cui è possibile dimostrare che appartengono alla classe antDi∋e sono gli individui che denotano oggetti appartenenti ad antDi∋e *in tutti i modelli* della KB: in questo modo si ottiene appunto *la più piccola* estensione transitiva di genDi. Tuttavia, la KB ammette anche modelli in cui la proprietà antDi non è la chiusura transitiva di genDi; supponiamo ad esempio di aggiungere all'ABox l'asserzione (fig. 10.6*b*)

antDi(c,f)

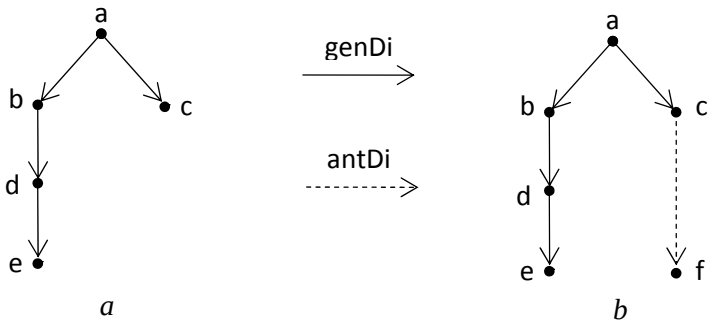
Se antDi fosse effettivamente la chiusura transitiva di genDi dovremmo essere in grado di dedurre che c è un genitore, perchè non si può essere antenati di qualcuno senza essere anche genitori di qualcuno. Dovremmo quindi avere:

?– ∃genDi (∗) ⇒ {a,b,c,d}

Invece la risposta all'interrogazione di retrieval è:

?– ∃genDi (∗) ⇒ {a,b,d}

Il motivo è semplice: gli assiomi 10.7 definiscono antDi come *una qualsiasi* estensione transitiva di genDi, non come *la più piccola* estensione del genere. Ciò significa che, per quanto ne sa la nostra KB, si può anche essere antenati di qualcuno senza essere genitori di nessuno.



a *b*

Figura 10.6. Genitori e antenati rappresentati nell'ABox.

Un altro modo di vedere la situazione consiste nell'osservare che gli assiomi 10.7 stabiliscono condizioni *necessarie*, ma non *sufficienti*, per l'appartenenza di una coppia ordinata di oggetti alla chiusura transitiva di genDi. Infatti per appartenere alla chiusura transitiva di genDi, appartenere a un'estensione transitiva di genDi è sicuramente necessario; ma non è sufficiente, perché la chiusura transitiva di genDi è la sua estensione transitiva *minima*. Quindi quando diciamo che antDi (definito da 10.7) *approssima* la chiusura transitiva di genDi, intendiamo dire che antDi rispetta condizioni necessarie, ma non sufficienti, per la chiusura transitiva di genDi. Naturalmente, il fatto di disporre di un'approssimazione di un concetto non significa automaticamente che tale approssimazione sia adeguata: ciò dipende dalla specifica applicazione e non può essere stabilito in generale. Ad esempio, essere Uomo è una condizione necessaria per essere Padre, e in questo senso possiamo dire che la classe Uomo approssima la classe Padre; tuttavia, è difficile immaginare un'applicazione concreta in cui un'approssimazione del genere sia accettabile.

Ritorniamo al concetto di chiusura. In generale, se *R* è una relazione binaria, e **F** è una qualità formale di relazioni binarie (come l'essere transitiva o l'essere una relazione di equivalenza) la *chiusura* $R^{\mathbf{F}}$ *di R* rispetto a **F** è definita come *la più piccola estensione di R, se esiste, che goda della qualità* **F**. Perché esista *la più piccola* estensione di *R* che goda della qualità **F** occorre che si verifichino due condizioni:

- deve esistere almeno un'estensione di *R* che goda della qualità **F**;
- tale qualità si deve conservare rispetto alle intersezioni arbitrarie: in altre parole, l'intersezione insiemistica di una collezione arbitraria (finita o infinita) di relazioni binarie che godano della proprietà **F** dev'essere a sua volta una relazione binaria che gode della proprietà **F**; la più piccola estensione di *R* che goda della qualità **F** coincide allora con l'intersezione insiemistica di tutte le estensioni di *R* che godono della qualità **F**.

Ad esempio non è possibile parlare di chiusura irriflessiva di una generica relazione *R*, perché se per qualche *a* vale *R*(*a*,*a*), nessuna estensione di *R* sarà irriflessiva. È invece possibile parlare di chiusura localmente riflessiva di una relazione *R*, perché ogni relazione ammette estensioni localmente riflessive e inoltre la riflessività locale si conserva rispetto alle intersezioni arbitrarie. Lo stesso vale, ad esempio, per la chiusura simmetrica; come esempio concreto consideriamo la classe delle persone, alcune delle quali sono state presentate ad altre persone:

presentatoA: Persona $\longrightarrow$ Persona

In condizioni normali è lecito assumere che ciascuna persona conosca se stessa, tutte le persone a cui è stata presentata, nonché tutte le persone che le sono state presentate. Se è così, conosce è un'estensione localmente riflessiva e simmetrica di presentatoA:

conosce: Persona $\longrightarrow$ Persona

Persona $\sqsubseteq \exists$conosce.Self

presentatoA $\sqsubseteq$ conosce

Sym(conosce)

10.6 Il tutto e le parti

Come abbiamo visto, in un'ontologia le classi formano, rispetto alla relazione di sottoclasse, una gerarchia in cui ogni specie si correla a uno o più generi. Oltre alla relazione di sottoclasse esistono altre relazioni rispetto a cui le classi possono formare una gerarchia. Fra queste è spesso importante la relazione che lega un tutto alle sue parti: ad esempio, un libro è costituito da una copertina e da un insieme di pagine; una colonna classica è composta da un fusto, una base e un capitello, a sua volta costituito da un àbaco e da un èchino; un ateneo è costituita da un certo numero di facoltà; e così via. La logica descrittiva non ha costrutti specifici per denotare la relazione parte-tutto: tale relazione andrà quindi rappresentata con una o più proprietà definite caso per caso[7].

[7] In altre parole, nelle DL la relazione parte-tutto, al contrario ad esempio della relazione di sottoclasse, non è già prevista dal linguaggio formale, ma va definita nel modello concettuale (in TBox e RBox).

Ad esempio, se ciò che ci interessa è la relazione fra le parti di un oggetto fisico e l'oggetto complessivo possiamo definire una proprietà

(10.8) parteDi: OggFisico ⟶ OggFisico

con la sua inversa,

(10.9) haParte ≡ parteDi⁻

Una parte di una parte di un tutto è ancora una parte del tutto: ad esempio, un abaco, in quanto parte di un capitello, è anche parte di una colonna. La relazione parteDi è quindi transitiva:

Tra(parteDi)

Non occorre dichiarare la transitività di haParte, perchè l'inversa di una relazione transitiva è a sua volta transitiva. Queste relazioni possono essere utilizzate per specificare la struttura di determinati oggetti fisici; ad esempio[8]:

Colonna ⊑ ∃haParte.Fusto ⊓ ∃haParte.Base ⊓ ∃haParte.Capitello
Capitello ⊑ ∃haParte.Abaco ⊓ ∃haParte.Echino

In questo modo si definisce una gerarchia delle parti (le parti di un tutto, le parti delle parti e così via), spesso denominata *gerarchia meronìmica* (dal greco *méros*, parte, e *ónyma*, nome). (Si noti che si può parlare di "gerarchia" solo in presenza di una relazione transitiva, come la relazione di sottoclasse o la proprietà parteDi.)

Al contrario della relazione di sottoclasse, la proprietà parteDi non comporta in generale un fenomeno di eredità (par. 10.1). Alcune qualità specifiche, tuttavia, vengono effettivamente ereditate anche lungo una gerarchia meronimica; ad esempio, in molte applicazioni sarà corretto assumere che se un oggetto fisico si trova in un luogo, anche le sue parti si trovano nello stesso luogo. L'ereditarietà della posizione dal tutto alle parti si potrà allora specificare come segue:

siTrovaIn: OggFisico ⟶ Luogo
parteDi ∘ siTrovaIn ⊑ siTrovaIn

Occorre però prestare attenzione a due fatti. Primo, certe proprietà, come ad esempio il peso di un oggetto, non sono ereditate dalle parti di un tutto; secondo, certe proprietà possono essere ereditate nel contesto di un'applicazione e non di un'altra: ad esempio, se il tutto di cui si parla è un servizio di bicchieri, le parti del tutto si possono trovare in luoghi diversi.

È importante non confondere la relazione parte-tutto con la relazione di sottoclasse. Ad esempio, una poesia è un tipo di testo (e quindi la classe Poesia sarà sottoclasse della classe Testo), ma un verso di una poesia non è un tipo di poesia, bensì una parte della poesia, e quindi fra un verso e una poesia sussisterà la proprietà parteDi. Infine, va notato che la proprietà parteDi, in quanto transitiva, è composita, e pertanto le espressioni che la contengono sono soggette ai relativi vincoli non strutturali (par. 4.8). In particolare:

- non è possibile specificare che ogni oggetto fisico è parte di se stesso, perché una proprietà compositá non può comparire in una restrizione di riflessività locale; per ragioni analoghe non è neppure possibile specificare il contrario, ovvero definire parteDi come irriflessiva e asimmetrica;
- su parteDi non è possibile esprimere restrizioni di cardinalità.

La seconda limitazione può essere problematica: ad esempio, potrebbe essere importante specificare che una colonna ha esattamente un capitello, che ogni università ha almeno due facoltà e così via. In un caso del genere è necessario abbandonare l'idea di definire una proprietà generale haParte (necessariamente transitiva), ricorrendo a proprietà più specifiche (non transitive).

[8] Sarebbe più corretto dire che una colonna ha *esattamente un* fusto, *esattamente una* base e così via; le restrizioni di cardinalità, tuttavia, non possono essere imposte a proprietà transitive (par. 4.8).

Ad esempio possiamo definire una colonna come:

$\mathsf{haFusto}: \mathsf{Colonna} \longrightarrow \mathsf{Fusto}$
$\mathsf{haBase}: \mathsf{Colonna} \longrightarrow \mathsf{Base}$
$\mathsf{haCapitello}: \mathsf{Colonna} \longrightarrow \mathsf{Capitello}$
$\mathsf{Colonna} \sqsubseteq {=}1\ \mathsf{haFusto} \sqcap {=}1\ \mathsf{haBase} \sqcap {=}1\ \mathsf{haCapitello}$

È anche possibile specificare che fusto, base e capitello sono parti della colonna, nel modo seguente:

$\mathsf{haFusto} \sqsubseteq \mathsf{haParte}$
$\mathsf{haBase} \sqsubseteq \mathsf{haParte}$
$\mathsf{haCapitello} \sqsubseteq \mathsf{haParte}$

Si noti che da questi assiomi si deriva, grazie agli assiomi 10.8 e 10.9, che colonne, fusti, basi e capitelli sono oggetti fisici.

11. Dal linguaggio ordinario ad OWL

Nelle applicazioni reali, l'ingegnere della conoscenza si trova spesso a dover specificare in OWL un 'mondo' inizialmente delimitato in modo vago e poco o per nulla strutturato. In genere il primo passo consiste nel descrivere questo mondo utilizzando il linguaggio ordinario (italiano, inglese, ...). Occorre però fare molta attenzione, perché le descrizioni in linguaggio ordinario possono celare delle ambiguità. Inoltre può capitare che queste descrizioni non siano rappresentabili con assiomi OWL; in questo caso è spesso possibile approssimare le descrizioni con assiomi più deboli, che catturano almeno in parte ciò che si desidera specificare. Naturalmente il fatto che un'approssimazione sia o non sia utile dipende dalla specifica applicazione di un'ontologia; di seguito vediamo alcuni esempi.

Condizioni necessarie e sufficienti

Gli assiomi di TBox possono esprimere condizioni necessarie, condizioni sufficienti, o condizioni necessarie e sufficienti. Ad esempio, l'assioma

$C \sqsubseteq C'$

dice che essere un C è condizione sufficiente per essere un C' oppure, se si preferisce, che essere un C' è condizione necessaria per essere un C. L'assioma

$C \equiv C'$

dice invece che essere un C è condizione necessaria e sufficiente per essere un C' (e viceversa).

Occorre fare molta attenzione alla differenza fra questi diversi tipi di condizioni, che a volte sono in qualche modo 'nascoste' nelle descrizioni espresse in linguaggio ordinario. Esempi:

- "Il rosso, il verde e il blu sono colori". Ciò significa che essere il rosso, il verde o il blu è condizione sufficiente, ma non necessaria, per essere un colore:
  $\{\mathsf{rosso},\mathsf{verde},\mathsf{blu}\} \sqsubseteq \mathsf{Colore}$
- "Il rosso, il verde e il blu sono *i* colori". Ciò significa che essere il rosso, il verde o il blu è condizione necessaria e sufficiente per essere un colore:
  $\{\mathsf{rosso},\mathsf{verde},\mathsf{blu}\} \equiv \mathsf{Colore}$
- "Gatti e tigri sono felini". Ciò significa che essere un gatto o essere una tigre è condizione sufficiente per essere un felino:
  $\mathsf{Gatto} \sqcup \mathsf{Tigre} \sqsubseteq \mathsf{Felino}$
- "Gli oggetti naturali sono gli animali, i vegetali e i minerali". Ciò significa che essere un animale, un vegetale o un minerale è condizione necessaria e sufficiente per essere un oggetto naturale:
  $\mathsf{Animale} \sqcup \mathsf{Vegetale} \sqcup \mathsf{Minerale} \equiv \mathsf{OggettoNaturale}$

Le cose stanno diversamente per gli assiomi di RBox contenenti catene di proprietà. In tal caso è possibile rappresentare condizioni sufficienti, ma non condizioni necessarie; ad esempio, la specifica

"lo zio è il fratello di un genitore" significa che essere fratello di un genitore è condizione necessaria e sufficiente per essere uno zio. Come abbiamo già rilevato, tuttavia, in OWL è esprimibile solo la condizione sufficiente:

$\mathsf{fratDi} \circ \mathsf{genDi} \sqsubseteq \mathsf{zioDi}$

Questo è un esempio di approssimazione: una condizione necessaria e sufficiente viene qui approssimata con una condizione soltanto sufficiente.

Occorre fare molta attenzione all'uso della congiunzione italiana "se": a rigore, il "se" dell'italiano corrisponde al *condizionale semplice* "$p \rightarrow q$" (se p, allora q) della logica proposizionale e all'operatore $\sqsubseteq$ delle DL. A volte, però, nel linguaggio di tutti i giorni si usa il "se" per esprimere il *bicondizionale* "$p \leftrightarrow q$" (p se, e solo se, q) della logica proposizionale o l'operatore $\equiv$ delle DL. Ad esempio, se Andrea dice a Barbara

(11.1) "se piove non vengo in montagna con te"

è probabile che intenda dire

"non vengo in montagna con te se, e solo se, piove",

il che implica che se non piove Andrea andrà in montagna con Barbara: ma questa implicazione non vale se interpretiamo il "se" come semplice "se ... allora ...". Quando si specifica in italiano il mondo di un'applicazione, sarà quindi importante utilizzare la congiunzione "se" soltanto come equivalente del condizionale semplice, e dire esplicitamente "... se, e solo se, ..." quando si vuole esprimere un bicondizionale. Infine si noti che una frase della forma "p se q" significa "$q \rightarrow p$" e non "$p \rightarrow q$"! Ad esempio, presa alla lettera la 11.1 è equivalente a:

"non vengo in montagna con te se piove".

Disgiunzione di proprietà

La relazione "genitore di" è l'unione disgiunta delle relazioni "madre di" e "padre di". Come abbiamo già osservato, in OWL non è possibile rappresentare l'unione disgiunta di due o più proprietà. Possiamo però approssimare la descrizione precedente asserendo che le proprietà madreDi e padreDi sono disgiunte:

DisPro(madreDi,padreDi)

Ciò che non riusciamo a dire in OWL è che "genitore di" è l'unione di "madre di" e "padre di": ovvero, che gli unici modi per essere genitore di qualcuno è essere sua madre o suo padre.

OWL non prevede operazioni booleane sulle proprietà; quindi negli assiomi non è possibile negare, congiungere o disgiungere proprietà. È però possibile ottenere una forma di disgiunzione implicita in un caso particolare, perché un assioma della forma (non consentita)

(!) $R_1 \sqcup R_2 \sqsubseteq S$

equivale ai due assiomi

$R_1 \sqsubseteq S$
$R_2 \sqsubseteq S$

Ad esempio:

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

Vincoli di cardinalità

Alle proprietà composite non è possibile applicare vincoli di cardinalità. Consideriamo l'esempio seguente: "ogni docente valuta gli esami dei suoi corsi". L'ontologia relativa si può specificare come segue

$\mathsf{docenteDi}: \mathsf{Persona} \longrightarrow \mathsf{Corso}$
$\mathsf{esameDi}: \mathsf{Esame} \longrightarrow \mathsf{Corso}$

valuta: Persona $\longrightarrow$ Esame

docenteDi $\circ$ esameDi^{-} $\sqsubseteq$ valuta

A questo punto la proprietà valuta risulta composita. Se ora tentiamo di dire che ogni esame è valutato esattamente da una persona,

(!) Esame $\sqsubseteq$ =1 valuta^{-}

ci troviamo a violare un vincolo non strutturale. Possiamo però approssimare questa specifica asserendo che ogni esame è valutato da almeno una persona:

Esame $\sqsubseteq \exists$valuta^{-}

Chiusura di una proprietà

Come abbiamo visto nel paragrafo 10.5 non è possibile specificare la chiusura $R^{\mathbf{F}}$ di una proprietà R rispetto a una qualità formale **F**. Se la qualità **F** è esprimibile in OWL, tuttavia, è possibile dire che una relazione R' è una **F**-estensione di R (ovvero, un'estensione di R che gode della qualità **F**):

$R \sqsubseteq R'$

F(R')

Specifiche parametriche

È possibile specificare in OWL la classe di tutte le persone che hanno più figlie che figli maschi? Intuitivamente dovremmo scrivere qualcosa come

(!) (=n genDi.Femmina $\sqcap$ =m genDi.Maschio) | n > m

Come sappiamo, tuttavia, le restrizioni di cardinalità richiedono che n ed m siano specificati come costanti, e non è quindi possibile trattarli come parametri variabili. Se sapessimo che ogni persona può avere al massimo, poniamo, quattro figli, potremmo cavarcela con la disgiunzione di tutti i casi possibili:

($\exists$genDi.Femmina $\sqcap$ $\neg\exists$genDi.Maschio)

$\sqcup$ ($\geq$2 genDi.Femmina $\sqcap$ $\leq$1 genDi.Maschio)

$\sqcup$ ($\geq$3 genDi.Femmina $\sqcap$ $\leq$2 genDi.Maschio)

$\sqcup$ (=4 genDi.Femmina $\sqcap$ $\leq$3 genDi.Maschio)

Ma in generale il problema non è risolubile, perché in OWL non possono essere rappresentate specifiche parametriche. Occorre però interpretare questa limitazione nel modo corretto: in molti casi, infatti, una specifica in linguaggio ordinario può contenere dei 'parametri', che però vengono eliminati quando si passa alla rappresentazione OWL. Ad esempio, possiamo specificare che "*x* è nonna di *y* se *x* è madre di *z* e *z* è genitore di *y*"; ma anche se abbiamo utilizzato i 'parametri' *x*, *y* e *z*, la specifica è facilmente rappresentabile in OWL come:

madreDi $\circ$ genDi $\sqsubseteq$ nonnaDi

In altre parole, non è detto che una specifica espressa in forma parametrica nel linguaggio ordinario richieda una rappresentazione parametrica in OWL.

12. Come interrogare una base di conoscenze

"Interrogare" una base di conoscenze significa verificare se un enunciato segue o non segue logicamente dalla base di conoscenze; come sappiamo, questa verifica può essere effettuata invocando un servizio di ragionamento. In questo paragrafo ci occupiamo della differenza fra l'interrogazione di una base di dati e di una base di conoscenze e presentiamo un procedimento che aiuta a costruire interrogazioni complesse.

12.1 Interrogazioni e assunzioni di chiusura

L'interrogazione di una base di conoscenza è certamente simile all'interrogazione di una base di dati; questa somiglianza è però ingannevole, perché l'assenza di assunzioni chiusura nelle DL può portare a risultati inattesi.

Consideriamo ad esempio la KB costituita dagli assiomi seguenti:

T1. genDi: Persona ⟶ Persona
A1. ≠(a,b,c,d,e)
A2. genDi(a,b)
A3. genDi(a,c)
A4. genDi(d,e)

Ecco alcune interrogazioni con i loro risultati:

– *chi ha almeno un figlio?* ?– ∃genDi(∗) ⇒ {a,d}
– *chi ha almeno due figli?* ?– ≥2 genDi(∗) ⇒ {a}
– *chi ha almeno tre figli?* ?– ≥3 genDi(∗) ⇒ { }

Consideriamo ora l'interrogazione seguente:

– *chi ha al massimo un figlio?* ?– ≤1 genDi(∗) ⇒ { }

Forse ci aspettavamo come risposta {b,c,d,e}; cerchiamo quindi di capire perché il servizio di retrieval non trova alcun individuo che soddisfi la classe ≤1 genDi. Per capire che cosa succede è necessario tener conto delle assunzioni di chiusura. Se le asserzioni A2–A4 fossero considerate come tuple di una tabella in una base di dati relazionale, l'interrogazione "*chi ha al massimo un figlio?*" dovrebbe effettivamente restituire come risultato {b,c,d,e}; infatti, grazie alle assunzioni dell'unicità dei nomi, della chiusura dell'universo e del mondo chiuso, le asserzioni A2–A4 determinano *un unico modello* (vedi il par. 6.2), il cui universo è costituito da cinque oggetti *a*, *b*, *c*, *d* ed *e* (rispettivamente corrispondenti agli individui a, b, c, d, e), con l'estensione di genDi data dall'insieme di coppie ordinate $\{\langle a,b\rangle,\langle a,c\rangle,\langle d,e\rangle\}$.

Quando si interroga una base di dati, la risposta è data sulla base di *ciò che è vero nell'unico modello descritto dalla base di dati*; quindi è giusto che la risposta all'interrogazione "*chi ha al più un figlio?*" sia {b,c,d,e}. Le interrogazioni di basi di conoscenze, invece, sono interpretate in modo diverso: tanto per cominciare non vengono ipotizzate le assunzioni di chiusura; inoltre la risposta a un'interrogazione è data non sulla base di *ciò che è vero in uno specifico modello*, bensì sulla base di *ciò che segue logicamente dalla base di conoscenze*, ovvero di *ciò che è vero in tutti i possibili modelli della base di conoscenze*. In assenza di assunzioni di chiusura, è compatibile con ciò che è stato affermato che ciascun individuo (ovvero a, b, c, d, e) denoti un oggetto *con più di un figlio*; infatti, l'assenza di un'asserzione dall'ABox non equivale ad assumerne la falsità. Pertanto nessun enunciato della forma ≤1 genDi(a_k), dove a_k è un individuo qualsiasi, segue logicamente dalla base di conoscenze.

Ci possiamo chiedere a questo punto perché le prime tre interrogazioni abbiano dato lo stesso risultato che ci saremmo aspettati da una base di dati. Il motivo è semplice: in questi casi le assunzioni di chiusura non fanno differenza; infatti le asserzioni A1–A4 impongono che in ogni modello della base di conoscenze gli individui a e d abbiano almeno un figlio e che l'individuo a abbia almeno due figli.

L'esempio analizzato mostra che è molto importante distinguere le interrogazioni alle basi di conoscenza dalle interrogazioni alle basi di dati: mentre le seconde, come abbiamo visto, riguardano ciò che vale *in un singolo modello* (determinato dalla base di dati grazie alle assunzioni di chiusura), le prime riguardano ciò che vale *in ogni modello* della base di conoscenze.

A questo punto ci si può chiedere se non sarebbe opportuno, per semplificare le applicazioni, basarsi sulle assunzioni di chiusura anche nell'ambito delle DL. Il fatto, però, è che le assunzioni di chiusura sono accettabili solo in condizioni di *conoscenza completa*; più in particolare:

- l'assunzione di unicità dei nomi (UNA) presuppone che si sia certi che due individui distinti non denotino mai lo stesso oggetto (per fare un esempio, siamo sicuri che giuseppe e beppe sono sempre e comunque persone diverse);
- l'assunzione di chiusura dell'universo (UCA) presuppone che sia stato assegnato un individuo a ciascun oggetto dell'universo;
- l'assunzione del mondo chiuso (CWA) presuppone che si conoscano tutti i fatti che sussistono nel mondo di cui ci si occupa.

Come si vede si tratta di ipotesi molto forti, che non possono essere assunte una volta per tutte; rinunciando a tali assunzioni, le DL risultano compatibili con le situazioni in cui la conoscenza del mondo è incompleta. È pur vero che in alcuni casi si potrebbe sapere a priori che le conoscenze rappresentate in una base di conoscenze sono complete, almeno sotto certi aspetti. È quindi corretto porsi il problema di come le assunzioni di chiusura possano essere importate in una base di conoscenza quando ciò sia ragionevole.

UNA

Imporre l'unicità dei nomi in un'ontologia non è difficile. L'opzione più ovvia è di impostare il ragionatore, se questo lo consente, in modo tale che la UNA sia rispettata. Il difetto di questa soluzione è che la stessa ontologia potrebbe produrre risposte diverse qualora fosse utilizzata con un ragionatore che non prevede questa opzione.

La soluzione più corretta consiste nell'asserire esplicitamente la diversità degli individui, utilizzando asserzioni del tipo

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

Questo risulta più agevole se si è provveduto a specificare tutte le relazioni di disgiunzione fra le classi con assiomi del tipo $DisCla(A_1, ..., A_n)$. Infatti non è necessario specificare la diversità di individui appartenenti a classi disgiunte.

UCA

L'assunzione di chiusura dell'universo non gioca solitamente un ruolo importante. Qualora fosse necessario 'chiudere' l'universo, si può sempre asserire che

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

Gli assiomi di questo tipo vanno però utilizzati con cautela, perché impediscono all'ontologia di essere importata in un'altra ontologia, o di importare un'altra ontologia. Se si vuole imporre una condizione del genere è meglio operare la chiusura non dell'intero universo, ma di una classe specifica, come ad esempio in:

$$\mathsf{MyObject} \equiv \{a_1, ..., a_n\}$$

In questo modo l'ontologia si mantiene compatibile con l'esistenza di altri oggetti, appartenenti all'universo Δ, anche se non appartengono all'estensione della classe MyObject.

CWA

Nelle interrogazioni (e in particolare nelle interrogazioni negative: vedi il prossimo paragrafo) l'assunzione del mondo chiuso è la più problematica. Al momento non esiste la possibilità di utilizzare un reasoner con l'assunzione CWA: qualora un'interrogazione la presupponga, occorrerà quindi regolarsi caso per caso. Un esempio è presentato nel prossimo paragrafo.

12.2 Interrogazioni complesse

Consideriamo la seguente ontologia:

- ogni corso è offerto da una facoltà (ovvero ingegneria, architettura, disegno industriale):

$\mathsf{Facoltà} \equiv \{\mathsf{ing},\mathsf{arc},\mathsf{dis}\}$	$\neq(\mathsf{ing},\mathsf{arc},\mathsf{dis})$
$\mathsf{offertoDa}: \mathsf{Corso} \longrightarrow \mathsf{Facoltà}$	$\mathsf{Corso} \sqsubseteq =1\ \mathsf{offertoDa}$

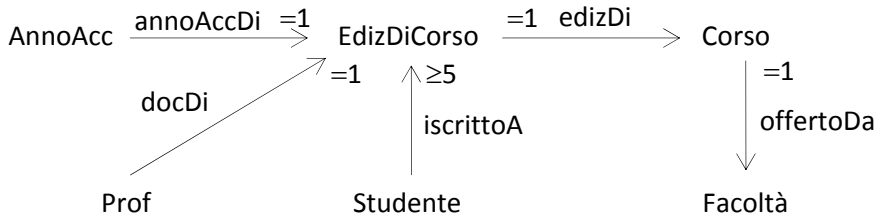


Figura 12.1. Diagramma di un'ontologia.

– ogni edizione di un corso ha luogo in un anno accademico:
  {2002-03,2003-04,2004-05,2005-06,2006-07} ⊑ AnnoAcc
  edizDi: EdizDiCorso ⟶ Corso
  annoAccDi: AnnoAcc ⟶ EdizDiCorso
  EdizDiCorso ⊑ =1 edizDi ⊓ =1 annoAccDi⁻
– ogni edizione di un corso ha un professore come docente e almeno cinque studenti iscritti:
  docDi: Prof ⟶ EdizDiCorso
  iscrittoA: Studente ⟶ EdizDiCorso
  EdizDiCorso ⊑ =1 docDi⁻ ⊓ ≥5 iscrittoA⁻

Percorrere il diagramma delle proprietà

La figura 12.1 mostra un diagramma dell'ontologia, che sfrutteremo per costruire interrogazioni di retrieval e di instance check. Iniziamo con l'interrogazione di retrieval

"quali corsi hanno avuto come docente il professor p07160 nell'anno accademico 2004-05?"

In un'interrogazione di retrieval, quindi della forma ?– *C*(∗), l'unica difficoltà consiste nella costruzione della classe *C*. Si può procedere nel modo seguente:

– tracciare un percorso, lungo le frecce del diagramma, dai dati d'uscita ai dati d'ingresso dell'interrogazione (fig. 12.2);
– costruire l'interrogazione seguendo il percorso dai dati d'uscita ai dati d'ingresso, tenendo conto dell'orientamento delle frecce: quando la freccia della proprietà *R* è diretta come la freccia del percorso, nell'interrogazione appare *R*; in caso contrario appare *R*⁻.

Nel nostro caso abbiamo:

?– ∃edizDi.(annoAccDi⁻ ϶ 2004-05} ⊓ docDi⁻ ϶ p07160}) (∗)

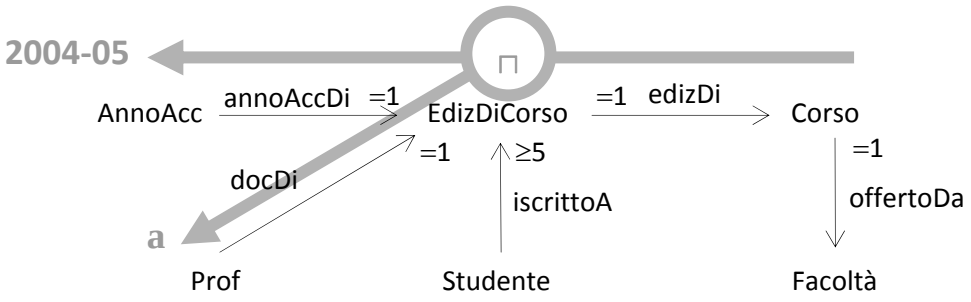


Figura 12.2. Costruzione di un'interrogazione.

Ecco altri esempi:

- quali corsi della facoltà di ingegneria hanno avuto come docente il professor p07160 nell'anno accademico 2004-05?

  ?– offertoDa ∋ ing ⊓ ∃edizDi⁻.(annoAccDi⁻ ∋ 2004-05} ⊓ docDi⁻p07160}) (∗)

- il professor p07160 è stato docente di almeno due edizioni di corsi della facoltà di architettura?

  ?– ≥2 docDi.(∃edizDi.(offertoDa ∋ arc)) (p07160)

- lo studente matr666666 è stato iscritto ad almeno un'edizione del corso c2045 tenuto dal professor p07160?

  ?– ∃iscrittoA.(edizDi ∋ c2045 ⊓ docDi⁻ ∋ p07160) (matr666666)

Interrogazioni negative

Particolare attenzione va dedicata alle interrogazioni che contengono componenti negative. Sempre rispetto alla precedente ontologia, consideriamo l'interrogazione:

"quali studenti non sono iscritti ad alcuna edizione del corso c2045?"

Il primo problema che affrontiamo è la costruzione dell'espressione che costituisce l'interrogazione. Un'interrogazione del tipo

?– ¬∃iscrittoA.(edizDi ∋ c2045) (∗)

peraltro equivalente a

?– ∀iscrittoA.¬(edizDi ∋ c2045) (∗)

sarebbe scorretta perché troverebbe tutti gli oggetti dell'universo, *che siano o non siano studenti*, che non sono iscritti ad alcuna edizione del corso c2045. Per ovviare a questo problema riscriviamo l'interrogazione come

?– Studente ⊓ ¬∃iscrittoA.(edizDi ∋ c2045) (∗)

A questo punto, però, ci imbattiamo in un altro problema, già analizzato in precedenza: il ragionatore ci restituirà l'elenco degli studenti per cui *è possibile provare* che *non sono iscritti* ad alcuna edizione del corso c2045. Probabilmente, però, ciò cui siamo interessati è piuttosto l'elenco degli studenti per cui *non è possibile provare* che *siano iscritti* a un'edizione del corso c2045. Ma qui entra in gioco di l'assunzione del mondo chiuso, CWA. Come abbiamo osservato nel parafrafo precedente dobbiamo escogitare un modo per aggirare il problema; costruiremo quindi la risposta desiderata nel modo seguente:

- con una prima interrogazione calcoliamo l'insieme di tutti gli individui che sono studenti, legandolo a una variabile *S*, che va considerata come una struttura dati esterna all'ontologia:

  ?– Studente (∗) ⇒ *S*

- con una seconda interrogazione calcoliamo l'insieme di tutti gli individui iscritti a un'edizione di c2045, legandolo alla variabile *I*, anch'essa considerata come una struttura dati esterna all'ontologia:

  ?– ∃iscrittoA.(edizDi ∋ c2045) (∗) ⇒ *I*

- infine, sempre esternamente all'ontologia, calcoliamo la differenza fra i due insiemi, che costituisce la risposta desiderata:

  $S \setminus I$

Questo metodo va al di là dell'uso di un ragionatore e richiede l'implementazione di una specifica applicazione software che acceda all'ontologia. A questo scopo è possibile sfruttare apposite librerie, come Jena (disponibile però soltanto per OWL 1 DL), o utilizzare le interfacce di programmazione previste per l'accesso a ontologie OWL 2 DL (OWL API, `http://owlapi.sourceforge.net/`).

Riferimenti bibliografici

Alesso, H.P., & C.F. Smith (2008). *Thinking on the Web: Berners Lee, Gödel and Turing*, Wiley Blackwell.

Allemang, D,. & J. Hendler (2008). *Semantic Web for the working ontologist: Effective modeling in RDFS and OWL*, Morgan Kaufmann.

Antoniou, G., & F. van Harmelen, (2008). *Semantic Web primer (2nd edition)*, MIT Press.

Berners-Lee, T., J. Hendler & O. Lassila (2001). The semantic web, *Scientific American*, May 2001, `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`.

BlackBurn, P. & M. Tzakova (1998). Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, 24 (1-4), 23–49. Online: `http://citeseer.ist.psu.edu/325862.html`

Breitman, K.K., M.A. Casanova & W. Truszkowski (2007). *Semantic Web: Concepts, technologies and applications*, Springer.

Ceri, S., G. Gottlob & L. Tanca (1989). What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1 (1), 146–166. Online: `http://portal.acm.org/citation.cfm?id=627357`.

Church, A. (1936). A Note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1, 40–41.

Davies, J., R. Studer & P. Warren (2006). *Semantic Web technologies: Trends and research in ontology-based systems*, Wiley Blackwell.

Davies, J., R. Studer, & P. Warren (2006). *Semantic Web technologies: Trends and research in ontology-based systems*, Wiley Blackwell.

Donini, F.M., M. Lenzerini, D. Nardi, A. Schaerf, & W. Nutt (1998). An epistemic operator for description logics. *Artificial Intelligence*, 100 (1-2), 225–274.

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

Fellbaum, C. (1998). *WordNet: An electronical lexical database*, MIT Press.

Fensel, D. (2005). *Spinning the Semantic Web: Bringing the World Wide Web to its full potential*, MIT Press.

Fensel, D. (2005). *Spinning the Semantic Web: Bringing the World Wide Web to its full potential*, MIT Press.

Hitzler, P., M. Krötzsch, & S. Rudolph (2009). *Foundations of Semantic Web Technologies*, Chapman & Hall/CRC.

Horridge, M., N. Drummond, J. Goodwin, A. Rector, R. Stevens, & H.H. Wang1 (2006). The Manchester OWL syntax. *Proceedings of the OWL Experiences and Directions Workshop* (OWLed'06). Online: `http://www.co-ode.org/resources/papers/`.

Horrocks, I., O. Kutz, & U. Sattler (2006). The even more irresistible $\mathcal{SROIQ}$. *Proceedings of KR 2006*, 452–457. Online: `http://www.cs.man.ac.uk/~okutz/sroiq-TR.pdf`.

Jackson, P. (1998). *Introduction to expert systems*, 3rd edition. Addison Wesley.

Kolovski, V., B. Parsia, & E. Sirin (2006). Extending the $\mathcal{SHOIQ}$(D) tableaux with DL-safe rules: First results. *Proceedings of the 2006 International Workshop on Description Logics (DL 06)*, 192–199. Online: `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-189/DL2006.pdf`.

Lacy, L.W. (2005). *QWL: Representing information using the Web Ontology Language*, Trafford Publishing.

Palmer, S.B. (2001). The semantic web, `http://infomesh.net/2001/swintro/`.

Passin, T.B. (2004). *Explorer's guide to the Semantic Web*, Manning Publications.

Pollock, J.T. (2009). *Semantic Web for dummies*, John Wiley & Sons.

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

Shadbolt, N., T., Berners-Lee & W. Hall (2006). The semantic web revisited. *IEEE Intelligent Systems* 21/3, 96–101, `http://www.w3.org/2001/sw/`.

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

Swartz, A. (2002). The semantic web in breadth, `http://logicerror.com/semanticWeb-long`.

Turing, A.M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings ofthe London Mathematical Society*, Series 2, 42, 230–265.

Appendici

Appendice I

Sintassi logica di $\mathcal{SROIQ}(\mathbf{D_n})$

A rappresenta una *classe atomica*
C, D, C_i rappresentano *classi arbitrarie (atomiche o complesse)*
$\mathcal{D}$ rappresenta *un dominio di dati*
R, S, S_i rappresentano *proprietà*
P, Q, P_i rappresentano *attributi*
a, b rappresentano *individui*
d rappresenta un *dato*
n rappesenta un *numero naturale* (0, 1, ...)
τ rappresenta un *enunciato di TBox*
ρ rappresenta un *enunciato di RBox*
α rappresenta un'*asserzione di ABox*

$$\tau \longrightarrow C \sqsubseteq D \mid C \equiv D \mid DisCla(C_1,...,C_n) \mid DisUni(A,C_1,...,C_n) \mid Dom(R,C) \mid Rng(R,C) \mid Fun(R) \mid InvFun(R) \mid Dom(P,C) \mid Rng(P,\mathcal{D}) \mid Fun(P)$$

$$\rho \longrightarrow R \sqsubseteq S \mid R \equiv S \mid S_1 \circ ... \circ S_n \sqsubseteq R \mid DisPro(R_1,...,R_n) \mid Ref(R) \mid Irr(R) \mid Sym(R) \mid Asy(R) \mid Tra(R) \mid P \sqsubseteq Q \mid P \equiv Q \mid DisPro(P_1,...,P_n)$$

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

$$C \longrightarrow \top \mid \bot \mid A \mid \neg C \mid (C \sqcap D) \mid (C \sqcup D) \mid \forall R.C \mid \exists R.C \mid R \ni a \mid \exists R.\mathrm{Self} \mid \leq nR.C \mid \geq nR.C \mid =nR.C \mid \{a\} \mid \forall P.\mathcal{D} \mid \exists P.\mathcal{D} \mid P \ni d \mid \leq nP.\mathcal{D} \mid \geq nP.\mathcal{D} \mid =nP.\mathcal{D} \mid \{d\}$$

$$R \longrightarrow \dot{\top} \mid \dot{\bot} \mid R \mid R^{-}$$

Nota: un'ontologia $\mathcal{SROIQ}(\mathrm{D_n})$ deve rispettare anche i *vincoli non strutturali* (par. 5.8).

Appendice II

Sintassi funzionale e sintassi di Manchester di $\mathcal{SROIQ}(\mathbf{D_n})$

Sintassi logica	*Sintassi funzionale*	*Sintassi di Manchester*
$C \sqsubseteq C'$	`SubClassOf(C C')`	*C* **SubClassOf** *C′*
$C \equiv C'$	`EquivalentClasses(C C')`	*C* **EquivalentTo** *C′*
DisCla(C_1,...,C_n)	`DisjointClasses(`C_1 ... C_n`)`	**DisjointClasses** C_1, ..., C_n
DisUni(*A*,C_1,...,C_n)	`DisjointUnion(A` C_1 ... C_n`)`	
$\top$	`owl:Thing`	owl:Thing
$\bot$	`owl:Nothing`	owl:Nothing
$\neg C$	`ObjectComplementOf(C)`	**not** *C*
$C \sqcap C'$	`ObjectIntersectionOf(C C')`	*C* **and** *C′* oppure *C* **that** *C′*
$C \sqcup C'$	`ObjectUnionOf(C C')`	*C* **or** *C′*
$\exists R.C$	`ObjectSomeValuesFrom(R C)`	*R* **some** *C*
$\forall R.C$	`ObjectAllValuesFrom(R C)`	*R* **only** *C*
$\exists R.$Self	`ObjectExistsSelf(R)`	*R* **some Self**
$\geq nR.C$	`ObjectMinCardinality(n R C)`	*R* **min** *n C*
$\leq nR.C$	`ObjectMaxCardinality(n R C)`	*R* **max** *n C*
$=nR.C$	`ObjectExactCardinality(n R C)`	*R* **exactly** *n C*
$\{a_1, ..., a_n\}$	`ObjectOneOf(`a_1 ... a_n`)`	$\{a_1, ..., a_n\}$
$\exists P.\mathcal{D}$	`DataSomeValuesFrom(P` $\mathcal{D}$`)`	*P* **some** xsd:$\mathcal{D}$
$\forall P.\mathcal{D}$	`DataAllValuesFrom(P` $\mathcal{D}$`)`	*P* **only** xsd:$\mathcal{D}$
$R \ni a$	`ObjectHasValue(R a)`	*R* **value** *a*
$P \ni d$	`DataHasValue(P d)`	*P* **value** *d*
$R \sqsubseteq S$	`SubObjectPropertyOf(R S)`	*R* **SubPropertyOf** *S*
$R \equiv S$	`EquivalentObjectProperties(R S)`	*R* **EquivalentTo** *S*
DisPro(R_1,...,R_n)	`DisjointObjectProperties(`R_1 ... R_n`)`	**DisjointProperties** R_1, ..., R_n
Dom(*R*,*C*)	`ObjectPropertyDomain(R C)`	*R* **Domain** *C*
Rng(*R*,*C*)	`ObjectPropertyRange(R C)`	*R* **Range** *C*
R^-	`InverseObjectProperty(R)`	**inverse**(*R*)
InvPrp(*R*,*S*)	`InverseObjectProperties(R S)`	*R* **InverseOf** *S*
DisPro(R_1,...,R_n)	`DisjointObjectProperties(`R_1 ... R_n`)`	**DisjointProperties** R_1, ..., R_n
Fun(*R*)	`FunctionalObjectProperty(R)`	*R* **Characteristics** Functional
InvFun(*R*)	`InverseFunctionalObjectProperty(R)`	*R* **Characteristics** InverseFunctional
Ref(*R*)	`ReflexiveObjectProperty(R)`	*R* **Characteristics** Reflexive
Irr(*R*)	`IrreflexiveObjectProperty(R)`	*R* **Characteristics** Irreflexive
Sym(*R*)	`SymmetricObjectProperty(R)`	*R* **Characteristics** Symmetric
Asy(*R*)	`AsymmetricObjectProperty(R)`	*R* **Characteristics** Asymmetric
Tra(*R*)	`TransitiveObjectProperty(R)`	*R* **Characteristics** Transitive
$P \sqsubseteq Q$	`SubDataPropertyOf(P Q)`	*P* **SubPropertyOf** *Q*
$P \equiv Q$	`EquivalentDataProperties(P Q)`	*P* **EquivalentTo** *Q*
DisPro(P_1,...,P_n)	`DisjointDataProperties(`P_1 ... P_n`)`	**DisjointProperties** P_1, ..., P_n
Dom(*P*,*C*)	`DataPropertyDomain(P C)`	*P* **Domain** *C*
Rng(*P*,*C*)	`DataPropertyRange(P C)`	*P* **Range** *C*
Fun(*P*)	`FunctionalDataProperty(P)`	*P* **Characteristics** Functional
$=(a_1, a_2, ..., a_n)$	`SameIndividual(`a_1 ... a_n`)`	a_1 **SameAs** a_2, ..., a_n
$\neq(a_1, a_2, ..., a_n)$	`DifferentIndividuals(`a_1 ... a_n`)`	**DifferentIndividuals** a_1, ..., a_n
C(*a*)	`ClassAssertion(a C)`	*a* **Type** *C*
R(*a*,*b*)	`ObjectPropertyAssertion(R a b)`	a **Fact** *R b*
$\neg R(a,b)$	`NegativeObjectPropertyAssertion(R a b)`	a **Fact not** *R b*
P(*a*,*d*)	`DataPropertyAssertion(P a d)`	a **Fact** *P d*
$\neg P(a,d)$	`NegativeDataPropertyAssertion(P a d)`	a **Fact not** *P d*

Appendice III

Semantica formale di $\mathcal{SROIQ}(\mathbf{D_n})$

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

Modelli

Un *modello* è una terna ordinata $M = \langle \Delta, \{\Delta_i\}, -^I \rangle$, dove l'*universo* $\Delta = \{x, y, ...\}$ è un insieme non vuoto di *oggetti*, ciascun Δ_i è un *dominio di dati*, e la *funzione d'interpretazione* $-^I$ associa:

- un oggetto $a^I \in \Delta$ a ciascun individuo a;
- un dato $d^I \in \Delta_i$ a ciascuna costante d di tipo i;
- un insieme $A^I \subseteq \Delta$ a ciascuna classe atomica A, con $\top^I = \Delta$ e $\bot^I = \varnothing$;
- una relazione binaria $R^I \subseteq \Delta \times \Delta$ a ciascuna proprietà R, con $\dot{\top}^I = \Delta^2$ e $\dot{\bot}^I = \varnothing$
- una relazione binaria $P^I \subseteq \Delta \times \Delta_i$ (per qualche i) a ciascun attributo P.

La funzione $-^I$ si estende alle classi complesse e alle proprietà inverse definendo:

- $(\neg C)^I = \Delta \setminus C^I$;
- $(C \sqcap D)^I = C^I \cap D^I$;
- $(C \sqcup D)^I = C^I \cup D^I$;
- $(\forall R.C)^I = \{x \in \Delta \mid \text{se } \langle x,y \rangle \in R^I, \text{ allora } y \in C^I\}$;
- $(\exists R.C)^I = \{x \in \Delta \mid \langle x,y \rangle \in R^I \text{ per qualche } y \in C^I\}$;
- $(\exists R.\mathsf{Self})^I = \{x \in \Delta \mid \langle x,x \rangle \in R^I\}$;
- $(R \ni a)^I = \{x \in \Delta \mid \langle x, a^I \rangle \in R^I\}$;
- $(\leq nR.C)^I = \{x \in \Delta \mid \#\{y \in C^I \mid \langle x,y \rangle \in R^I\} \leq n\}$;
- $(\geq nR.C)^I = \{x \in \Delta \mid \#\{y \in C^I \mid \langle x,y \rangle \in R^I\} \geq n\}$;
- $(=nR.C)^I = \{x \in \Delta \mid \#\{y \in C^I \mid \langle x,y \rangle \in R^I\} = n\}$;
- $\{a\}^I = \{a^I\}$;
- $(R^-)^I = (R^I)^{-1}$.

Verità

Un enunciato o asserzione φ è *vero in un modello M*, in simboli

$M \models \varphi$,

sotto le seguenti condizioni:

$M \models C \sqsubseteq D$	se e solo se $C^I \subseteq D^I$,
$M \models C \equiv D$	se e solo se $C^I = D^I$,
$M \models \mathit{DisCla}(C_1,...,C_n)$	se e solo se $C_i^I \cap C_j^I = \varnothing$ per $1 \leq i < j \leq n$,
$M \models \mathit{DisUni}(D,C_1,...,C_n)$	se e solo se $D^I = C_1^I \cup ... \cup C_n^I$ e $C_i^I \cap C_j^I = \varnothing$ per $1 \leq i < j \leq n$,
$M \models R \sqsubseteq S$	se e solo se $R^I \subseteq S^I$,
$M \models R \equiv S$	se e solo se $R^I = S^I$,
$M \models S_1 \circ ... \circ S_n \sqsubseteq R$	se e solo se $S_1^I \circ ... \circ S_n^I \subseteq R^I$,
$M \models \mathit{DisPro}(R_1,...,R_n)$	se e solo se $R_i^I \cap R_j^I = \varnothing$ per $1 \leq i < j \leq n$,
$M \models \mathit{Fun}(R)$	se e solo se R^I è funzionale,
$M \models \mathit{InvFun}(R)$	se e solo se $(R^I)^{-1}$ è funzionale,
$M \models \mathit{Ref}(R)$	se e solo se R^I è globalmente riflessiva,

$M \vDash Irr(R)$	se e solo se R^I è irriflessiva,
$M \vDash Sym(R)$	se e solo se R^I è simmetrica,
$M \vDash Asy(R)$	se e solo se R^I è asimmetrica,
$M \vDash Tra(R)$	se e solo se R^I è transitiva,
$M \vDash P \sqsubseteq Q$	se e solo se $P^I \subseteq Q^I$,
$M \vDash P \equiv Q$	se e solo se $P^I = Q^I$,
$M \vDash DisPro(P_1,...,P_n)$	se e solo se $P_i^I \cap P_j^I = \varnothing$ per $1 \le i < j \le n$,
$M \vDash Fun(P)$	se e solo se P^I è funzionale,
$M \vDash C(a)$	se e solo se $a^I \in C^I$,
$M \vDash R(a,b)$	se e solo se $\langle a^I,b^I\rangle \in R^I$,
$M \vDash \neg R(a,b)$	se e solo se $\langle a^I,b^I\rangle \notin R^I$,
$M \vDash a = b$	se e solo se $a^I = b^I$,
$M \vDash a \neq b$	se e solo se $a^I \neq b^I$.

Conseguenza logica e validità

Una base di conoscenze K è un insieme finito di enunciati o asserzioni. Un modello M *soddisfa* K (o *è un modello di* K) se e solo se:

$M \vDash \varphi$ per ogni assioma o asserzione $\varphi \in K$,

Un enunciato o asserzione φ *segue logicamente* da K,

$K \vDash \varphi$,

se e solo se $M \vDash \varphi$ per ogni modello M che soddisfi K. Un enunciato φ è *valido*,

$\vDash \varphi$,

se e solo se $\varnothing \vDash \varphi$, dove $\varnothing$ indica la base di conoscenze vuota.

Definizione semantica di alcuni compiti di ragionamento

Data una base di conoscenze K:

- il controllo di consistenza di K restituisce *true* se K ammette almeno un modello e *false* in caso contrario;
- la verifica della relazione di sottoclasse $C \sqsubseteq D$ restituisce *true* se $K \vDash C \sqsubseteq D$, e *false* in caso contrario;
- la verifica dell'equivalenza $C \equiv D$ restituisce *true* se $K \vDash C \equiv D$, e *false* in caso contrario;
- la verifica della soddisfacibilità di C restituisce *true* se $K \nvDash C \sqsubseteq \bot$, e *false* in caso contrario;
- l'*instance check* di C ed a restituisce *true* se $K \vDash C(a)$, e *false* in caso contrario;
- il *retrieval* relativo a C restituisce $\{a_1, ..., a_n\}$, con $n \ge 0$, se e solo se $a_1, ..., a_n$ sono tutti e soli gli individui occorrenti in K tali che $K \vDash C(a_k)$.

Appendice IV

Regole di traduzione da $\mathcal{SROIQ}(\mathbf{D_n})$ a FOL

Ogni classe, proprietà, enunciato e asserzione può essere tradotto in una formula di FOL equivalente. Le classi si traducono come predicati a un argomento, ovvero come formule contenenti esattamente una variabile libera (con una o più occorrenze). Le proprietà si traducono come simboli predicativi biargomentali applicati a due variabili libere. Gli enunciati e le asserzioni si traducono come formule chiuse, ovvero prive di occorrenze libere di variabili.

La traduzione FOL di un'espressione φ viene di seguito indicata come $[\varphi]$ (nel caso di formule chiuse), $[\varphi]_x$ (nel caso di formule contenenti x come variabile libera) o $[\varphi]_{xy}$ (nel caso di formule contenenti x e y come variabili libere).

L'espressione $sost(a,x,\varphi)$ indica il risultato della sostituzione di ogni occorrenza della variabile x nella formula φ con un'occorrenza della costante a.

$$
\begin{aligned}
[C \sqsubseteq D] &= \forall x\,([C]_x \rightarrow [D]_x)\\
[C \equiv D] &= \forall x\,([C]_x \leftrightarrow [D]_x)\\
[DisCla(C_1,\ldots,C_n)] &= \bigwedge_{1 \le i < j \le n} \forall x\,([C_1]_x \rightarrow \sim[C_n]_x)\\
[DisUni(A,C_1,\ldots,C_n)] &= \forall x\,(A(x) \leftrightarrow [C_1]_x \wedge \ldots \wedge [C_n]_x) \bigwedge_{1 \le i < j \le n} \forall x\,([C_1]_x \rightarrow \sim[C_n]_x)
\end{aligned}
$$

$$
\begin{aligned}
[R \sqsubseteq S] &= \forall x\,\forall y\,([R]_{xy} \rightarrow [S]_{xy})\\
[R \equiv S] &= \forall x\,\forall y\,([R]_{xy} \leftrightarrow [S]_{xy})\\
[DisPro(R_1,\ldots,R_n)] &= \bigwedge_{1 \le i < j \le n} \forall x\,\forall y\,([R_i]_{xy} \rightarrow \sim[R_j]_{xy})\\
[Fun(R)] &= \forall x\,\exists_{\le 1} y\,R(x,y)\\
[InvFun(R)] &= \forall x\,\exists_{\le 1} y\,R(y,x)\\
[Ref(R)] &= \forall x\,R(x,x)\\
[Irr(R)] &= \forall x \sim R(x,x)\\
[Sym(R)] &= \forall x\,\forall y\,(R(x,y) \rightarrow R(y,x))\\
[Asy(R)] &= \forall x\,\forall y\,(R(x,y) \rightarrow \sim R(y,x))\\
[Tra(R)] &= \forall x\,\forall y\,\forall z\,(R(x,y) \wedge R(y,z) \rightarrow R(x,z))\\
[P \sqsubseteq Q] &= \forall x\,\forall y\,(P(x,y) \rightarrow Q(x,y))\\
[P \equiv Q] &= \forall x\,\forall y\,(P(x,y) \leftrightarrow Q(x,y))\\
[DisPro(P_1,\ldots,P_n)] &= \bigwedge_{1 \le i < j \le n} \forall x\,\forall y\,(P_i(x,y) \rightarrow \sim P_j(x,y))\\
[Fun(P)] &= \forall x\,\forall y\,\forall z\,(P(x,y) \wedge P(y,z) \rightarrow P(x,z))
\end{aligned}
$$

$$
\begin{aligned}
[C(a)] &= sost(a,x,[C]_x)\\
[R(a,b)] &= R(a,b)\\
[\neg R(a,b)] &= \sim R(a,b)\\
[a = b] &= (a = b)\\
[a \neq b] &= (a \neq b)
\end{aligned}
$$

$$
\begin{aligned}
[\top]_x &= (x = x)\\
[\bot]_x &= (x \neq x)
\end{aligned}
$$

$[A]_x = A(x)$

$[\neg C]_x = {\sim}[C]_x$

$[C \sqcap D]_x = [C]_x \wedge [D]_x$

$[C \sqcup D]_x = [C]_x \vee [D]_x$

$[\forall R.C]_x = \forall y\,([R]_{xy} \rightarrow [C]_y)$

$[\exists R.C]_x = \exists y\,([R]_{xy} \wedge [C]_y)$

$[R \ni a]_x = [R]_{xa}$

$[\exists R.\mathrm{Self}]_x = [R]_{xx}$

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

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

$[=n R.C]_x = \exists_{=n} y\,([R]_{xy} \wedge [C]_y)$

$[\forall P.\mathcal{D}]_x = \forall y\,(P(x,y) \rightarrow \mathcal{D}(y))$

$[\exists P.\mathcal{D}]_x = \exists y\,(P(x,y) \wedge \mathrm{D}(y))$

$[\leq 1\, P.\mathcal{D}]_x = \exists_{\leq 1} y\,(P(x,y) \wedge \mathcal{D}(y))$

$[\{a\}]_x = (x = a)$

$[R]_{xy} = R(x,y)$

$[R^-]_{xy} = [R]_{yx}$

$[R_1 \circ \ldots \circ R_n]_{xy} = \exists z_1 \ldots \exists z_{n-1}\,([R_1]_{xz_1} \wedge \ldots \wedge [R_n]_{z_{n-1}y})$

Esempio:

le madri sono le donne che hanno almeno un figlio

$\mathrm{Madre} \equiv \mathrm{Donna} \sqcap \exists \mathrm{figlioDi}^-$

$[\mathrm{Madre} \equiv \mathrm{Donna} \sqcap \exists \mathrm{figlioDi}^-] = \forall x\,([\mathrm{Madre}]_x \leftrightarrow [\mathrm{Donna} \sqcap \exists \mathrm{figlioDi}^-]_x)$

$= \forall x\,(\mathrm{Madre}(x) \leftrightarrow [\mathrm{Donna}]_x \wedge [\exists \mathrm{figlioDi}^-]_x)$

$= \forall x\,(\mathrm{Madre}(x) \leftrightarrow \mathrm{Donna}(x) \wedge \exists y\,([\mathrm{figlioDi}^-]_{xy} \wedge [\top]_y))$

$= \forall x\,(\mathrm{Madre}(x) \leftrightarrow \mathrm{Donna}(x) \wedge \exists y\,([\mathrm{figlioDi}]_{yx} \wedge (y = y)))$

$= \forall x\,(\mathrm{Madre}(x) \leftrightarrow \mathrm{Donna}(x) \wedge \exists y\,\mathrm{figlioDi}(y,x))$