Verifica automatica di proprietà su ontologie *DL-Lite*$_{\mathcal{A}}$

Valerio Del Grande, Ilaria Forte

Dipartimento di Informatica e Sistemistica
Sapienza, Università di Roma
valerio.delgrande@gmail.com
forteilaria@hotmail.com

14 marzo 2009

"Save the cheerleader, save the World"
Hero Nakamura

Abstract

Questo paper ha lo scopo di introdurre la Decription Logic *DL-Lite*$_\mathcal{A}$, un sottoinsieme della First Order Logic particolarmente adatto a descrivere e ragionare su ontologie.

Nel capitolo 2 saranno descritte in dettaglio la sintassi e la semantica del linguaggio, i concetti di TBox (livello intensionale di una base di conoscenza) e ABox (livello estensionale di una base di conoscenza) e verrà mostrato superficialmente come si effettua il query answering su ontologie e quali sono i vantaggi di tale approccio.

Nel capitolo 3 mostriamo a puro scopo didattico, al fine di aiutare il lettore nella comprensione delle ontologie, la corrispondenza tra i classici diagrammi concettuali Entity Relationship (ER) e l'espressione dello stesso dominio in una ontologia *DL-Lite*$_\mathcal{A}$: questo procedimento di traduzione torna particolarmente utile qualora si vogliano migrare sistemi di basi di dati standard per cui è disponibile un diagramma concettuale in sistemi Ontology Based Data Access (OBDA).

Il capitolo 4 è invece il punto cruciale del lavoro: vogliamo mostrare come effettuare la verifica di proprietà statiche su un'ontologia espressa in *DL-Lite*$_\mathcal{A}$, ovvero rilevare eventuali inconsistenze o errori nella progettazione concettuale: allo stato attuale della ricerca, non esistono strumenti software che lavorano direttamente su *DL-Lite*$_\mathcal{A}$ per effettuare verifica di proprietà statiche, se non banali. Mostreremo come, traducendo la base di conoscenza *DL-Lite*$_\mathcal{A}$ in First Order Logic, sia possibile invece verificare qualsiasi proprietà esprimibile al primo ordine utilizzando theorem prover strumenti preesistenti come OTTER/mace.

La sezione 5 è dedicata allo svolgimento di alcuni specifici casi di studio, per i diversi scenari illustrati ai capitoli 3 e 4. Nel primo esempio si mostra come a partire da una specifica in linguaggio naturale si progetta il diagramma ER che poi viene tradotto in formule di *DL-Lite*$_\mathcal{A}$. Il secondo esempio mostra come a partire da una ontologia in *DL-Lite*$_\mathcal{A}$ si effettua la traduzione in FOL al fine di utilizzare Otter/mace per la verifica di proprietà.

Il sofware di traduzione automatica da *DL-Lite*$_\mathcal{A}$ a FOL prende in input un'ontologia descritta in XML secondo il formato proprietario di QuOnto e la traduce in una base di conoscenza FOL scritta nel formato di Otter.

Indice

Capitolo 1

Introduzione

Un'ontologia è una concettualizzazione del dominio di interesse di un sistema informativo espressa in un qualche linguaggio formale.

Gli schemi concettuali (UML e ER) sono particolari tipi di ontologie detti Graph-based pensati per rappresentare un singolo modello logico (rispettivamente Software e Basi di dati). Esistono diversi linguaggi formali che permettono di descrivere un ontologia, primo fra tutti la First Order Logic. Sono di particolare interesse per la loro utilità pratica, ontologie descritte utilizzando le logiche descrittive (Description Logics), una famiglia di formalismi utilizzati per rappresentare la conoscenza di un dominio applicativo strutturato (*structured knowledge*). In particolare *DL-Lite*$_\mathcal{A}$ garantisce un compromesso accettabile tra potere espressivo e complessità computazionale rispetto alla dimensione dei dati (*data complexity*): rispondere ad una UCQ (*Union of Conjunctive Queries*) su un'ontologia espressa in *DL-Lite*$_\mathcal{A}$ è LOGSPACE rispetto alla dimensione dei dati, cioè ha la stessa complessità di una query SQL effettuata una base di dati relazionale mediante un rDBMS. Per questo motivo *DL-Lite*$_\mathcal{A}$ rappresenta un'ottima scelta per la descrizione di ontologie in sistemi di reasoning per query answering (OBDA – Ontology-based Data Access) e sistemi per data integration (OBDI – Ontology-based Data Integration). Una implementazione di sistema OBDA è QuOnto ([4]) e la sua estensione al data integration Mastro-I.

Un'ontologia espressa in una Description Logic è composta da un livello intensionale (TBox) che rappresenta le proprietà intrinseche del dominio modellato e un livello estensionale (ABox) che rappresenta appunto l'estensione della TBox.
Gli elementi del livello intensionale sono:

Concept: elemento dell'ontologia che rappresenta una collezione di istanze (es. Persona)

Concept attribute: elemento dell'ontologia che qualifica un concept (relazione binaria tra un concept e un valore)

Role: elemento dell'ontologia che esprime un'associazione tra due concepts (relazione binaria tra due concept)

Role attribute: elemento dell'ontologia che qualifica un'associazione tra concepts (relazione ternaria tra due concepts e un valore)

Assertion: esprime a livello intensionale una condizione che deve essere soddisfatta a livello estensionale

Gli elementi del livello estensionale sono:

Instance: rappresenta un oggetto che è estensione di un Concept

Fact: rappresenta una relazione presente tra due istanze

Nel capitolo 2 viene descritta in dettaglio la sintassi e la semantica delle espressioni elementari *DL-Lite*$_{\mathcal{A}}$, per poi costruire con esse le asserzioni intensionali ed estensionali che compongono rispettivamente la Tbox e l'Abox.

É interessante notare che, essendo *DL-Lite*$_{\mathcal{A}}$ un sottoinsieme della Logica del Primo Ordine, è possibile in ogni momento tradurre un'espressione *DL-Lite*$_{\mathcal{A}}$ in una espressione della FOL. Quindi allo scopo di verificare se una data proprietà è rispettata o meno da un'ontologia si potrà utilizzare un dimostratore di teoremi standard per la FOL (vedi capitolo 4).

Capitolo 2

DL-Lite$_{\mathcal{A}}$

Le Description Logics sono una famiglia di formalismi per rappresentare la conoscenza in modo strutturato, classificando il mondo di interesse in *Concepts*, che denotano insiemi di oggetti rilevanti per il dominio, e *Roles*, che specificano le proprietà dei concepts.

Un'ontologia in DL è composta da una Terminological Box (TBox) e una Assertion Box (ABox): la TBox rappresenta il livello intensionale del dominio ed è il modello concettuale, espresso in modo formale, di un frammento di realtà; l'ABox rappresenta il livello estensionale della conoscenza: è il livello concreto, espresso in modo formale, di un frammento di realtà in quanto contiene conoscenze sotto forma di asserzioni, ovvero le istanze di Concepts e Roles.

La famiglia di linguaggi *DL-Lite* è pensata appositamente per catturare i modelli concettuali dei dati (Entity-Relationship) e i formalismi Object-Oriented (UML class diagrams). *DL-Lite*$_{\mathcal{A}}$ presenta una novità importante rispetto alle altre Description Logics in quanto, per la prima volta, viene presa considerazione la distinzione tra oggetti e valori. *DL-Lite*$_{\mathcal{A}}$ distingue infatti:

- Concepts da Value-domains: un concept è l'astrazione di un insieme di oggetti mentre un value-domain denota un insieme di valori concreti,
- Attributes da Roles: un role denota una relazione binaria tra oggetti mentre un attribute denota una relazione binaria tra oggetti e valori.

2.1 Sintassi *DL-Lite*$_{\mathcal{A}}$

Come in ogni logica le espressioni *DL-Lite*$_{\mathcal{A}}$ sono costruite a partire da un alfabeto.

Alfabeto

- Denotiamo con **A** gli Atomic Concepts, cioè i concetti primitivi; un Atomic Concept è una collezione di instanze di oggetti dello stesso tipo, denotata da un nome. Ad esempio l'atomic concept *Person* rappresenta la collezione di oggetti di tipo person.

- Denotiamo con **P** gli Atomic Roles, cioè i ruoli primitivi; un Atomic Role esprime un'associazione binaria tra due Concepts ed è denotata da un nome. Ad esempio dati due atomic concepts *Person* e *City* l'atomic role *cityOfBirth* rappresenta la relazione binaria che esiste tra essi.

- Denotiamo con $\mathbf{T} = \{T_1, ...T_n\}$ l'insieme dei Value-Domains, dove ogni T_i rapprensenta il range di un attributo, cioè l'insieme di valori concreti che l'attributo può assumere. Ogni value-domain può essere uno degli RDF-Data-Types, ad esempio: String, Integer...

- Denotiamo con $\mathbf{U_C}$ gli Atomic Concept Attributes, cioè gli attributi primitivi che qualificano i concepts, denotati da un nome. Ad esempio l'atomic concept attribute *name* potrebbe rapprensentare una proprietà del concept *Person*.

- Denotiamo con $\mathbf{U_R}$ gli Atomic Role Attributes, cioè gli attributi primitivi che qualificano i roles, denotati da un nome. Ad esempio l'atomic role attribute *dateOfBirth* potrebbe rapprensentare una proprietà del role *cityOfBirth*.

- Denotiamo con $\mathbf{\Gamma}$ l'insieme dei simboli di costante. Tale insieme è partizionato in due sottoinsiemi:

  - $\mathbf{\Gamma_O}$, simboli di costante per gli oggetti
  - $\mathbf{\Gamma_V}$, simboli di costante per i valori, a sua volta partizionato in n sottoinsiemi Γ_{V_1},...Γ_{V_n} dove ogni Γ_{V_i} è l'insieme di costanti per i valori del value-domain T_i.

Esempio:
Atomic Concepts: Person, City
Atomic Roles: cityOfBirth
Atomic Concept Attributes: name,surname,cityname
Atomic Role Attributes: dateOfBirth
$T = \{String, Date\}$
$\Gamma_O = \{o_1, o_2, o_3, ...\}$
$\Gamma_V = \{v_1, v_2, v_3, ...\}$

2.1.1 Notazione

Nel seguito verrà utilizzata la seguente notazione:

- A atomic concept, B basic concept, C general concept, $\top_C$ concept universale, dove B e C sono espressioni costruite secondo la grammatica illustrata al par. 2.1.2, a partire da elementi atomici.
- P atomic role, Q basic role, R general role, dove Q e R sono espressioni costruite secondo la grammatica illustrata al par. 2.1.2, a partire da elementi atomici.
- T_i value domain, E basic value-domain, F value-domain expression, $\top_D$ value-domain universale, dove E e F sono espressioni costruite secondo la grammatica illustrata al par. 2.1.2, a partire da elementi atomici.
- U_C atomic concept attribute, V_C general concept attribute, dove V_C è un'espressione costruita secondo la grammatica illustrata al par. 2.1.2, a partire da elementi atomici.
- U_R atomic role attribute, V_R general role attribute, dove V_R è un'espressione costruita secondo la grammatica illustrata al par. 2.1.2, a partire da elementi atomici.

Dato un atomic concept attribute U_C chiamiamo *dominio* di U_C, denotato con $\delta(U_C)$, l'insieme di oggetti (Concepts) che U_C mette in relazione con i valori; chiamiamo invece *range* di U_C, denotato con $\rho(U_C)$ l'insieme di valori (value-domains) che U_C mette in relazione con oggetti.

> Esempio: un atomic concept attribute è una relazione binaria tra un concept e un valore quindi se consideriamo name=(Person, String) allora $\delta(name) = Person$ $\rho(name) = String$, cioè dominio e codominio della relazione binaria.

Dato un atomic role attribute U_R chiamiamo *dominio* di U_R, denotato con $\delta(U_R)$, l'insieme di coppie di oggetti (Concepts) che U_R mette in relazione con i valori; chiamiamo invece *range* di U_R, denotato con $\rho(U_R)$ l'insieme di valori (value-domains) che U_R mette in relazione con la coppia di oggetti.

> Esempio: un atomic role attribute è una relazione ternaria tra due concepts e un valore quindi se consideriamo dateOfBirth=(Person, City, Date) allora $\delta(dateOfBirth) = (Person, City)$ $\rho(dateOfBirth) = Date$.

2.1.2 Espressioni

Possiamo definire ora le espressioni *DL-Lite*$_\mathcal{A}$ costruite a partire dall'alfabeto

1. Concept expressions:

$$B \ ::= \ A \mid \exists Q \mid \delta(U_C)$$

$$C \ ::= \ \top_C \mid B \mid \neg B$$

2. Value-domain expressions:

$$E \ ::= \ \rho(U_C) \mid \rho(U_R)$$

$$F \ ::= \ \top_D \mid T_1 \mid \cdots \mid T_n$$

3. Role expressions:

$$Q \ ::= \ P \mid P^- \mid \delta(U_R) \mid \delta(U_R)^-$$

$$R \ ::= \ Q \mid \neg Q$$

4. Attribute expressions:

$$V_C \ ::= \ U_C \mid \neg U_C$$

$$V_R \ ::= \ U_R \mid \neg U_R$$

2.2 Semantica *DL-Lite*$_\mathcal{A}$

Il significato di ogni espressione *DL-Lite*$_\mathcal{A}$ è definito dalla semantica, fornita in termini di interpretazioni della First Order Logic.
Ogni value-domain T_i è interpretato come l'insieme $val(T_i)$ di valori del corrispondente data type (RDF data type) e ogni costante $c_i \in \Gamma_V$ è interpretata come uno specifico valore in $val(T_i)$ denotato con $val(C_i)$. É importante notare che se $i \neq j$ allora $val(T_i) \cap val(T_j) = \emptyset$.

Un'interpretazione è una coppia $I = (\Delta^I, \cdot^I)$ dove:

- Δ^I è il dominio di interpretazione, unione disgiunta di due insiemi non vuoti: Δ^I_O dominio degli oggetti e Δ^I_V dominio dei valori e a sua volta $\Delta^I_V = \bigcup val(T_i)$.
- $\cdot^I$ è la funzione di interpretazione che assegna un elemento di Δ^I ad ogni costante in Γ, un sottoinsieme di Δ^I ad ogni concetto e value-domain, e un sottoinsieme di $\Delta^I \times \Delta^I$ ad ogni role e attribute in modo che

- per ogni $a \in \Gamma_V$, $a^I = val(a)$,
- per ogni $a \in \Gamma_O$, $a^I \in \Delta_O^I$,
- per ogni $a, b \in \Gamma$, $a \neq b$ implica $a^I \neq b^I$,
- per ogni T_i, $T_i^I = val(T_i)$,
- sono soddisfatte le seguenti condizioni ($o, o' \in \Delta_O^I$, e $v \in \Delta_V^I$):
  $\top_C^I = \Delta_O^I$
  $\top_D^I = \Delta_V^I$
  $A^I \subseteq \Delta_O^I$
  $P^I \subseteq \Delta_O^I \times \Delta_O^I$
  $U_C^I \subseteq \Delta_O^I \times \Delta_V^I$
  $U_R^I \subseteq \Delta_O^I \times \Delta_O^I \times \Delta_V^I$
  $(\neg U_C^I) = (\Delta_O^I \times \Delta_V^I) \setminus U_C^I$
  $(\neg U_R^I) = (\Delta_O^I \times \Delta_O^I \times \Delta_V^I) \setminus U_R^I$
  $(\rho(U_C))^I = \{v | \exists o.(o, v) \in U_C^I\}$
  $(\rho(U_R))^I = \{v | \exists o, o'.(o, o', v) \in U_R^I\}$
  $(\delta(U_C))^I = \{o | \exists v.(o, v) \in U_C^I\}$
  $(\delta(U_R))^I = \{(o, o') | \exists v.(o, o', v) \in U_R^I\}$
  $(\delta(U_R)^-)^I = \{(o, o') | \exists v.(o', o, v) \in U_R^I\}$
  $(P^-)^I = \{(o, o') | (o', o) \in P^I\}$
  $(\exists Q)^I = \{o | \exists o'.(o, o') \in Q^I\}$
  $(\neg Q)^I = (\Delta_O^I \times \Delta_O^I) \setminus Q^I$
  $(\neg B^I) = \Delta_O^I \setminus B^I$
  $(\neg E^I) = \Delta_V^I \setminus E^I$

É importante notare che costanti distinte sono interpretate diversamente ovvero *DL-Lite*$_\mathcal{A}$ adotta l'assunzione di nome unico (*Unique name assumption*).

Esempio :
Alfabeto:
Atomic Concepts: $Person, City$
Atomic Roles: $cityOfBirth$
Atomic Concept Attributes: $name, cityname$
Atomic Role Attributes: $dateOfBirth$
$\Gamma_O = \{pers1, pers2, cit1, cit2\}$
$\Gamma_V = \Gamma_{STRING} \cup \Gamma_{DATE}$
$\Gamma_{STRING} = \{str1, str2, str3, str4\}$
$\Gamma_{DATE} = \{d1, d2\}$

Dominio di interpretazione:
$\Delta_O^I = \{o_1, o_2, o_3, o_4\}$
$\Delta_V^I = val(STRING) \cup val(DATE) =$

$= \{Ilaria, Valerio, Roma, Lucera, 02/02/1983, 06/08/1983...\}$

Funzione di interpretazione:
$str1^I = Ilaria$
$str2^I = Valerio$
$str3^I = Roma$
$str4^I = Lucera$
$d1^I = 02/02/1983$
$d2^I = 06/08/1983$

$pers1^I = o_1$
$pers2^I = o_2$
$cit1^I = o_3$
$cit2^I = o_4$

$\top_C^I = \{o_1, o_2, o_3, o_4\}$
$\top_D^I = \{Ilaria, Valerio, Roma, Lucera, 02/02/1983, 06/08/1983...\}$

$Person^I = \{o_1, o_2\}$
$City^I = \{o_3, o_4\}$

$cityOfBirth^I = \{(o_1, o_4), (o_2, o_3)\}$
$(cityOfBirth^-)^I = \{(o_4, o_1), (o_3, o_2)\}$

$name^I = \{(o_1, Ilaria), (o_2, Valerio)\}$
$cityname^I = \{(o_3, Roma), (o_4, Lucera)\}$

$dateOfBirth^I = \{(o_1, o_4, 02/02/1983), (o_2, o_3, 06/08/1983)\}$

$(\rho(name))^I = \{Ilaria, Valerio\}$
$(\rho(cityname))^I = \{Roma, Lucera\}$

$(\delta(name))^I = \{o_1, o_2\}$
$(\delta(cityname))^I = \{o_3, o_4\}$

$(\rho(dateOdBirth))^I = \{02/02/1983, 06/08/1983\}$

$(\delta(dateOdBirth))^I = \{(o_1, o_4), (o_2, o_3)\}$
$(\delta(dateOdBirth)^-)^I = \{(o_4, o_1), (o_3, o_2)\}$

$(\exists cityOfBirth)^I = \{o_1, o_2\}$
$(\exists cityOfBirth^-)^I = \{o_3, o_4\}$

2.3 Ontologie *DL-Lite*$_\mathcal{A}$

Un'ontologia *DL-Lite*$_\mathcal{A}$ è una coppia $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, dove $\mathcal{T}$ e $\mathcal{A}$ sono rispettivamente una *DL-Lite*$_\mathcal{A}$ TBox e una *DL-Lite*$_\mathcal{A}$ ABox definite come segue.

2.3.1 Terminological Box: Livello intensionale

DL-Lite$_\mathcal{A}$ *intensional assertions* :

Inclusion Assertions:

B	$\sqsubseteq$	C	(*concept inclusion assertion*)
Q	$\sqsubseteq$	R	(*role inclusion assertion*)
E	$\sqsubseteq$	F	(*value-domain inclusion assertion*)
U_C	$\sqsubseteq$	V_C	(*concept attribute inclusion assertion*)
U_R	$\sqsubseteq$	V_R	(*role attribute inclusion assertion*)

Una concept inclusion assertion esprime che un Basic Concept B è incluso in un General Concept C. Analogamente per le altre inclusion assertion. Nell'esempio di TBox a pag 11 le asserzioni (2.1), (2.2), (2.4), (2.6), (2.8), (2.10), (2.11) e (2.12) sono Concept Inclusion Assertions; le asserzioni (2.3), (2.7) e (2.15) sono Value-Domain Inclusion Assertions; l'asserzione (2.14) è una Role Inclusion Assertion. Il significato di queste asserzioni è commentato a pag 11.

Tramite queste asserzioni sarà possibile modellare le proprietà semantiche del dominio: per una maggiore comprensione si veda il capitolo 3 che mostra la corrispondenza tra asserzioni *DL-Lite*$_\mathcal{A}$ e i ben noti diagrammi ER.

Functionality assertions:

$(funct\ P)$	(*role functionality assertion*)
$(funct\ P^-)$	(*inverse role functionality assertion*)
$(funct\ U_C)$	(*concept attribute functionality assertion*)
$(funct\ U_R)$	(*role attribute functionality assertion*)

Una role functionality assertion esprime la funzionalità globale di un Atomic Role P, cioè se un concept partecipa come dominio nella relazione binaria P, allora vi partecipa una sola volta. Analogamente per le altre functionality assertions.
Nell'esempio di TBox a pag 11 le asserzioni (2.5) e (2.9) sono concept attribute functionality assertion; l'asserzione (2.13) è una role functionality assertion. Il significato di queste asserzioni è commentato a pag 11.

Per una maggiore comprensione si veda la semantica al paragrafo successivo e il capitolo 3 che mostra la corrispondenza tra asserzioni *DL-Lite*$_{\mathcal{A}}$ e i ben noti diagrammi ER.

Identification assertions:

$$(id\ B\ I_1, ..., I_n) \qquad (\textit{identification assertion})$$

dove B è un basic concept e ogni I_i nella *identification list* è o un atomic attribute o un basic role.

Una identification assertion significa che la combinazione delle proprietà $I_1, ..., I_n$ identifica le istanze di B, ovvero impone che non possono esistere due istanze di B con gli stessi "valori" per la identification list.

Semantica :
Un'interpretazione $I = (\Delta^I, \cdot^I)$ soddisfa

- una concept inclusion assertion $B \sqsubseteq C$, se $B^I \subseteq C^I$;
- una role inclusion assertion $Q \sqsubseteq R$, se $Q^I \subseteq R^I$;
- una value-domain inclusion assertion $E \sqsubseteq F$, se $E^I \subseteq F^I$;
- una concept attribute inclusion assertion $U_C \sqsubseteq V_C$, se $U_C^I \subseteq V_C^I$;
- una role attribute inclusion assertion $U_R \sqsubseteq V_R$, se $U_R^I \subseteq V_R^I$;
- una role functionality assertion $(funct\ P)$, se per ogni $o_1, o_2, o_3 \in \Delta_O^I$, $(o_1, o_2) \in P^I$ e $(o_1, o_3) \in P^I$ implica $o_2 = o_3$;
- una inverse role functionality assertion $(funct\ P^-)$, se per ogni $o_1, o_2, o_3 \in \Delta_O^I$, $(o_1, o_2) \in P^{-I}$ e $(o_1, o_3) \in P^{-I}$ implica $o_2 = o_3$;
- una concept attribute functionality assertion $(funct\ U_C)$, se per ogni $o \in \Delta_O^I$ e $v_1, v_2 \in \Delta_V^I$, $(o, v_1) \in U_C^I$ e $(o, v_2) \in U_C^I$ implica $v_1 = v_2$;
- una role attribute functionality assertion $(funct\ U_R)$, se per ogni $o_1, o_2 \in \Delta_O^I$ e $v_1, v_2 \in \Delta_V^I$, $(o_1, o_2, v_1) \in U_R^I$ e $(o_1, o_2, v_2) \in U_R^I$ implica $v_1 = v_2$;
- una identification assertion $(id\ B\ I_1, ..., I_n)$, se per ogni $o_1, o_2 \in B^I$ e per ogni $f_1^1 ... f_1^n$, $f_2^1 ... f_2^n$ si ha che $(o_1, f_1^i) \in I_i^I$ e $(o_2, f_2^i) \in I_i^I$, per ogni $i \in [1, n]$, implica $o_1 = o_2$.

I è un modello di $\mathcal{T}$, scritto $I \models \mathcal{T}$, se e solo se I soddisfa tutte le intensional assertions in $\mathcal{T}$.

Def.* DL-Lite$_\mathcal{A}$ *TBox :
Un atomic attribute U è detto *identifying property* se la TBox contiene un'asserzione di funzionalità ($funct\ U$). Un atomic role P (o P^-) è detto *identifying property* se la TBox contiene un'asserzione di funzionalità ($funct\ P$) (o ($funct P^-$)). Un atomic attribute o basic role X si dice *primitivo* se non appare positivamente nel lato destro di una inclusion assertion ($Y \sqsubseteq X$) e non appare in un'espressione della forma $\exists Q.C$.

> Una *DL-Lite$_\mathcal{A}$* TBox è un insieme finito $\mathcal{T}$ di asserzioni intensionali che soddisfano la condizione che ogni identifying property in $\mathcal{T}$ è primitiva in $\mathcal{T}$.

Questo significa in pratica che, in *DL-Lite$_\mathcal{A}$* non è possibile specializzare una *identifying property*, cioè un attributo o ruolo funzionale.

Esempio TBox :
Atomic Concepts= $\{Student, Person\}$
Atomic Concept Attributes = $\{name, stud_id\}$
Atomic Roles = $\{cityOfBirth\}$
Atomic Role Attributes = $\{dateOfBirth\}$

$$Student \sqsubseteq Person \tag{2.1}$$
$$\delta(name) \sqsubseteq Person \tag{2.2}$$
$$\rho(name) \sqsubseteq xsd : string \tag{2.3}$$
$$Person \sqsubseteq \delta(name) \tag{2.4}$$
$$(funct\ name) \tag{2.5}$$
$$\delta(stud_id) \sqsubseteq Student \tag{2.6}$$
$$\rho(stud_id) \sqsubseteq xsd : integer \tag{2.7}$$
$$Student \sqsubseteq \delta(stud_id) \tag{2.8}$$
$$(funct\ stud_id) \tag{2.9}$$
$$\exists cityOfBirth \sqsubseteq Person \tag{2.10}$$
$$\exists cityOfBirth^- \sqsubseteq City \tag{2.11}$$
$$Person \sqsubseteq \exists cityOfBirth \tag{2.12}$$
$$(funct\ cityOfBirth) \tag{2.13}$$
$$\delta(dateOfBirth) \sqsubseteq cityOfBirth \tag{2.14}$$
$$\rho(dateOfBirth) \sqsubseteq xsd : date \tag{2.15}$$

La (2.1) esprime che ogni istanza di *Student* è anche una istanza di *Person*; la (2.2) esprime che *name* è un attributo di *Person*; la (2.3) esprime che *name* è un attributo di tipo *string*; la (2.4) esprime che ogni *Person*

ha obbligatoriamente un *name*; la (2.5) eprime che *name* è funzionale ovvero ogni $Person$ può avere un solo *name*; (2.6), (2.7), (2.8) e (2.9) sono analoghe alle precedenti; (2.10) e (2.11) tipizzano rispettivamente il dominio e il codominio del role $cityOfBirth$; la (2.12) esprime che ogni istanza di $Person$ deve partecipare obbligatoriamente al role $cityOfBirth$; la (2.13) esprime la funzionalità del role $cityOfBirth$ ovvero impone che data una istanza dell'oggetto del dominio ($Person$) questa non possa partecipare due volte alla relazione, ovvero ogni $Person$ ha una sola $cityOfBirth$; la (2.14) esprime che $dateOfBirth$ è un attributo del role $cityOfBirth$; la (2.15) esprime che l'attributo $dateOfBirth$ è di tipo *date*.

Si noti ad esempio che in questa TBox una persona può nascere in una sola città ma potrebbe avere più di una data di nascita, dal momento che il role attribute $dateOfBirth$ non è dichiarato come funzionale. Questo potrebbe essere un tipico errore nella modellazione concettuale, individuabile mediante la metodologia esposta al capitolo 4.

2.3.2 Assertions Box: Livello estensionale

DL-Lite$_\mathcal{A}$ *membership assertions* :

$A(a), \qquad P(a,b), \qquad U_C(a,c), \qquad U_R(a,b,c)$

dove a e b sono costanti dell'alfabeto in Γ_O e c è una costamte in Γ_V.

Per una maggiore comprensione delle membership assertion si veda la semantica al paragrafo successivo e l'esempio di ABox a pag.13.

Semantica :
Un'interpretazione $I = (\Delta^I, \cdot^I)$ soddisfa una membership assertion

- $A(a)$ se $a^I \in A^I$;
- $P(a,b)$ se $(a^I, b^I) \in P^I$;
- $U_C(a,c)$ se $(a^I, c^I) \in U_C^I$;
- $U_R(a,b,c)$ se $(a^I, b^I, c^I) \in U_R^I$;

I è un modello di $\mathcal{A}$, scritto $I \models \mathcal{A}$, se e solo se I soddisfa tutte le membership assertions in $\mathcal{A}$.

Def.* DL-Lite$_\mathcal{A}$ *ABox :
Una *DL-Lite*$_\mathcal{A}$ ABox è un insieme finito $\mathcal{A}$ di membership assertions.

Esempio ABox : In grassetto le costanti in Γ_O (oggetti), in corsivo le costanti in Γ_V (valori).

$$Student(\textbf{stud1}) \tag{2.16}$$
$$name(\textbf{stud1}, \textit{Valerio}) \tag{2.17}$$
$$stud_id(\textbf{stud1}, \texttt{798134}) \tag{2.18}$$
$$City(\textbf{city1}) \tag{2.19}$$
$$cityOfBirth(\textbf{stud1}, \textbf{city1}) \tag{2.20}$$
$$date(\textbf{stud1}, \textbf{city1}, \textit{06/08/1983}) \tag{2.21}$$

La (2.16) afferma che esiste a livello estensionale una istanza stud1 del Concept *Student*; la (2.17) afferma che l'istanza stud1 ha un attributo *name* con valore Valerio; la (2.18) è analoga alla precedente; la (2.19) afferma che esiste una istanza city1 di *City*; le (2.20) e (2.21) affermano che lo studente stud1 è nato nella città city1 in data 06/08/1983.

2.3.3 *Semantica Ontologie in* DL-Lite$_{\mathcal{A}}$

Un'interpretazione $I = (\Delta^I, \cdot^I)$ è un modello di un'ontologia $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, scritto $I \models \mathcal{O}$, se e solo se $I \models \mathcal{T}$ e $I \models \mathcal{A}$.

Capitolo 3

Traduzione da ER a *DL-Lite*$_{\mathcal{A}}$

Scenario: questo capitolo ha lo scopo di mostrare, più che altro a scoppo didattico, la corrispondenza tra i diagrammi ER per la modellazione concettuale dei dati e le ontologie espresse in *DL-Lite*$_{\mathcal{A}}$. Viene mostrato in dettaglio la traduzione di ogni costrutto esprimibile in ER verse le corrisponti asserzioni ontologiche. Da un punto di vista pratico, questo procedimento di traduzione torna particolarmente utile qualora si vogliano migrare sistemi di basi di dati standard per cui è disponibile un diagramma concettuale (in fase di analisi) verso sistemi moderni Ontology Based Data Access (OBDA) per cui è ovviamente necessaria la descrizione formale del dominio.

3.1 Diagrammi Entity-Relationship (ER)

Entità: rappresenta una classe di oggetti di interesse per l'applicazione che hanno proprietà comuni. Viene rappresentata graficamente mediante un rettangolo all'interno del quale viene scritto il nome dell'entità (unico nel diagramma).

Attributi di entità: rapprensentano proprietà locali di un'entità, il loro nome è unico nell'ambito dell'entità. Sono rappresentati graficamente mediante un cerchio collegato all'entità a cui appartengono.

Tipi degli attributi: sono una collezione di valori (stringa, interi ecc..) a cui è associato il dominio dell'attributo. Generalmente vengono tralasciati nella rappresentazione grafica, oppure si scrivono accanto al nome: es. Cognome/stringa.

Molteplicità degli attributi: un attributo in ER è una funzione totale dalle istanze dell'entità agli elementi del dominio, cioè ogni attributo ha uno e un solo valore: molteplicità (1,1) si omette dal

diagramma. E' possibile modificare la semantica degli attributi specificando una cardinalità minima 0 (attributo opzionale — funzione non totale) oppure una cardinalità massima diversa da 1 (attributi multivalore — relazione invece di funzione).

Indentificatore di entità: insieme di proprietà (attributi e/o relazioni) che permettono di identificare univocamente una entità a livello estensionale, ovvero tale che non esistono due istanze di entità che assumono lo stesso valore per tutte le proprietà che formano l'identificatore.

interno: formato da soli attributi dell'entità. Se è un solo attributo si rappresenta con il cerchio dell'attributo annerito, se sono più attributi si uniscono con una linea che termina con un cerchio annerito.

esterno: formato da attributi dell'entità (anche nessuno) e ruoli che coinvolgono l'entità. Si rappresenta unendo atrtibuti e ruoli con una linea che termina con un cerchio annerito.

Relazioni: rappresentano un legame tra due o più entità (relazione n-arie). Si rappresentano graficamente mediante un rombo collegato alle entità interessate, all'interno del quale è scritto il nome della relazione (unico nel diagramma). Dal punto di vista semantico una relazione ER è una relazione matematica. Se la relazione insiste sulla stessa entità è necessario specificare i ruoli, il cui nome viene scritto vicino al collegamento della relazione.

Cardinalità di relazioni: se non specificata la cardinalità delle relazioni è (0,n) da entrambe le parti (relazione matematica). Possono essere specificati vincoli di cardinalità più restrittivi sia minima che massima.

Attributi di relazione: rappresentano proprietà locali di una relazione. Sono rappresentati graficamente con un cerchio collegato al rombo della relazione.

Molteplicità di attributi di relazione: come per gli attributi di entità la molteplicità è (1,1) se non diversamente specificato.

Relazione ISA tra entità: sussiste tra due entità di un diagramma ed esprime una relazione di sottoinsieme, ovvero che ogni istanza della sott-oentità è anche istanza della super-entità. Si rappresenta graficamente con una freccia dalla sotto-entità all'entità padre. Un'entità può avere al massimo un padre (non è ammessa ereditarietà multipla). *Ereditarietà*: le istanze della sottoentità ereditano tutte le proprietà della super-entità (attributi e relazioni) e possono averne delle altre.

Generalizzazione: è una relazione ISA tra un padre e più sotto-entità rispetto ad un unico criterio. Le sottoentità sono disgiunte a coppie. La generalizzazione può essere completa (l'unione delle istanze delle sotto-entità sono tutte le istanze dell'entità padre) oppure non completa. Graficamente si rappresenta con una freccia non annerità se la generalizzazione è non completa oppure con una freccia annerita se è completa.

ISA tra relazioni: se R ISA Q le relazioni devono avere lo stesso grado, gli stessi ruoli e, per ogni ruolo U, l'entità corrispondente a U in R è entità figlia dell'entità che corrisponde a U in Q.

Specializzazioni: Ogni vincolo di cardinalità massima di Q viene ereditato da R che però può specificare una cardinalità massima minore. I vincoli di cardinalità minima sono invece scorrelati.

In ogni schema si assume che le entità che non hanno un padre sono in realtà sottoentità di una generalizzazione completa con una entità padre detta TOP e sono quindi disgiunte a coppie.

3.2 Limitazioni *DL-Lite*$_\mathcal{A}$

Il potere espressivo di *DL-Lite*$_\mathcal{A}$ è limitato rispetto ai diagrammi ER, in particolare non è possibile esprimere:

- Completezza di generalizzazioni
- Vincoli di cardinalità delle relazioni diversi da 0, 1 o n
- Vincoli di molteplicità di attributi diversi da 0, 1 o n
- Relazioni n-arie tra entità (con $n > 2$)

3.3 Traduzione da ER in *DL-Lite*$_\mathcal{A}$

In Figura 3.1 è riportato un semplice diagramma ER di esempio con le corrispondenti asserzioni *DL-Lite*$_\mathcal{A}$. Vediamo ora in dettaglio come si effettua la traduzione.

Entità: Per ogni entità ER si dichiara il relativo Atomic Concept $a \in A$ nell'alfabeto

Attributi: per ogni attributo si dichiara il relativo Atomic Concept Attribute $u \in U_C$ nell'alfabeto. Ogni attributo deve avere un

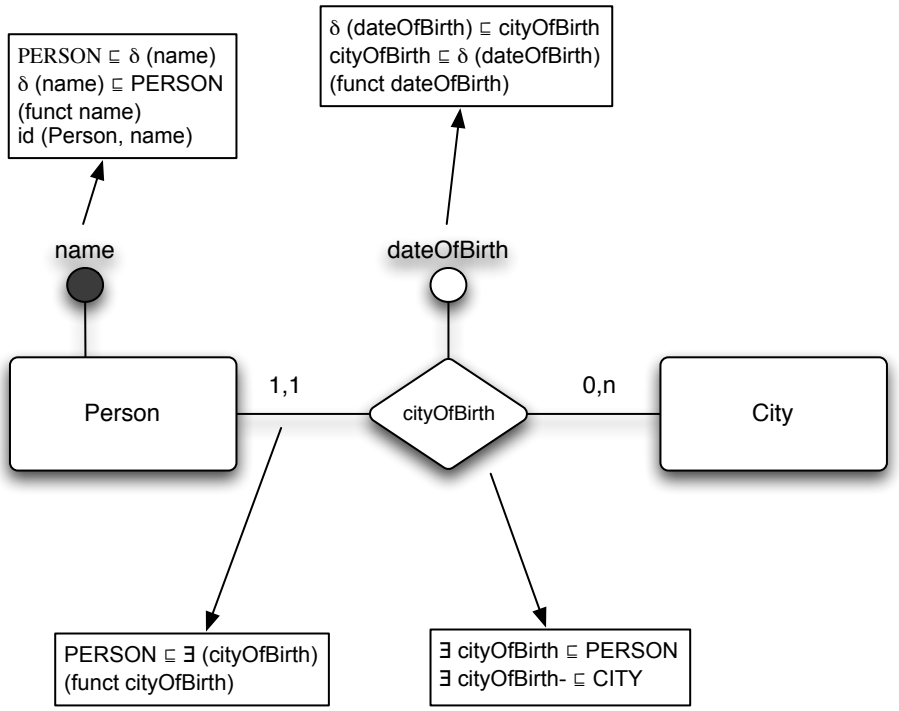


Figura 3.1: Esempio diagramma ER e DL-LiteA TBox

nome diverso, unico nella TBox, mentre in ER il nome dell'attributo è unico nell'ambito dell'entità.
Nella TBox è presente una concept inclusion assertion del tipo

$$\delta(U_C) \sqsubseteq A \tag{3.1}$$

Esempio: $\delta(name) \sqsubseteq Person$, cioè il dominio dell'attributo name è Person.

Tipi degli attributi: nella TBox è presente una value-domain inclusion assertion del tipo

$$\rho(U_C) \sqsubseteq T_i \tag{3.2}$$

Esempio: $\rho(name) \sqsubseteq xsd : string$, cioè il codominio di name sono stringhe.

Molteplicità degli attributi: senza altre asserzioni rispetto alle due precedenti gli attributi hanno molteplicità (0,n). Per tradurre la normale semantica (1,1) degli attributi ER sono necessarie le due asserzioni seguenti:
Molteplicità minima 1: nella TBox è presente una concept inclusion Assertion del tipo

$$A \sqsubseteq \delta(U_C) \tag{3.3}$$

Esempio: $Person \sqsubseteq \delta(name)$, cioè ogni persona è contenuta nel dominio dell'attributo name (ogni persona ha almeno un nome).
Molteplicità massima 1: nella TBox è presente una concept attribute functionality assertion del tipo

$$(funct\, U_C) \tag{3.4}$$

Esempio: $(funct\ name)$, cioè name è unico (ogni persona ha al più un nome).

Relazioni: si possono esprimere solo relazioni ER binarie. Per ogni relazione si dichiara il relativo Atomic Role $p \in P$ nell'alfabeto.
Tipizzazione del role: due concept inclusion assertions del tipo

$$\exists P \sqsubseteq A \tag{3.5}$$

$$\exists P^- \sqsubseteq A \tag{3.6}$$

Esempio: $\exists cityOfBirth \sqsubseteq Person$ e $\exists cityOfBirth^- \sqsubseteq City$, cioè cityOfBirth è una relazione binaria il cui dominio è Person e il cui codominio è City.

Molteplicità relazioni: senza ulteriori asserzioni rispetto alle precedenti le relazioni sono (0,n).

Molteplicità minima 1: nella TBox è presente una concept inclusion assertion del tipo

$$A \sqsubseteq \exists Q \tag{3.7}$$

Esempio: $Person \sqsubseteq \exists cityOfBirth$.

Molteplicità massima 1: nella TBox è presente una role functionality assertion del tipo

$$(funct\ P) \tag{3.8}$$

Esempio: $(funct\ cityOfBirth)$.

Attributi di relazioni: per ogni attributo si dichiara il relativo atomic role attribute $u \in U_R$ nell'alfabeto.
Nella TBox è presente una role inclusion assertion del tipo

$$\delta(U_R) \sqsubseteq P \tag{3.9}$$

Esempio: $\delta(date) \sqsubseteq cityOfBirth$.

Tipo degli attributi: nella TBox è presente una value-domain inclusion assertion del tipo

$$\rho(U_R) \sqsubseteq T_i \tag{3.10}$$

Esempio: $\rho(date) \sqsubseteq xsd : date$.

Molteplicità minima 1: nella TBox è presente una role inclusion Assertion del tipo

$$P \sqsubseteq \delta(U_R) \tag{3.11}$$

Esempio: $cityOfBirth \sqsubseteq \delta(date)$.

Molteplicità massima 1: nella TBox è presente una role attribute functionality assertion del tipo

$$(funct\ U_R) \tag{3.12}$$

Esempio: $(funct\ date)$.

Relazione ISA : sia a l'entità padre e a' l' entità figlia, allora nella TBox è presente una concept inclusion assertion

$$a' \sqsubseteq a \tag{3.13}$$

con $a, a' \in A$. Esempio: $Student \sqsubseteq Person$

Generalizzazione: per ogni coppia padre a e figlio a_i, nella TBox è presente una concept inclusion assertion

$$a_i \sqsubseteq a \tag{3.14}$$

con $a, a_i \in A$
e per ogni coppia di sotto-entità a_i, a_j una concept inclusion assertion del tipo

$$a_i \sqsubseteq \neg a_j \tag{3.15}$$

con $a_i, a_j \in A$
Esempio: $Student \sqsubseteq Person$, $Employee \sqsubseteq Person$, $Student \sqsubseteq \neg Employee$.
La completezza della generalizzazione non si può esprimere in *DL-Lite*$_\mathcal{A}$.

Vincoli di identificazione: una identification assertions nella TBox del tipo

$$(id\ B\ I_1, ..., I_n) \tag{3.16}$$

interna: ogni I_i è un atomic attribute.

esterna: ogni I_i è un basic roles oppure (opzionale) un atomic attribute.

TBox completa dell'esempio di figura 3.1:

$$\delta(name) \sqsubseteq Person \tag{3.17}$$
$$\rho(name) \sqsubseteq xsd : string \tag{3.18}$$
$$Person \sqsubseteq \delta(name) \tag{3.19}$$
$$(funct\ name) \tag{3.20}$$
$$\exists cityOfBirth \sqsubseteq Person \tag{3.21}$$
$$\exists cityOfBirth^- \sqsubseteq City \tag{3.22}$$
$$Person \sqsubseteq \exists cityOfBirth \tag{3.23}$$
$$(funct\ cityOfBirth) \tag{3.24}$$
$$\delta(dateOfBirth) \sqsubseteq cityOfBirth \tag{3.25}$$
$$\rho(dateOfBirth) \sqsubseteq xsd : date \tag{3.26}$$
$$cityOfBirth \sqsubseteq \delta(dateOfBirth) \tag{3.27}$$
$$(funct\ dateOfBirth) \tag{3.28}$$
$$id(Person, name) \tag{3.29}$$
$$\tag{3.30}$$

Capitolo 4

Traduzione da *DL-Lite*$_\mathcal{A}$ in FOL

Scenario: vogliamo verificare in modo automatico proprietà di una base di conoscenza espressa in *DL-Lite*$_\mathcal{A}$. Siamo interessati alla verifica di proprietà qualunque esprimibili al primo ordine al fine di individuare eventuali errori nella modellazione concettuale del dominio. Allo stato attuale della ricerca esiste un solo DL-reasoner che opera su ontologie espresse in *DL-Lite*$_\mathcal{A}$ (QuOnto/Mastro); tale strumento è pensato principalmente per il query answering ovvero il task di valutare una Conjunctive Query sul livello estensionale. Se fosse possibile valutare una conjunctive query booleana sul solo livello intensionale dell'ontologia, potremmo verificare proprietà (limitatamente a quelle esprimibili in CQ) tramite QuOnto, tuttavia questo non è possibile: QuOnto offre soltanto alcuni servizi di reasoning intensionale che sono limitati al check di sussunzione tra elementi dell'ontologia.

Poichè *DL-Lite*$_\mathcal{A}$ è un sottoinsieme decidibile della FOL, possiamo tradurre l'insieme di asserzioni intensionali che compongono la TBox in una teoria della FOL equivalente e utilizzare strumenti preesistenti come il theorem prover OTTER o il model finder MACE per verificare una qualsiasi proprietà dell'ontologia di partenza.

Tale procedimento di traduzione è diretto e automatizzabile quindi, comportando uno sforzo minimo, rende interessante questo approccio alla verifica di proprietà su ontologie.

4.1 Traduzione verso FOL

Atomic Concept: per ogni Atomic Concept $C \in A$ nell'alfabeto allora esiste un simbolo di predicato unario $C/1$. Esempio: Person/1.

Atomic Concept Attribute: per ogni Atomic Concept Attribute $a \in U_C$ nell'alfabeto allora esiste un simbolo di predicato binario $a/2$. Esem-

pio: name/2.

Atomic Role: per ogni Atomic Role $R \in P$ nell'alfabeto allora esiste un simbolo di predicato binario $r/2$. Esempio: cityOfBirth/2.

Atomic Role Attribute: per ogni Atomic Role Attribute $b \in U_R$ nell'alfabeto allora esiste un simbolo di predicato ternario $b/3$. Esempio: dateOfBirth/3.

Value Domain: per ogni Value Domain T_i esiste un simbolo di predicato unario con il nome del corrispondente data type $T_i/1$. Esempio: String/1.

Nel segiuto viene mostrato come si traducono in FOL le più significative intensional assertions *DL-Lite*$_\mathcal{A}$. Ovviamente data la sintassi al capitolo 2, paragrafi 2.1 e 2.1.2, è possibile scrivere altre asserzioni legali del linguaggio che si traducono in modo simile.

Attributo di concetto: $\delta(a) \sqsubseteq C$ e . Formula FOL:

$$\forall XY\ a(X,Y) \rightarrow C(X) \tag{4.1}$$

Tipo attributo: $\rho(a) \sqsubseteq T$. Formula FOL:

$$\forall XY\ a(X,Y) \rightarrow T(Y) \tag{4.2}$$

Molteplicità minima 1: $C \sqsubseteq \delta(a)$. Formula FOL:

$$\forall X\ C(X) \rightarrow \exists Y\ a(X,Y) \tag{4.3}$$

Molteplicità massima 1: $(funct\ a)$. Formula FOL:

$$\forall XYZ\ C(X) \wedge a(X,Y) \wedge a(X,Z) \rightarrow Y = Z \tag{4.4}$$

Ruolo tra due Concept: $\exists R \sqsubseteq C_1$ e $\exists R^- \sqsubseteq C_2$. Formula FOL:

$$\forall XY\ R(X,Y) \rightarrow (C_1(X) \wedge C_2(Y)) \tag{4.5}$$

Cardinalità minima 1: $C_1 \sqsubseteq \exists R$. Formula FOL:

$$\forall X\ C_1(X) \rightarrow \exists Y\ R(X,Y) \tag{4.6}$$

Cardinalità minima 1: $C_2 \sqsubseteq \exists R^-$. Formula FOL:

$$\forall X\ C_2(X) \rightarrow \exists Y\ R(Y,X) \tag{4.7}$$

Cardinalità massima 1: $(funct\ R)$. Formula FOL:

$$\forall XYZ\ C_1(X) \wedge R(X,Y) \wedge R(X,Z) \rightarrow Y = Z \tag{4.8}$$

Cardinalità massima 1: $(funct\ R^-)$. Formula FOL:

$$\forall XYZ\ C_2(X) \wedge R(Y,X) \wedge R(Z,X) \rightarrow Y = Z \tag{4.9}$$

Attributo di ruolo e tipo: $\delta(b) \sqsubseteq R$. Formula FOL:

$$\forall XYZ\ b(X,Y,Z) \rightarrow R(X,Y) \tag{4.10}$$

Tipo attributo: $\rho(b) \sqsubseteq T$. Formula FOL:

$$\forall XYZ\ b(X,Y,Z) \rightarrow T(Z) \tag{4.11}$$

Molteplicità minima 1: $R \sqsubseteq \delta(b)$. Formula FOL:

$$\forall XY\ R(X,Y) \rightarrow \exists Z\ b(X,Y,Z) \tag{4.12}$$

Molteplicità massima 1: $(funct\ b)$. Formula FOL:

$$\forall XYZW\ R(X,Y) \wedge b(X,Y,Z) \wedge b(X,Y,W) \rightarrow Z = W \tag{4.13}$$

ISA tra Concepts: $C_1 \sqsubseteq C_2$. Formula FOL:

$$\forall X\ C_1(X) \rightarrow C_2(X) \tag{4.14}$$

ISA tra Roles: $R_1 \sqsubseteq R_2$. Formula FOL:

$$\forall XY\ R_1(X,Y) \rightarrow R_2(X,Y) \tag{4.15}$$

Disgiunzione tra Concept: $C_1 \sqsubseteq \neg C_2$. Formula FOL:

$$\forall X\ C_1(X) \rightarrow \neg C_2(X) \tag{4.16}$$

Chiavi interne: $(id\ C\ a)$. Formula FOL:

$$\forall XYZ\ C(X) \wedge C(Y) \wedge a(X,Z) \wedge a(Y,Z) \rightarrow X = Y \tag{4.17}$$

Chiavi esterne: $(id\ C\ R)$. Formula FOL:

$$\forall XYZ\ C(X) \wedge C(Y) \wedge R(X,Z) \wedge R(Y,Z) \rightarrow X = Y \tag{4.18}$$

Chiavi miste: $(id\ C\ R, a)$. Formula FOL:

$$\forall XYZW\ C(X) \wedge C(Y) \wedge R(X,Z) \wedge R(Y,Z) \wedge a(X,W) \wedge a(Y,W) \rightarrow X = Y \tag{4.19}$$

I value-domain T_i sono tra loro disgiunti e le disgiunzioni vanno specificate con formule FOL del tipo:

$$\forall X\ Integer(X) \rightarrow \neg String(X) \tag{4.20}$$

Vanno specificate inoltre tutte le disgiunzioni tipi e concepts con formule FOL del tipo:

$$\forall X\ Person(X) \rightarrow \neg String(X) \tag{4.21}$$

Per ogni coppia di concepts che non è in relazione ISA o gerarchia va specificata la disgiunzione:

$$\forall X\ Person(X) \rightarrow \neg City(X) \tag{4.22}$$

4.2 Verifica automatica di proprietà

4.2.1 Nozioni di base

Sia Φ l'insieme di formule della FOL che descrivono il dominio. Una proprietà, rappresentata al primo ordine da una formula ψ, si dice *conseguenza implicita* di Φ se è una conseguenza logica cioè:

$$\Phi \models \psi \tag{4.23}$$

Proposizione 1 *Siano Φ e ψ due formule del prim'ordine. Allora le seguenti affermazioni sono equivalenti:*

- $\Phi \models \psi$
- $\Phi \rightarrow \psi$ *è una tautologia*
- $\Phi \wedge \neg\psi$ *è insoddisfacibile*

Per dimostrare la (4.23) si può quindi procedere dimostrando che:

$$\Phi \wedge \neg\psi \tag{4.24}$$

è insoddisfacibile.

Teorema 1 *(Teorema di Church) La logica del primo ordine è indecidibile, ovvero non esiste alcuna macchina di Turing in grado di determinare se una formula della logica del prim'ordine è valida oppure no.*

Non esistono quindi algoritmi corretti e completi per verificare la (4.24). Normalmente si procede per refutazione, ovvero cercando di dimostrare che $\Phi \wedge \neg\psi$ è insoddisfacibile. Questo procedimento può portare a un successo oppure no: nel primo caso si può affermare che $\Phi \models \psi$, mentre nel secondo caso non si può dire nulla.

4.2.2 Programmi disponibili

All'indirizzo web `http://www.cs.unm.edu/~mccune/otter/` è possibile effettuare il download di due software:

- OTTER: dimostratore di teoremi per FOL basato sulla tecnica della risoluzione
- MACE: ricercatore di modelli su dominio finito basato sull'algoritmo DPLL

Entrambi i sotware accettano in input files scritti con la stessa sintassi.

Esempio di utilizzo Sia Φ l'insieme di formule che descrive la base di conoscenza e ψ la proprietà da verificare. Allora nel file di input si scriveranno tutte le formule di Φ e infine $\neg\psi$.

OTTER cercherà di dimostrare che $\Phi \wedge \neg\psi$ è insoddisfacibile, cercando di generare una clausola vuota per risoluzione. Se ci riesce darà in output *That finishes the proof of the theorem.* Si potrà quindi affermare che ψ è logicamente implicata dalla base di conoscenza, ovvero $\Phi \models \psi$.

Se OTTER non termina si può provare allora a cercare un controesempio passando lo stesso input a MACE che cercherà un modello (su un dominio di interpretazione di dimensione finita) per $\Phi \wedge \neg\psi$.

Quindi se voglio verificare che

$$\Phi \models \psi$$

provo ad utilizzare OTTER.
Se invece voglio verificare che

$$\Phi \not\models \psi$$

cerco un modello con MACE.

4.2.3 Proprietà interessanti

Consistenza di un Concept: Un concept C si dice consistente se esiste almeno una istanziatione di $\mathcal{T}$ in cui C ha un numero di istanze maggio di zero, cioè quando:

$$\Phi \not\models \forall X\ \neg C(X)$$

Sussunzione tra Concept: Un concept C_1 sussume un concept C_2 se:

$$\Phi \models \forall X\ C_2(X) \rightarrow C_1(X)$$

Equivalenza tra Concept: Un concept C_1 è equivalente a un concept C_2 se:

$$\gamma_1 :\ \forall X\ C_2(X) \rightarrow C_1(X)$$

$$\gamma_2 :\ \forall X\ C_1(X) \rightarrow C_2(X)$$

$$\Phi \models \gamma_1 \wedge \gamma_2$$

Capitolo 5

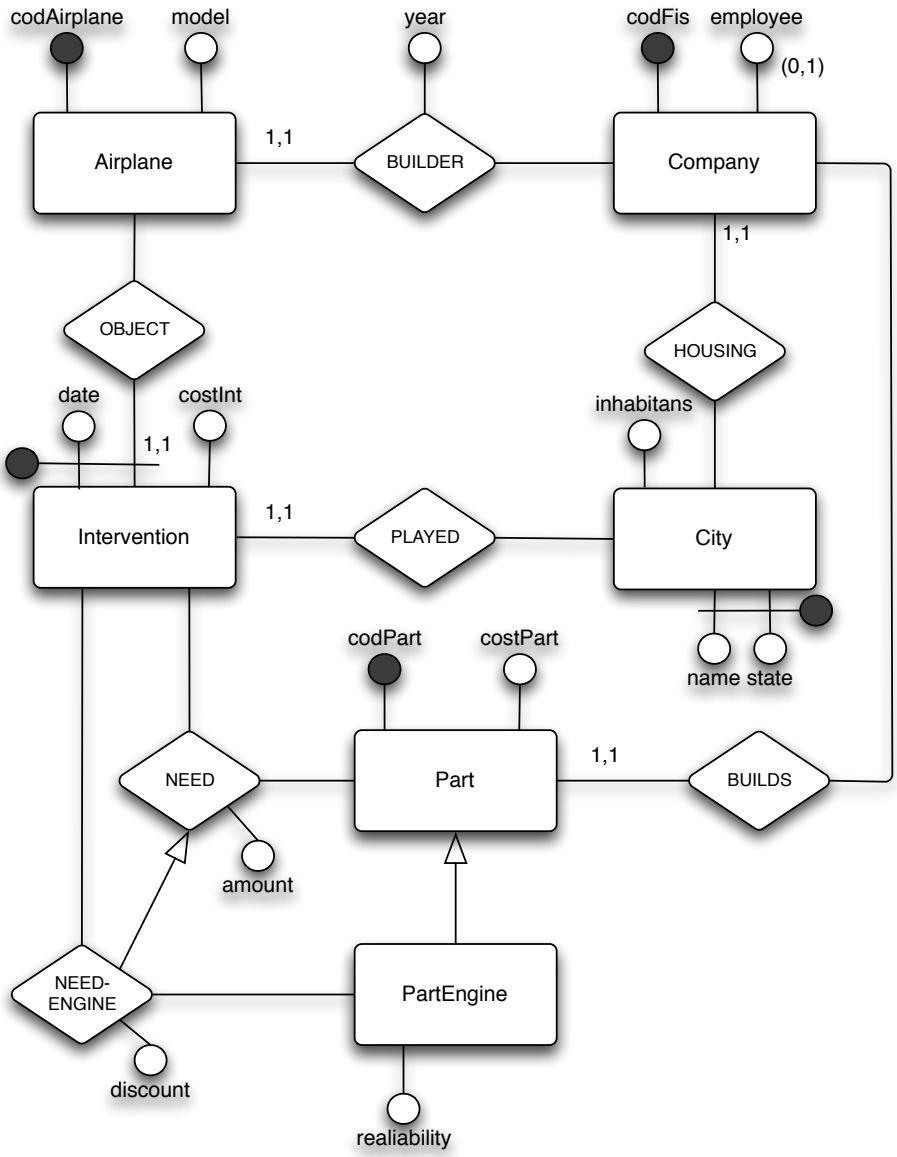
Case Studies: esempi pratici

5.1 Esame Basi di Dati 15/09/2005

5.1.1 Specifica

Si richiede di progettare lo schema concettuale Entità-Relazione di un'applicazione relativa agli interventi di manutenzione su aerei. Di ogni aereo interessa il codice (identificatore), il modello, la ditta costruttrice, e l'anno di costruzione. Di ogni ditta costruttrice interessa il codice fiscale (identificatore), la città in cui ha sede, ed il numero di dipendenti (quest'ultima informazione può però non essere disponibile). Di ogni città interessa il nome (unico nell'ambito dello stato), lo stato ed il numero di abitanti. Gli aerei sono soggetti ad interventi di manutenzione: di ogni intervento interessa l'aereo oggetto dell'intervento, la data, il costo di manodopera dell'intervento e la città in cui si [è svolto. Un aereo non può essere oggetto di più di un intervento di manutenzione al giorno. Un intervento di manutenzione può richiedere la sostituzione di pezzi. Di ogni pezzo interessa il codice (identificatore), il costo unitario, e la ditta costruttrice. Se il pezzo è un pezzo del motore, interessa sapere anche l'indice di affidabilità (intero positivo). Infine, per ogni pezzo oggetto di sostituzione nell'ambito di un intervento di manutenzione interessa sapere la quantità sostituita (ad esempio, in un intervento di manutenzione può essere necessario sostituire 3 poggiatesta dell'aereo), e nel caso di pezzo di motore, lo sconto praticato sul prezzo unitario.

5.1.2 Diagramma ER



5.1.3 Ontologia *DL-Lite*$_{\mathcal{A}}$

Atomic Concepts:
Airplane
Company
City
Intervention
Part

$PartEngine$

Atomic Roles:
$BUILDER$
$OBJECT$
$NEED$
$NEED - ENGINE$
$PLAYED$
$HOUSING$
$BUILDS$

Atomic Concept Attributes:
$codAirplane\ model$
$codFis$
$employees$
$data$
$costInt$
$codPart$
$costPart$
$reliability$
$name$
$state$
$inhabitants$

Atomic Role Attributes:
$year$
$amount$
$discount$

TBox Assertions:
$PartEngine \sqsubseteq Part$
$NEED - ENGINE \sqsubseteq NEED$

$\exists OBJECT \sqsubseteq Intervention$
$\exists OBJECT^- \sqsubseteq Airplane$
$\exists NEED \sqsubseteq Intervention$
$\exists NEED^- \sqsubseteq Part$
$\exists NEED - ENGINE \sqsubseteq Intervention$
$\exists NEED - ENGINE^- \sqsubseteq PartEngine$
$\exists PLAYED \sqsubseteq Intervention$
$\exists PLAYED^- \sqsubseteq City$
$\exists BUILDER \sqsubseteq Airplane$
$\exists BUILDER^- \sqsubseteq Company$
$\exists HOUSING \sqsubseteq Company$

$\exists HOUSING^{-} \sqsubseteq City$
$\exists BUILDS \sqsubseteq Part$
$\exists BUILDS^{-} \sqsubseteq Company$

$Airplane \sqsubseteq \exists BUILDER$
$Intervention \sqsubseteq \exists OBJECT$
$Intervention \sqsubseteq \exists PLAYED$
$Company \sqsubseteq \exists HOUSING$
$Part \sqsubseteq \exists BUILDS$

$(funct\ BUILDER)$
$(funct\ OBJECT)$
$(funct\ PLAYED)$
$(funct\ HOUSING)$
$(funct\ BUILDS)$

$\rho(codAirplane) \sqsubseteq xsd : string$
$\rho(model) \sqsubseteq xsd : string$
$\rho(data) \sqsubseteq xsd : date$
$\rho(costInt) \sqsubseteq xsd : float$
$\rho(codFis) \sqsubseteq xsd : string$
$\rho(employees) \sqsubseteq xsd : integer$
$\rho(codPart) \sqsubseteq xsd : string$
$\rho(costPart) \sqsubseteq xsd : float$
$\rho(name) \sqsubseteq xsd : string$
$\rho(state) \sqsubseteq xsd : string$
$\rho(inhabitants) \sqsubseteq xsd : integer$
$\rho(reliability) \sqsubseteq xsd : integer$

$\delta(codAirplane) \sqsubseteq Airplane$
$\delta(model) \sqsubseteq Airplane$
$\delta(data) \sqsubseteq Intervention$
$\delta(costInt) \sqsubseteq Intervention$
$\delta(codFis) \sqsubseteq Company$
$\delta(employees) \sqsubseteq Company$
$\delta(codPart) \sqsubseteq Part$
$\delta(costPart) \sqsubseteq Part$
$\delta(name) \sqsubseteq City$
$\delta(state) \sqsubseteq City$
$\delta(inhabitants) \sqsubseteq City$
$\delta(reliability) \sqsubseteq PartEngine$

$(funct\ codAirplane)$
$(funct\ model)$

$(funct\ data)$
$(funct\ costInt)$
$(funct\ codFis)$
$(funct\ employees)$
$(funct\ codPart)$
$(funct\ costPart)$
$(funct\ name)$
$(funct\ state)$
$(funct\ inhabitants)$
$(funct\ reliability)$

$Airplane \sqsubseteq \delta(codAirplane)$
$Airplane \sqsubseteq \delta(model)$
$Intervention \sqsubseteq \delta(data)$
$Intervention \sqsubseteq \delta(costInt)$
$Company \sqsubseteq \delta(codFis)$
$Company \sqsubseteq \delta(employees)$
$Part \sqsubseteq \delta(codPart)$
$Part \sqsubseteq \delta(costPart)$
$City \sqsubseteq \delta(name)$
$City \sqsubseteq \delta(state)$
$City \sqsubseteq \delta(inhabitants)$
$PartEngine \sqsubseteq \delta(reliability)$

$\rho(year) \sqsubseteq xsd : integer$
$\rho(amount) \sqsubseteq xsd : integer$
$\rho(discount) \sqsubseteq xsd : float$

$\delta(year) \sqsubseteq BUILDER$
$\delta(amount) \sqsubseteq NEED$
$\delta(discount) \sqsubseteq NEED - ENGINE$

$(funct\ year)$
$(funct\ amount)$
$(funct\ discount)$

$BUILDER \sqsubseteq \delta(year)$
$NEED \sqsubseteq \delta(amount)$
$NEED - ENGINE \sqsubseteq \delta(discount)$

$id(Airplane \quad codAirplane)$
$id(Intervention \quad data,\ OBJECT)$
$id(Company \quad codFis)$
$id(Part \quad codPart)$

$id(City \quad name,\ state)$

5.1.4 Traduzione in FOL

Simboli di predicato:
$Airplane/1$
$Company/1$
$City/1$
$Intervention/1$
$Part/1$
$PartEngine/1$
$BUILDER/2$
$OBJECT/2$
$NEED/2$
$NEED-ENGINE/2$
$PLAYED/2$
$HOUSING/2$
$BUILDS/2$
$codAirplane/2$
$model/2$
$codFis/2$
$employees/2$
$data/2$
$costInt/2$
$codPart/2$
$costPart/2$
$reliability/2$
$name/2$
$state/2$
$inhabitants/2$
$year/3$
$amount/3$
$discount/3$
$String/1$
$Integer/1$
$Float/1$
$Date/1$

Formule FOL:
$\forall X\ PartEngine(X) \rightarrow Part(X)$
$\forall XY\ NEED-ENGINE(X,Y) \rightarrow NEED(X,Y)$

$\forall XY\ OBJECT(X,Y) \rightarrow (Intervention(X) \wedge Airplane(Y))$

$\forall XY\ NEED(X,Y) \rightarrow (Intervention(X) \wedge Part(Y))$
$\forall XY\ NEED-ENGINE(X,Y) \rightarrow (Intervention(X) \wedge PartEngine(Y))$
$\forall XY\ PLAYED(X,Y) \rightarrow (Intervention(X) \wedge City(Y))$
$\forall XY\ BUILDER(X,Y) \rightarrow (Airplane(X) \wedge Company(Y))$
$\forall XY\ HOUSING(X,Y) \rightarrow (Company(X) \wedge City(Y))$
$\forall XY\ BUILDS(X,Y) \rightarrow (Part(X) \wedge Company(Y))$

$\forall X\ Airplane(X) \rightarrow \exists Y\ BUILDER(X,Y)$
$\forall X\ Intervention(X) \rightarrow \exists Y\ OBJECT(X,Y)$
$\forall X\ Intervention(X) \rightarrow \exists Y\ PLAYED(X,Y)$
$\forall X\ Company(X) \rightarrow \exists Y\ HOUSING(X,Y)$
$\forall X\ Part(X) \rightarrow \exists Y\ BUILDS(X,Y)$

$\forall XYZ\ Airplane(X) \wedge BUILDER(X,Y) \wedge BUILDER(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Intervention(X) \wedge OBJECT(X,Y) \wedge OBJECT(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Intervention(X) \wedge PLAYED(X,Y) \wedge PLAYED(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Company(X) \wedge HOUSING(X,Y) \wedge HOUSING(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Part(X) \wedge BUILDS(X,Y) \wedge BUILDS(X,Z) \rightarrow Y = Z$

$\forall XY\ Airplane(X) \wedge codAirplane(X,Y) \rightarrow String(Y)$
$\forall XY\ Airplane(X) \wedge model(X,Y) \rightarrow String(Y)$
$\forall XY\ Intervention(X) \wedge data(X,Y) \rightarrow Date(Y)$
$\forall XY\ Intervention(X) \wedge costInt(X,Y) \rightarrow Float(Y)$
$\forall XY\ Company(X) \wedge codFis(X,Y) \rightarrow String(Y)$
$\forall XY\ Company(X) \wedge employees(X,Y) \rightarrow Integer(Y)$
$\forall XY\ Part(X) \wedge codPart(X,Y) \rightarrow String(Y)$
$\forall XY\ Part(X) \wedge costPart(X,Y) \rightarrow Float(Y)$
$\forall XY\ City(X) \wedge name(X,Y) \rightarrow String(Y)$
$\forall XY\ City(X) \wedge state(X,Y) \rightarrow String(Y)$
$\forall XY\ City(X) \wedge inhabitants(X,Y) \rightarrow Integer(Y)$
$\forall XY\ PartEngine(X) \wedge reliability(X,Y) \rightarrow Integer(Y)$

$\forall XYZ\ Airplane(X) \wedge codAirplane(X,Y) \wedge codAirplane(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Airplane(X) \wedge model(X,Y) \wedge model(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Intervention(X) \wedge data(X,Y) \wedge data(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Intervention(X) \wedge costInt(X,Y) \wedge costInt(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Company(X) \wedge codFis(X,Y) \wedge codFis(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Company(X) \wedge employees(X,Y) \wedge employees(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Part(X) \wedge codPart(X,Y) \wedge codPart(X,Z) \rightarrow Y = Z$
$\forall XYZ\ Part(X) \wedge costPart(X,Y) \wedge costPart(X,Z) \rightarrow Y = Z$
$\forall XYZ\ City(X) \wedge name(X,Y) \wedge name(X,Z) \rightarrow Y = Z$
$\forall XYZ\ City(X) \wedge state(X,Y) \wedge state(X,Z) \rightarrow Y = Z$
$\forall XYZ\ City(X) \wedge inhabitants(X,Y) \wedge inhabitants(X,Z) \rightarrow Y = Z$
$\forall XYZ\ PartEngine(X) \wedge reliability(X,Y) \wedge reliability(X,Z) \rightarrow Y = Z$

$\forall X\ Airplane(X) \rightarrow \exists Y\ codAirplane(X,Y)$
$\forall X\ Airplane(X) \rightarrow \exists Y\ model(X,Y)$
$\forall X\ Intervention(X) \rightarrow \exists Y\ data(X,Y)$
$\forall X\ Intervention(X) \rightarrow \exists Y\ costInt(X,Y)$
$\forall X\ Company(X) \rightarrow \exists Y\ codFis(X,Y)$
$\forall X\ Company(X) \rightarrow \exists Y\ employees(X,Y)$
$\forall X\ Part(X) \rightarrow \exists Y\ codPart(X,Y)$
$\forall X\ Part(X) \rightarrow \exists Y\ costPart(X,Y)$
$\forall X\ City(X) \rightarrow \exists Y\ name(X,Y)$
$\forall X\ City(X) \rightarrow \exists Y\ state(X,Y)$
$\forall X\ City(X) \rightarrow \exists Y\ inhabitants(X,Y)$
$\forall X\ PartEngine(X) \rightarrow \exists Y\ reliability(X,Y)$

$\forall XYZ\ BUILDER(X,Y) \wedge year(X,Y,Z) \rightarrow Integer(Z)$
$\forall XYZ\ NEED(X,Y) \wedge amount(X,Y,Z) \rightarrow Integer(Z)$
$\forall XYZ\ NEED-ENGINE(X,Y) \wedge discount(X,Y,Z) \rightarrow Float(Z)$

$\forall XYZW\ BUILDER(X,Y) \wedge year(X,Y,Z) \wedge year(X,Y,W) \rightarrow Z = W$
$\forall XYZW\ NEED(X,Y) \wedge amount(X,Y,Z) \wedge amount(X,Y,W) \rightarrow Z = W$
$\forall XYZW\ NEED{-}ENGINE(X,Y) \wedge discount(X,Y,Z) \wedge discount(X,Y,W) \rightarrow Z = W$

$\forall XY\ BUILDER(X,Y) \rightarrow \exists Z\ year(X,Y,Z)$
$\forall XY\ NEED(X,Y) \rightarrow \exists Z\ amount(X,Y,Z)$
$\forall XY\ NEED-ENGINE(X,Y) \rightarrow \exists Z\ discount(X,Y,Z)$

$\forall XYZ\ Airplane(X) \wedge Airplane(Y) \wedge codAirplane(X,Z) \wedge codAirplane(Y,Z) \rightarrow X = Y$
$\forall XYZW\ Intervention(X) \wedge Intervention(Y) \wedge OBJECT(X,Z) \wedge OBJECT(Y,Z) \wedge data(X,W) \wedge data(Y,W) \rightarrow X = Y$
$\forall XYZ\ Company(X) \wedge Company(Y) \wedge codFis(X,Z) \wedge codFis(Y,Z) \rightarrow X = Y$
$\forall XYZ\ Part(X) \wedge Part(Y) \wedge codPart(X,Z) \wedge codPart(Y,Z) \rightarrow X = Y$
$\forall XYZW\ City(X) \wedge City(Y) \wedge name(X,Z) \wedge name(Y,Z) \wedge state(X,W) \wedge state(Y,W) \rightarrow X = Y$

$\forall X\ String(X) \rightarrow \neg Integer(X)$
$\forall X\ String(X) \rightarrow \neg Float(X)$
$\forall X\ String(X) \rightarrow \neg Date(X)$
$\forall X\ Integer(X) \rightarrow \neg Float(X)$
$\forall X\ Integer(X) \rightarrow \neg Date(X)$
$\forall X\ Float(X) \rightarrow \neg Date(X)$

$\forall X\ Airplane(X) \rightarrow \neg Company(X)$
$\forall X\ Airplane(X) \rightarrow \neg City(X)$
$\forall X\ Airplane(X) \rightarrow \neg Intervention(X)$
$\forall X\ Airplane(X) \rightarrow \neg Part(X)$
$\forall X\ Airplane(X) \rightarrow \neg PartEngine(X)$
$\forall X\ Company(X) \rightarrow \neg City(X)$
$\forall X\ Company(X) \rightarrow \neg Intervention(X)$
$\forall X\ Company(X) \rightarrow \neg Part(X)$
$\forall X\ Company(X) \rightarrow \neg PartEngine(X)$
$\forall X\ City(X) \rightarrow \neg Intervention(X)$
$\forall X\ City(X) \rightarrow \neg Part(X)$
$\forall X\ City(X) \rightarrow \neg PartEngine(X)$
$\forall X\ Intervention(X) \rightarrow \neg Part(X)$
$\forall X\ Intervention(X) \rightarrow \neg PartEngine(X)$

$\forall X\ Airplane(X) \rightarrow \neg String(X)$
$\forall X\ Airplane(X) \rightarrow \neg Integer(X)$
$\forall X\ Airplane(X) \rightarrow \neg Float(X)$
$\forall X\ Airplane(X) \rightarrow \neg Date(X)$
$\forall X\ Company(X) \rightarrow \neg String(X)$
$\forall X\ Company(X) \rightarrow \neg Integer(X)$
$\forall X\ Company(X) \rightarrow \neg Float(X)$
$\forall X\ Company(X) \rightarrow \neg Date(X)$
$\forall X\ City(X) \rightarrow \neg String(X)$
$\forall X\ City(X) \rightarrow \neg Integer(X)$
$\forall X\ City(X) \rightarrow \neg Float(X)$
$\forall X\ City(X) \rightarrow \neg Date(X)$
$\forall X\ Intervention(X) \rightarrow \neg String(X)$
$\forall X\ Intervention(X) \rightarrow \neg Integer(X)$
$\forall X\ Intervention(X) \rightarrow \neg Float(X)$
$\forall X\ Intervention(X) \rightarrow \neg Date(X)$
$\forall X\ Part(X) \rightarrow \neg String(X)$
$\forall X\ Part(X) \rightarrow \neg Integer(X)$
$\forall X\ Part(X) \rightarrow \neg Float(X)$
$\forall X\ Part(X) \rightarrow \neg Date(X)$

5.1.5 Input Otter/mace

```
set(auto).
formula_list(usable).

%% TBOX assertions
```

```
all X (PartEngine(X) -> Part(X)).
all XY (NEEDENGINE(X,Y) -> NEED(X,Y)).

all XY (OBJECT(X,Y) -> (Intervention(X) & Airplane(Y))).
all XY (NEED(X,Y) -> (Intervention(X) & Part(Y))).
all XY (NEEDENGINE(X,Y) -> (Intervention(X) & PartEngine(Y))).
all XY (PLAYED(X,Y) -> (Intervention(X) & City(Y))).
all XY (BUILDER(X,Y) -> (Airplane(X) & Company(Y))).
all XY (HOUSING(X,Y) -> (Company(X) & City(Y))).
all XY (BUILDS(X,Y) -> (Part(X) & Company(Y))).

all X (Airplane(X) -> (exists Y BUILDER(X,Y))).
all X (Intervention(X) -> (exists Y OBJECT(X,Y))).
all X (Intervention(X) -> (exists Y PLAYED(X,Y))).
all X (Company(X) -> (exists Y HOUSING(X,Y))).
all X (Part(X) -> (exists Y BUILDS(X,Y))).

all XYZ (Airplane(X) & BUILDER(X,Y) & BUILDER(X,Z) -> Y=Z).
all XYZ (Intervention(X) & OBJECT(X,Y) & OBJECT(X,Z) -> Y=Z).
all XYZ (Intervention(X) & PLAYED(X,Y) & PLAYED(X,Z) -> Y=Z).
all XYZ (Company(X) & HOUSING(X,Y) & HOUSING(X,Z) -> Y=Z).
all XYZ (Part(X) & BUILDS(X,Y) & BUILDS(X,Z) -> Y=Z).

all XY (Airplane(X) & codAirplane(X,Y) -> String(Y)).
all XY (Airplane(X) & model(X,Y) -> String(Y)).
all XY (Intervention(X) & data(X,Y) -> Date(Y)).
all XY (Intervention(X) & costInt(X,Y) -> Float(Y)).
all XY (Company(X) & codFis(X,Y) -> String(Y)).
all XY (Company(X) & employees(X,Y) -> Integer(Y)).
all XY (Part(X) & codPart(X,Y) -> String(Y)).
all XY (Part(X) & costPart(X,Y) -> Float(Y)).
all XY (City(X) & name(X,Y) -> String(Y)).
all XY (City(X) & state(X,Y) -> String(Y)).
all XY (City(X) & inhabitants(X,Y) -> Integer(Y)).
all XY (PartEngine(X) & reliability(X,Y) -> Integer(Y)).

all XYZ (Airplane(X) & codAirplane(X,Y) & codAirplane(X,Z) -> Y=Z).
all XYZ (Airplane(X) & model(X,Y) & model(X,Z) -> Y=Z).
all XYZ (Intervention(X) & data(X,Y) & data(X,Z) -> Y=Z).
all XYZ (Intervention(X) & costInt(X,Y) & costInt(X,Z) -> Y=Z).
all XYZ (Company(X) & codFis(X,Y) & codFis(X,Z) -> Y=Z).
all XYZ (Company(X) & employees(X,Y) & employees(X,Z) -> Y=Z).
all XYZ (Part(X) & codPart(X,Y) & codPart(X,Z) -> Y=Z).
```

```
all XYZ (Part(X) & costPart(X,Y) & costPart(X,Z) -> Y=Z).
all XYZ (City(X) & name(X,Y) & name(X,Z) -> Y=Z).
all XYZ (City(X) & state(X,Y) & state(X,Z) -> Y=Z).
all XYZ (City(X) & inhabitants(X,Y) & inhabitants(X,Z) -> Y=Z).
all XYZ (PartEngine(X) & reliability(X,Y) & reliability(X,Z) -> Y=Z).

all X (Airplane(X) -> (exists Y codAirplane(X,Y))).
all X (Airplane(X) -> (exists Y model(X,Y))).
all X (Intervention(X) -> (exists Y data(X,Y))).
all X (Intervention(X) -> (exists Y costInt(X,Y))).
all X (Company(X) -> (exists Y codFis(X,Y))).
all X (Company(X) -> (exists Y employees(X,Y))).
all X (Part(X) -> (exists Y codPart(X,Y))).
all X (Part(X) -> (exists Y costPart(X,Y))).
all X (City(X) -> (exists Y name(X,Y))).
all X (City(X) -> (exists Y state(X,Y))).
all X (City(X) -> (exists Y inhabitants(X,Y))).
all X (PartEngine(X) -> (exists Y reliability(X,Y))).

all XYZ (BUILDER(X,Y) & year(X,Y,Z) -> Integer(Z)).
all XYZ (NEED(X,Y) & amount(X,Y,Z) -> Integer(Z)).
all XYZ (NEEDENGINE(X,Y) & discount(X,Y,Z) -> Float(Z)).

all XYZW (BUILDER(X,Y) & year(X,Y,Z) & year(X,Y,W) -> Z=W).
all XYZW (NEED(X,Y) & amount(X,Y,Z) & amount(X,Y,W) -> Z=W).
all XYZW (NEEDENGINE(X,Y) & discount(X,Y,Z) & discount(X,Y,W)
-> Z=W).

all XY (BUILDER(X,Y) -> (exists Z year(X,Y,Z))).
all XY (NEED(X,Y) -> (exists Z amount(X,Y,Z))).
all XY (NEEDENGINE(X,Y) -> (exists Z discount(X,Y,Z))).

all XYZ (Airplane(X) & Airplane(Y) & codAirplane(X,Z) & codAirplane(Y,Z)
-> X = Y).
all XYZW (Intervention(X) & Intervention(Y) & OBJECT(X,Z) & OBJECT(Y,Z)
  & data(X,W) & data(Y,W) -> X = Y).
all XYZ (Company(X) & Company(Y) & codFis(X,Z) & codFis(Y,Z) -> X = Y).
all XYZ (Part(X) & Part(Y) & codPart(X,Z) & codPart(Y,Z) -> X = Y).
all XYZW (City(X) & City(Y) & name(X,Z) & name(Y,Z) & state(X,W)
& state(Y,W) -> X = Y).

all X (String(X) -> -Integer(X)).
all X (String(X) -> -Float(X)).
all X (String(X) -> -Date(X)).
```

```
all X (Integer(X) -> -Float(X)).
all X (Integer(X) -> -Date(X)).
all X (Float(X) -> -Date(X)).

all X (Airplane(X) -> -Company(X)).
all X (Airplane(X) -> -City(X)).
all X (Airplane(X) -> -Intervention(X)).
all X (Airplane(X) -> -Part(X)).
all X (Airplane(X) -> -PartEngine(X)).
all X (Company(X) -> -City(X)).
all X (Company(X) -> -Intervention(X)).
all X (Company(X) -> -Part(X)).
all X (Company(X) -> -PartEngine(X)).
all X (City(X) -> -Intervention(X)).
all X (City(X) -> -Part(X)).
all X (City(X) -> -PartEngine(X)).
all X (Intervention(X) -> -Part(X)).
all X (Intervention(X) -> -PartEngine(X)).

all X (Airplane(X) -> -String(X)).
all X (Airplane(X) -> -Integer(X)).
all X (Airplane(X) -> -Float(X)).
all X (Airplane(X) -> -Date(X)).
all X (Company(X) -> -String(X)).
all X (Company(X) -> -Integer(X)).
all X (Company(X) -> -Float(X)).
all X (Company(X) -> -Date(X)).
all X (City(X) -> -String(X)).
all X (City(X) -> -Integer(X)).
all X (City(X) -> -Float(X)).
all X (City(X) -> -Date(X)).
all X (Intervention(X) -> -String(X)).
all X (Intervention(X) -> -Integer(X)).
all X (Intervention(X) -> -Float(X)).
all X (Intervention(X) -> -Date(X)).
all X (Part(X) -> -String(X)).
all X (Part(X) -> -Integer(X)).
all X (Part(X) -> -Float(X)).
all X (Part(X) -> -Date(X)).

%% CONSEGUENZE LOGICHE (VERE: USARE OTTER)
%PartEngine eredita l'attributo codPart di tipo stringa
C1 <-> (all XY (PartEngine(X) & codPart(X,Y) -> String(Y))).
```

```
%% CONSEGUENZE LOGICHE (FALSE: USARE MACE)
%Consistenza Concept Intervention
C2 <-> (all X (-Intervention(X))).

%% TESI
-(
C1
%C2
).

end_of_list.
```

5.1.6 Verifica di proprietà

Per dimostrare la proprietà che $\Phi \models C1$ si usa OTTER invocando:
`$ ./otter < casestudy1.ott`
In output abbiamo:

```
---------------- PROOF ----------------

1 [] -PartEngine(x)|Part(x).
38 [] -Part(X)| -codPart(X,Y)|String(Y).
140 [] C1| -String(Y).
142 [] -C1.
143 [] C1|PartEngine(X).
144 [copy,143,unit_del,142] PartEngine(X).
145 [] C1|codPart(X,Y).
146 [copy,145,unit_del,142] codPart(X,Y).
149 [hyper,144,1] Part(X).
153 [hyper,146,38,149] String(Y).
159 [hyper,153,140] C1.
160 [binary,159.1,142.1] $F.

------------ end of proof -------------

That finishes the proof of the theorem.
```

Per dimostrare invece che $\Phi \not\models C2$ si usa MACE invocando:
`$ ./mace2 -n 4 -p < casestudy1.ott`
In output abbiamo:

```
The set is satisfiable (1 model(s) found).

Intervention :
        0 1 2 3
    -----------
        T F F F
```

Capitolo 6

OntologyConverter

6.1 Software di traduzione automatica

OntologyConverter è uno strumento software che prende in input una TBox DL-LiteA descritta in formato XML secondo le stesse DTD's accettate dal reasoner **QuOnto/Mastro** (`http://wwww.dis.uniroma1.it/quonto`) e ne effettua la traduzione automatica in una base di conoscenza equivalente in First Order Logic, fornendo in output un file nel formato accettato da **Otter/Mace**.

6.1.1 Sintassi XML QuOnto

Di seguito viene riportata la DTD per la descrizione di una TBox *DL-Lite*$_{\mathcal{A}}$:

```
<!-- level 1 ********************************************-->

<!ELEMENT ontology (alphabet, tbox)>

<!-- level 2 ********************************************-->

<!ELEMENT alphabet (atomicC | atomicV | atomicR | atomicCA |
atomicRA)+>
<!ELEMENT tbox (inclusionAssertion | funct)*>

<!-- level 3 ********************************************-->

<!ELEMENT atomicC (#PCDATA)>
<!ELEMENT atomicV (#PCDATA)>
<!ELEMENT atomicR (#PCDATA)>
<!ELEMENT atomicCA (#PCDATA)>
<!ELEMENT atomicRA (#PCDATA)>
```

```
<!ELEMENT inclusionAssertion ((basicC, generalC+) |
(basicV, generalV+) | (basicR, generalR+) |
(atomicCA, generalCA+) | (atomicRA, generalRA+))>
<!ELEMENT funct (basicR | atomicCA | atomicRA)>

<!-- level 4 ********************************************-->

<!ELEMENT basicC (atomicC | exists | CADomain)>
<!ELEMENT generalC (topC | signedC | qualifiedExists |
CAQualifiedDomain | existsRAQualifiedDomain)>
<!ELEMENT basicV (atomicV | ARange)>
<!ELEMENT generalV (topD | signedV | predefinedV)>
<!ELEMENT basicR (atomicR | RADomain)>
<!ATTLIST basicR dir (direct | inverse) #REQUIRED>
<!ELEMENT generalR (signedR | RAQualifiedDomain)>
<!ELEMENT generalCA (atomicCA)>
<!ATTLIST generalCA sign (positive | negative) #REQUIRED>
<!ELEMENT generalRA (atomicRA)>
<!ATTLIST generalRA sign (positive | negative) #REQUIRED>

<!-- level 5 ********************************************-->

<!ELEMENT exists (basicR)>
<!ELEMENT CADomain (atomicCA)>
<!ELEMENT topC EMPTY>
<!ELEMENT signedC (basicC)>
<!ATTLIST signedC sign (positive | negative) #REQUIRED>
<!ELEMENT qualifiedExists (basicR, generalC)>
<!ELEMENT CAQualifiedDomain (atomicCA, generalV)>
<!ELEMENT existsRAQualifiedDomain (RAQualifiedDomain)>
<!ELEMENT ARange (atomicCA | atomicRA)>
<!ELEMENT topD EMPTY>
<!ELEMENT signedV (basicV)>
<!ATTLIST signedV sign (positive | negative) #REQUIRED>
<!ELEMENT predefinedV (#PCDATA)>
<!ELEMENT RADomain (atomicRA)>
<!ELEMENT signedR (basicR)>
<!ATTLIST signedR sign (positive | negative) #REQUIRED>
<!ELEMENT RAQualifiedDomain (atomicRA, generalV)>
<!ATTLIST RAQualifiedDomain dir (direct | inverse) #REQUIRED>

<!--********************************************************-->
```

6.1.2 Diagrammi UML delle classi

Le figure 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8 e 6.9 rappresentano i diagrammi UML del software OntologyConverver.

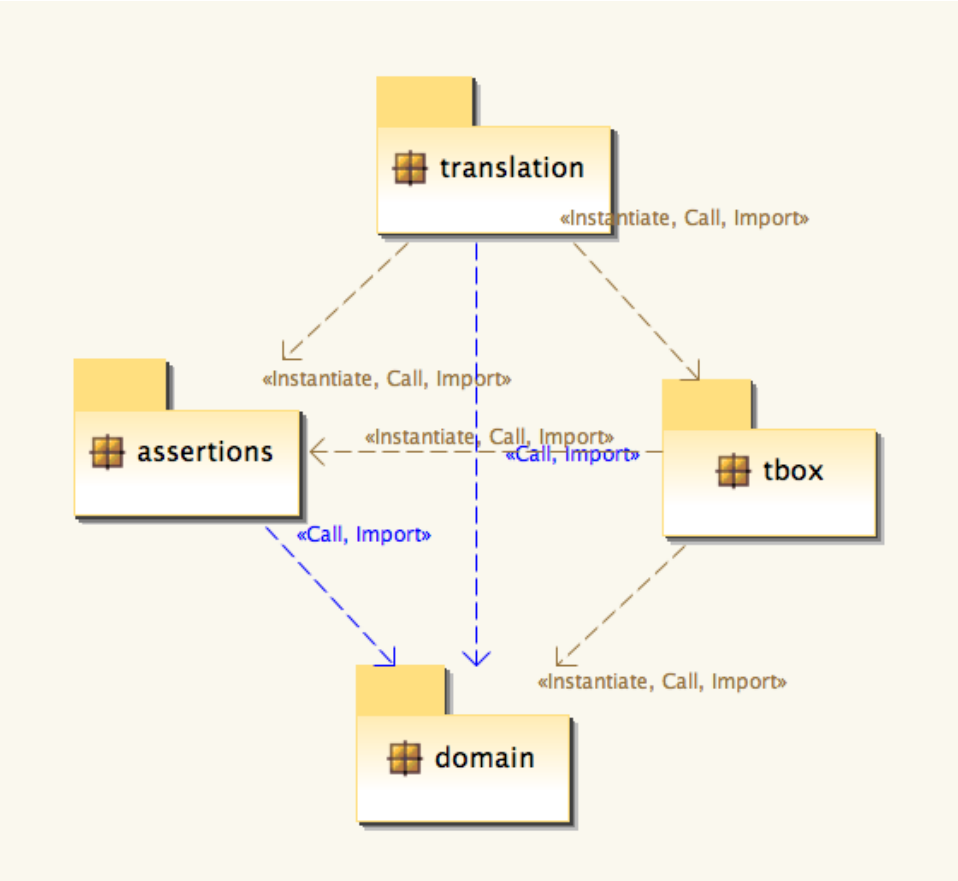


Figura 6.1: Diagramma dei Package

6.2 IMDB Ontology: un esempio

Mostriamo ora una reale applicazione del software OntologyConverter su una ontologia *DL-Lite*$_{\mathcal{A}}$ di dimensioni considerevoli.

L'ontologia descrive il dominio applicativo del sito web The Internet Movie Database (`http://wwww.imdb.com`), la più grande sorgente dati al mondo riguardante la cinematografia.

La TBox dell'ontologia consiste di:

- 11 Atomic Concepts
- 56 Atomic Concept Attributes

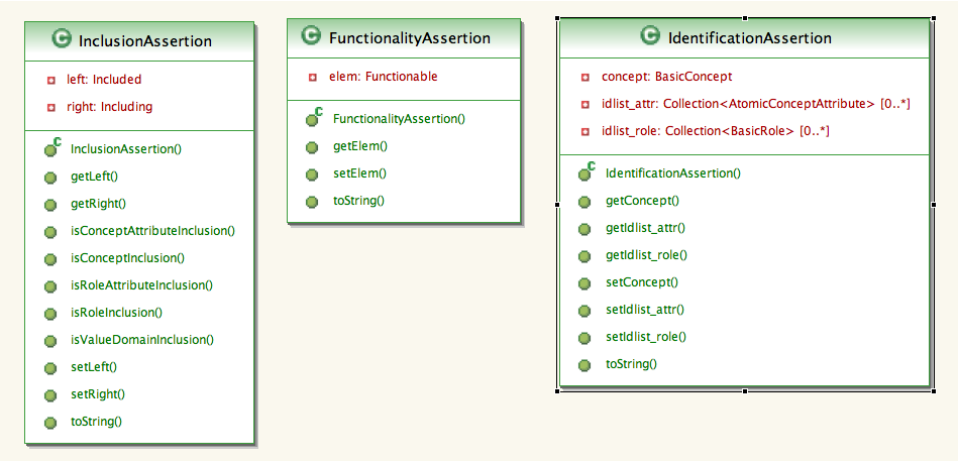


Figura 6.2: Classi del package assertions

Figura 6.3: Classi del package tbox

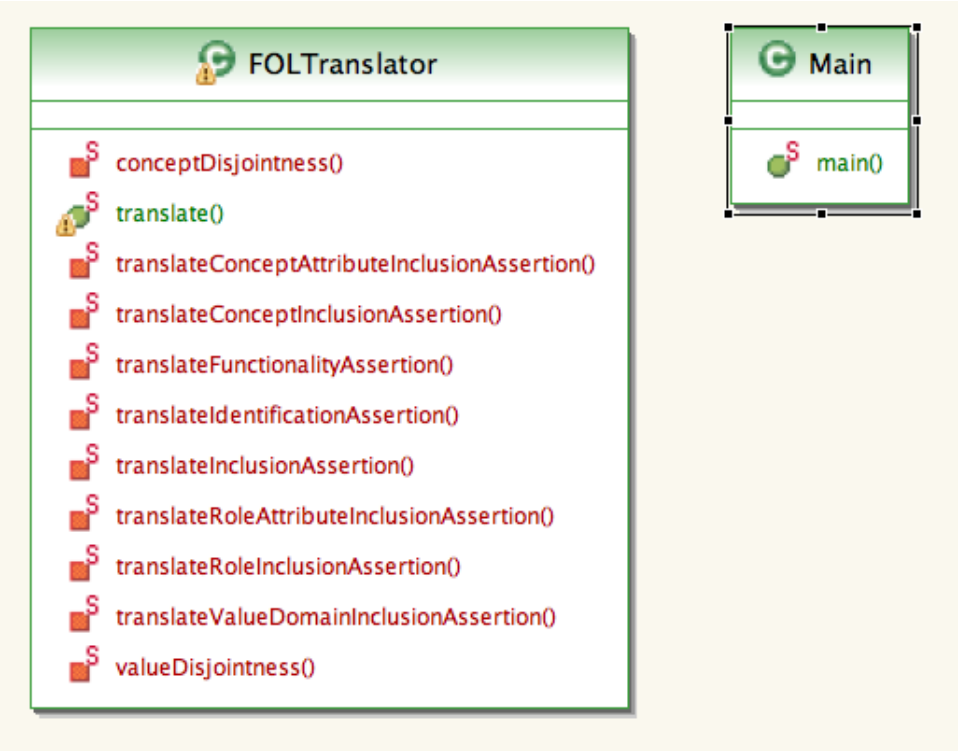


Figura 6.4: Classi del package translation

- 57 Atomic Roles
- 38 Atomic Role Attributes
- 470 Intensional Assertions

Tale ontologia è stata utilizzata per il testing delle performance di query answering di QuOnto/MastroI dopo l'opportuno mapping (in merito si veda l'articolo [9]) su un database relazione di 32 milioni di tuple.

In questa sede siamo interessati invece alla verifica delle proprietà statiche dell'ontologia, come spiegato al capitolo 4.

6.2.1 TBox IMDB

$PERSON \sqsubseteq \delta(name)$ $(functbirthDate)$
$\delta(name) \sqsubseteq PERSON$ $\delta(birthName) \sqsubseteq PERSON$
$(functname)$ $(functbirthName)$
$PERSON \sqsubseteq \delta(surname)$ $\delta(height) \sqsubseteq PERSON$
$\delta(surname) \sqsubseteq PERSON$ $(functheight)$
$(functsurname)$ $\delta(biography) \sqsubseteq PERSON$
$PERSON \sqsubseteq \delta(birthDate)$ $(functbiography)$
$\delta(birthDate) \sqsubseteq PERSON$ $\delta(trademark) \sqsubseteq PERSON$

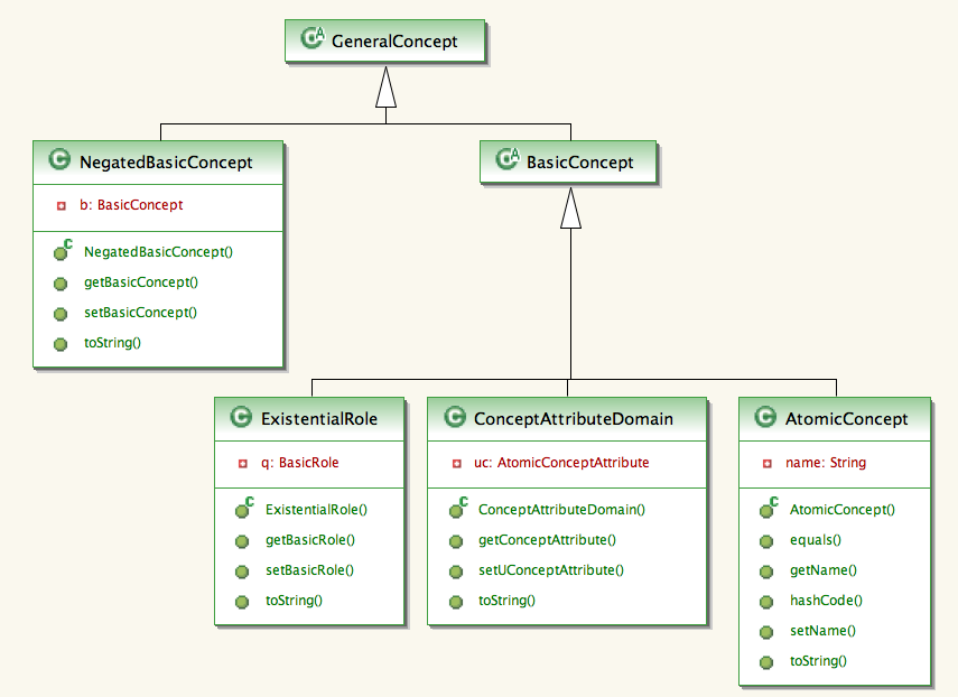


Figura 6.5: Classi del package domain: Concepts

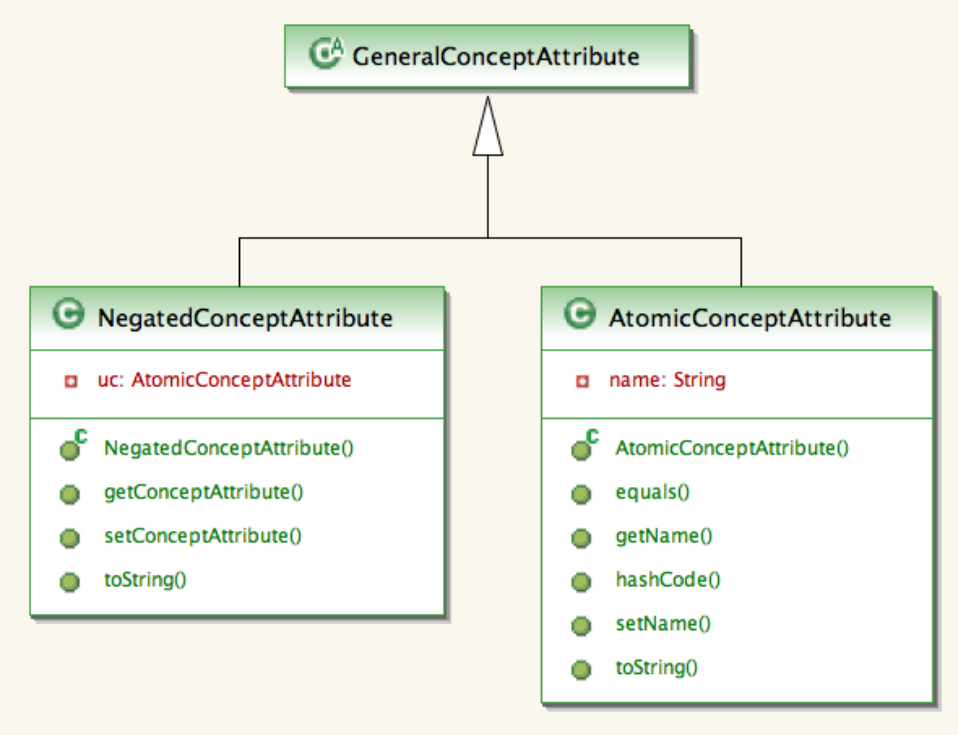


Figura 6.6: Classi del package domain: Concept Attributes

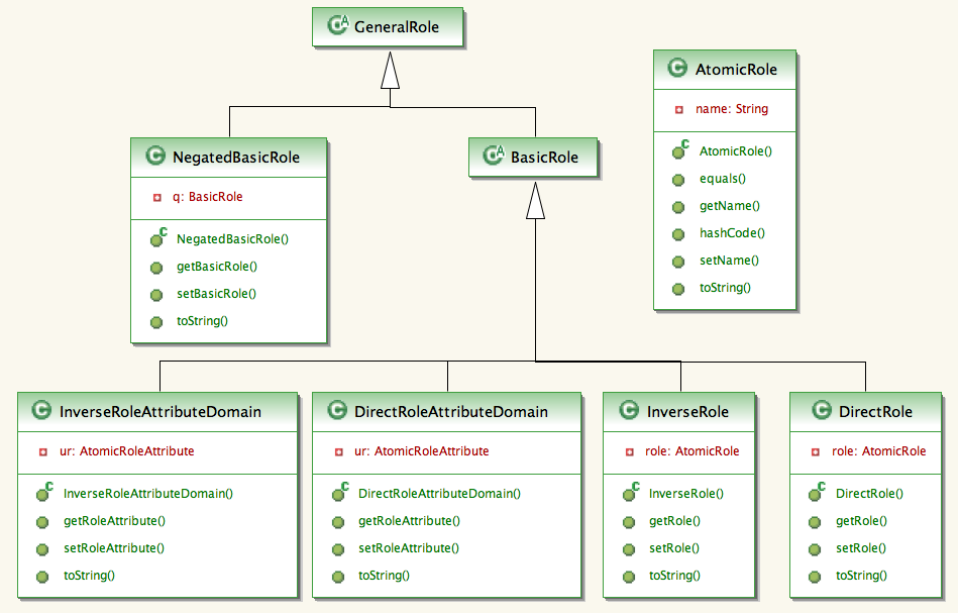


Figura 6.7: Classi del package domain: Roles

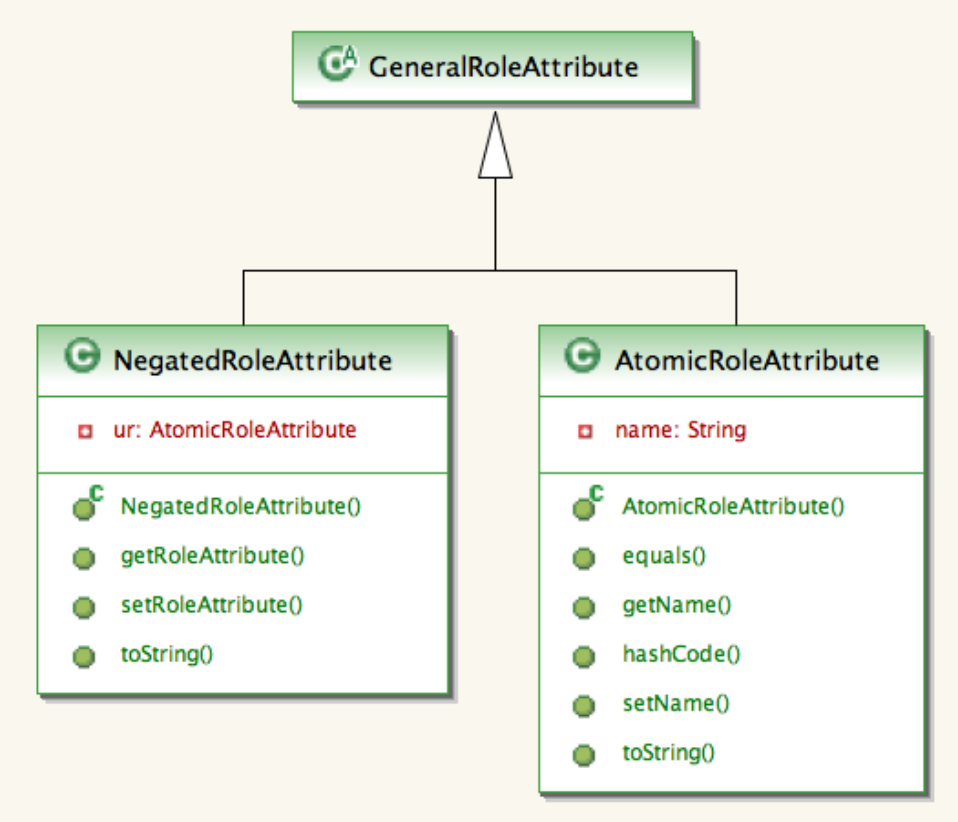


Figura 6.8: Classi del package domain: Role Attributes

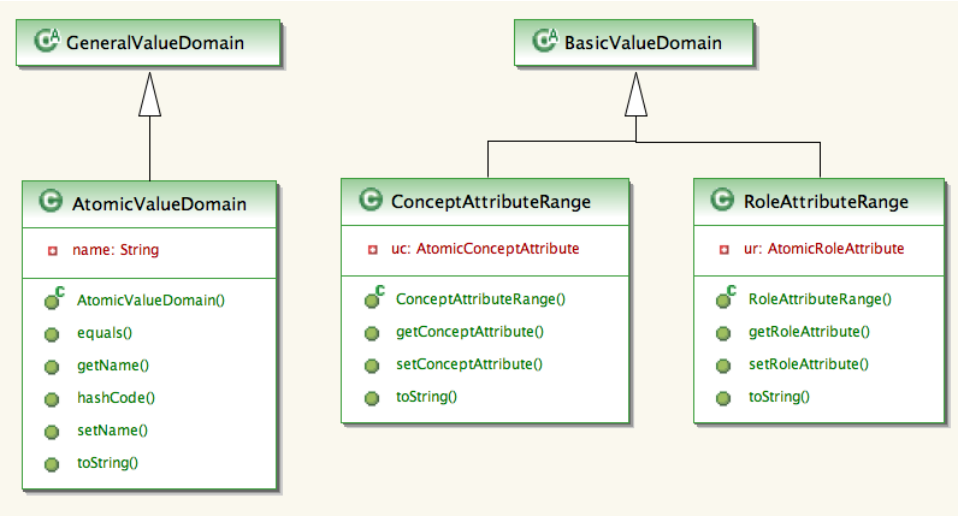


Figura 6.9: Classi del package domain: Value Domains

$\delta(trivia) \sqsubseteq PERSON$
$\delta(personalQuotes) \sqsubseteq PERSON$
$\delta(lastAppearances) \sqsubseteq PERSON$
$\delta(akaNames) \sqsubseteq PERSON$

$\exists cityOfBirth \sqsubseteq PERSON$
$\exists cityOfBirth- \sqsubseteq CITY$
$PERSON \sqsubseteq \exists(cityOfBirth)$
$(functcityOfBirth)$

$\exists salary \sqsubseteq PERSON$
$\exists salary- \sqsubseteq MOVIE$
$salary \sqsubseteq \delta(money)$
$\delta(money) \sqsubseteq salary$
$(functmoney)$

$\exists directedBy \sqsubseteq PERSON$
$\exists directedBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists directedBy-$
$\delta(role1) \sqsubseteq directedBy$
$(functrole1)$

$\exists writtenBy \sqsubseteq PERSON$
$\exists writtenBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists writtenBy-$
$\delta(role2) \sqsubseteq writtenBy$
$(functrole2)$

$\exists producedBy \sqsubseteq PERSON$
$\exists producedBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists producedBy-$
$\delta(role3) \sqsubseteq producedBy$
$(functrole3)$

$\exists editedBy \sqsubseteq PERSON$
$\exists editedBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists editedBy-$
$\delta(role4) \sqsubseteq editedBy$
$(functrole4)$

$\exists composedBy \sqsubseteq PERSON$
$\exists composedBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists composedBy-$
$\delta(role5) \sqsubseteq composedBy$
$(functrole5)$

$\exists cinematographyBy \sqsubseteq PERSON$
$\exists cinematographyBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists cinematographyBy-$
$\delta(role6) \sqsubseteq cinematographyBy$
$(functrole6)$

$\exists costumeDesignBy \sqsubseteq PERSON$
$\exists costumeDesignBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists costumeDesignBy-$
$\delta(role7) \sqsubseteq costumeDesignBy$
$(functrole7)$

$\exists productionDesignBy \sqsubseteq PERSON$
$\exists productionDesignBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists productionDesignBy-$
$\delta(role8) \sqsubseteq productionDesignBy$
$(functrole8)$

$\exists castingyBy \sqsubseteq PERSON$
$\exists castingBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists castingBy-$
$\delta(role9) \sqsubseteq castingBy$
$(functrole9)$

$\exists originalMusicBy \sqsubseteq PERSON$
$\exists originalMusicBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists originalMusicBy-$
$\delta(role10) \sqsubseteq originalMusicBy$
$(functrole10)$

$\exists artDirectionBy \sqsubseteq PERSON$
$\exists artDirectionBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists artDirectionBy-$
$\delta(role11) \sqsubseteq artDirectionBy$
$(functrole11)$

$\exists setDecorationBy \sqsubseteq PERSON$
$\exists setDecorationBy- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists setDecorationBy-$
$\delta(role12) \sqsubseteq setDecorationBy$

$(functrole12)$

$\exists makeupDept \sqsubseteq PERSON$
$\exists makeupDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists makeupDept-$
$\delta(role13) \sqsubseteq makeupDept$
$(functrole13)$

$\exists artDept \sqsubseteq PERSON$
$\exists artDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists artDept-$
$\delta(role14) \sqsubseteq artDept$
$(functrole14)$

$\exists soundDept \sqsubseteq PERSON$
$\exists soundDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists soundDept-$
$\delta(role15) \sqsubseteq soundDept$
$(functrole15)$

$\exists cameraDept \sqsubseteq PERSON$
$\exists cameraDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists cameraDept-$
$\delta(role16) \sqsubseteq cameraDept$
$(functrole16)$

$\exists animationDept \sqsubseteq PERSON$
$\exists animationDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists animationDept-$
$\delta(role17) \sqsubseteq animationDept$
$(functrole17)$

$\exists castingDept \sqsubseteq PERSON$
$\exists castingDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists castingDept-$
$\delta(role18) \sqsubseteq castingDept$
$(functrole18)$

$\exists costumeDept \sqsubseteq PERSON$
$\exists costumeDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists costumeDept-$
$\delta(role19) \sqsubseteq costumeDept$
$(functrole19)$

$\exists editorialDept \sqsubseteq PERSON$
$\exists editorialDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists editorialDept-$
$\delta(role20) \sqsubseteq editorialDept$
$(functrole20)$

$\exists musicDept \sqsubseteq PERSON$
$\exists musicDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists musicDept-$
$\delta(role21) \sqsubseteq musicDept$
$(functrole21)$

$\exists transportationDept \sqsubseteq PERSON$
$\exists transportationDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists transportationDept-$
$\delta(role22) \sqsubseteq transportationDept$
$(functrole22)$

$\exists stuntmanDept \sqsubseteq PERSON$
$\exists stuntmanDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists stuntmanDept-$
$\delta(role23) \sqsubseteq stuntmanDept$
$(functrole23)$

$\exists specialEffectDept \sqsubseteq PERSON$
$\exists specialEffectDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists specialEffectDept-$
$\delta(role24) \sqsubseteq specialEffectDept$
$(functrole24)$

$\exists visualEffectDept \sqsubseteq PERSON$
$\exists visualEffectDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists visualEffectDept-$
$\delta(role25) \sqsubseteq visualEffectDept$
$(functrole25)$

$\exists productionManagementDept \sqsubseteq PERSON$
$\exists productionManagementDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists productionManagementDept-$
$\delta(role26) \sqsubseteq productionManagementDept$
$(functrole26)$

$\exists assistantDirectorDept \sqsubseteq PERSON$
$\exists assistantDirectorDept- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists assistantDirectorDept-(functsex)$
$\delta(role27) \sqsubseteq assistantDirectorDept$
$(functrole27)$

$\exists otherCrew \sqsubseteq PERSON$
$\exists otherCrew- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists otherCrew-$
$\delta(role28) \sqsubseteq otherCrew$
$(functrole28)$

$\exists thanksTo \sqsubseteq PERSON$
$\exists thanksTo- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists thanksTo-$
$\delta(role29) \sqsubseteq thanksTo$
$(functrole29)$

$\exists songWrittenBy \sqsubseteq PERSON$
$\exists songWrittenBy- \sqsubseteq SONG$
$SONG \sqsubseteq \exists songWrittenBy-$

$\exists songPerformedBy \sqsubseteq PERSON$
$\exists songPerformedBy- \sqsubseteq SONG$
$SONG \sqsubseteq \exists songPerformedBy-$

$\exists songSingedBy \sqsubseteq PERSON$
$\exists songSingedBy- \sqsubseteq SONG$
$SONG \sqsubseteq \exists songSingedBy-$

$\exists award2person \sqsubseteq AWARD$
$\exists award2person- \sqsubseteq PERSON$
$AWARD \sqsubseteq \exists award2person$
$(functaward2person)$

$\exists host \sqsubseteq CEREMONY$
$\exists host- \sqsubseteq PERSON$
$CEREMONY \sqsubseteq \exists host$
$(functhost)$

$ACTOR \sqsubseteq PERSON$
$ACTOR \sqsubseteq \delta(sex)$
$\delta(sex) \sqsubseteq ACTOR$

$\exists cast \sqsubseteq ACTOR$
$\exists cast- \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \exists cast-$
$cast \sqsubseteq \delta(character)$
$\delta(character) \sqsubseteq cast$

$MOVIE \sqsubseteq \delta(movieTitle)$
$\delta(movieTitle) \sqsubseteq MOVIE$
$(functmovieTitle)$
$MOVIE \sqsubseteq \delta(year)$
$\delta(year) \sqsubseteq MOVIE$
$(functyear)$
$MOVIE \sqsubseteq \delta(genre)$
$\delta(genre) \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \delta(tagline)$
$\delta(tagline) \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \delta(plotOutline)$
$\delta(plotOutline) \sqsubseteq MOVIE$
$(functplotOutline)$
$MOVIE \sqsubseteq \delta(plotSynopsis)$
$\delta(plotSynopsis) \sqsubseteq MOVIE$
$(functplotSynopsis)..$
$MOVIE \sqsubseteq \delta(keyword)..$
$\delta(keyword) \sqsubseteq MOVIE$
$\delta(quotes) \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \delta(mpaa)..$
$\delta(mpaa) \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \delta(runTime)$
$\delta(runTime) \sqsubseteq MOVIE$
$MOVIE \sqsubseteq \delta(language)$
$\delta(language) \sqsubseteq MOVIE$
$\delta(color) \sqsubseteq MOVIE$
$(functcolor)$
$\delta(aspectRatio) \sqsubseteq MOVIE$
$(functaspectRatio)$
$\delta(laboratory) \sqsubseteq MOVIE$
$\delta(cinematographicProcess) \sqsubseteq$

$MOVIE$
$\delta(negativeFormat) \sqsubseteq MOVIE$
$\delta(printedFormat) \sqsubseteq MOVIE$
$\delta(parentsGuide) \sqsubseteq MOVIE$
$\delta(soundMix) \sqsubseteq MOVIE$
$\delta(movieTrivia) \sqsubseteq MOVIE$
$\delta(goofs) \sqsubseteq MOVIE$
$\delta(tv) \sqsubseteq MOVIE$
$(functtv)$
$\delta(video) \sqsubseteq MOVIE$
$(functvideo)$
$\delta(rating) \sqsubseteq MOVIE$
$(functrating)$
$\delta(votes) \sqsubseteq MOVIE$
$(functvotes)$
$\delta(certificate) \sqsubseteq MOVIE$

$\exists releasedIn \sqsubseteq MOVIE$
$\exists releasedIn- \sqsubseteq COUNTRY$
$MOVIE \sqsubseteq \exists releasedIn$
$releasedIn \sqsubseteq \delta(releaseDate)$
$\delta(releaseDate) \sqsubseteq releasedIn$
$(functreleaseDate)$
$releasedIn \sqsubseteq \delta(akaTitle)$
$\delta(akaTitle) \sqsubseteq releasedIn$
$(functakaTitle)$

$\exists producedIn \sqsubseteq MOVIE$
$\exists producedIn- \sqsubseteq COUNTRY$
$MOVIE \sqsubseteq \exists producedIn$

$\exists filmedIn \sqsubseteq MOVIE$
$\exists filmedIn- \sqsubseteq LOCATION$
$MOVIE \sqsubseteq \exists filmedIn$

$\exists followedBy \sqsubseteq MOVIE$
$\exists followedBy- \sqsubseteq MOVIE$

$\exists remakeOf \sqsubseteq MOVIE$
$\exists remakeOf- \sqsubseteq MOVIE$

$\exists spinOffFrom \sqsubseteq MOVIE$
$\exists spinOffFrom- \sqsubseteq MOVIE$

$\exists references \sqsubseteq MOVIE$
$\exists references- \sqsubseteq MOVIE$

$\exists referencedIn \sqsubseteq MOVIE$
$\exists referencedIn- \sqsubseteq MOVIE$

$\exists features \sqsubseteq MOVIE$
$\exists features- \sqsubseteq MOVIE$

$\exists featuredIn \sqsubseteq MOVIE$
$\exists featuredIn- \sqsubseteq MOVIE$

$\exists soundtrack \sqsubseteq MOVIE$
$\exists soundtrack- \sqsubseteq SONG$

$\exists productionCompany \sqsubseteq MOVIE$
$\exists productionCompany- \sqsubseteq COMPANY$
$MOVIE \sqsubseteq \exists productionCompany$
$\delta(role30) \sqsubseteq productionCompany$
$(functrole30)$

$\exists distributionCompany \sqsubseteq MOVIE$
$\exists distributionCompany- \sqsubseteq COMPANY$
$MOVIE \sqsubseteq \exists distributionCompany$
$distributionCompany \sqsubseteq \delta(distributionYear)$
$\delta(distributionYear) \sqsubseteq distributionCompany$
$(functdistributionYear)$
$distributionCompany \sqsubseteq \delta(type)$
$\delta(type) \sqsubseteq distributionCompany$
$(functtype)$

$\exists sfxCompany \sqsubseteq MOVIE$
$\exists sfxCompany- \sqsubseteq COMPANY$
$MOVIE \sqsubseteq \exists sfxCompany$
$\delta(role31) \sqsubseteq sfxCompany$
$(functrole31)$

$\exists otherCompany \sqsubseteq MOVIE$
$\exists otherCompany- \sqsubseteq COMPANY$
$MOVIE \sqsubseteq \exists otherCompany$
$\delta(role32) \sqsubseteq otherCompany$
$(functrole32)$

$\exists award2movie \sqsubseteq AWARD$
$\exists award2movie- \sqsubseteq MOVIE$
$AWARD \sqsubseteq \exists award2movie$
$(functaward2movie)$

$TVSERIE \sqsubseteq MOVIE$
$TVSERIE \sqsubseteq \delta(seasonNo)$
$\delta(seasonNo) \sqsubseteq TVSERIE$
$(functseasonNo)$
$TVSERIE \sqsubseteq \delta(episodeNo)$
$\delta(episodeNo) \sqsubseteq TVSERIE$
$(functepisodeNo)$
$TVSERIE \sqsubseteq \delta(episodeTitle)$
$\delta(episodeTitle) \sqsubseteq TVSERIE$
$(functepisodeTitle)$

$LOCATION \sqsubseteq \delta(address)$
$\delta(address) \sqsubseteq LOCATION$
$(functaddress)$
$LOCATION \sqsubseteq \delta(pointOfInterest)$
$\delta(pointOfInterest) \sqsubseteq LOCATION$
$(functpointOfInterest)$

$\exists loc2city \sqsubseteq LOCATION$
$\exists loc2city- \sqsubseteq CITY$
$LOCATION \sqsubseteq \exists loc2city$
$(functloc2city)$

$\exists located \sqsubseteq CEREMONY$
$\exists located- \sqsubseteq LOCATION$
$CEREMONY \sqsubseteq \exists located$
$(functlocated)$

$CITY \sqsubseteq \delta(cityName)$
$\delta(cityName) \sqsubseteq CITY$
$(functcityName)$
$CITY \sqsubseteq \delta(region)$
$\delta(region) \sqsubseteq CITY$
$(functregion)$
$CITY \sqsubseteq \delta(state)$
$\delta(state) \sqsubseteq CITY$
$(functstate)$

$\exists city2country \sqsubseteq CITY$
$\exists city2country- \sqsubseteq COUNTRY$
$CITY \sqsubseteq \exists city2country$
$(functcity2country)$

$COUNTRY \sqsubseteq \delta(countryName)$
$\delta(countryName) \sqsubseteq COUNTRY$
$(functcountryName)$

$COMPANY \sqsubseteq \delta(companyName)$
$\delta(companyName) \sqsubseteq COMPANY$
$(functcompanyName)$

$SONG \sqsubseteq \delta(songTitle)$
$\delta(songTitle) \sqsubseteq SONG$
$(functsongTitle)$
$\delta(copyrightInfo) \sqsubseteq SONG$
$(functcopyrightInfo)$

$AWARD \sqsubseteq \delta(category)$
$\delta(category) \sqsubseteq AWARD$
$(functcategory)$
$AWARD \sqsubseteq \delta(result)$
$\delta(result) \sqsubseteq AWARD$
$(functresult)$
$AWARD \sqsubseteq \delta(motivation)$
$\delta(motivation) \sqsubseteq AWARD$
$(functmotivation)$

$\exists award2ceremony \sqsubseteq AWARD$
$\exists award2ceremony- \sqsubseteq CEREMONY$
$AWARD \sqsubseteq \exists award2ceremony$
$(functaward2ceremony)$

$CEREMONY \sqsubseteq \delta(ceremonyDate)$
$\delta(ceremonyDate) \sqsubseteq CEREMONY$
$(functceremonyDate)$
$CEREMONY \sqsubseteq \delta(awardType)$
$\delta(awardType) \sqsubseteq CEREMONY$

$(functawardType)$
$CEREMONY \sqsubseteq \delta(academy)$
$\delta(academy) \sqsubseteq CEREMONY$
$(functacademy)$

$\rho(name) \sqsubseteq xsd : string$
$\rho(surname) \sqsubseteq xsd : string$
$\rho(birthDate) \sqsubseteq xsd : string$
$\rho(birthName) \sqsubseteq xsd : string$
$\rho(trademark) \sqsubseteq xsd : string$
$\rho(trivia) \sqsubseteq xsd : string$
$\rho(personalQuotes) \sqsubseteq xsd : string$
$\rho(biography) \sqsubseteq xsd : string$
$\rho(height) \sqsubseteq xsd : string$
$\rho(lastAppearances) \sqsubseteq xsd : string$
$\rho(akaNames) \sqsubseteq xsd : string$
$\rho(sex) \sqsubseteq xsd : string$
$\rho(movieTitle) \sqsubseteq xsd : string$
$\rho(year) \sqsubseteq xsd : string$
$\rho(genre) \sqsubseteq xsd : string$
$\rho(tagline) \sqsubseteq xsd : string$
$\rho(plotOutline) \sqsubseteq xsd : string$
$\rho(plotSynopsis) \sqsubseteq xsd : string$
$\rho(keyword) \sqsubseteq xsd : string$
$\rho(quotes) \sqsubseteq xsd : string$
$\rho(mpaa) \sqsubseteq xsd : string$
$\rho(runTime) \sqsubseteq xsd : string$
$\rho(language) \sqsubseteq xsd : string$
$\rho(color) \sqsubseteq xsd : string$
$\rho(aspectRatio) \sqsubseteq xsd : string$
$\rho(laboratory) \sqsubseteq xsd : string$
$\rho(cinematographicProcess) \sqsubseteq xsd : string$
$\rho(negativeFormat) \sqsubseteq xsd : string$
$\rho(printedFormat) \sqsubseteq xsd : string$
$\rho(parentsGuide) \sqsubseteq xsd : string$
$\rho(soundMix) \sqsubseteq xsd : string$
$\rho(movieTrivia) \sqsubseteq xsd : string$
$\rho(goofs) \sqsubseteq xsd : string$
$\rho(tv) \sqsubseteq xsd : string$
$\rho(video) \sqsubseteq xsd : string$
$\rho(rating) \sqsubseteq xsd : string$
$\rho(votes) \sqsubseteq xsd : string$

$\rho(certificate) \sqsubseteq xsd : string$
$\rho(seasonNo) \sqsubseteq xsd : string$
$\rho(episodeNo) \sqsubseteq xsd : string$
$\rho(episodeTitle) \sqsubseteq xsd : string$
$\rho(address) \sqsubseteq xsd : string$
$\rho(pointOfInterest) \sqsubseteq xsd : string$
$\rho(cityName) \sqsubseteq xsd : string$
$\rho(region) \sqsubseteq xsd : string$
$\rho(state) \sqsubseteq xsd : string$
$\rho(countryName) \sqsubseteq xsd : string$
$\rho(companyName) \sqsubseteq xsd : string$
$\rho(songTitle) \sqsubseteq xsd : string$
$\rho(copyrightInfo) \sqsubseteq xsd : string$
$\rho(category) \sqsubseteq xsd : string$
$\rho(result) \sqsubseteq xsd : string$
$\rho(motivation) \sqsubseteq xsd : string$
$\rho(ceremonyDate) \sqsubseteq xsd : string$
$\rho(awardType) \sqsubseteq xsd : string$
$\rho(academy) \sqsubseteq xsd : string$
$\rho(money) \sqsubseteq xsd : string$
$\rho(character) \sqsubseteq xsd : string$
$\rho(releaseDate) \sqsubseteq xsd : string$
$\rho(akaTitle) \sqsubseteq xsd : string$
$\rho(distributionYear) \sqsubseteq xsd : string$
$\rho(type) \sqsubseteq xsd : string$
$\rho(role1) \sqsubseteq xsd : string$
$\rho(role2) \sqsubseteq xsd : string$
$\rho(role3) \sqsubseteq xsd : string$
$\rho(role4) \sqsubseteq xsd : string$
$\rho(role5) \sqsubseteq xsd : string$
$\rho(role6) \sqsubseteq xsd : string$
$\rho(role7) \sqsubseteq xsd : string$
$\rho(role8) \sqsubseteq xsd : string$
$\rho(role9) \sqsubseteq xsd : string$
$\rho(role10) \sqsubseteq xsd : string$
$\rho(role11) \sqsubseteq xsd : string$
$\rho(role12) \sqsubseteq xsd : string$
$\rho(role13) \sqsubseteq xsd : string$
$\rho(role14) \sqsubseteq xsd : string$
$\rho(role15) \sqsubseteq xsd : string$
$\rho(role16) \sqsubseteq xsd : string$
$\rho(role17) \sqsubseteq xsd : string$
$\rho(role18) \sqsubseteq xsd : string$
$\rho(role19) \sqsubseteq xsd : string$

$\rho(role20) \sqsubseteq xsd : string$
$\rho(role21) \sqsubseteq xsd : string$
$\rho(role22) \sqsubseteq xsd : string$
$\rho(role23) \sqsubseteq xsd : string$
$\rho(role24) \sqsubseteq xsd : string$
$\rho(role25) \sqsubseteq xsd : string$
$\rho(role26) \sqsubseteq xsd : string$
$\rho(role27) \sqsubseteq xsd : string$
$\rho(role28) \sqsubseteq xsd : string$
$\rho(role29) \sqsubseteq xsd : string$
$\rho(role30) \sqsubseteq xsd : string$
$\rho(role31) \sqsubseteq xsd : string$
$\rho(role32) \sqsubseteq xsd : string$

6.2.2 Traduzione

Di seguito è riportato un breve esempio di traduzione automatica. L'input XML per OntologyConverter è:

```
<inclusionAssertion>
    <basicC>
        <atomicC>Person</atomicC>
    </basicC>
    <generalC>
        <signedC sign="positive">
            <basicC>
                <CADomain>
                    <atomicCA>name</atomicCA>
                </CADomain>
            </basicC>
        </signedC>
    </generalC>
</inclusionAssertion>

<inclusionAssertion>
    <basicC>
        <CADomain>
            <atomicCA>name</atomicCA>
        </CADomain>
    </basicC>
    <generalC>
        <signedC sign="positive">
            <basicC>
                <atomicC>Person</atomicC>
            </basicC>
        </signedC>
    </generalC>
</inclusionAssertion>
```

```
<funct>
    <atomicCA>name</atomicCA>
</funct>
```

In output si ottengono le seguenti formule della FOL:

```
all X (Person(X) -> (exists Y name(X,Y))).
all XY (name(X,Y) -> Person(X)).
all XYZ (name(X,Y) & name(X,Z) -> Y=Z).
```

6.2.3 Verifica proprietà

Il file ottenuto con OntologyCoverter è nel formato di Otter/mace quindi per l'implicazione logica di una proprietà basta aggiungerla negata in fondo al file (vedi 4.2.2).

Di seguito sono riportati alcuni semplici esempi di proprietà da verificare sull'ontologia IMDB:

Proprietà vere: usare otter per verificare

```
%% Actor ha un attributo name
P <-> (all XY (Actor(X) & name(X,Y) -> STRING(Y))).
```

Proprietà false: usare mace per trovare un modello

```
%% Person ha un attributo biography obbligatorio
P <-> (all XY (Person(X) -> (exists Y biography(X,Y)))).
```

Bibliografia

[1] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Linking Data to Ontologies: The Description Logic DL-Lite$_{\mathcal{A}}$. In *OWLED*, 2006.

[2] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi. Data Integration through DL-Lite$_{\mathcal{A}}$ Ontologies. In *SDKB*, pages 26–47, 2008.

[3] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Can OWL Model Football Leagues? In *OWLED*, 2007.

[4] Claudio Corona, Domenico Fabio Savio, and Marzo Ruzzi. The QuOnto/Mastro family of systems: a technical overview. http://www.dis.uniroma1.it/quonto, 2008.

[5] Giuseppe De Giacomo. DL-Lite$_{\mathcal{A}}$. Dispense del corso di Seminari di Ingegneria del Software, 2008.

[6] Valerio Del Grande and Junio Valerio Franchi. The Internet Movie Database DL-Lite$_{\mathcal{A}}$ Ontology. Tesina del corso di Seminari di Ingegneria del Software, 2008.

[7] Maurizio Lenzerini. La Progettazione Concettuale. Dispense del corso di Basi di Dati, 2009.

[8] Toni Mancini and Marco Cadoli. Diagrammi UML delle classi. Dispense del corso di Metodi Formali nell'Ingegneria del Software, 2007.

[9] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking Data to Ontologies. *J. Data Semantics*, 10:133–173, 2008.

[10] Dmitry Tsarkov and Ian Horrocks. DL Reasoner vs. First-Order Prover. In *Description Logics*, 2003.