Rappresenteazione di diagrammi UML in DLLite$_A$ attraverso il software QToolkit

Dott. Claudio Corona

Prerequisiti

- UML (corso di Progettazione del Software)
- $DLLite_A$ (slide del corso)
- Union of Conjunctive Queries (slide del corso)
- Epistemic Queries* (slide del corso)

* Per la sintassi concreta delle query epistemiche (SparSQL) si rimanda a questo articolo

Obiettivo dell’esercitazione

- **Parte A**: Individuare il diagramma delle classi UML; tradurre il diagramma delle classi in DLLite$_A$ e scriverlo in sintassi funzionale su QToolkit. Dopo aver avviato il sistema con l’ontologia definita, sfruttare i servizi di ragionamento per verificare se ci sono classi vuote, classi equivalenti, ecc.
- **Parte B**: Generare una istanziazione parziale del diagramma (una ABox) e verificare la consistenza dell'istanziazione stessa.
- **Parte C**: A partire dalla specifica degli use case, costruire delle query congiuntive o epistemiche da porre al sistema.

Parte A: specifica

L'applicazione da progettare riguarda una parte del sistema di gestione di un asilo per il corrente anno di iscrizione. Ogni classe è caratterizzata da un nome (una stringa), dai bambini ad essa assegnati e dalle maestre che vi insegnano. In una classe insegna esattamente una maestra. Ogni bambino ha un nome e un'età (compresa tra 0 e 5 anni) ed è assegnato ad esattamente una classe. Ogni maestra ha un nome ed una anzianità di servizio (un intero). Alcune classi sono classi di scolarizzazione e ad esse vengono assegnati almeno 1 bambino non-scolarizzati. Dei bambini non-scolarizzati interessa sapere se portano ancora il pannolino (un booleano). Come per le classi normali, anche in una classe di scolarizzazione insegna esattamente una maestra.

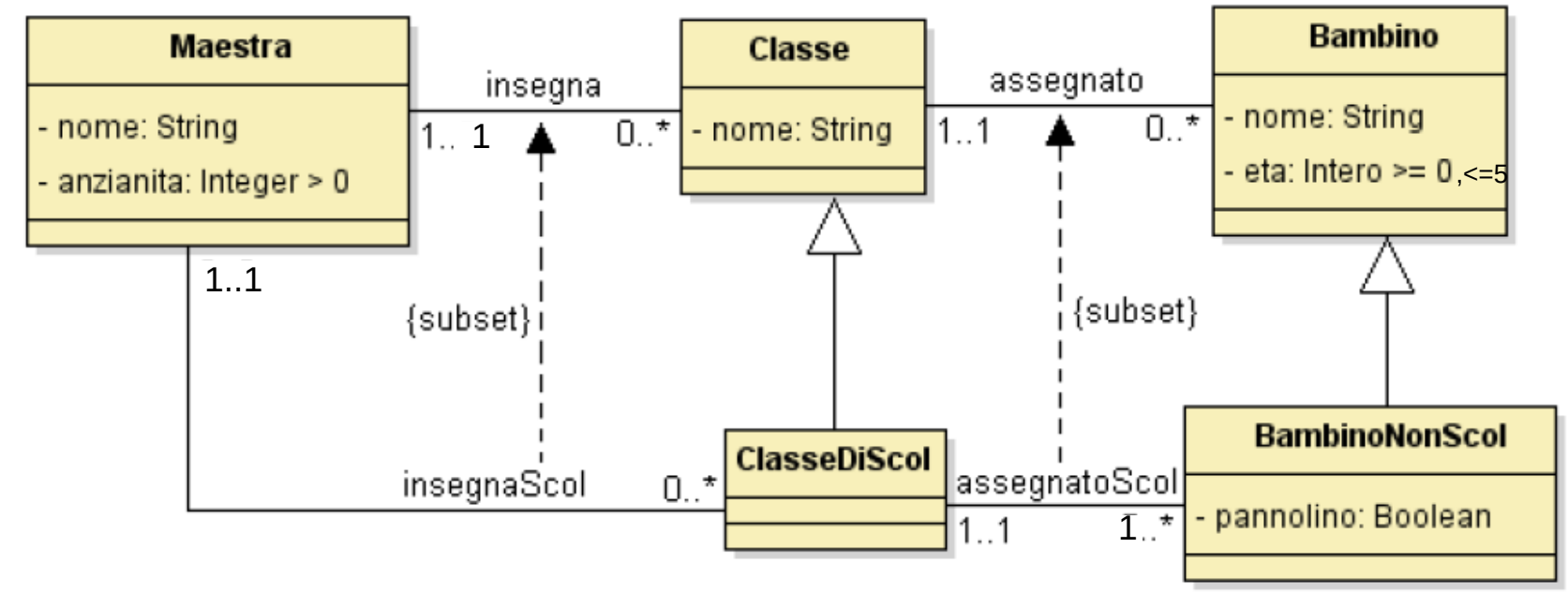
Parte A: specifica(2)

Il coordinatore didattico è interessato ad effettuare diversi controlli sulle classi, in particolare:

- dato un insieme di classi *s*, restituire il sottoinsieme formato dalle classi *problematiche* di *s*: dove una classe è problematica se è una classe di scolarizzazione tale che tutti i bambini assegnati ad essa sono non-scolarizzati;
- data una classe *c*, restituire l'età media dei bambini ad essa assegnati.

Parte A: diagramma delle classi UML

Diagramma UML delle classi



Parte A: passaggio da UML a sintassi funzionale $DLLite_A$

Vedere file in input al QToolkit

[Esercitazione\TBox.tbox] + [Esercitazione\EBox.cbox]

Nota: in $DLLite_A$ valgono le seguenti restrizioni sulla TBox:

1. Se $Q \sqsubseteq P$ o $Q \sqsubseteq P^-$ è in T, allora (**funct** P) e (**funct** P^-) non sono in T
2. Se $U1 \sqsubseteq U2$ è in T, allora (**funct** U2) non è in T

Nel diagramma UML le relazioni “insegna” e “assegnato” sono funzionali (cardinalità max = 1) e sono specializzate da “insegnaScol” e “assegnatoScol”, rispettivamente. In questo esercizio si sceglie di mantenere le 2 isa tra ruoli e di implementare le 2 funzionalità come vincoli epistemici

Parte B: istanziazione del diagramma delle classi (= ABox)

Vedere file in input al QToolkit

[Esercitazione\ABox.abox]

Parte C: use-case

1. dato un insieme di classi *s*, restituire il sottoinsieme formato dalle classi *problematiche* di s: dove una classe è problematica se è una classe di scolarizzazione tale che tutti i bambini assegnati ad essa sono non-scolarizzati

Vedere file in input al Qtoolkit

[Esercitazione\ UseCase_prog1.txt]

Parte C: use-case (2)

2. data una classe *c*, restituire l'età media dei bambini ad essa assegnati

Vedere file in input al Qtoolkit

[Esercitazione\ UseCase_prog2.txt]

Vincoli di covering

- Per concludere, vediamo come sia possibile implementare un vincolo di covering (derivante ad esempio da una generalizzazione completa) attraverso un vincolo epistemico
- Vorremmo imporre sull'ontologia il seguente vincolo: ogni persona è un maschio o una femmina. Non potendo imporre questo vincolo in una TBox DLLite, ci si "accontenta" di imporre il seguente vincolo epistemico: ogni persona *nota* deve essere un maschio *noto* o una femmina *nota*

$\text{Person} \sqsubseteq \text{Male} \cup \text{Female}$ diventa $\mathbf{K}\,\text{Person} \sqsubseteq \mathbf{K}\,\text{Male} \cup \mathbf{K}\,\text{Female}$

```
VERIFY not exists (     SELECT persons.x
                        FROM SparqlTable(    SELECT ?x
                                             WHERE {?x rdf:type 'Person'}) persons
                        EXCEPT      (
                                    SELECT males.x
                                    FROM SparqlTable(    SELECT ?x
                                                         WHERE {?x rdf:type 'Male'}) males
                                    UNION
                                    SELECT females:x
                                    FROM SparqlTable(    SELECT ?x
                                                         WHERE {?x rdf:type 'Female']) females
                                    )
                   )
```

Vincoli di covering (2)

- Si noti che le query rosse, che sono union of conjunctive queries espresse in SPARQL, hanno il compito di estrarre la *conoscenza* dall'ontologia (tale conoscenza viene restituita sottoforma di *risposte certe*);
- Visto che sulla conoscenza estratta si ha informazione completa (≈ una tupla **è o non è** nelle risposte certe), è possibile interrogare tale conoscenza come se fosse un database: questo è il ruolo svolto dalla parte della query SparSQL in nero, ovvero la query SQL
- Riassumendo: le query in rosso (UCQ) interrogano l'ontologia, estraendone conoscenza che viene manipolata/interrogata attraverso la query SQL (parte in nero)