Ontologia prototipo: definizione e struttura delle Parti

Obiettivo

Definire una rappresentazione gerarchica delle Parti che

- sia *generale*, *modulare* e facilmente *estendibile*, e
- permetta di *ordinare* e *selezionare* una Parte date alcune delle sue caratteristiche.

L'ontologia proposta si focalizza sul primo punto fornendo una rappresentazione delle scenario. Per il secondo punto, si prevede un componente *esterno* che sfrutti l'ontologia per selezionare una Parte secondo un certo *ranking* di caratteristiche. Questa assunzione viene fatta per motivi tecnici (stabilità, licenze, risorse, modularità, flessibilità, etc.) ma il ragionamento implementato nel componente esterno potrebbe essere incorporato nell'ontologia (ad esempio nel dominio fuzzy).

Ontologie OWL

L'Ontology Web Language (OWL) formalizza la conoscenza in un Terminological Box (Tbox), dove vengono definiti concetti e relazioni astratte (in pratica: i simboli usati e la loro semantica), ed un Assertion Box (Abox), che contiene particolari istanze caratterizzate da alcuni dei simboli definiti nel Tbox. Entrambi i box contengono *assiomi logici* che possono essere rappresentate in forma di grafo (spesso come albero).

Il Tbox contiene due tipi di entità

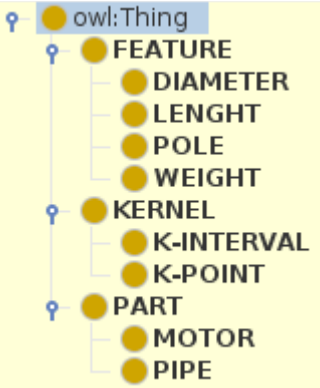
- i *concetti* (o classi) che definiscono i tipi di istanze (o individui),
- i *ruoli* (o proprietà) che relazionano individui tra di loro.

Da notare che quantità (es., numeri) sono considerati come individui speciali che sono istanze di *concetti concreti* (es., numeri reali, booleani, stringhe di testo, etc.).

L'ontologia è dotata di un ragionatore che controlla la consistenza logica degli assiomi nei box e, se c'è un'inconsistenza, è in grado di darne delle spiegazioni. Inoltre, il ragionatore può trarre conclusioni logiche per classificare istanze in concetti o assegnare proprietà. Chiamerò questa componente: *ragionatore interno*, per distinguerlo da quello esterno introdotto precedentemente per il ranking e la selezione delle Parti.

Un prototipo di Tbox e Abox

Si definiscono i seguenti concetti generali: FEATURE, KERNEL e PART. A loro volta, FEATURE contiene diverse caratteristiche (es., WEIGHT, LENGHT, etc.), mentre KERNEL specifica un metodo di selezione e ranking per ogni FEATURE. Infine, ogni PART è formata da una serie di FEATURE.



Gerarchia di concetti.

Ogni FEATURE è definita da una proprietà (es., `F1 hasLenght 30`, la quale viene usata automaticamente per classificare l'istanza F1 nel concetto LENGH). Inoltre, ogni istanza del concetto FEATURE è anche istanza di un KERNEL, il quale specifica il tipo di confronto che il ragionatore esterno deve usare per la selezione delle Parti date alcune delle loro caratteristiche.

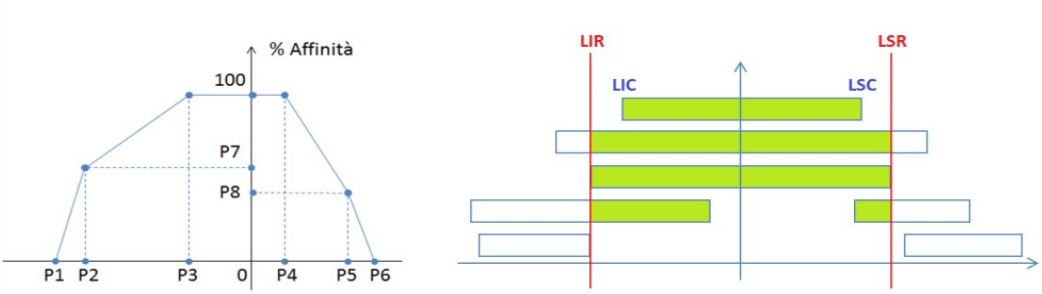
Ogni PART è composta da delle caratteristiche (es., `X hasFeature F1`) e, alcune di esse possono essere specificate per la classificazione della Parte, es., `X hasMotorFeature F2` dove `F2 hasPole 3` e il ragionatore interno concluderà che `X: MOTOR`. Dato che i sotto-concetti di PART sono tra di loro disgiunti, ne consegue che ogni istanza di PART potrà avere proprietà di tipo hasFeature, dovrà averne alcune di tipo hasΔFeature (dove Δ rappresenta la Parte) e nessuna proprietà hasΓFeature dove Γ≠Δ. Tuttavia, se necessario, si possono considerare anche rappresentazioni di Parti gerarchiche più complesse dove, ad esempio, Γ è costituito da più Parti di tipo Δ.

Alcune relazioni tra i concetti dell'ontologia.

Query e ragionatore esterno

Questa struttura ontologica è pensata per essere flessibile e permette di identificare una Parte. In particolare, è possibile chiedere al ragionatore interno tutte le Parti coerenti con specifiche tipi di caratteristiche, es., trovare quelle istanze di PART che sono rappresentate attraverso un numero di poli. Queste descriveranno indirettamente anche i kernel da utilizzare per analizzare ogni caratteristica ed ordinare le Parti trovate per selezionare le migliori

I kernel si possono rappresentare con diverse funzioni (fuzzy, a soglia, probabilistiche etc.) che sono processate dal ragionatore esterno. Quest'ultimo potrà fare anche operazioni di più alto livello, come pesare i contributi delle diverse caratteristiche di una Parte. Con questo approccio, il sistema risulta essere flessibile per quanto riguarda altri kernel e politiche di ranking. In altri termini, l'interfaccia tra il ragionatore esterno ed interno dipende solo dalla definizione dei kernel, che sono rappresentati in modo generico nell'ontologia.



Due esempi di kernel. K-POINT a sinistra e K-INTERVAL a destra.

Creazione di nuove Parti

Ogni istanza che descrive una nuova Parte si può rappresentare nell'ontologia aggiungendo assiomi nell'Abox. In particolare si dovrà aggiungere:

1. un'insieme di individui `{F1,F2,...Fn}: FEATURE` ognuno con
   - una specifica proprietà (es., `F1 hasWeight 0.8`),
   - e la referenza ad un kernel (es., `F1: K-POINT`);
2. un individuo (es., `X: PART`) con
   - alcune proprità `X hasFeature Fi` relazionate la punto precedente,
   - e almeno una proprietà specifica per la relativa Parte (`X hasPipeLenght F2`). Queste proprietà verranno usate per discriminare `X: PIPE` rispetto agli altri sotto-concetti di PART.

Se invece si vuole definire una nuova possibile Parte si dovrà cambiare il Tbox. In particolare:

1. aggiungere sotto-concetti di FEATURE e relativa relazione (es., hasDiameter) se non già presenti;
2. aggiungere un sotto-concetto di PART (es., WHEEL), e la relativa proprietà (es., hasWheelFeature).

Aspetti computazionali ed architetturali

Il ragionatore interno di un'ontologia è esponenzialmente lento all'incremento del numero di assiomi che l'Abox ed il TBox contengono. Per questo, se il numero di FEATURE, PART e le loro istanze aumentano molto, il tempo di computazione potrebbe diventare proibitivo.

Tuttavia, grazie al ragionatore esterno, si può invocare il ragionatore interno solamente quando nuove Parti vengono create, il che si assume essere un evento relativamente sporadico. Nel dettaglio, sarà necessario aggiornare il ragionatore interno quando l'Abox o il Tbox cambiano (e, per grandi ontologie, questo richiederà un tempo di calcolo non trascurabile), ma quando il ragionatore esterno farà le query per selezionare la Parte più appropriata la computazione sarà ridotta.

Inoltre, si può pensare di limitare il numero di assiomi nell'ontologia, e quindi il tempo di ragionamento, dividendo la rappresentazioni in diverse ontologie indipendenti, ognuna con un dedicato ragionatore interno; es., un'ontologia che descrive Parti fisiche (come nelle figure), un'altra per utilizzi di pezzi fatti su progetti precedenti, etc. Però, lo svantaggio di questo approccio è dato dal dover identificare rappresentazioni ontologiche sufficientemente espressive e, allo stesso tempo, indipendenti tra di loro.

Dal punto di vista architetturale, ogni ontologia ed il relativo ragionatore interno si comportano come un database che rimane sempre attivo e può rispondere a delle query specifiche attraverso un'interfaccia WEB. Quest'ultima è utilizzata dal ragionatore esterno, che può essere implementato come un micro-servizio stateless. Inoltre, per creare nuove Parti, l'ontologia richiede un'altra interfaccia WEB che aggiunga assiomi nell'Abox e Tbox e che aggiorni il reasoner interno per trovare nuove inferenze.