SISTEMI

BASATI SU CONOSCENZA

E

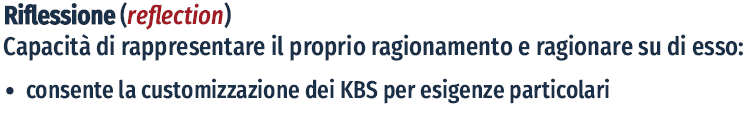
ONTOLOGIE

Vedremo un uso dei metalinguaggi o meta-interpreti scritti tramite delle clausole per poter ragionare sul linguaggio stesso con il quale andiamo a scrivere i meta-interpreti. Questo significa che il linguaggio stesso andrà a modificare la sua struttura interna per modificare la modalità di ragionamento effettuata. Tutta la base di conoscenza sarà rappresentata dal meta-linguaggio che lavorerà ad un livello superiore rispetto a quello del linguaggio stesso.

Affronteremo anche l’aspetto della conoscenza distribuita su database aziendali. In questi settori usando linguaggi standard è possibile usare e condividere la conoscenza a livello globale a patto di aderire agli standard imposti dal W3C e IEEE.

Questo capitolo riguarda l’implementazione dei sistemi basati su conoscenza. Se un enunciato può essere manipolato da un linguaggio di programmazione ovvero ad esempio al prolog permettiamo di manipolare le clausole come fossero termini o costanti allora otteniamo un meta linguaggio ed è possibile modificare il ragionamento nel linguaggio stesso. (esempio meta compilatori)In questa maniera si costruiscono dei meta interpreti del linguaggio stesso.

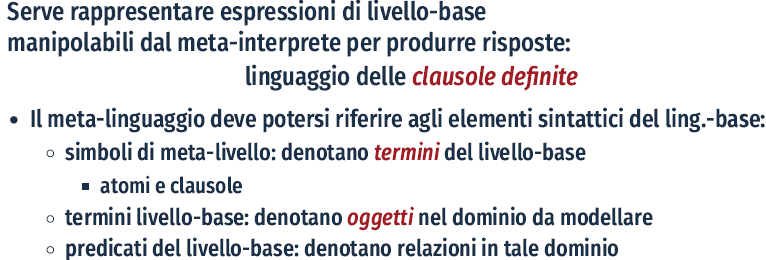
In molti linguaggi di programmazione è permessa la riflessione che permette di manipolare la struttura interna del linguaggio e delle variabili. La riflessione permette al linguaggio di riconoscere il tipo di classe a run time.



Un meta-linguaggio serve a dettagliare meglio alcuni aspetti del linguaggio affinché se ne aumenti l’efficienza.

Graphical user interface, text, application

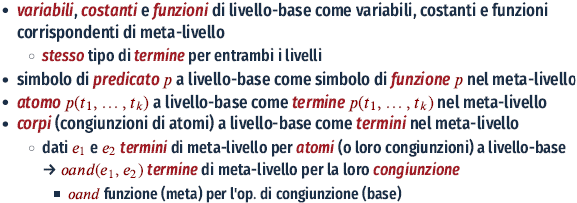
Description automatically generated



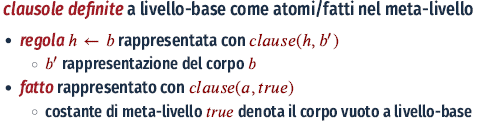
La scelta più importante da fare nella costruzione di un meta-linguaggio sta nel scegliere come il linguaggio debba auto rappresentarsi attraverso clausole definite e le possibili rappresentazioni sono:

1. RAPPRESENTAZIONE NON-GROUND
2. RAPPRESENTAZIONE GROUND

Affronteremo solo la rappresentazione NON-GROUND nella quale il meta-linguaggio serve a rappresentare tutti i costrutti di livello base:



Adesso si necessità oggettivare una clausola affinché sia maneggiabile dal meta-linguaggio:



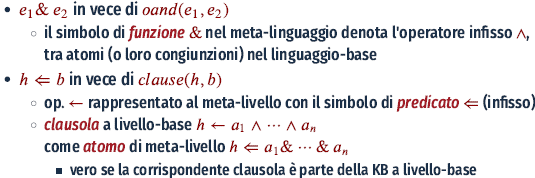
Vediamo un esempio di rappresentazione delle clausole nel livello base e successivamente nel meta-livello:

Text, letter

Description automatically generated

Nell’ultimo esempio si può notare come l’operatore oand necessità due termini o argomenti su cui agire, per cui per rappresentare il corpo che è un insieme di congiunti viene ad essere rappresentato come funzione di funzione in modo ricorsivo.

Poiché l’applicazione dell’operatore oand risulta complicata e poco leggibile riadattiamo la sintassi con una rappresentazione alternativa che ci permette di facilitare la scrittura dell’operatore oand (tramite un processo di zucchero sintattico) ove l’operatore da prefisso diventa infisso con il simbolo & che sostituisce oand.



Inoltre, invece di usare clause(h,b) per definire le regole o i fatti semplifichiamo la lettura usando la freccia di implicazione con doppia barra. Con questa nuova sintassi riadattata per una migliore lettura i precedenti esempi diventano come segue:

Text

Description automatically generated

RAPPRESENTAZIONE NON-GROUND: RIEPILOGO DELLE REGOLE DA USARE PER PASSARE DAL LIVELLO BASE(DEL LINGUAGGIO) AL META-LIEVELLO(DEL META-INTERPRETE):

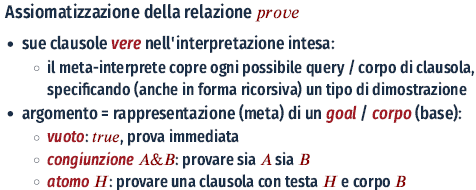
Table

Description automatically generated

I simboli usati per la rappresentazione & e <=, sono arbitrari se ne possono usare degli altri per ridefinirli.

Adesso vediamo il primo esempio di Meta-interprete che lavora sul linguaggio base, ma agendo sul meta-livello della rappresentazione precedentemente vista.

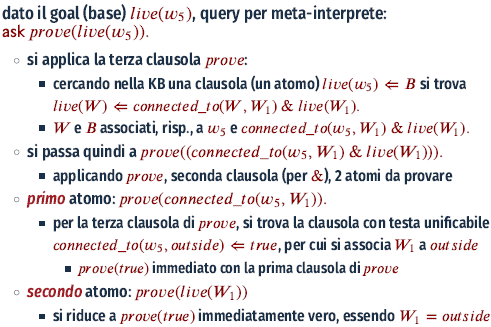
Il meta-interprete basilare ovvero il minimo interprete che possiamo realizzare prende il nome di VANILLA (ovvero il più basilare) e per farlo andiamo ad assiomatizzare il ragionamento del linguaggio(ovvero come si dimostra una conseguenza logica):



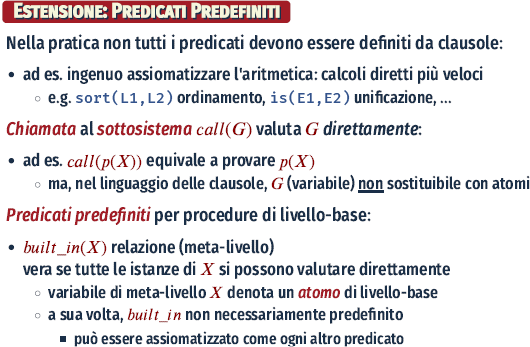
Text

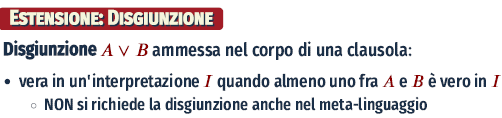
Description automatically generated

Dato il meta livello è possibile chiedere query al meta interprete nel seguente modo:



Il linguaggio-base può essere ampliato oppure ristretto andando a modificare il meta-interprete base che abbiamo visto in particolare andremo ad estendere il linguaggio base aggiungendo nuove clausole al meta-interprete, mentre andremo a ridurre il linguaggio base andando ad aggiungere delle condizioni alle clausole del meta-interprete.

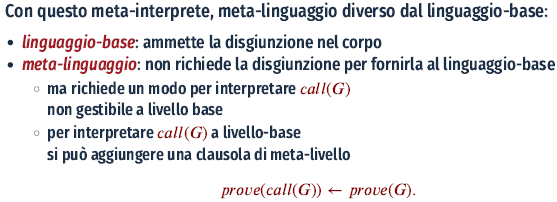




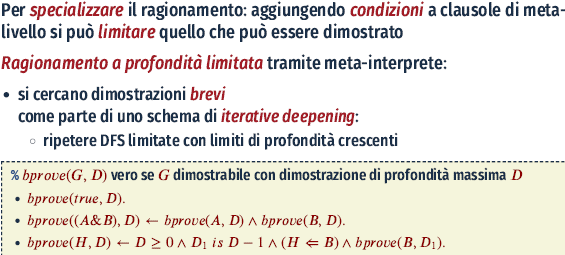
Text

Description automatically generated with medium confidence

Il sistema restituirà sempre o true o false una volta chiamato.



Adesso invece vediamo come usare il meta-linguaggio per restringere l’orizzonte del ragionamento e quindi aggiungendo condizioni al meta-interprete. Si usa nel caso in cui la memoria dell’agente intelligente sia limitata e quindi dobbiamo gestirla in maniera efficiente. Possiamo quindi implementare un ragionamento a profondità limitata.



Viene ad essere codificata nel meta linguaggio la profondità del ragionamento andando a porre la condizione che sia provata la clausola al limite di profondità scelto.

META-INTERPRETI LIMITATI

Si realizza usando una diversa misura sugli alberi di dimostrazione e quindi si adottano come limitazioni il numero dei nodi nell’albero oppure assumendo che ci sia un costo sulle congiunzioni.

Per implementare la domanda how il meta-interprete conserva una rappresentazione della dimostrazione, questa rappresentazione usata è quella dell’albero dimostrazionale che va a navigare per fornire spiegazioni su come abbia ragionato per provare determinate clausole.

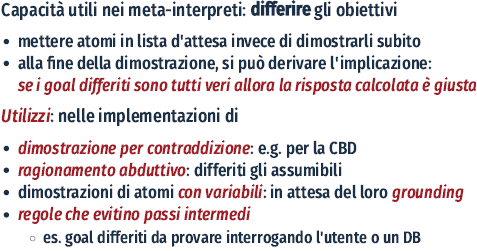
Text

Description automatically generated

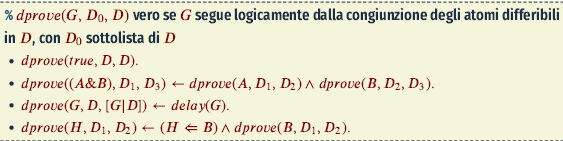
Text, letter

Description automatically generated

Possibilità di differire i goal con i meta-interpreti



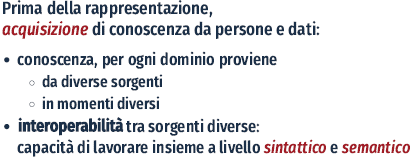
Nel meta-linguaggio vengono codificate le informazioni per poter differire gli atomi da dimostrare in un secondo momento:



CONDIVISIONE DELLA CONOSCENZA BASI DI CONOSCENZA DISTRIBUITE

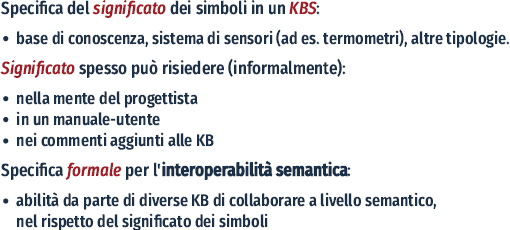
Cercheremo di costruire sistemi che siano in grado di utilizzare la conoscenza sparsa nel mondo nei vari sistemi intelligenti.

Nota: web3.0 è il web dei dati



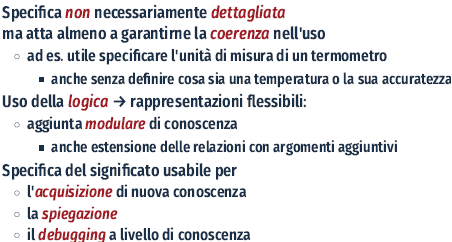
ONTOLOGIA

L’obbiettivo è condividere il significato dei simboli che vengono utilizzati in una KBS.



Un ontologia serve a specificare il significato della terminologia utilizzata per usare la conoscenza.

Usando la logica nelle KB possiamo estendere la conoscenza in maniera modulare ed incrementale e questo ci permette di aumentare i dettagli della conoscenza e allo stesso tempo di renderla flessibile alle varie necessità:



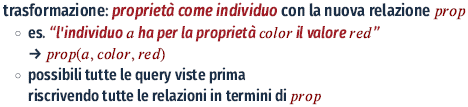
Alla base delle moderne ontologie abbiamo rappresentazioni flessibili supportate dalla logica tra la quale si usano i predicati del primo ordine e si utilizzano individui e relazioni, rappresentazioni grafiche e classi.

Il primo passo da fare è la scelta di cosa sia un individuo e cosa invece debba essere una relazione che va ad agire su un individuo o su più individui.

Text

Description automatically generated

Nella rappresentazione possiamo avere problemi sia nel mappare un concetto come individuo che come relazione per sopperire a queste problematiche si utilizza un nuovo tipo di relazione prop che utilizza una semantica del tipo soggetto – predicato – oggetto questa relazione ternaria ci potrà permettere di dire che un individuo x ha una determinata proprietà y la quale ha un valore z.



A picture containing text

Description automatically generated

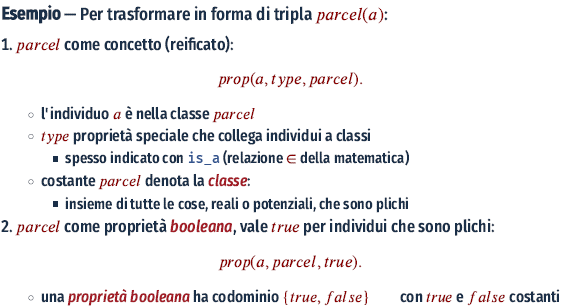
Questa rappresentazione permette di dare vita ai grafi di conoscenza i quali utilizzano queste relazioni ternarie per gestire la conoscenza.

In una tripla, il verbo è una proprietà p che ha un:

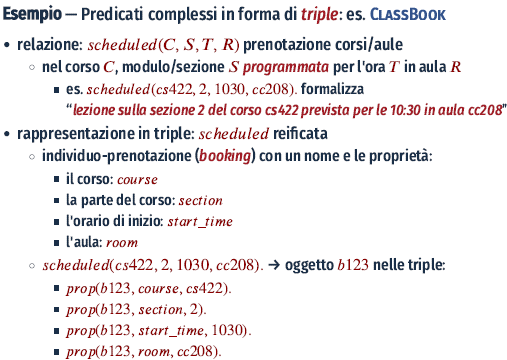
1. Dominio insieme di individui che possono essere i soggetti delle triple
2. Codominio insieme dei valori che possono essere oggetti delle triple

Text

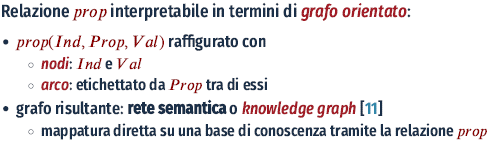
Description automatically generated with medium confidence



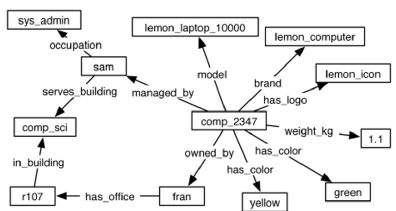
Con la rappresentazione attraverso triple è possibile scomporre un predicato complesso in triple indipendentemente dal numero di argomenti, i quali saranno opportunamente reificati (un po’ come avviene nei processi di concettualizzazione e progettazione delle basi di dati). Per scomporre una relazione n-aria la scompongo in n parti ognuna riferita ad un argomento e poi lego le reificazioni fatte in maniera binaria:



La rappresentazione in triple permette una notevole modularità con che non si riesce ad ottenere con un database relazionale in quanto per andare a modificare uno schema ci necessita una operazione di notevole overhead, mentre aggiungere proprietà con la rappresentazione a triple risulta pratico e rapido. In questo caso si parla di triple store (magazzini di triple) con gestione a grafo e permette di gestire database non relazionali.



Esempio di una rete semantica anche detta knowledge graph:

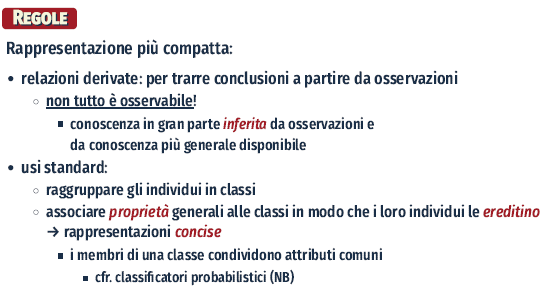


La notazione grafica permette una facile visualizzazione delle proprietà date dalle triple

Text

Description automatically generated

Adesso vediamo come compattare la conoscenza attraverso l’uso di regole, poiché più regole avremo e meno sforzo dovrà essere fatto per spiegare un concetto o simbolo rappresentato dalla KB.



Graphical user interface, text, application, email

Description automatically generated

Text

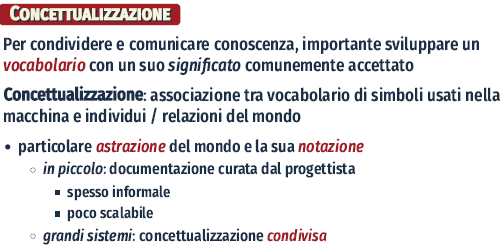
Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

ONTOLOGIE E CONDIVISIONE DELLA CONOSCENZA

L’idea è costruire più basi di conoscenza che abbiamo tutte un linguaggio standard per comunicare di modo che sia possibile la condivisione della conoscenza tra più sistemi.



XML non ci dice nulla sulla semantica definisce solo la sintassi di come è fatta una risorsa.

È inutile creare un nuovo schema di conoscenza quando ci sono standard condivisi.

Un vocabolario è utilizzato per rendere i simboli equivalenti. Esso è parte dell’ontologia e permette di avere una distinzione di categorie da rappresentare e distinguere proprietà da classi.

All’interno di ogni categoria facente parte dell’ontologia vi è un organizzazione delle categorie(ex. Pensa al sistema bibliotecario con la numerazione decimale di Dewey) che permette di gestire al meglio le varie categorie implementando un organizzazione gerarchica basata sul principio dell’ereditarietà (uguale a quello dei linguaggi OOP). La gerarchia dà vita ad una TASSONOMIA delle categorie.

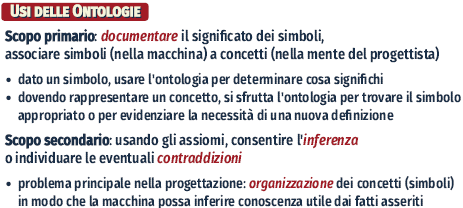
Infine, all’interno di un’ontologia ritroviamo un insieme di assiomi che vincolano la definizione di alcuni simboli per riflettere meglio il significato inteso.

Questi assiomi, che permettono di vincolare le definizioni, permettono di esprimere la transitività di una proprietà, restringere il dominio o il codominio di una proprietà.

Graphical user interface, text

Description automatically generated

Lo scopo principale delle ontologie è quello di documentare come visto con i vocabolari andando ad associare simboli a concetti e se questa documentazione viene ad essere effettuata in modo formale allora posso effettuare sulle ontologie stesse un processo di inferenza o individuare ove presenti contraddizioni concettuali. Con l’inferenza possiamo dire che delle cose sono verificate implicitamente grazie ad altre.



Text

Description automatically generated

Text

Description automatically generated

Le ontologie e il loro utilizzo sono alla base della conoscenza distribuita che prende il nome di web semantico, che permette di avere una conoscenza distribuita attraverso l’infrastruttura di internet.

Graphical user interface, text, application, email

Description automatically generated

RDF(impropriamente chiamato linguaggio in realtà è un protocollo) risulta essere molto simili a SQL ed è conosciuto come SPARQL.

RDF è un linguaggio a livello semantico che sfrutta la sintassi di XML per andare a definire informazione semantica sotto forma di triple <sog, pred, ogg>, RDF-Schema invece permette tramite il linguaggio RDF di andare a definire le relazioni di classe sotto-classe e il concetto di ereditarietà tutte relazioni necessarie per la costruzione di un ontologia.

Per avere la massima potenza espressiva si utilizza il linguaggio nato per gestire le ontologie sul web ed esso è OWL ontology web language.

Gli oggetti, le proprietà e le classi di cui parlavamo alla base della costruzione di un ontologia saranno denotati in modo univoco come risorse URI nel web.

URI 🡪 UNIFORM RESOURCE IDENTIFIER (permette di identificare in modo univoco una risorsa sul web)

Graphical user interface, text, application

Description automatically generated with medium confidence

IRI -> International resource identifier è la forma generale di un URI ed è di fatto lo standard a livello mondiale proposto dallo IETF(internet engineering task force) e a differenza del URI è costituito da una sequenza di caratteri UCS(universal character set). L’utilizzo degli IRI al posto degli URI ha vantaggi e svantaggi, tra i vantaggi è possibile vedere la risorsa in qualsiasi lingua utilizzata da internet, mentre lo svantaggio è che l’uso di diverse codifiche comporta rischi di sicurezza.

Un URI può essere classificato come “URL” o “URN” od entrambi:

1. un URL (Uniform Resource Locator) è un URI che, oltre ad identificare una risorsa, fornisce dei mezzi per agire su di essa o per ottenerne una rappresentazione; inoltre, esso descrive il suo meccanismo di accesso primario e la sua “location” (ubicazione) in una rete.
2. un URN (Uniform Resource Name) è un URI che identifica una risorsa mediante un “name” (nome) in un particolare dominio di nomi che viene detto “namespace”. Un URN può essere utilizzato per descrivere una risorsa senza lasciare intendere la sua ubicazione o su come ottenerne una rappresentazione.

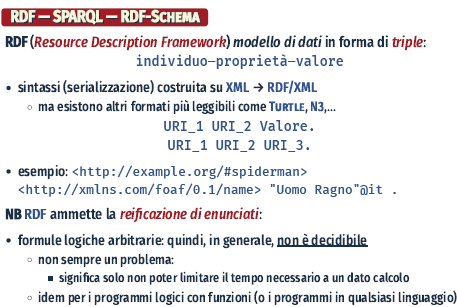
Diagram, venn diagram

Description automatically generatedPer fare un esempio, mentre un URN può essere paragonato al nome di una specifica persona, l’URL può essere pensato come l’indirizzo della residenza di quella persona. Quindi, mentre l’URN identifica univocamente la persona, l’URL fornisce un mezzo per trovarla. Così facendo, gli URL e gli URN hanno scopi complementari anche se entrambi rientrano nell’ambito generale della “resource identification”.

Esempio di un ontologia usata per identificare persone e loro caratteristiche

Graphical user interface, text, application

Description automatically generated



La reificazione degli enunciati permette flessibilità e potenza espressiva che viene pagata in un aumento della complessità che si sposta da decidibile a indecidibile.

Text

Description automatically generated

Vediamo un esempio di sintassi delle query che avviene utilizzando SPARQL:

Timeline

Description automatically generated

Con RDF Schema posso descrivere dei fatti che valgono per più tipi



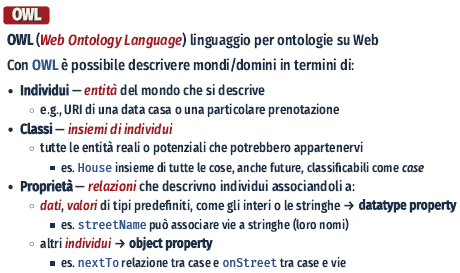


LOGICHE DESCRITTIVE

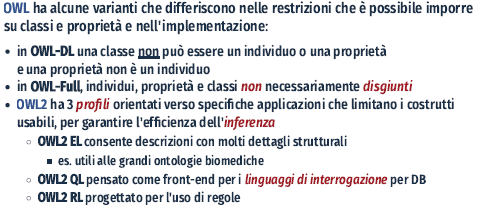
Text

Description automatically generated with medium confidence

Con le logiche descrittive è possibile utilizzare i simboli dell’insiemistica con lo stesso significato e tali simboli hanno un corrispettivo nel linguaggio OWL.



OWL [web ontology language] in realtà è una famiglia di linguaggi



Graphical user interface, text, application, email

Description automatically generated

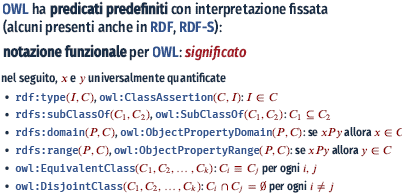
Owl non usa univocità dei nomi poiché stessi nomi possono avere significati diversi perché definiti da risorse differenti.

Text

Description automatically generated with medium confidence

Table

Description automatically generated



Graphical user interface, text, application

Description automatically generated

Alcuni esempi di Ontologie attualmente esistenti:

