OWL (Ontology Web Language)

Creato per **aumentare e migliorare l'espressività di RDF** e RDFS.

- Definire proprietà transitive e riflessive
- Definire assiomi di cardinalità sulle classi

È un linguaggio di markup per **rappresentare esplicitamente significato e semantica di termini con vocabolari e relazioni tra gli stessi.**
Esistono varie versioni del linguaggio, che differiscono molto tra di loro.

Lo scopo di OWL è **descrivere** delle **KB** (*Knowledge Base*), effettuare delle deduzioni su di esse e integrarle con i contenuti delle pagine Web.
OWL intende rendere possibile, ad esempio:

- ricerche nel Web che superino i problemi di omonimia e ambiguità presenti nelle normali ricerche testuali;
- applicazioni che effettuino delle deduzioni sui dati.

La **rappresentazione dei termini e** delle relative **relazioni** è chiamata **ontologia**. Insieme a RDF, di cui è un'estensione, OWL fa parte del progetto del "web semantico".

Il linguaggio OWL è costituito da una **semantica formale** costruita dal consorzio W3C **sulla base dello standard XML** chiamato **RDF**. I due maggiori rilasci di specifiche di tale semantica sono chiamate OWL1 e OWL2.
OWL2 è raccomandato dal W3C ed è completamente compatibile con OWL1.

Decidibilità

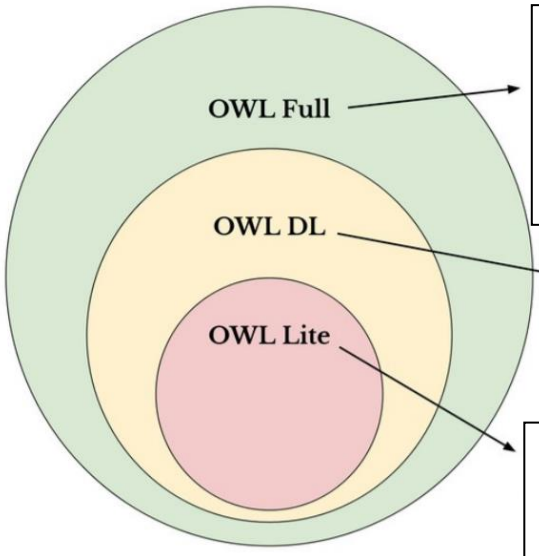
Problema di ***decidibilità*** di un'ontologia (→ ovvero esiste un algoritmo sempre terminante che determina se una soluzione esiste oppure no) risolto basando OWL su logiche descrittive (DL).
RDF è invece basato su *First Order Logic.*
Esiste una sintassi astratta che può essere serializzata nei formati RDF/XML, OWL/XML e Turtle.

Uno dei problemi maggiori delle ontologie è la scelta della potenza espressiva.
La **logica del primo** ordine (RDF) è **molto potente ma non è decidibile**, ossia non è possibile costruire un algoritmo che, dati uno o più assiomi, dica in tempo finito se un'affermazione segue o non segue logicamente da tali assiomi.

Ovviamente questo è sconveniente dal punto di vista dell'utilizzo pratico, perché si vorrebbe avere una **KB che risponda in maniera prevedibile alle richieste**, e non dia informazioni in base alle deduzioni fatte fino a quel momento. Gli approcci al problema sono dunque:

- selezionare una logica meno potente della logica del primo ordine che abbia **meno poter espressivo ma** sia **decidibile**;
- rinunciare alla decidibilità operando in un contesto di affidabilità parziale della base di dati.

Tre principali varianti di OWL 1 (ovvero la prima versione, 2004)



Estensione semantica di RDF.
Linguaggio risultante dall'estensione di RDFS con classi e proprietà necessarie per la codifica della sintassi astratta (molto espressivo ma perde decidibilità). Utilizza la logica del primo ordine, addirittura espandendola con altri predicati. È quindi molto espressivo ma indecidibile.

Linguaggio risultante dalla codifica della sintassi astratta in notazione concreta. Include DL e quindi è più espressivo → permette ragionamento).
È decidibile e ha procedure di deduzione con complessità ben conosciute.

Sottoinsieme sintattico (potere espressivo limitato).
Semplice da implementare ma ha basso potere espressivo.

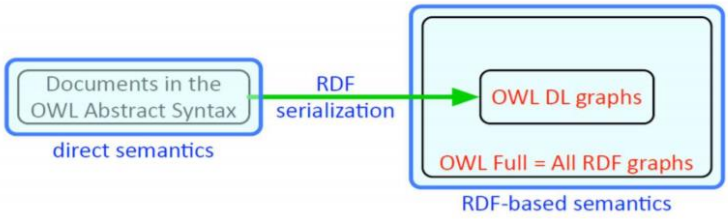
*… in the theory of Description Logics, which calls the original semantic network as "ABox" (for "**Assertional Box**"), the ontology as "TBox" (for "**Terminological Box**") and the pair formed by the two as "Knolwedge Base".*
***In OWL, this distinction is abandoned: a DL knowledge base is simply called "ontology",** and includes both the original network and what we have called "ontology" so far.*

…the term "OWL ontology" is synonymous with "knowledge base" in DLs and includes the two networks discussed above.

Due semantiche

1. Diretta
Definita dalla sintassi astratta e basata su DL.
Non può essere applicata a OWL Full perché il potere espressivo di quest'ultimo non ha senso in DL.

2. Basata su RDF
Definita sulla codifica di OWL Full nella sintassi astratta RDF e bastata sul modello di RDF (*first order logic*). Può essere applicata alle ontologie OWL DL codificate in RDF.



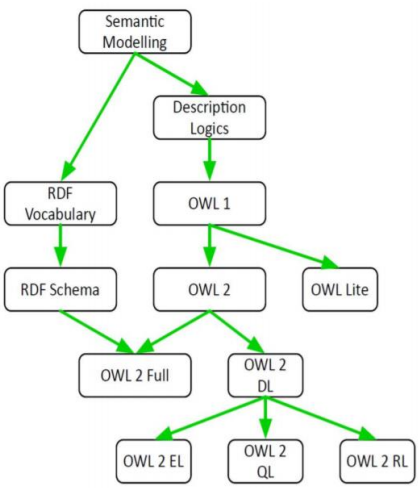
Definizioni

Quantificatore esistenziale	"esistono" o "esiste almeno un". Diverso da quantificatore universale che esprime una proprietà che vale per tutti i membri del dominio.
Ontologia inconsistente	Contiene una contraddizione interna
Class expression subsumption	Calcolo della gerarchia della classe.
Soddisfacibilità dell'espressione della classe	Se esiste un'interpretazione
Controllo dell'istanza	calcolo della classe più specifica per un individuo
Query congiunte	forma limitata di query del primo ordine che utilizza l'operatore di congiunzione logica

Profili

OWL DL è ***intrattabile*** → non ha soluzioni in tempo polinomiale (è EXPTIME-hard).
Per questo motivo è fornito di tre profili più semplici, ognuno pensato pe runa specifica classe di applicazioni.

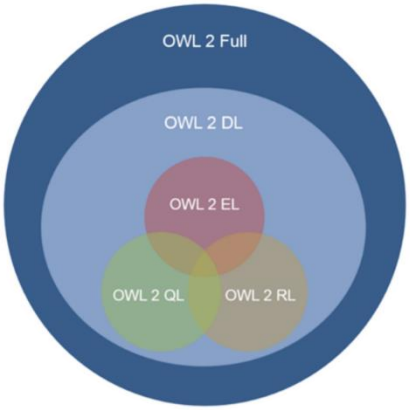
	Rivolto ad applicazioni che…	Risoluzione problemi	Acronimo
OWL 2 EL	Impiegano ontologie le quali contengono un numero molto elevato di proprietà e / o classi. Cattura la potenza espressiva usata da molte di queste ontologie	Problemi di ragionamento di base possono essere eseguiti in tempo che è polinomiale rispetto alla dimensione della KB.	Deriva dalla famiglia EL di logiche descrittive, che forniscono solo la quantificazione esistenziale
OWL 2 QL	utilizzano volumi molto grandi di dati di istanza e in cui la risposta alle query è l'attività di ragionamento più importante.	La risposta alle query congiunte può essere implementata utilizzando sistemi di database relazionali convenzionali; su LOGSPACE può essere eseguita una risposta completa alle query congiunte in relazione alla dimensione dei dati. Come in EL possono essere utilizzati algoritmi in tempo polinomiale per implementare la coerenza ontologica e i problemi c*lass expression subsumption*	Deriva dal fatto che la risposta alle query può essere implementata riscrivendole in un linguaggio di query (QL) relazionale standard.
OWL 2 RL	• richiedono un ragionamento scalabile senza sacrificare troppa potenza espressiva • possono scambiare la piena espressività del linguaggio per efficienza • applicazioni RDF(S) che richiedono una maggiore espressività.	La coerenza ontologica, la soddisfacibilità dell'espressione della classe, la sussunzione dell'espressione della classe, il controllo dell'istanza e la risposta a query congiunte possono essere risolti in tempi polinomiali rispetto alla dimensione del KB.	Il ragionamento può essere implementato usando un linguaggio a regole (RL) standard (sistemi di ragionamento *rule-based*)



OWL Full mantiene la completa compatibilità con la sintassi RDF → la sintassi astratta di OWL non è sufficientemente ricca per esprimere ontologie OWL Full.

Tuttavia OWL Full è ***indecidibile*** → non esiste un algoritmo sicuramente terminante che determina se una soluzione esiste o no.

La famiglia di linguaggi OWL offre più varianti per offrire opzioni nella via di mezzo tra espressività e trattabilità



Ontologia OWL 2 DL

Descrizione formale di un dominio di interesse. È essa stessa una risorsa identificata da un IRI. Consiste in tre categorie sintattiche

- ## Entità (elementi base dell'ontologia)

  - ### Classi (tra cui due classi built-in: owl:Thing, la classe di tutti gli individui, e owl:Nothing, la classe vuota)

  - ### Proprietà
    - *Object prop* → connette due individui
    - *Data prop* → collega individui e literals
    - *Annotation prop*

  - ### Individui
    - *Named*, gli oggetti del dominio
    - Anonimi, equivalenti ai blank node di RDF

  - ### Datatypes

    Sono forniti in OWL allo stesso modo di RDF.

    Ognuno è identificato da un IRI e definito dalle componenti:

    - *value space*
    - *lexical space*
    - *facet space*

    Il *datatype map* di OWL 2 è una lista dei datatype che possono essere usati nelle ontologie OWL 2. Molti sono presi dal XML Schema e da RDF.

Entità + literals = Vocabolario

Entities, literals, anonymous individuals and datatype facets form the vocabulary of an OWL DL ontology

- ## Espressioni: nozioni complesse che catturano l'intensione di classi e proprietà

<table>
<tr><th>In OWL 2 possiamo usare…</th><th>… per creare …</th><th>Sintatticamente:</th><th>Semanticamente:</th></tr>
<tr><td rowspan="2">Proprietà</td><td rowspan="2">property expressions</td><td>Sono uguali ma vi si applicano dei costruttori (sono termini più complessi)</td><td>Hanno un significato relazionale per collegare coppie di individui</td></tr>
<tr><td colspan="2">In OWL c'è solo un tipo di espressione di proprietà: la proprietà inversa, che può essere applicata solo alle object properties. Es. se ho prop <code>childOf</code>, può essere utile usare la sua inversa.</td></tr>
<tr><td>Datatypes</td><td>data ranges</td><td>Come sopra; termini complessi formati applicando:
• intersection
• union
• complement
• enumeration
• restriction</td><td>Come i datatype, ovvero insieme di valori usati come range di proprietà</td></tr>
</table>

<table>
<tr>
<td rowspan="2">Class expressions: servono a definire l’intensione di una classe in modo più specifico rispetto a RDFS

• Set-theoretic expressions: combinare classi usando operatori insiemistici
  ○ <code>intersectionOf</code> (and)
  ○ <code>unionOf</code> (or)
  ○ <code>complementOf</code> (not)
  ○ Enumeration (or…or)

• Property restrictions: restringono il valore di una proprietà quando è applicata ad una certa classe
  • (object) property restrictions:
  ○ Existential Quantification: <code>someValuesFrom</code> (almeno uno, some)
  ○ Universal Quantification: <code>allValuesFrom</code> (tutti, only)
  ○ Individual Value Restriction
  ○ Self Restriction

  • (data) property restrictions (uguale a sopra con l’eccezione che non esiste una data property inverse e che la restrizione è definita su un range di dati invece che su una class expr):
  ○ Existential quantif
  ○ Universal quantif
  ○ Literal value restriction

  • Property Cardinality restrictions
  • (object) property cardinality restrictions: restrizioni sul numero di relazioni dello stesso tipo che un individuo può avere
  ○ cardinalità minima (<code>owl:minCardinality</code>)
  ○ cardinalità massima (<code>owl:maxCardinality</code>)
  ○ cardinalità esatta (<code>owl:cardinality</code>)
  • (data) property cardinality restrictions (uguale a sopra, ma la restrizione è definita su un range di dati invece che su una class expr).</td>
<td rowspan="2">class intensions</td>
<td colspan="2">In RDF identificavamo intensioni in classi che ne indicano le caratteristiche (es. Parent= “resource that is a person and has at least one child”). Ma così si perdono delle inferenze potenzialmente interessanti.
In OWL possiamo dare una descrizione più accurata usando costruttori (es. Parent: Person and at least one <code>hasChild</code> who is a Person)</td>
</tr>
<tr>
<td>Sono termini più complessi formati applicando costruttori a classi, proprietà o espressioni.</td>
<td>selections of features (intension) that in every interpretation denote sets of individuals</td>
</tr>
</table>

Un’ontologia può importare altre ontologie specificandone l’IRI. ***Import closure*** = insieme che contiene l’ontologia e tutte le ontologie che essa importa; non può contenere versioni differenti della stessa ontologia o ontologie aventi annotazione owl:incompatibleWith.

Ontologia associata ad un **documento** in quale la contiene fisicamente; esso è costituito da:

- (zero o più) dichiarazioni di prefissi
- Parola chiave “Ontology” + IRI della versione
- (zero o più) importazioni (parola chiave “Import” + IRI)
- (zero o più) annotazioni
- (zero o più) assiomi

Annotazioni

- ## Contestuali

  Incorporate nell'ontologia, in assiomi o in altre annotazioni. Riguarda qualcosa che emerge dal contesto in cui l'annotazione stessa è incorporata.

  Alcune proprietà delle annotazioni contestuali sono offerte da OWL DL, es. rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy.

  Altre proprietà: owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, owl:incompatibleWith .

➢ Assiomi

= affermazioni che sono vere nel dominio modellato dall'ontologia.

Ontologia = insieme di assiomi. Ne sono il contenuto logico.
Axiom closure = insieme di assiomi che costituiscono l'ontologia.

Tipi di assiomi:

- Dichiarazioni: due scopi
  - dichiarare che un IRI ovvero una entità esiste, cioè che è parte del vocabolario
  - dichiarazione che associa l'IRI ad un tipo (classe, obj prop, data prop, individuo, …)

- **Assiomi sulle (espressioni di) classi**: stabiliscono relazioni tra espressioni di classi
  - Assiomi di sottoclasse (presenti anche in RDFS ma lì connettevano semplicemente classi e invece in OWL connettono espressioni di classi) → `subClassOf`
  - Assiomi di classi equivalenti (come sopra) es. "persona", "person", "personne" → `equivalentClass`
  - Assiomi di classi disgiunte, es. boy, girl → `disjointWith`
  - Unione disgiunta, es. child, boy, girl → `disjointUnionOf`

- **Assiomi su (object) properties**: stabilire relazioni tra object property expressions
  - object sub-property (è amico ←è migliore amico)
  - complex role inclusion (ha un padre che ha un fratello → ha uno zio)
  - equivalent
  - disjoint (ha fratelli, ha sorelle)
  - inverse (è figlio di, è genitore di) → `inverseOf`
  - domain
  - range
  - functional = al massimo un valore (es. *Italy hasPresident sergioMattarella*)
  - inverse functional (*sergioMattarella isPresidentOf Italy*)
  - reflexive (fa riferimento a se stessa) / irreflexive
  - symmetric (è amico di) / asymmetric (è madre di)
  - transitive (A è parte di B e B è parte di C allora A è parte di C)

- **Assiomi su (data) properties** (simili a sopra)
  - data sub-property
  - equivalent (has name , si chiama)
  - disjoint (ha nome, ha indirizzo)
  - domain (ha nome, persona)
  - range (ha nome, xsd:string)
  - functional (ha codice fiscale)

- **Definizioni di datatype**: definisce un nuovo datatype semanticamente equivalente ad un certo range di dati (es. definizione datatype "targa" con restrizioni: due lettere, tre numeri, due lettere)

- **Keys**: insieme di *obj/data prop expr* i cui valori identificano univocamente un'istanza di una *class expr* (es. codice fiscale)

- **Asserzioni** (fatti): stabiliscono relazioni tra individui o tra individui e class/prop expr.
  - same/different individual (batman, Bruce Wayne / batman, superman)
  - class (*Carlo rdf:type Person*)
  - negative/positive *obj prop*
  - negative/ positive *data prop*

- Assiomi sulle annotazioni
  - asserzioni
  - sottoproprietà
  - dominio
  - range

Problemi di inferenza

Una ontologia OWL 2 è soddisfatta in una certa interpretazione se tutti gli assiomi nella *axiom closure* dell'ontologia sono soddisfatti nella stessa interpretazione. Un'interpretazione che soddisfa un'ontologia è un modello dell'ontologia stessa; il contrario non è sempre vero: un modello di un'ontologia non è detto che la soddisfi. Possiamo però "costringerla" a soddisfarla modificando le interpretazioni degli individui anonimi.

Dati:

- D un *datatype map*
- V un vocabolario su D
- CE = *class expression*

Consistenza di ontologie. Una ontologia è **consistente** (o **soddisfacibile**) rispetto a D se esistono un modello dell'ontologia rispetto a D e un vocabolario.

Implicazione di ontologie. Una ontologia ne implica un'altra rispetto a D se ogni modello della prima ontologia rispetto a D e V è anche un modello della seconda rispetto a D e V.

Equivalenza di ontologie. Due ontologie sono equivalenti rispetto a D se la prima implica la seconda (rispetto a D) e viceversa.

Soddisfacibilità della *class expression*. Una CE è soddisfacibile rispetto a O e D se esistono un modello di O (rispetto a D) e un V, tali che

Sussunzione della *class expression*

Controllo delle istanze.

Restrizioni globali sugli assiomi

Per garantire la decidibilità dei problemi di reasoning base.

Axiom closure = il più piccolo insieme contenente, per ogni *object property*, essa stessa e la sua inversa.

L'*axiom closure* di ogni ontologia OWL 2 DL deve soddisfare restrizioni globali necessarie per ottenere un linguaggio decidibile. La definizione formale di queste condizioni si basa sulle nozioni di gerarchia di proprietà e di *simple object property expressions.*

Una ***object property expression*** è **composta** se essa, o la sua inversa, è il lato destro (→) di un *Complex Role Inclusion Axiom* (o se è una delle due object properties build-in, top e bottom).
Se, al contrario, una *object property expression* (o la sua inversa) non hanno nessuna sottoproprietà che è il lato destro di un *Complex Role Inclusion Axiom* allora essa si dice **semplice**.
La relazione → cattura la relazione esplicita di sottoproprietà tra *object property expressions* (ovvero la *obj prop* e la sua inversa).

La relazione di sottoproprietà è riflessiva e transitiva.

Un *axiom closure* soddisfa le restrizioni globali se soddisfa tutte le seguenti restrizioni:

1. `owl:topDataProperty`: la restrizione proibisce di definire superproprietà di essa.
   Esiste un teorema che garantisce che aggiungere datatype al datatype map D non altera l'implicazione e ha delle conseguenze importanti:
   - all'ontologia possono essere applicate semantiche dirette considerando solo i datatype espliciti che occorrono nell'ontologia.
   - Riferendoci ai vari problemi di reasoning, D non necessita di essere fornito in maniera esplicita perché è sufficiente considerare un D implicito contenente solo i datatype dell'ontologia data.
   - I reasoners possono fornire datatype non esplicitamente specificati senza paura di cambiare il significato delle ontologie.
2. **Datatypes**
   Ogni datatype che compare nell'*axiom closure* soddisfa esattamente una delle seguenti condizioni:
   - È un `rdfs:Literal`
   - È contenuto nel D
   - È definito da un singolo assioma di definizione di datatype

   Questo assicura che ad ogni datatype sia data una interpretazione ben definita e che le definizioni di datatype non ridefiniscono i datatype del D.

   Le definizioni di datatype sono acicliche: se un datatype è usato nella definizione di un altro, allora questo non può essere usato nella definizione del primo.

3. **Ruoli semplici**
   Ogni class expression e ogni assioma dei seguenti tipi contiene solo *object properties* semplici:
   `ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, ObjectHasSelf, FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, DisjointObjectProperties`

4. **Gerarchia di proprietà**
   Prevenire definizioni cicliche che coinvolgono assiomi di *object subproperty* con catene di proprietà. Per questo è definita una relazione irriflessiva e transitiva (ovvero uno *strict partial order*, <). Es. se per definire la sottoproprietà di "zio" uso "fratello" non posso usare "fratello" per definire "zio", altrimenti gli assiomi sono ciclici.

5. **Individui anonimi**: non occorrono in assiomi di tipo `SameIndividual, DifferentIndividuals, NegativeObjectPropertyAssertion, NegativeDataPropertyAssertion` e nelle class expression di tipo `ObjectOneOf, ObjectHasValue`

These restrictions limit the expressivity of the language but at the same time they guarantee full control.

OWL DL 2 is based on the description logic SROIQ, which is the most expressive decidable description logic that offers a level of expressivity adequate to realistic applications.