

# Introduzione all'Intelligenza Artificiale

879276 \* A.A. 2023/2024



# Introduzione all'Intelligenza Artificiale

Appunti del Corso ✎ Anno Accademico 2023-2024 ✎ 879276

---

25.09.2023

L'**intelligenza** è un concetto molto generico, include le capacità di ragionare, comprendere idee complesse, apprendere velocemente e apprendere dall'esperienza

È anche definibile come capacità di adattarsi all'**ambiente**, cambiando esso o se stessi.

Non è un singolo processo mentale ma una **combinazione** di vari processi improntati all'adattamento

Esistono due spazi, **pensare** e **agire**, e due assi, **umano** e **razionale**

Pensare come un umano	Pensare razionalmente	Dal Russel-Norvig
Agire come un umano	Agire razionalmente	

## Storia dell'AI

Come "anno zero" per l'intelligenza artificiale si indica l'agosto del 1956, quando si definì la prima **congettura** alla base di tutta la disciplina:

Ogni aspetto dell'apprendimento è dell'intelligenza può in principio essere definito così precisamente che una macchina è in grado di realizzarli

John McCarty  
Marvin Minsky  
Allan Newell  
Herbert Simon

L'AI nasce con un approccio profondamente **multidisciplinare**

Ancora prima di questo evento, Alan Turing, ha definito un test per "verificare" che una macchina sia o meno intelligente

Il test prevede l'interazione tra tre parti umane: A una donna, B, un uomo, e C. C comunica tramite scambio di messaggi con A e B, e deve definire quale dei due è la donna. Ad insaputa di C, A viene sostituita con una macchina; se dopo sufficienti iterazioni C stabilisce correttamente che A è la donna, sia prima che dopo lo scambio con la macchina, allora la macchina è considerabile intelligente

Esistono diversi approcci logici all'AI

### Approccio Simbolico

### Approccio Sub-Simbolico

Secondo l'approccio simbolico, il primo ad essere stato proposto, costruire sistemi in grado di effettuare operazioni su simboli è sufficiente ad effettuare un ragionamento

Gli approcci simbolici sono utili per gestire sistemi con dei vincoli, un insieme finito di azioni che possono modellare lo stato del sistema; situazioni in cui è formalmente definibile il problema.

I **sistemi esperti** sono una delle prime applicazioni dell'approccio simbolico

I sistemi ad approccio simbolico sono hanno come caratteristica la **composizionalità**

L'approccio simbolico è soggetto a vari **limiti**, il numero di assiomi o regole, quindi il **costo della codificazione** della conoscenza, è molto elevato e difficili da gestire, e non è sempre sicuro che si disponga delle conoscenze necessarie. Sono **incerti**

Secondo la seconda corrente, quella degli approcci sub-simbolici, non è necessario manipolare i simboli per ragionare, ma effettuare calcoli in base ad alcuni principi dimostrati capaci di risolvere problemi

Il sistema effettua dei calcoli, poi in base al risultato si misura la bontà della risposta, e si cambia la base di conoscenza

Il primo esempio di sistemi sub-simbolici è stata la **rete neurale artificiale**

Ciascun neurone biologico ha una prima componente che riceve delle informazioni da altri neuroni, e in base ad esse produce un certo output e lo invia ad un altro neurone

Fin dal 1943 sono stati proposti modelli basati sulla struttura dei neuroni biologici

L'idea è che, attraverso un sistema di **pesi** calibrati, posso elaborare i segnali di input per generarne uno di output

Concatenando più neuroni logici in modo che il segnale in uscita da uno sia quello in entrata di un altro, è l'idea alla base del modello a rete neurale multi layer

I **pesi** sono la componente centrale del modello. È possibile **apprendere** come calibrare i pesi, ad esempio in maniera **supervisionata**

La back-propagation è uno dei metodi più importanti proposti per l'addestramento del modello, l'output viene propagato indietro al sistema per permettergli di calibrarsi

A partire da questa definizione di **machine learning** sono stati proposti moltissimi modelli

#### Apprendimento Supervisionato

- **Classificazione**
- **Regressione**

#### Apprendimento non Supervisionato

- **Clustering**
- **Riduzione di dimensionalità**

#### Apprendimento per Rinforzo

Le reti neurali di oggi sono dette **profonde**, reti di deep learning

La profondità è data da un elevato numero di layer di processamento, ad ognuno del quale si analizza un dettaglio differente

**28.09.2023**

#### Agentificazione

Un **agente intelligente** è il tema unificante di questa disciplina; è un agente che percepisce informazioni dall'**ambiente** ed esegue **azioni** di conseguenza. Avviene una **mappatura** tra gli stimoli che riceve e le azioni che svolge

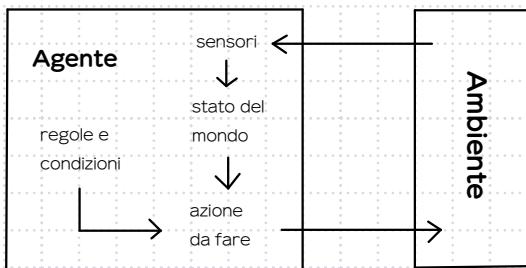
Un agente è qualcuno entità che possa essere vista come **percepente** che svolge azioni che hanno un **effetto** su sé o sul mondo

Un agente autonomo, tramite **sensori** genera una sequenza di percezioni potenzialmente infinita, la funzione che mappa le percezioni alle azioni definisce l'**architettura** dell'agente

Gli agenti possono essere classificati in diverse categorie

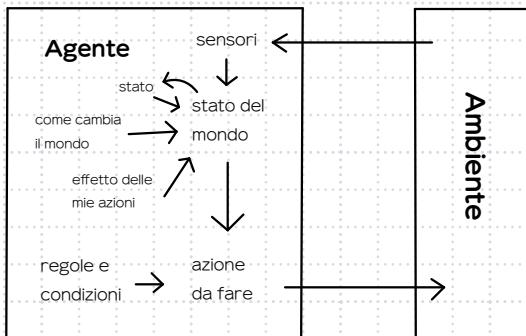
- Simple-Reflex Agents
- Model-Based Goal-Based Agents
- Model-Based Reflex Agents
- Model-Based Utility-Based Agents

### Simple-Reflex Agents



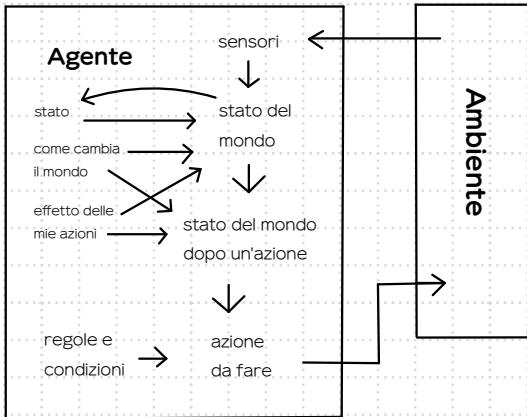
L'agente non è consapevole dello stato del mondo nel tempo e dell'effetto delle sue azioni

### Model-Based Reflex Agents



L'agente ricorda i cambiamenti di stato del mondo nel tempo e conosce l'effetto delle sue azioni

## Model-Based Goal-Based Agents



L'agente sceglie le sue azioni in base al cambiamento che provocherebbero sul mondo al fine di raggiungere un obiettivo di cui è a conoscenza

Un **problema di ricerca** consiste in:

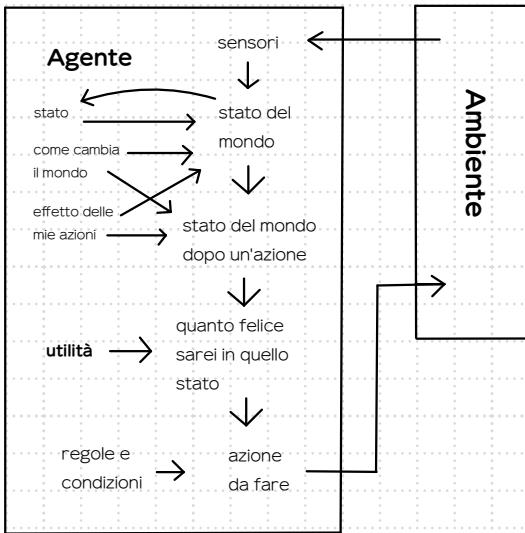
- uno **spazio degli stati**
- una **funzione** successore (con azioni e costi)
- uno **stato iniziale** e un **goal test**

Lo **stato del mondo** contiene ogni singolo dettaglio sul mondo; un **search state** contiene tutti i dettagli necessari a pianificare l'azione successiva (**abstraction**)

Per rappresentare lo spazio degli **stati di ricerca** si utilizza un albero la cui **radice** è lo stato iniziale, e di cui ogni nodo ha tanti figli quante sono le azioni ammissibili dallo stato che rappresenta

Un'altra possibile rappresentazione è quella (meno usata e raramente possibile) a **grafo**, dove ogni nodo rappresenta una configurazione del mondo

## Model-Based Utility-Based Agents



Se posso associare a un cambio di stato un valore numerico che indica il grado di **gradibilità** dello stato successivo, l'agente è in grado di riconoscere lo stato più **utile**, e di conseguenza capace di comportamenti più sofisticati

### Ambiente

Un ambiente può essere **accessibile** o **inaccessibile**; in un ambiente **inaccessibile**, l'agente non ha **informazioni complete**, accurate e aggiornate sul suo stato

Un'altra caratteristica è il **determinismo**; un ambiente è deterministico se le azioni degli agenti hanno un **effetto** unico e **garantito**

In un ambiente **episodico** l'esperienza degli agenti può essere divisa in step **atomici**, ai quali l'agente risponde con una singola azione la cui scelta dipende dal singolo **episodio** o step, senza pensare agli step passati o futuri

Si dice invece **sequenziale** se l'agente ha memoria delle esperienze passate e sceglie la sua prossima azione in base a tutto il contesto temporale dell'ambiente

In un ambiente **statico** mantiene il suo stato finché l'agente non esegue l'azione, ovvero il suo cambiamento di stato dipende unicamente dall'esecuzione dell'azione da parte dell'agente

In un ambiente **dinamico** invece, ci sono più processi che ne modificano lo stato, e il cambiamento dell'ambiente è fuori dal controllo del singolo agente

Un ambiente si dice **discreto** se il numero di azioni e percezioni in esso è finito e fisso; un ambiente **continuo** non viene spesso associato con i sistemi di elaborazione, è quindi possibile "discretizzare" un ambiente attraverso limitazioni.

## Autonomia

La parola **autonomia** ha vari significati e accezioni; la **capacità di decidere** riguardo la propria azione, la **capacità di controllare il proprio stato**. Una definizione forte, proposta da Russel-Norvig, è l'**abilità di decidere basandosi solamente sulle proprie esperienze personali e non conoscenza pre-programmata**

29.09.2023

02.10.2023

Il modello a grafo di conoscenza diretto e etichettato che rappresenta triple **RDF** come la relazione nodo-arco-nodo è ottimale per comprendere la **semantica** dietro i dati. Questo metodo di rappresentazione permette di utilizzare query SPARQL per interrogare la base di dati

## Resource Description Framework (RDF)

L'RDF è uno standard raccomandato dal W3C, fa parte dello stack tecnologico del Semantic Web. È un modello di rappresentazione dati nel web fondamentale per inter-operare con certi formati di dati.

Formalmente, è basato su tre aspetti fondamentali:

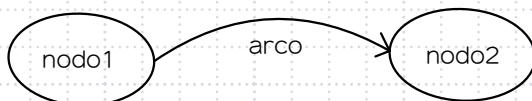
**Triple** (unità base di informazione)

**Grafi etichettati diretti**

**URI**

Una **tripla** RDF, detta anche **statement**, è l'unità atomica di informazione, composta da tre elementi. Nella rappresentazione a **grafo**, i due elementi agli altri estremi (soggetto e oggetto) rappresentano l'etichetta di due nodi, e quello c'centrale (predicato) l'etichetta dell'arco che li collega

nodo1 arco nodo2 .



Un **URI** è un identificatore univoco di risorse, può essere di tipo URL o URN; più generalmente si parla del concetto di **IRI**, ovvero URI che contengono caratteri non solo ASCII ma anche internazionali.

Il protocollo sottostante al modello RDF è HTTP, in modo tale da rendere i dati accessibili via web. Per richiedere un certo dato, viene effettuata una **negoziazione HTTP**

[https://dbpedia.org/resource/some\\_resource](https://dbpedia.org/resource/some_resource)

Una richiesta HTTP di questo tipo viene inoltrata tramite protocollo https da un browser web ad un server che si trova all'indirizzo dbpedia.org

Tale server è un Server Virtuoso, un "negozi" di triple, il quale ritorna automaticamente al browser la risorsa richiesta in un formato congeniale, ovvero una pagina dall'indirizzo:

[https://dbpedia.org/page/some\\_resource](https://dbpedia.org/page/some_resource)

Questo è un esempio di negoziazione di dati

RDF prevede la definizione di **alias**, da poter usare come prefissi per identificare le risorse nelle triple

**dbr:** for <https://dbpedia.org/resource>

**dbr/some\_resource** → [https://dbpedia.org/resource/some\\_resource](https://dbpedia.org/resource/some_resource)

Esistono alcuni tag con funzioni specifiche come ad esempio `rdf:type`

Oltre alle risorse identificate da IRI, esistono anche dei tipi primitivi (**literal**) in RDF, come numeri, stringhe, date...

Le triple RDF sono **human and machine readable**, e sono sempre nella forma:

subject	predicate	object	.
IRI o Blank	IRI	IRI, Blank o Literal	punto

Esistono alcuni prefissi che puntano a risorse esterne (come ad esempio: **foaf**, **dcterms**); queste risorse esterne sono a tutti gli effetti **vocabolari** che mettono a disposizione significati semantici condivisi (**foaf:Person**, **foaf:name**, **foaf:img**)

**FOAF, Friend of a Friend**, è un progetto nato per permettere di descrivere e rappresentare persone, collegamenti tra esse e ciò di cui si occupano, creando pagine web human and machine readable

Foaf, come gli altri vocabolari, sono una **convenzione** per descrivere le risorse

Un altro esempio più moderno, supportato da attori come Google, Microsoft e Yahoo, è **Schema.org**. Questa iniziativa definisce **proprietà** per descrivere più approfonditamente certi concetti, aggiungendo dettagli

Proprietà e attributi di questo tipo sono utilizzati da **motori di ricerca** per interpretare i metadati delle varie entry. Tutte le informazioni visualizzate a seguito di una ricerca possono essere visualizzate in un certo formato o con una certa semantica proprio grazie al modo in cui tali informazioni sono descritte e memorizzate

I formati di cui abbiamo parlato in precedenza sono parte delle **best practices** per rappresentare e **pubblicare** nel web dati provenienti da diversi domini di conoscenza, **self-described** e **connessi tra di loro (Linked Open Data)**

Gli IRI sono i **nomi** delle cose, ed è possibile effettuare richieste HTTP per rendere gli utenti in grado di **cercare** i nomi

Il risultato di una ricerca deve contenere **informazioni utili** riguardo al nome ricercato, in uno dei formati supportati (**RDF**, **SPARQL**), e eventualmente contenere ulteriori IRI per permettere all'utente di conoscere più cose

Per interconnettere banche dati differenti, ovvero per far sì che risorse equivalenti identificate da IRI diversi su piattaforme diverse vengano associate, si può utilizzare la proprietà **owl:sameAs**

dbr:some\_resource      owl:sameAs      wikipedia:the\_same\_resource

RDF, insomma, si può utilizzare per modellare grafi di conoscenza per dati condivisi, ma ha anche ulteriori usi, come ad esempio **incorporare** in pagine web tramite formati di **mark-up** (XML, JSON-LD, RDFa, microdata, hCard...) informazioni codificate in RDF, in modo che strumenti di ricerca possano trarne vantaggio per mostrare in modo efficace e intelligente rich-**snippet** o informazioni aggiuntive nei suoi risultati

06.10.2023

## SPARQL - Query ai Grafi di Conoscenza RDF

**SPARQL** è il linguaggio di interrogazione associato al data model RDF, l'ente responsabile per la sua standardizzazione è il W3C, sul cui sito è possibile trovarne la documentazione

Di seguito, un esempio di query SPARQL

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX mo: <http://purl.org/ontology/mo/>
```

```
SELECT ?album
FROM <http://musicbrainz.org/20130302>
WHERE {
    dbpedia:The_Beatles foaf:made ?album .
    ?album a mo:Record ; dc:title ?title
}
ORDER BY ?title
```

definizione dei **prefissi**

Selezione ?album dal grafo indicato, recuperando le triple che presentano il **pattern** indicato, dove il valore di ?album è di tipo record, e il relativo titolo

Nella specifica 1.1 di SPARQL sono state introdotte funzionalità di gestione dei dati tipiche dei database tradizionali, oltre alla semplice interrogazione dei dati presente nella versione 1.0, tra cui anche condizioni di attraversamento dei grafi

Un **Grafo RDF** è un grafo diretto ed etichettato, un insieme di asserzioni RDF

Un **DataSet RDF** è un insieme di triple, composto da un grafo di default ed eventuali altri grafi nominati; il grafo di default può utilizzare i nomi degli altri grafi come soggetti nelle sue triple

Uno **SPARQL Protocol Client** è un client HTTP che è in grado di mandare richieste SPARQL, mentre un uno **SPARQL Protocol Service** è l'agente che accetta le richieste del client e le elabora su uno **SPARQL Endpoint**

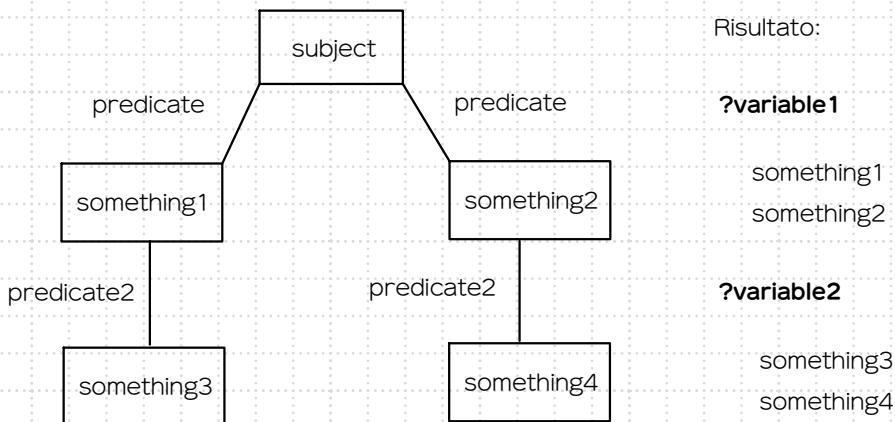
## Query SPARQL

Per eseguire query, si utilizzano degli **RDF Triple Pattern**, ovvero delle triple in cui sono presenti variabili, indicate dalla presenza di un ? O \$ prima del loro nome

subject predicate ?variable .

Il meccanismo base dietro le interrogazioni SPARQL è il **matching di pattern**, una query descrive un **sottografo** dove ogni tripla rispetta il pattern della query; è possibile combinare più triple patterns:

subject predicate1 ?variable1 .  
?variable1 predicate2 ?variable2 .



Nota: "a" si può usare come abbreviazione del predicato "rdf:type"

Un **Graph Pattern** è un grafo dove oltre a costanti e etichette, si utilizzano variabili per identificare i nodi



Per trovare il risultato, si esegue un **mapping** dalle variabili alle costanti presenti nel grafo, ovvero si assegnano alle variabili una costante, e si verifica poi che il grafo pattern risultante sia un valido sottografo del grafo iniziale. In caso positivo, le sostituzioni consistono in un corretto risultato della query

È possibile eseguire join e altre operazioni per eseguire query più complesse, si parla in questo caso di **complex graph pattern**

## Path Queries

Le path query sono query SPARQL che richiedono di **esplorare** il grafo per ottenere il risultato, come ad esempio trovare nodi che sono a distanza di n nodi passando tramite un certo arco (sequence), oppure nodi collegati da diversi tipi di archi (alternatives)

Matcha uno o entrambi i predicati indicati

```
{ :book1 dc:title|rdfs:label ?displayString }
```

**Alternatives**

Trova i nomi delle persone distanti due foaf:knows da Alice

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name .
}
```

**Sequence**

Trova i nodi collegati tramite foaf:name ai nodi che sono foaf:knows/foaf:knows distanti dal nodo ?x, che ha come foaf:mbox la mail indicata (?x è Alice)

Inverte la direzione della proprietà, scambiando oggetto e soggetto

```
{ <mailto:alice@example> ^foaf:mbox ?x }
```

**Inverse Property Paths**

Trovare tutti i nodi che conoscono qualcuno che ?x conosce

```
{
  ?x foaf:knows/^foaf:knows ?y .
  FILTER(?x != ?y)
}
```

**Inverse Sequence Path**

Trova i nomi di tutti i nodi raggiungibili tramite foaf:knows partendo da Alice

```
{
  ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name .
}
```

**Arbitrary Length Match**

**09.10.2023**

Esistono ulteriori query form oltre a SELECT:

**SELECT**

Restituisce le variabili richieste e i loro binding corretti

**ASK**

Testa se un query pattern ha o meno soluzione

**DESCRIBE**

Ritorna un grafo RDF contenente i dati della risorsa

**CONSTRUCT**

Ritorna un grafo RDF del graph pattern specificato

## Esercizi RDF

### 1. Indicare quali triple sono valide

	Subject	Predicate	Object
1	<http://example.org/alice>	rdf:type	foaf:Person
2	_:alice	_:Type	_:Person
3	ex:bob	rdf:type	foaf:Person
4	_:Alice	rdf:type	foaf:Person
5	"bob"^^xsd:string	rdf:type	_:Person
6	_:alice	rdf:type	"Person"^^xsd:string
7	_:bob	rdf:type	_:Person
8	_:Bob	_:Type"^^xsd:string	foaf:Person

**Validi:** 1, 3, 4, 6, 7 (Errori evidenziati)

### 2. Traduzione di frasi in RDF Turtle

Senza **blank** nodes

```
unimib:disco nome "Dipartimento di Informatica Sistemistica e Comunicazione"@it .  
unimib:disco address unimib:indirizzo_disco  
unimib:indirizzo_disco via "Viale Sarca 336"  
unimib:indirizzo_disco citta "Milano"  
unimib:disco part_of unimib .
```

Con **blank** nodes

```
unimib:disco nome "Dipartimento di Informatica Sistemistica e Comunicazione"@it .  
unimib:disco address [ via "Viale Sarca 336"; citta "Milano"] .  
unimib:disco part_of unimib .
```

### 3. Dato il grafo RDF, dare il risultato delle query

```
@prefix : <http://example.org/> . :Student rdfs:subClassOf :Person .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . :MathStudent rdfs:subClassOf :Student .

:bob :knows :alice . :bob :occupation :Pianist .
:carol :knows :alice . :alice :occupation :Guitarist .

:alice :name "Alice" . :alice :parentOf :bob .
:carol :name "Carol" . :bob :parentOf :carol .
:david :parentOf :david .

:bob a :Person . :bob :age 49 .
:carol a :Student . :alice :age 21 .
:david a :MathStudent . :carol :age 33 .
```

1. select ?x ?y where { ?x :knows ?y . }
2. select ?x where { ?x a :Person . }
3. select ?x ?y where { ?x :knows ?y . ?x a :Student . }
4. select ?x ?z where { ?x :knows ?y ; a :Student . ?y :name ?z }
5. select ?x ?y ?z where { ?x :parentOf ?y . OPTIONAL {?y :occupation ?z} }
6. select ?x ?y where { ?x rdfs:subClassOf/rdfs:subClassOf ?y }

1 :bob :alice      2 :bob      3 :carol :alice      4 :carol "Alice"  
:carol :alice

5 :alice :bob :pianist      6 :MathStudent :Person  
:bob :carol  
:carol :david

## Resource Description Framework Schema - RDFS

Introduciamo il concetto di **Ontologia** nel contesto dell'informatica

Un **ontologia assiomatica** è la specificazione formale di concetti tramite un **linguaggio** e un insieme di **assiomi logici**

RDFS è il linguaggio più semplice in una famiglia di linguaggi, che ci permette di effettuare **inferenze** sui dati

In RDF non si ha la possibilità di distinguere tra relazioni "attributo-valore" o relazioni tra risorse con IRI. Per introdurre un semantica è necessario aggiungere vincoli, RDFS fa questo, definendo **classi** e **proprietà** allo scopo di descrivere semanticamente oggetti

<b>rdfs:Resource</b>	ogni cosa descritta tramite RDF è una risorsa di default
<b>rdfs:Class</b>	una classe è un gruppo di risorse, sue istanze
<b>rdfs:Literal</b>	questa classe descrive i letterali, come stringhe o interi
<b>rdfs:Datatype</b>	ci permette di descrivere i tipi di letterali, è un gruppo di risorse

Esistono ulteriori classi come ad esempio **rdfs:subClassOf**, **rdfs:subPropertyOf**,

**rdfs:domain** e **rdfs:range** per indicare relazioni di ereditarietà tra le risorse

In base alla classe, sono definiti e posso definire dei vincoli (**regole di entailment**) per introdurre la possibilità di modellare assiomi logici ed effettuare inferenze

Questo set di costrutti base permette di implementare alcuni task di **ragionamento**

Tuttavia, data la mancanza della rappresentazione della **disgiunzione** tra classi, RDFS non è abbastanza espressivo per trarre **contraddizioni**, ed è quindi possibile ottenere intersezioni non volute