UNICAL - A.A. 2006-2007

Gestione della Conoscenza

Prof. Massimo Ruffolo

Ing. Marco Manna

Capitolo 5

- …
- Linguaggi per il Web Semantico
  - Introduzione
  - Descrizioni di Classi
  - Assiomi di Classe
  - Le Proprietà
  - Individui e Fatti
  - La “wine ontology”
  - Servizi di Ragionamento
- …

Linguaggi per il Web Semantico

Introduzione

Le Logiche Descrittive (1)

- Come abbiamo già detto, le ***DL*** sono linguaggi di rappresentazione adatti alla definizione di ontologie
- Fino ad ora, le ***DL*** più utilizzate a questo scopo sono note come $\mathcal{SHIQ}$ **e** $\mathcal{SHOIN}(\mathrm{D_n})$
- Vediamo ora che cosa significhino questi acronimi

Le Logiche Descrittive (2)

- La lettera $\mathcal{S}$ indica la possibilità di scrivere
  - Enunciati di Sussunzione $\sqsubseteq$ e di Equivalenza $\equiv$ utilizzando i Termini $\top$, $\bot$, $\neg \mathbf{C}$, $\mathbf{C} \sqcap \mathbf{D}$, $\mathbf{C} \sqcup \mathbf{D}$, $\forall \mathbf{R}.\mathbf{C}$, $\exists \mathbf{R}.\mathbf{C}$,
  - Assiomi di transitività dei Ruoli $\boldsymbol{Tr}(\mathbf{R})$
- La lettera $\mathcal{H}$ *(role hierarchy)* indica la possibilità di definire
  - Relazioni di inclusione fra Ruoli $\mathbf{R} \sqsubseteq \mathbf{S}$
- La lettera $\mathcal{O}$ *(one of)* indica la possibilità di definire
  - Termini per Enumerazione $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$

Le Logiche Descrittive (3)

- La lettera $\mathcal{N}$ indica la possibilità di definire
  - Cardinalità non qualificate $\leq nR$, $\geq nR$, $= nR$
- La lettera $\mathcal{Q}$ indica la possibilità di definire
  - Cardinalità qualificate $\leq nR.C$, $\geq nR.C$, $= nR.C$
- Infine $\mathcal{D}_n$ indica la possibilità di utilizzare
  - domini concreti

Il linguaggio OWL (1)

- **OWL** *(Web Ontology Language)* è lo standard proposto dal **W3C** per la definizione di ontologie per il web semantico
  - http://www.w3.org/TR/owl-ref/
- Sviluppato a partire da **DAML+OIL**, il quale
  - implementa la logica $\mathcal{SHIQ}$
  - è basato sui linguaggi **OIL** e **DAML-ONT**
- Prevede tre livelli di complessità crescente
  - **OWL Lite**, semplice da usare e implementare ma scarsamente espressivo
  - **OWL DL**, su logica $\mathcal{SHOIN}(\mathrm{D_n})$, abbastanza espressivo, decidibile e dotato di procedure di ragionamento di complessità nota, studiate ed ottimizzate
  - **OWL Full**, oltre FOL, molto espressivo ma ***"semi-decidibile"***

Note di complessità

- Il linguaggio delle formule valide della logica del primo ordine non è decidibile, bensì **semi-decidibile**: esiste un algoritmo in grado di valutare la validità di una formula
  - Nel caso in cui la formula sia valida l'algoritmo è in grado di terminare restituendo come prova la dimostrazione della sua validità
  - In caso contrario, se la formula non è valida, l'algoritmo non è in grado di accorgersene e continua a eseguire calcoli (si dice che diverge), senza mai fornire una risposta
- Il linguaggio di tutte le formule universalmente valide della logica del secondo ordine non è **neppure semi-decidibile**. Questa è una conseguenza del teorema di incompletezza di Gödel
  - Un eventuale algoritmo che prende in input una formula potrebbe divergere anche nel caso in cui la formula sia valida

Il linguaggio OWL (2)

- Un’ontologia **OWL** si articola in una **TBox** e una **ABox** ambedue rappresentate come **grafi RDF** *(insiemi di **triple RDF**)*
- Diversi costrutti di **RDFS** sono direttamente adottati da **OWL** *(con eventuali limitazioni in OWL DL, e invece con completa libertà d’uso in OWL Full)*
- **OWL** introduce inoltre costrutti propri, non presenti in **RDFS**, comunque rappresentati come **triple RDF**

RDF & OWL

- La rappresentazione RDF di ontologie OWL presenta
  - vantaggi sull'interoperabilità delle applicazioni
  - svantaggi sulla *"chiarezza"* a causa delle numerose varianti consentite da RDF
- D'altra parte gli strumenti per la definizione di ontologie utilizzano in genere interfacce grafiche che nascondono all'utente la rappresentazione RDF
- Di seguito
  - continueremo ad utilizzare i ***simboli tipici delle DL***
  - ma introdurremo e useremo la terminologia specifica di OWL *(che spesso si discosta dalla terminologia più diffusa nel campo delle DL)*

Terminologia

- In **OWL**
  - I **Termini** sono denominati ***descrizioni di classi***
  - Gli **Operatori** per la definizione di termini sono denominati ***costruttori di classi***
  - I **Ruoli** sono denominati ***proprietà***
  - Le **Definizioni Terminologiche** della *TBox* sono dette ***assiomi di classe***
  - Le **Asserzioni** dell'*ABox* sono detti ***fatti***

Linguaggi per il Web Semantico

Descrizioni di Classi

Identificatore

- Ogni ***descrizione di classe*** descrive una **risorsa** di tipo `owl:Class`
  - Nel caso più semplice la descrizione consta di un *identificatore della classe (Unique Resource Identifier = URI)*, corrispondente a un **termine atomico** delle ***DL***
  - Sintassi ***RDF***
    - `<owl:Class rdf:ID="ClassName" />`
- Due identificatori di classi sono già predefiniti in OWL
  - `owl:Thing` = l'insieme di tutti gli individui ($\top$ per la *classe universale*)
    - Ogni classe OWL è sottoclasse di `owl:Thing`
  - `owl:Nothing` = l'insieme vuoto ($\bot$ per la *classe vuota*)
    - La classe `owl:Nothing` è una sottoclasse di ogni classe

Enumerazione

- Una classe **A** può essere descritta *dall'enumerazione* di un numero finito di nominali a_1, …, a_n tramite l'operatore **owl:oneOf**
- Sintassi ***DL***
  - $A \equiv \{a_1, \ldots, a_n\}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#a1" />
    ...
    <owl:Thing rdf:about="#an" />
  </owl:oneOf>
</owl:Class>
```

Restrizioni di proprietà (1)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno** un individuo della classe **C** tramite l'operatore **owl:someValuesFrom**
- Sintassi ***DL***
  - $A \equiv \exists R.C$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:someValuesFrom rdf:resource="#C" />
  </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (2)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **soli individui** di classe **C** tramite l'operatore **owl:allValuesFrom**
- Sintassi ***DL***
  - $A \equiv \forall R.C$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:allValuesFrom rdf:resource="#C" />
  </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (3)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** **sul solo individuo** a tramite l'operatore **owl:hasValue**
- Sintassi ***DL***
  - $A \equiv \forall R.\{a\}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:hasValue rdf:resource="#a" />
  </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (4)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **al massimo n individui** tramite l'operatore **owl:maxCardinality**
- Sintassi ***DL***
  - $A \equiv \leq nR$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
   <owl:onProperty rdf:resource="#R" />
   <owl:maxCardinality rdf:datatype=
     "&xsd;nonNegativeInteger">n</owl:maxCardinality>
  </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (5)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **almeno n individui** tramite l'operatore **owl:minCardinality**
- Sintassi ***DL***
  - $A \equiv \geq nR$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
    <owl:onProperty rdf:resource="#R" />
    <owl:minCardinality rdf:datatype=
      "&xsd;nonNegativeInteger">n</owl:minCardinality>
  </owl:Restriction>
</owl:Class>
```

Restrizioni di proprietà (6)

- Una classe **A** può essere descritta come *restrizione* su un insieme di Individui con ruolo **R** su **esattamente n individui** tramite l'operatore **owl:cardinality**
- Sintassi ***DL***
  - **A ≡ =nR**
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:Restriction>
   <owl:onProperty rdf:resource="#R" />
   <owl:cardinality rdf:datatype=
     "&xsd;nonNegativeInteger">n</owl:cardinality>
  </owl:Restriction>
</owl:Class>
```

Intersezione

- Una classe **A** può essere descritta come *intersezione* di un numero finito di classi C_1, …, C_n tramite l'operatore **owl:intersectionOf**
- Sintassi ***DL***
  - $A \equiv C_1 \sqcap \ldots \sqcap C_n$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#C1" />
    ...
    <owl:Class rdf:about="#Cn" />
  </owl:intersectionOf>
</owl:Class>
```

Unione

- Una classe **A** può essere descritta come *unione* di un numero finito di classi C_1, …, C_n attraverso l'operatore **owl:unionOf**
- Sintassi ***DL***
  - $A \equiv C_1 \sqcup \ldots \sqcup C_n$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#C1" />
    ...
    <owl:Class rdf:about="#Cn" />
  </owl:unionOf>
</owl:Class>
```

Complemento

- Una classe **A** può essere descritta come *complemento* di un'altra classe **B** attraverso l'operatore **owl:complementOf**
- Sintassi ***DL***
  - $A \equiv \neg B$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="A">
  <owl:complementOf>
    <owl:Class rdf:about="#B" />
  </owl:complementOf>
</owl:Class>
```

Linguaggi per il Web Semantico

Assiomi di classe

Sottoclassi

- Fra due descrizioni di classi **C** e **D** si può definire una relazione di sottoclasse attraverso l'operatore **rdfs:subClassOf**
- Sintassi ***DL***
  - $C \sqsubseteq D$
- Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <rdfs:subClassOf rdf:resource="#D" />
</owl:Class>
```

Equivalenza

- Fra due descrizioni di classi **C** e **D** si può definire una relazione di equivalenza attraverso l'operatore **owl:equivalentClass**
- Sintassi ***DL***
  - **C ≡ D**
- Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <owl:equivalentClass rdf:resource="#D" />
</owl:Class>
```

Disgiunzione

- Fra due descrizioni di classi **C** e **D** si può dichiarare una relazione di disgiunzione attraverso l'operatore **owl:disjointWith**
- Sintassi ***DL***
  - $C \sqcap D \equiv \bot$
- Sintassi ***RDF***

```
<owl:Class rdf:about="#C">
  <owl:disjointWith rdf:resource="#D" />
</owl:Class>
```

Linguaggi per il Web Semantico

Proprietà

Proprietà

- Coerentemente con quanto previsto da *RDFS*, in **OWL** anche le ***proprietà*** *(corrispondenti ai ruoli nelle DL)* possono essere viste come ***particolari classi***
- Possono quindi avere sottoproprietà ed essere combinate con vari costruttori
- Così come ogni classe di individui è una risorsa di tipo `owl:Class`, tutte le proprietà sono risorse di tipo `rdf:Property`
- In OWL le proprietà possono essere risorse di due tipi
  - `owl:ObjectProperty` *(proprietà di individui, cioè fra elementi di classi OWL)*
  - `owl:DatatypeProperty` *(proprietà di dati appartenenti a tipi di dati RDFS)*

Sottoproprietà

- Una proprietà **R** può essere definita come sottoproprietà di un'altra proprietà **S** attraverso l'operatore **rdfs:subPropertyOf**
- Sintassi ***DL***
  - $R \sqsubseteq S$
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:subPropertyOf rdf:resource="#S" />
</owl:ObjectProperty>
```

Dominio

- Di una proprietà **R** può essere specificato il dominio attraverso l'operatore **rdfs:domain**
- Sintassi ***DL***
  - $\top \sqsubseteq \forall$ **R.C** *(definizione del dominio* **D***)*
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:domain>
   <owl:Class rdf:about="#C" />
  </rdfs:domain>
</owl:ObjectProperty>
```

Codominio

- Di una proprietà **R** può essere specificato il codominio attraverso l'operatore **rdfs:range**
- Sintassi ***DL***
  - $\top \sqsubseteq \forall R^{-}.D$ *(definizione del codominio* **C***)*
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <rdfs:range>
   <owl:Class rdf:about="#D" />
  </rdfs:range>
</owl:ObjectProperty>
```

Proprietà equivalente

- Una proprietà **R** può essere definita come equivalente a un'altra proprietà **S** attraverso l'operatore **owl:equivalentProperty**
- Sintassi ***DL***
  - **R ≡ S**
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="R">
  <owl:equivalentProperty rdf:resource="#S" />
</owl:ObjectProperty>
```

Proprietà inversa

- Data una proprietà **R** si può definire la proprietà inversa **S** attraverso l'operatore **owl:inverseOf**
- Sintassi ***DL***
  - $S \equiv R^-$
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="S">
  <owl:inverseOf rdf:resource="#R" />
</owl:ObjectProperty>
```

Funzionalità

- Una proprietà **R** è funzionale se soddisfa il vincolo di cardinalità globale espresso dall'operatore **owl:FunctionalProperty**
- Sintassi ***DL***
  - $\top \sqsubseteq \leq 1 R$
- Sintassi ***RDF***

```
<owl:FunctionalProperty rdf:about="#R" />
…
<owl:ObjectProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#C" />
</owl:ObjectProperty>
```

Funzionalità Inversa

- Una proprietà **R** è funzionale inversa se soddisfa il vincolo di cardinalità globale espresso dall'operatore **owl:InverseFunctionalProperty**
- Sintassi ***DL***
  - $\top \sqsubseteq \leq 1 R^-$
- Sintassi ***RDF***

```
< owl:InverseFunctionalProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#C" />
</owl:InverseFunctionalProperty>
```

Transitività

- In OWL è possibile dichiarare che una proprietà transitiva tramite l'operatore **`owl:TransitiveProperty`**
- Sintassi ***DL***
  - ***Tr*(R)**
- Sintassi ***RDF***

```
<owl:TransitiveProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#D" />
</owl:TransitiveProperty>
```

Simmetria

- In OWL è possibile dichiarare che una proprietà simmetrica tramite l'operatore **owl:SymmetricProperty**
- Sintassi ***DL***
  - $R \sqsubseteq R^-$
- Sintassi ***RDF***

```
<owl:SymmetricProperty rdf:ID="R">
  <rdfs:domain rdf:resource="#D" />
  <rdfs:range rdf:resource="#D" />
</owl:SymmetricProperty>
```

Linguaggi per il Web Semantico

Individui e fatti

Appartenenza a una classe

- In OWL è possibile specificare che un individuo **a** appartiene a una classe **C**
- Sintassi ***DL***
  - **C(a)**
- Sintassi ***RDF***

```
<C rdf:ID="a">
  ...
</C>
```

Valori di proprietà

- In OWL è possibile specificare che una proprietà **R** di un individuo **a** ha valore **b**
- Sintassi ***DL***
  - **R(a,b)**
- Sintassi ***RDF***

```
<C rdf:ID="a">
  <R rdf:resource="#b" />
  ...
</C>
```

Identità (1)

- Il linguaggio **OWL** non assume che gli individui abbiano nome unico. Quindi è possibile asserire che due nomi fanno riferimento allo stesso individuo tramite l'operatore **owl:sameAs**
- Sintassi ***DL***
  - a = b
- Sintassi ***RDF***

```
<rdf:Description rdf:about="#a">
  <owl:sameAs rdf:resource="#b" />
</rdf:Description>
```

Identità (2)

- Analogamente è possibile asserire che due nomi fanno riferimento al individui distinti tramite l'operatore **owl:differentFrom**
- Sintassi ***DL***
  - $a \neq b$
- Sintassi ***RDF***

```
<C rdf:ID="a">
  <owl:differentFrom rdf:resource="#b"/>
  …
</C>
```

Identità (3)

- È anche possibile asserire che **n** individui sono tutti distinti fra loro tramite l'operatore **owl:AllDifferent**
- Sintassi ***DL***
  - **a1 ≠ ... ≠ an**
- Sintassi ***RDF***

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <C rdf:about="#a1"/>
    ...
    <C rdf:about="#an"/>
  </owl:distinctMembers>
</owl:AllDifferent>
```

Linguaggi per il Web Semantico

La “wine ontology”

owl:oneOf

- La classe **WineColor** è descritta *dall'enumerazione* di **White**, **Rose**, **Red**
- Sintassi ***DL***
  - **WhineColor ≡ {White, Rose, Red}**
- Sintassi ***RDF***

```
<owl:Class rdf:ID="WineColor">
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White" />
    <owl:Thing rdf:about="#Rose" />
    <owl:Thing rdf:about="#Red" />
  </owl:oneOf>
</owl:Class>
```

owl:someValuesFrom

- La classe **Wine** è descritta come sottoclasse sulla *restrizione* su un insieme di Individui con ruolo **locatedIn** su **almeno** un individuo della classe **Region**
- Sintassi ***DL***
  - $\mathtt{Wine} \sqsubseteq \exists \mathtt{locatedIn}.\mathtt{Region}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="Wine">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn"/>
      <owl:someValuesFrom rdf:resource="&vin;Region"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:allValuesFrom

- La classe **Wine** è descritta come sottoclasse sulla *restrizione* su un insieme di Individui con ruolo **hasMaker** su **soli individui** di classe **Winery**
- Sintassi ***DL***
  - $\mathbf{Wine} \sqsubseteq \forall \mathbf{hasMaker}.\mathbf{Winery}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="Wine">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:allValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:hasValue

- La classe **WhiteWine** è descritta come intersezione tra gli individui della classe **Wine** e la *restrizione* su un insieme di Individui con ruolo **hasColor** **sul solo individuo White**
- Sintassi ***DL***
  - $\mathbf{WhiteWine} \equiv \mathbf{Wine} \sqcap \forall \mathbf{hasColor}.\{\mathbf{White}\}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasColor" />
      <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

owl:maxCardinality

- La classe **WhiteBurgundy** è definita come sottoclasse sulla *restrizione* su un insieme di Individui con ruolo **madeFromGrape** su **al massimo 1 individuo**
- Sintassi ***DL***
  - $\mathtt{WhiteBurgundy} \sqsubseteq \leq 1 \mathtt{madeFromGrape}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="WhiteBurgundy"> ...</owl:Class>

<owl:Class rdf:about="#WhiteBurgundy">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape" />
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:minCardinality

- La classe **Wine** è descritta come sottoclasse sulla *restrizione* su un insieme di Individui con ruolo **madeFromGrape** su **almeno 1 individuo**
- Sintassi ***DL***
  - $\mathbf{Wine} \sqsubseteq \geq 1 \mathbf{madeFromGrape}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="Wine">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape" />
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:cardinality

- La classe **Wine** è descritta come sottoclasse sulla *restrizione* su un insieme di Individui con ruolo **hasMaker** su **esattamente 1 individuo**
- Sintassi ***DL***
  - $\mathbf{Wine} \sqsubseteq =1\mathbf{hasMaker}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="Wine">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

owl:intersectionOf

- La classe **WhiteBurgundy** è descritta come *intersezione* di delle classi **Burgundy** e **WhiteWine**
- Sintassi ***DL***
  - $\mathbf{WhiteBurgundy} \equiv \mathbf{Burgundy} \sqcap \mathbf{WhiteWine}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="WhiteBurgundy">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Burgundy" />
    <owl:Class rdf:about="#WhiteWine" />
  </owl:intersectionOf>
</owl:Class>
```

owl:unionOf

- La classe **WineDescriptor** è descritta come *unione* delle classi **WineTaste** e **WineColor**
- Sintassi ***DL***
  - $\mathbf{WineDescriptor} \equiv \mathbf{WineTaste} \sqcup \mathbf{WineColor}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="WineDescriptor">
  ...
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#WineTaste" />
    <owl:Class rdf:about="#WineColor" />
  </owl:unionOf>
</owl:Class>
```

rdfs:subClassOf

- La classi **Wine** è definita come sottoclasse della classe **PotableLiquid**
- Sintassi ***DL***
  - $\mathbf{Wine} \sqsubseteq \mathbf{PotableLiquid}$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf
    rdf:resource="&food;PotableLiquid" />
  ...
</owl:Class>
```

owl:disjointWith

- Fra le due descrizioni di classi **LateHarvest** e **EarlyHarvest** si può dichiarare una relazione di disgiunzione
- Sintassi ***DL***
  - $\mathbf{LateHarvest} \sqcap \mathbf{EarlyHarvest} \equiv \bot$
- Sintassi ***RDF***

```
<owl:Class rdf:ID="LateHarvest">
  <owl:disjointWith rdf:resource="#EarlyHarvest" />
</owl:Class>
```

rdfs:subPropertyOf

- La proprietà **madeFromGrape** è definita come sottoproprietà della proprietà **madeFromFruit**
- Sintassi ***DL***
  - $\mathbf{madeFromGrape} \sqsubseteq \mathbf{madeFromFruit}$
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:subPropertyOf rdf:resource="&food;madeFromFruit" />
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineGrape" />
</owl:ObjectProperty>
```

rdfs:domain

- Della proprietà **madeFromGrape** è specificato il dominio
- Sintassi ***DL***
  - $\top \sqsubseteq \forall$ **madeFromGrape.Wine** *(definizione del dominio* **WineGrape***)*
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:subPropertyOf
    rdf:resource="&food;madeFromFruit" />
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineGrape" />
</owl:ObjectProperty>
```

rdfs:range

- Della proprietà **madeFromGrape** è specificato il codominio
- Sintassi ***DL***
  - $\top \sqsubseteq \forall$ **madeFromGrape$^-$.WineGrape** *(definizione del codominio* **Wine***)*
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:subPropertyOf
    rdf:resource="&food;madeFromFruit" />
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineGrape" />
</owl:ObjectProperty>
```

owl:inverseOf

- Data la proprietà **madeFromGrape** è definita la proprietà inversa **madeIntoWine**
- Sintassi ***DL***
  - $\mathbf{madeIntoWine} \equiv \mathbf{madeFromGrape}^{-}$
- Sintassi ***RDF***

```
<owl:ObjectProperty rdf:ID="madeIntoWine">
  <owl:inverseOf rdf:resource="#madeFromGrape" />
</owl:ObjectProperty>
```

Appartenenza a una classe

- Si specifica che l'individuo **LoireRegion** appartiene alla classe **Region**
- Sintassi ***DL***
  - **Region(LoireRegion)**
- Sintassi ***RDF***

```
<Region rdf:ID="LoireRegion">
  <locatedIn rdf:resource="#FrenchRegion" />
</Region>
```

Valori di proprietà

- Si specificare che la proprietà **locatedIn** dell'individuo **LoireRegion** ha valore **FrenchRegion**
- Sintassi ***DL***
  - **locatedIn(LoireRegion,FrenchRegion)**
- Sintassi ***RDF***

```
<Region rdf:ID="LoireRegion">
  <locatedIn rdf:resource="#FrenchRegion" />
</Region>
```

owl:differentFrom

- Si asserisce che due nomi fanno riferimento ad individui distinti
- Sintassi ***DL***
  - OffDry ≠ Dry
  - OffDry ≠ Sweet
- Sintassi ***RDF***

```
<WineSugar rdf:ID="OffDry">
  <owl:differentFrom rdf:resource="#Dry" />
  <owl:differentFrom rdf:resource="#Sweet" />
</WineSugar>
```

owl:AllDifferent

- È anche possibile asserire che **n** individui sono tutti distinti fra loro
- Sintassi ***DL***
  - **Red ≠ White ≠ Ros**
- Sintassi ***RDF***

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <vin:WineColor rdf:about="#Red" />
    <vin:WineColor rdf:about="#White" />
    <vin:WineColor rdf:about="#Rose" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

Linguaggi per il Web Semantico

Servizi di Ragionamento

Reasoning

- L'aspetto che distingue nettamente una base di conoscenze da una base di dati è la possibilità di condurre ***ragionamenti*** in modo automatico
- Poiché una base di conoscenze **KB** è costituita da una *TBox* **T** e da una *ABox* **A** scriveremo in generale
  - **KB = ‹T,A›**
- Nel contesto della logica, quando si parla di "*ragionamento*" ci si riferisce sempre a ragionamenti di tipo **deduttivo**, o più semplicemente **deduzioni**
- In generale, quindi, un ragionamento è un procedimento che porta a verificare se un enunciato **X** *(ad esempio la sussunzione o l'equivalenza di due termini)* è ***conseguenza logica*** di una base di conoscenza

Conseguenza logica (1)

- Intuitivamente un enunciato X è conseguenza logica di una base di conoscenze **KB** quando X è certamente vero in ogni situazione in cui siano veri gli assiomi terminologici e le asserzioni contenuti in **KB**.
- Più precisamente, un enunciato X è conseguenza logica di una base di conoscenze **KB** quando X è vero in ogni modello *(nel senso di FOL)* degli assiomi terminologici e delle asserzioni contenuti in **KB**
- In tal caso scriviamo
  - $\mathbf{KB} \models X$
  - **KB** implica logicamente X *(X è conseguenza logica di* **KB***)*

Conseguenza logica (2)

- Consideriamo ad esempio la *TBox* $\mathbf{T}$ contenente le definizioni seguenti:
  - $\mathbf{T1.\ GENITORE \equiv PERSONA \sqcap \exists GenDi}$
  - $\mathbf{T2.\ GenDi : PERSONA \rightarrow PERSONA}$
  - $\mathbf{T3.\ DONNA \equiv PERSONA \sqcap FEMMINA}$
  - $\mathbf{T4.\ UOMO \equiv PERSONA \sqcap \neg FEMMINA}$
  - $\mathbf{T5.\ MADRE \equiv GENITORE \sqcap FEMMINA}$
  - $\mathbf{T6.\ PADRE \equiv GENITORE \sqcap \neg FEMMINA}$
- Il contenuto di $\mathbf{T}$ implica logicamente che certi enunciati, pur non essendo contenuti esplicitamente in $\mathbf{T}$, sono necessariamente veri sotto l'ipotesi che sia vero il contenuto di $\mathbf{T}$. Ad esempio:
  - ogni madre è una persona nonché una donna
  - ogni padre è una persona nonché un uomo
  - la classe delle madri e la classe dei padri sono disgiunte

Conseguenza logica (3)

- Ogni madre è una persona nonché una donna
  - $\mathbf{MADRE} \sqsubseteq \mathbf{PERSONA}$
  - $\mathbf{MADRE} \sqsubseteq \mathbf{DONNA}$
- Ogni padre è una persona nonché un uomo
  - $\mathbf{PADRE} \sqsubseteq \mathbf{PERSONA}$
  - $\mathbf{PADRE} \sqsubseteq \mathbf{UOMO}$
- La classe delle madri e la classe dei padri sono disgiunte
  - $\mathbf{MADRE} \sqcap \mathbf{PADRE} \equiv \bot$

Conseguenza logica (4)

- Per segnalare che questi enunciati sono conseguenze logiche di **T** scriviamo ad esempio
  - $\mathbf{T} \models \mathtt{MADRE} \sqsubseteq \mathtt{PERSONA}$
- Altri enunciati, invece, non sono conseguenza logica di **T**. Ad esempio dalla *TBox* precedente non segue logicamente che una persona abbia almeno due genitori. Per esprimere questo fatto scriveremo:
  - $\mathbf{T} \not\models \mathtt{PERSONA} \sqsubseteq \mathtt{=2GenDi}^{-}$

Tipi di ragionamento

- ***Compito*** di *ragionamento (reasoning task)*
  - è caratterizzato dal tipo di enunciati che si desidera dedurre da una base di conoscenze
- ***Procedura*** di *ragionamento*
  - l'algoritmo che consente la deduzione degli enunciati
- ***Servizio*** di *ragionamento*
  - un servizio effettivamente implementato da uno strumento e messo a disposizione delle applicazioni che accedono alla base di conoscenze.

Compiti di ragionamento *(TBox)*

- *Sussunzione*
  - data una *TBox* $\mathbf{T}$, stabilire se una sussunzione $\mathtt{C} \sqsubseteq \mathtt{D}$ è conseguenza logica di $\mathbf{T}$, ovvero stabilire se $\mathbf{T} \models \mathtt{C} \sqsubseteq \mathtt{D}$
- *Equivalenza*
  - data una *TBox* $\mathbf{T}$, stabilire se un'equivalenza $\mathtt{C} \equiv \mathtt{D}$ è conseguenza logica di $\mathbf{T}$, ovvero stabilire se $\mathbf{T} \models \mathtt{C} \equiv \mathtt{D}$
- *Soddisfacibilità*
  - data una *TBox* $\mathbf{T}$, stabilire se un termine $\mathtt{C}$ è soddisfacibile, cioè che esiste almeno un modello di $\mathbf{T}$ in cui non è vuoto l'insieme degli individui che soddisfano $\mathtt{C}$
- *Disgiunzione*
  - data una *TBox* $\mathbf{T}$, stabilire se due termini $\mathtt{C}$ e $\mathtt{D}$ sono disgiunti, cioè che in ogni modello di $\mathbf{T}$ è vuoto l'insieme degli individui che soddisfano entrambi i termini $\mathtt{C}$ e $\mathtt{D}$

Riduzione alla sussunzione

- Si vede facilmente che i quattro compiti di ragionamento fondamentali per le *TBox* possono essere ridotti alla sola sussunzione
- *Equivalenza*
  - $\mathbf{T} \models C \equiv D$ equivale a $\mathbf{T} \models C \sqsubseteq D$ e $\mathbf{T} \models D \sqsubseteq C$
- *Soddisfacibilità*
  - $\mathbf{T} \not\models C \sqsubseteq \bot$
- *Disgiunzione*
  - $\mathbf{T} \models C \sqcap D \sqsubseteq \bot$
- Questa è la strada che si segue per implementare i servizi di ragionamento per le ***DL poco espressive***

Riduzione alla soddisfacibilità

- I quattro compiti di ragionamento fondamentali per le *TBox* possono anche essere ridotti alla sola soddisfacibilità
- *Sussunzione* $\mathbf{T} \models C \sqsubseteq D$
  - $\mathbf{T} \models C \sqcap \neg D$ **è insoddisfacibile**
- *Equivalenza* $\mathbf{T} \models C \equiv D$
  - $\mathbf{T} \models C \sqcap \neg D$ **è insoddisfacibile** and
  - $\mathbf{T} \models \neg C \sqcap D$ **è insoddisfacibile**
- *Disgiunzione*
  - $\mathbf{T} \models C \sqcap D$ **è insoddisfacibile**
- Questa è la strada che si segue per implementare i servizi di ragionamento per le ***DL molto espressive***, es. $\mathcal{SHOIN}(\mathrm{D_n})$

La procedura SAT

- Per le ***DL decidibili*** – come $\mathcal{SHOIN}(\mathrm{D_n})$ – si può formulare una procedura che prende in ingresso una *TBox* arbitraria $\mathbf{T}$ e un termine arbitrario $\mathbf{C}$ e, in un numero finito di passi, stabilisce se $\mathbf{C}$ è o non è soddisfacibile *(tenendo conto ovviamente delle definizioni terminologiche di* $\mathbf{T}$*)*
- Nelle sue versioni più diffuse questa procedura, che chiameremo **SAT**, è basata sul cosiddetto metodo dei tableaux da tempo studiato e applicato nell'ambito di ***FOL***

Compiti di ragionamento *(ABox)*

- Ci occuperemo ora dei servizi di ragionamento che coinvolgono non soltanto assiomi terminologici della *TBox*, ma anche ***asserzioni*** dell'*ABox*.
- Come abbiamo già notato le ***asserzioni*** contenute in un'*ABox* possono essere *basate su termini* o *basate su ruoli*; ovvero, le ***asserzioni*** possono assumere una delle due forme seguenti:
  - `C(a)` *(`C` termine arbitrario; `a` nominale)*
  - `R(a,b)` *(`R` ruolo; `a`, `b` nominali)*

Compiti di ragionamento *(ABox)*

- *Instance check*
  - dati una *TBox* $\mathbf{T}$, una *ABox* $\mathbf{A}$, un termine arbitrario C e un nominale a, stabilire se si ha $\mathbf{T},\mathbf{A} \models C(a)$
- *Retrieval*
  - dati una *TBox* $\mathbf{T}$, una *ABox* $\mathbf{A}$ e un termine arbitrario C, fra tutti i nominali presenti nella base di conoscenze trovare tutti i nominali $a_1, \ldots, a_n$ tali che $\mathbf{T},\mathbf{A} \models C(a_k)$
- *Realizzazione*
  - dati una *TBox* $\mathbf{T}$, una *ABox* $\mathbf{A}$, un insieme di termini arbitrari $\{C_1, \ldots, C_n\}$ e un nominale a, determinare gli ***m*** termini $\{C_{i1}, \ldots, C_{im}\}$ ***più specifici*** *(sussunzione)* in $\{C_1, \ldots, C_n\}$ per cui si ha $\mathbf{T},\mathbf{A} \models C_k(a)$

Riduzione alla soddisfacibilità

- Un compito di ***instance check*** può essere ridotto a un problema di *soddisfacibilità*
- Un compito di ***retrieval***, poi, almeno in linea di principio può essere ridotto a un compito di *instance check* per ciascun nominale presente nella base di conoscenze
- Un compito di ***realizzazione*** può essere ridotto a una serie di compiti di *instance check* e a una serie di compiti di *sussunzione*
- Ciò significa che, almeno in linea di principio, tutti i compiti di ragionamento che abbiamo esaminato possono essere ridotti a problemi di *soddisfacibilità*

Invocazione dei servizi (1)

- Le specifiche di **OWL** definiscono in modo rigoroso la sintassi dei termini e degli enunciati logici ammissibili, ma curiosamente non stabiliscono una modalità standard per l'invocazione dei servizi di ragionamento fondamentali
- Scrittura convenzionale
  - `?- interrogazione → risposta`

Invocazione dei servizi (2)

- In particolare esprimeremo le invocazioni dei servizi di ragionamento presentati nel modo seguente:
  - *Sussunzione* **?- C ⊑ D → yes/no**
  - *Equivalenza* **?- C ≡ D → yes/no**
  - *Soddisfacibilità* **?- C → yes/no**
  - *Disgiunzione* **?- C ⊓ D ⊑ ⊥ → yes/no**
  - *Instance check* **?- C(a) → yes/no**
  - *Retrieval* **?- C(*) → $\{a_1, \ldots, a_n\}$**
  - *Realizzazione* **?- a:$C_1, \ldots, C_n$ → $\{C_{i1}, \ldots, C_{im}\}$**

Esempio (I)

- Consideriamo la *TBox* **T** seguente:
  - **T1. GENITORE ≡ PERSONA ⊓ ∃GenDi**
  - **T2. GenDi : PERSONA → PERSONA**
  - **T3. DONNA ≡ PERSONA ⊓ FEMMINA**
  - **T4. UOMO ≡ PERSONA ⊓ ¬FEMMINA**
  - **T5. MADRE ≡ GENITORE ⊓ FEMMINA**
  - **T6. PADRE ≡ GENITORE ⊓ ¬FEMMINA**
  - **T7. STATO ≡ {au, ch, de, es, fr, it, uk}**
  - **T8. CittDi : PERSONA → STATO**
  - **T9. ITAL ≡ PERSONA ⊓ ∃CittDi.{it}**
  - **T10. BRIT ≡ PERSONA ⊓ ∃CittDi.{uk}**

Esempio (II)

- Definiamo l'*ABox* **A** seguente:
  - **A1. DONNA(anna)**
  - **A2. DONNA(cecilia)**
  - **A3. UOMO(bob)**
  - **A4. GenDi(anna,cecilia)**
  - **A5. GenDi(bob,cecilia)**
  - **A6. CittDi(anna,it)**
  - **A7. CittDi(bob,uk)**
  - **A8. CittDi(cecilia,it)**
  - **A9. CittDi(cecilia,uk)**

Esempio (III)

- *Instance check*
  - Dati il termine **FEMMINA ⊓ GenDi** e il nominale **anna** si ha:
  - **?- FEMMINA ⊓ ∃GenDi(anna) → yes**
- *Retrieval*
  - Dato il termine **GENITORE** si ha:
  - **?- GENITORE ⊓ ∃GenDi(anna) → {anna, bob}**