



**UNIVERSITÀ**  
**DI TORINO**

**di.unito.it**

DIPARTIMENTO  
DI INFORMATICA

# **Editing di ontologie tramite il linguaggio di programmazione funzionale $\mathbb{C}$ Duce**

**Candidato:**  
**Daide Camino**

**Relatrice:**  
**Viviana Bono**

# Introduzione

Definizioni e presentazione degli strumenti usati

# Introduzione

- Ontologie
- Editing
- C<sub>D</sub>uce



# Introduzione

- Ontologie
- Editing
- C<sub>D</sub>uce



# Introduzione

- **Ontologie**

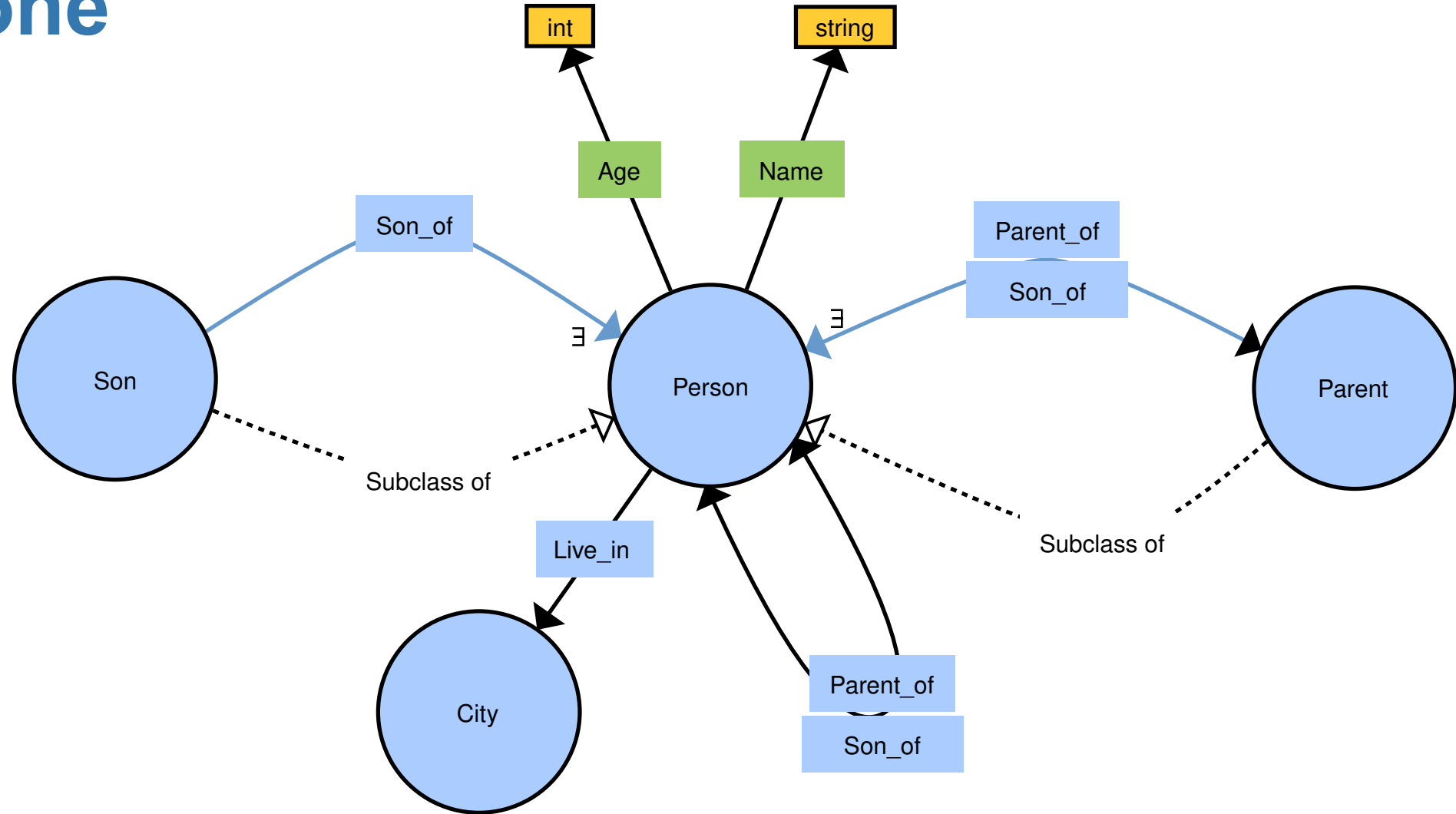
- Editing

- Duce

- Basi di conoscenza (KB)
- Classi e individui
- Gerarchia di classi ed ereditarietà multipla
- Relazioni tra classi
- Salvate come documenti XML con tag OWL

# Introduzione

- **Ontologie**
- Editing
- C<sub>D</sub>uce



# Introduzione

- Ontologie
- Editing
- C<sub>D</sub>uce



# Introduzione

- Ontologie

- Editing

- Duce

- Refactor

- Traduzione partendo da altre basi di conoscenza

- Merge



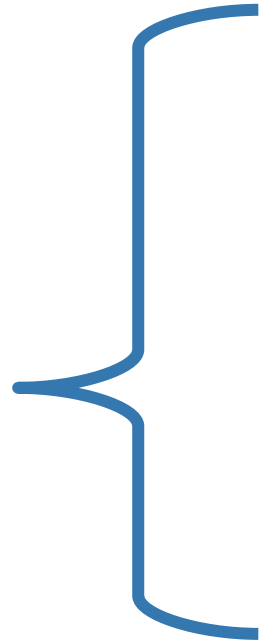


# Introduzione

- Ontologie

- Editing

- Duce



- Refactor

- Traduzione partendo da altre basi di conoscenza

- Merge



# Introduzione

- Ontologie
- Editing
- **CDuce**



# Introduzione

- Ontologie

- Editing

- **CDuce**

- Sviluppato specificatamente per manipolare documenti XML

- Utilizzabile in modo interattivo o tramite script

- Controllo statico dei tipi



# Introduzione

- Ontologie
- Editing
- **CDuce**

## Funzioni

- Pattern Matching
- Overloading
- Funzioni di ordine superiore
- Applicazioni parziali

```
1 fun f (t1 → s1; ... ; tn → sn)
2   | p1 → e1
3   ...
4   | pm → em
```

# Introduzione

- Ontologie

- Editing

- **CDuce**

## Manipolazione di liste

- map

```
1 map e with
2   | p1 → e1
3   ...
4   | pn → en
```

- transform

```
1 transform e with
2   | p1 → e1
3   ...
4   | pn → en
```

# Introduzione

- Ontologie
- Editing
- **CDuce**

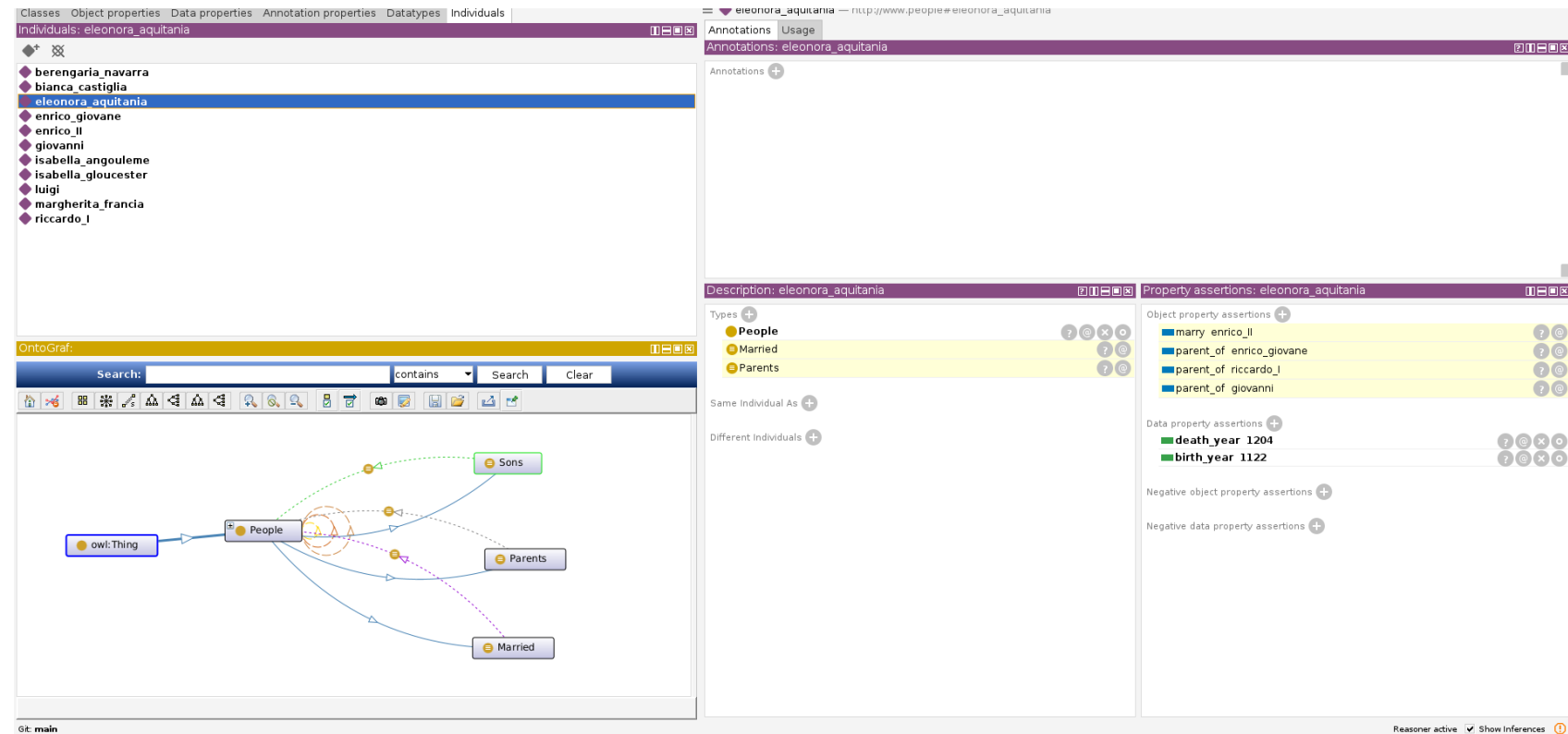
## Query e proiezioni

```
1  select e from
2      p1 in e1,
3      p2 in e2,
4      :
5      pn in en
6  where c
```

# Introduzione

- Ontologie
- Editing
- **ⓈDuce**

## Confronto con Protégè



# Traduzione da un'altra base di conoscenza

Trasformare un tesoro in un ontologia in modo da aumentarne l'espressività



# Cos'è un tesaurus?

- Dizionario tassonomico
- Forniscono sinonimi, contrari e traduzioni
- Relazione “broader term”
- Tag “SKOS”



# Cos'è un tesauo?

Europeana Fashion Thesaurus



europ<sup>e</sup>ana  
fashion

# Cos'è un tesauro?

## fashion objects

- ++ fashion objects
  - -- costume accessories
    - -- costume accessories worn
      - amulet
      - -- accessories worn on arms or hands
        - armband
        - -- handwear
          - mittens (handwear)
          - mitts (fingerless handwear)
          - glove
          - muff
          - sports glove
          - gauntlet
      - -- accessories worn on the legs or feet
        - chaps
        - -- footwear



# Parsing del tesauro

```
1  type Thesaurus = <rdf:RDF>[Desc *]
2  type Desc = <rdf:Description rdf:about=String>[DescAtt *]
3
4  type DescAtt = AltLabel | InScheme | PrefLabel | ScopeNote
                    | ExactMatch | Broader | Type
5  type AltLabel = <skos:altLabel xml:lang=String> String
6  type InScheme = <skos:inScheme rdf:resource=String> []
7  type PrefLabel = <skos:prefLabel xml:lang=String> String
8  type ScopeNote = <skos:scopeNote xml:lang=String> String
9  type ExactMatch = <skos:exactMatch rdf:resource=String> []
10 type Broader = <skos:broader rdf:resource=String> []
11 type Type = <rdf:type rdf:resource=String> [];;
```



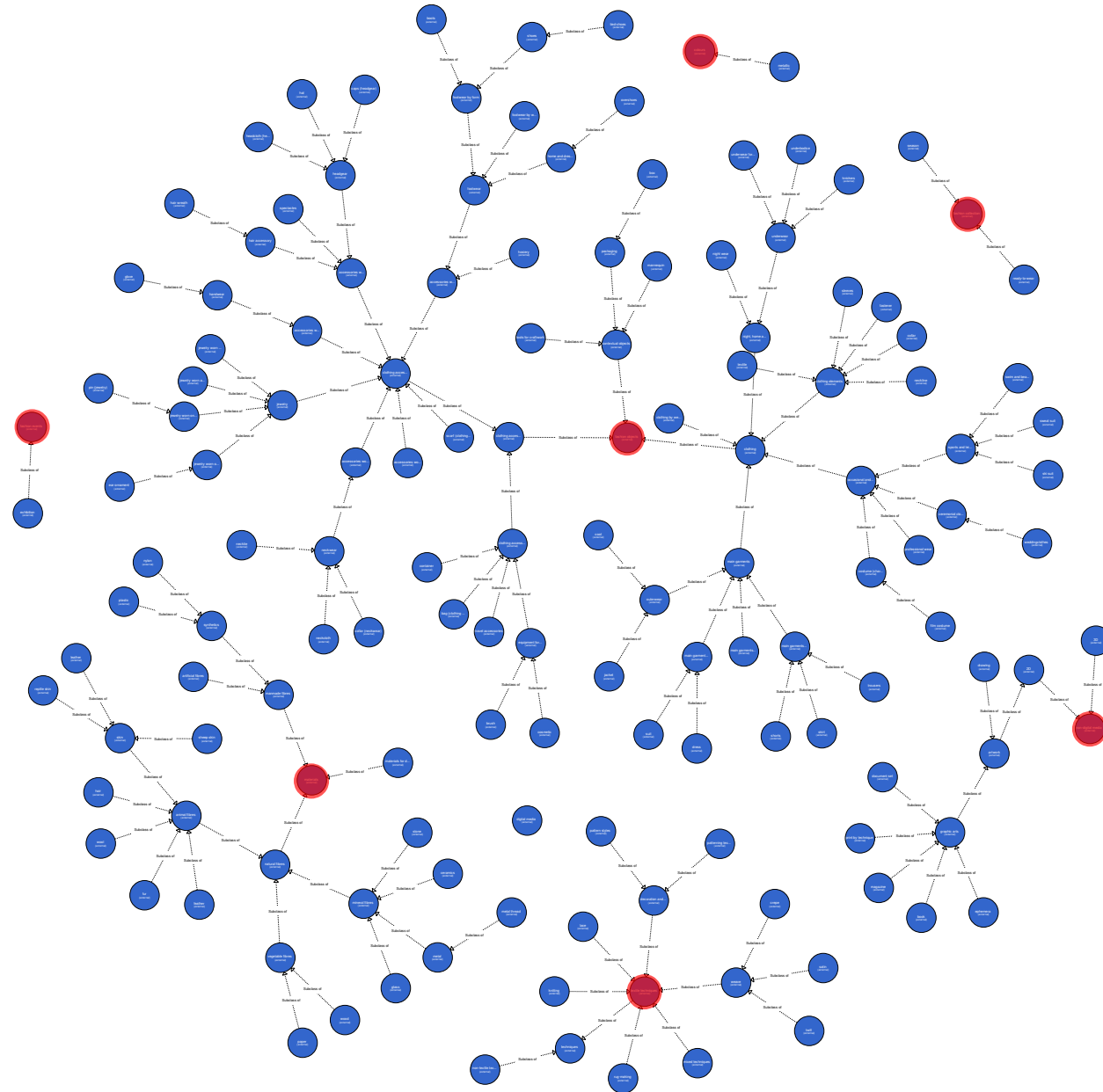
# Struttura dell'ontologia

```
1  type Ontology = <rdf:RDF xml:base=String> [ Thing * ]
2  type Thing = Ont | Class | Individual
3
4  type Ont = <owl:Ontology rdf:about=String> []
5
6  type Class = <owl:Class rdf:about=String> [ ClassAtt * ]
7  type ClassAtt = SubClass | Label | Note | Dictionary
8  type SubClass = <rdfs:subClassOf rdf:resource=String> []
9  type Label = <rdfs:label xml:lang=String> String
10 type Note = <skos:scopeNote xml:lang=String> String
11 type Dictionary = <skos:exactMatch rdf:resource=String> []
12
13 type Individual = <owl:NamedIndividual rdf:about=String>
                        [ IndAtt * ]
14 type IndAtt = IndClass | IndProp
15 type IndClass = <rdf:type rdf:resource=String> []
16 type IndProp = <_ rdf:resource=String> [];;
```

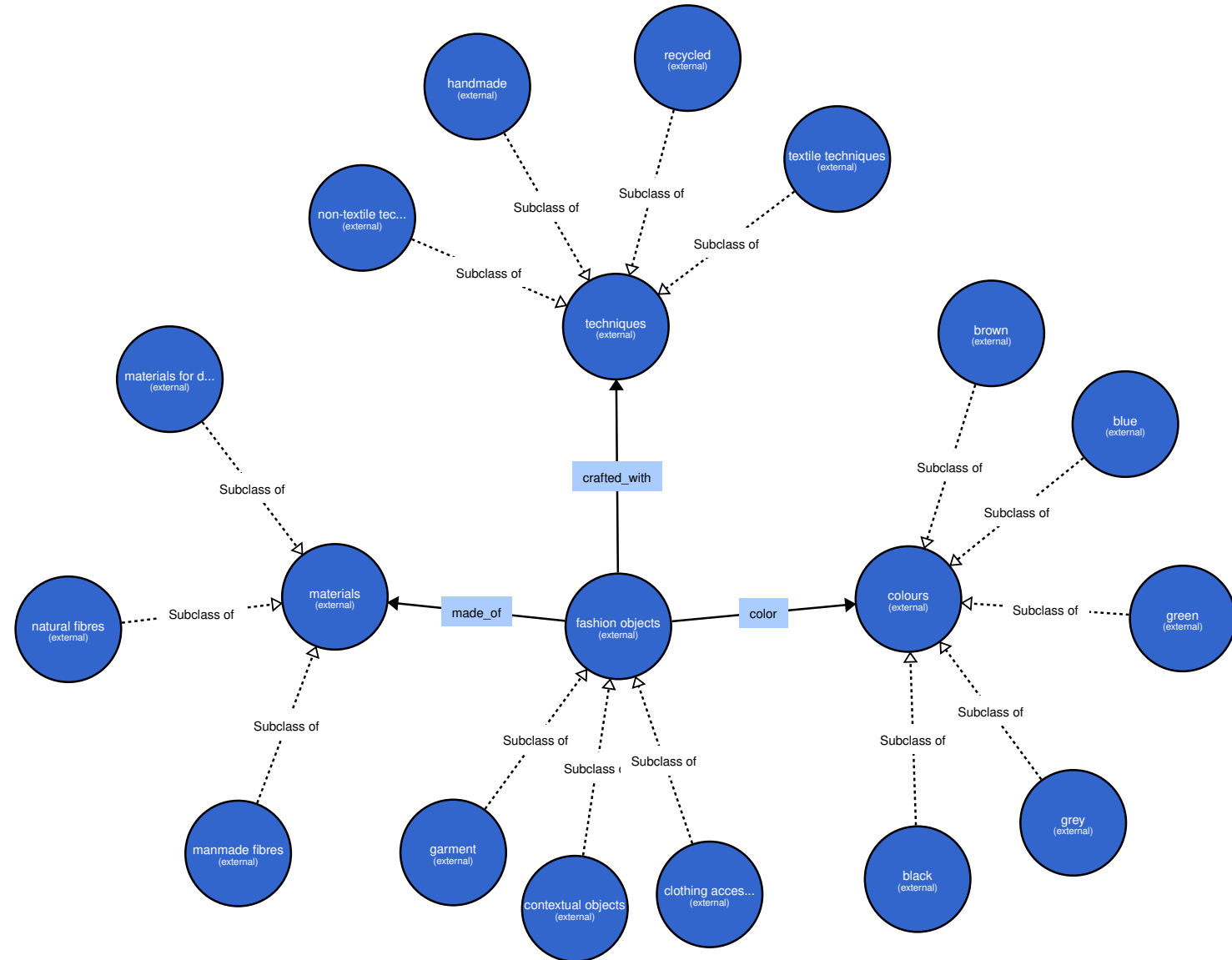
# Traduzione

```
1 let fun thesaurusToOntology (Thesaurus → Ontology)
2   <rdf:RDF>[ (concepts :: Desc) * ] →
3     let newClasses = map concepts with
4       <rdf:Description rdf:about=ab> [ (descAttr :: DescAtt)* ] →
5         let classAttr = transform descAttr with
6           | x & ScopeNote → [x]
7           | x & ExactMatch → [x]
8           | <skos:prefLabel xml:lang=l> lab
9             → [<rdfs:label xml:lang=l> lab]
10          | <skos:altLabel xml:lang=l> lab
11            → [<rdfs:label xml:lang=l> lab]
12          | <skos:broader rdf:resource=res> []
13            → [<rdfs:subClassOf rdf:resource=res> []]
14          in
15            <owl:Class rdf:about=ab> classAttr
16        in
17          <rdf:RDF xml:base="http://www.semanticweb.org/OntEur">
18            ( [ <owl:Ontology rdf:about="OntEur"> [ ] ] @ newClasses );;
```

# Risultato



# Risultato





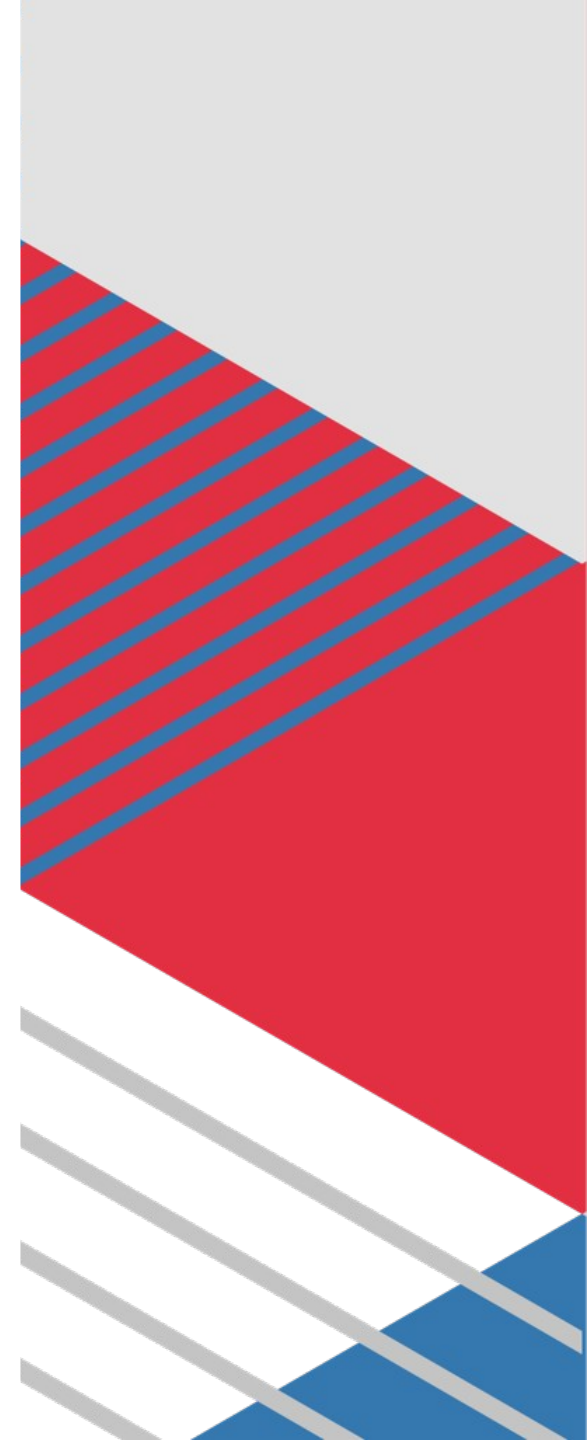
# Merge di Ontologie

Unione ontologie semplici per creare una descrizione di un dominio più complesso

# Obbiettivo del merge

**Rappresentazione  
della conoscenza**

- Accorpare classi equivalenti
- Definire nuove relazioni
- Importare classi e individui già definiti
- Mostrare funzioni più sofisticate
- Derivare proprietà senza reasoner esterni

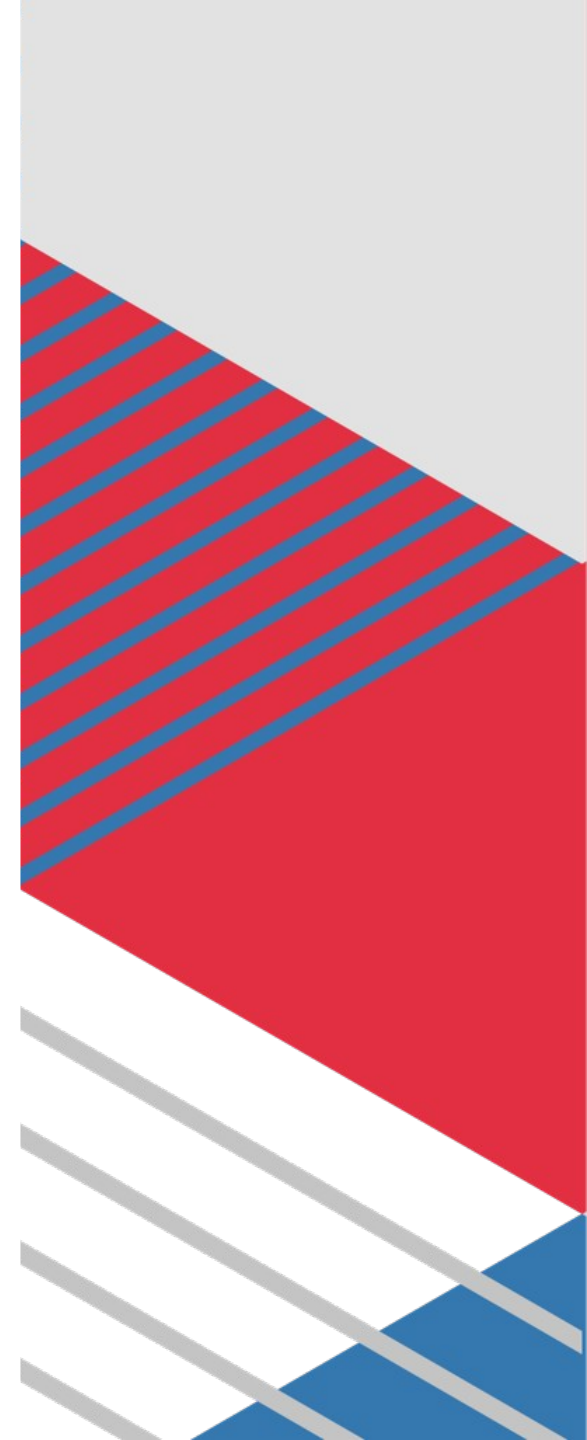


# Obbiettivo del merge

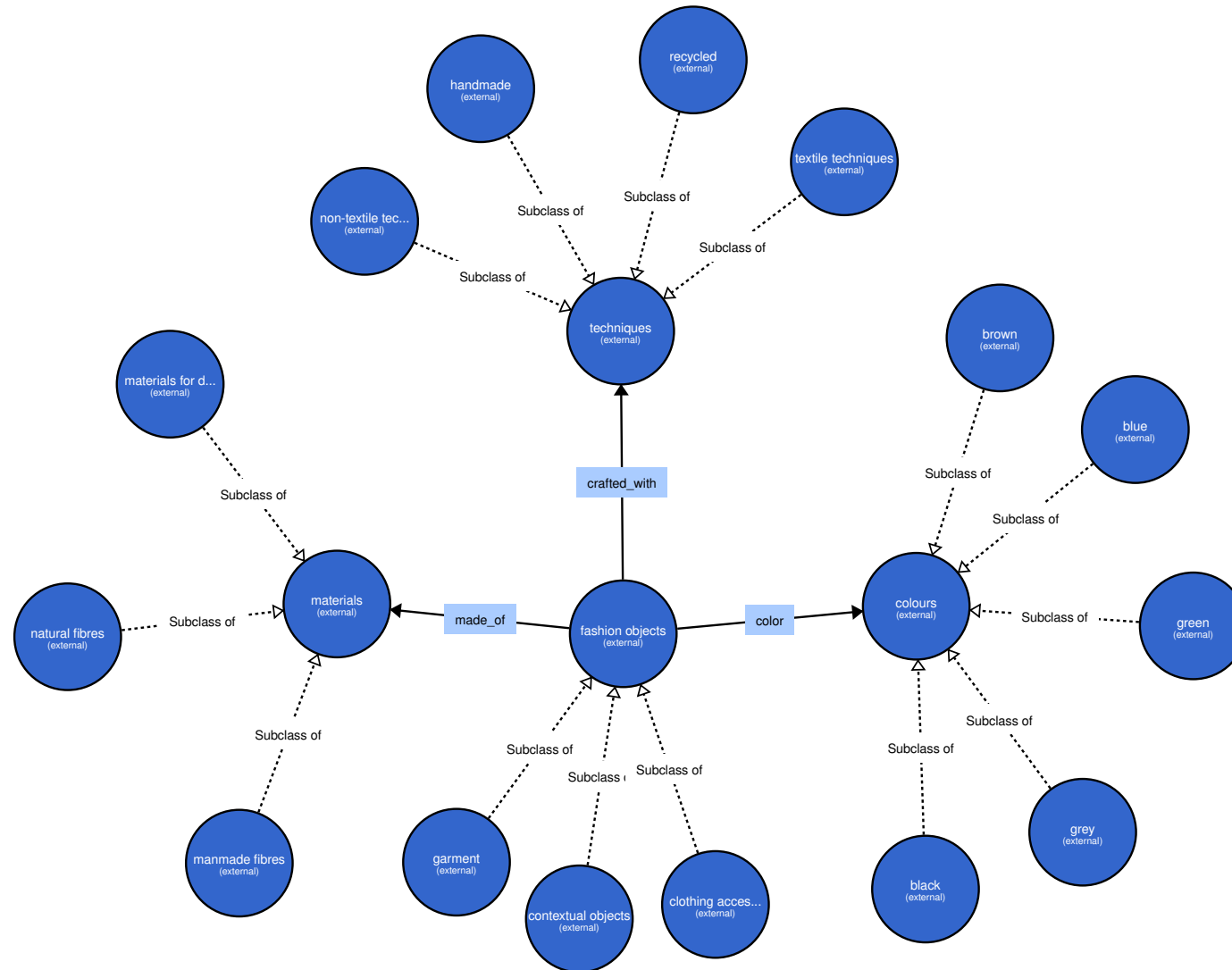
- Accorpare classi equivalenti
- Definire nuove relazioni
- Importare classi e individui già definiti

Ⓒ **Duce**

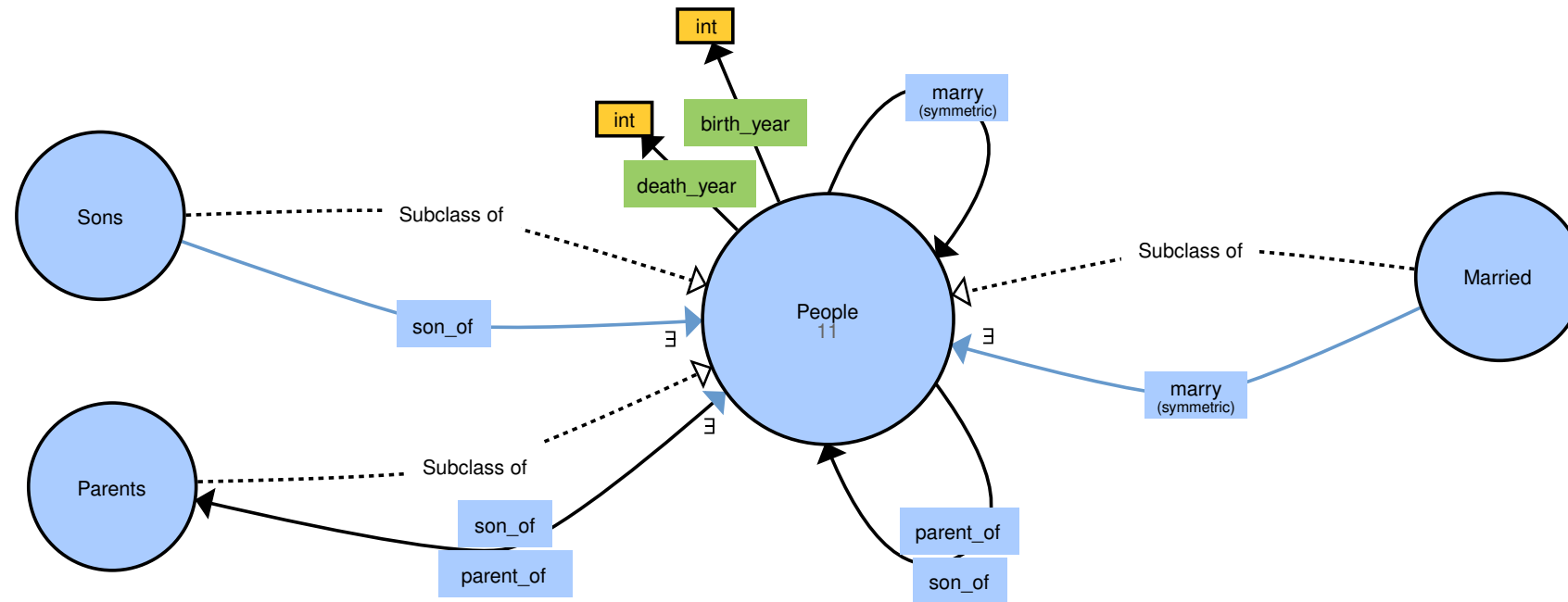
- Mostrare funzioni più sofisticate
- Derivare proprietà senza reasoner esterni



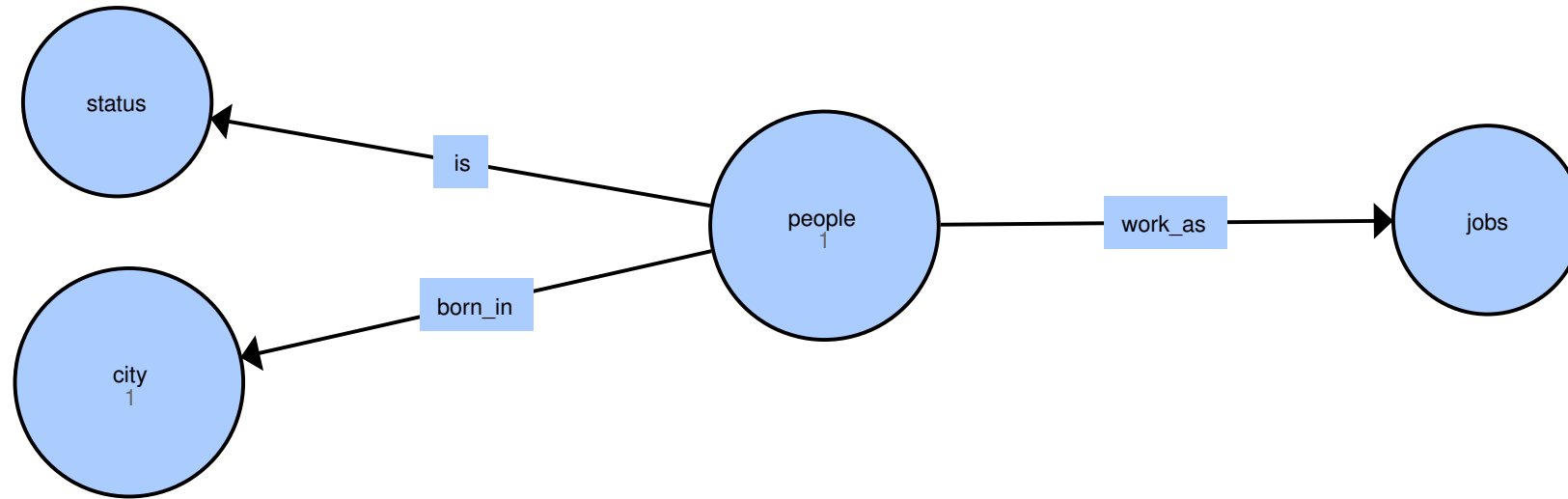
# Ontologie di partenza



# Ontologie di partenza

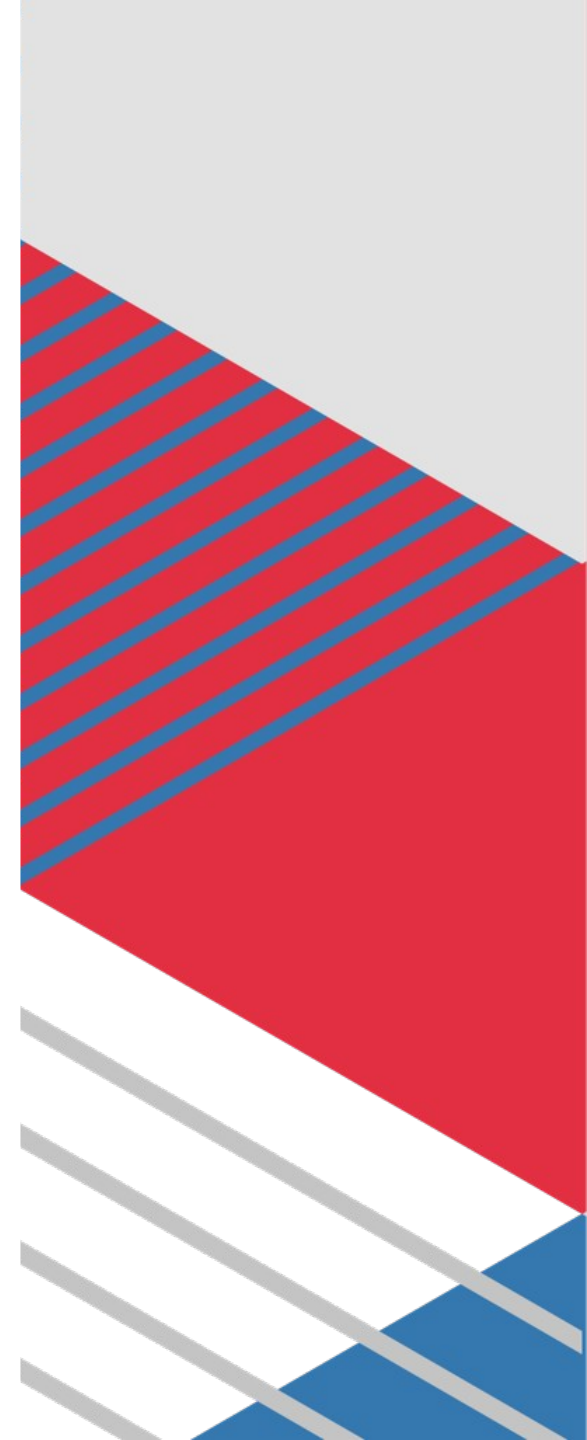


# Ontologie di partenza



# Obbiettivo del merge

- Importare tutte le persone vissute un un dato periodo storico
- Importare gli abiti e i materiali presenti in quel periodo
- Mantenere tutte le relazioni definite in precedenza
- Definirne di nuove per descrivere usi e costumi della società del periodo storico di interesse





# Funzioni ricorsive

```
1  let fun andList ( [ Bool* ] → Bool )
2    | ([ x ] @ y) → (x && (andList y))
3    | [] → `true;;
```

```
1  let fun subClasses (Class → Ontology → [ Class* ])
2    <owl:Class rdf:about=ab> [ _* ] → fun (Ontology → [ Class* ])
3    ont →
4      select x from
5        x in [ont]/Class,
6        y in [x]/SubClass
7        where (y = <rdfs:subClassOf rdf:resource=ab> []);;
8
9  let fun subClassesRec ( Class → Ontology → [ Class* ])
10    cl → fun (Ontology → [ Class* ])
11    ont →
12      let subCl = subClasses cl ont in
13      subCl @ flatten (map subCl with y → subClassesRec y ont)
```



# Reasoning

```
1 let fun madeOf (Individual → Ontology → [Individual*])
2   <owl:NamedIndividual ..> [ (ip::IndProp | _) *] → fun (Ontology → [Individual*])
3   ont →
4     transform ip with <ns1:made_of rdf:resource=str> [] →
5     transform [ont]/Individual with x →
6     match x with <owl:NamedIndividual rdf:about=a>[ _* ] → if a = str then [x] else [];;
```

```
1 let fun isArtificial (Individual → [Class*] → [Class*] → Ontology → Bool)
2   ind → fun ([Class*] → [Class*] → Ontology → Bool)
3   materials → fun ([Class*] → Ontology → Bool)
4   artificialMats → fun (Ontology → Bool)
5   ont → if (isInClasses ind materials ont)
6     then (isInClasses ind artificialMats ont)
7     else
8       let matMade = madeOf ind ont in
9       orList (map matMade with x → isInClasses x artificialMats ont);;
```

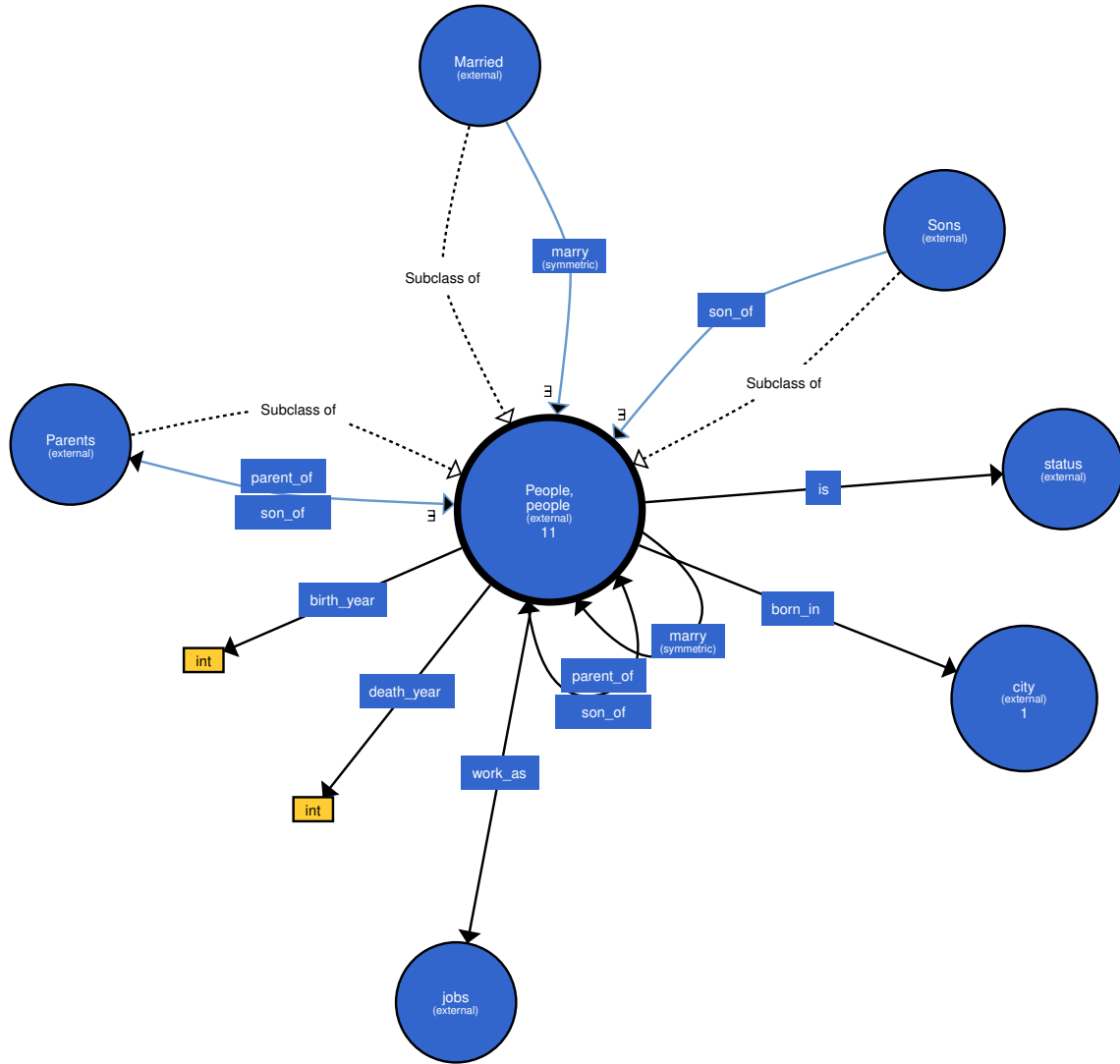
# Ristrutturazione e assemblaggio

```
1  let newMaterials : [Class*] =
2      select x
3      from x in materials
4      where (not contains x artificialMats);;
5
6  let newFashionIndividual : [Individual*] =
7      select x
8      from x in [fashion]/Individual
9      where (not isArtificial x materials artificialMats fashion);;
10
11 let socPeople :? Class =
12     match socPeople with
13     <owl:Class rdf:about=str> [ (attr::ClassAtt)* ] →
14     <owl:Class rdf:about=str> (attr @ [ <owl:equivalentClass
15         rdf:resource="http://www.people#People"> [] ]) ;;
16
17 let newPeople : [Thing*] = [people]/(Thing \ Ont);;
```

# Ristrutturazione e assemblaggio

```
1  let newThing : [Thing*] =  
2    [ newOnt ] @ newSociety @ [ socPeople ] @ newFashionIndividual @ newMaterials  
   @ newPeople;;  
3  
4  let newOntology :? Ontology =  
5    <rdf:RDF xml:base="http://www.semanticweb.org/society_merged"> newThing ;;  
6  
7  dump_to_file_utf8 "society_merged.rdf" (print_xml_utf8 newOntology);;
```

# Risultato



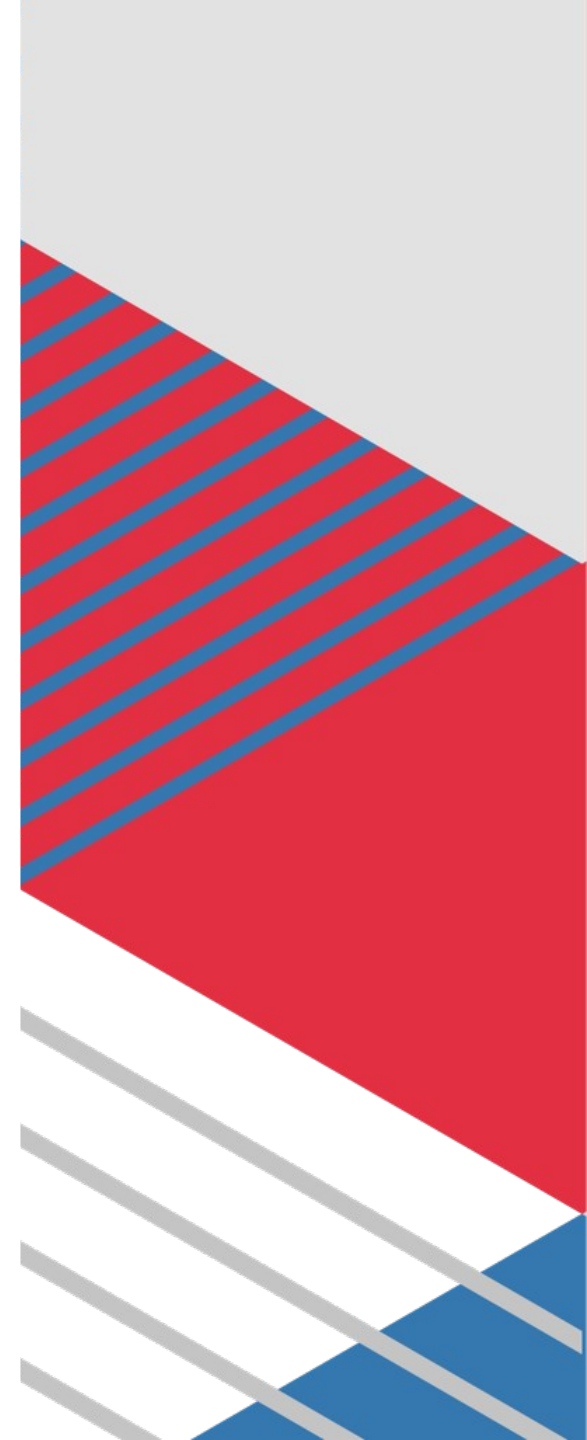
- Classe People unione delle due definizioni
- Possibilità di definire relazioni tra persone e vestiario
- Selezionati solo vestiti realizzati con materiali presenti nel XII secolo

# Conclusioni

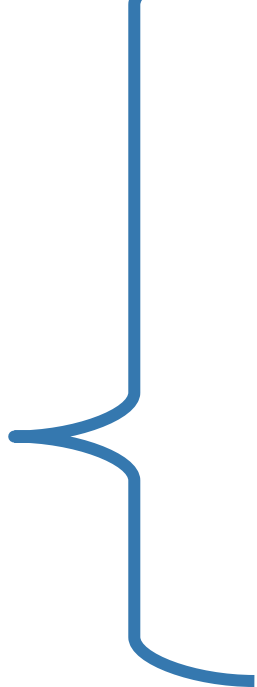
Vantaggi e svantaggi dell'uso di Cduce e possibili sviluppi futuri

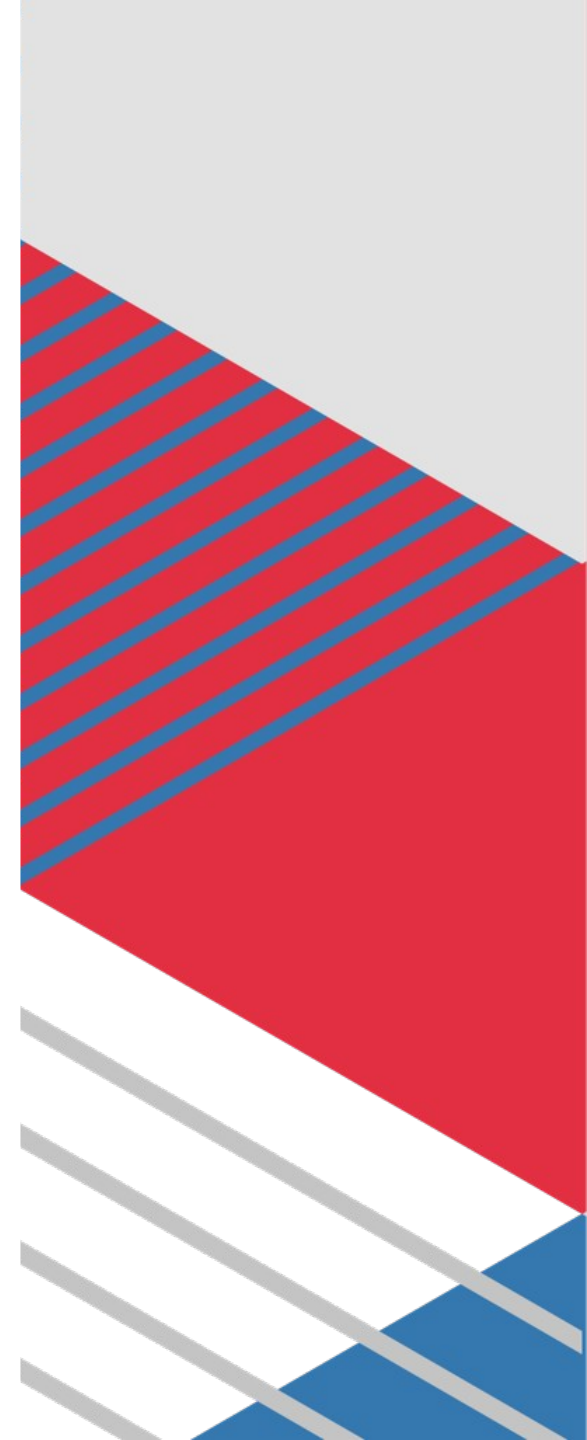
# Svantaggi

- Progetto di interesse accademico
- Piccole dimensioni
- Aggiornamenti non costanti



# Svantaggi

- Progetto di interesse accademico
  - Piccole dimensioni
  - Aggiornamenti non costanti
- 
- Ottenere C<sub>D</sub>uce
  - Compatibilità
  - Formattazione dell'output
  - Reperire informazioni



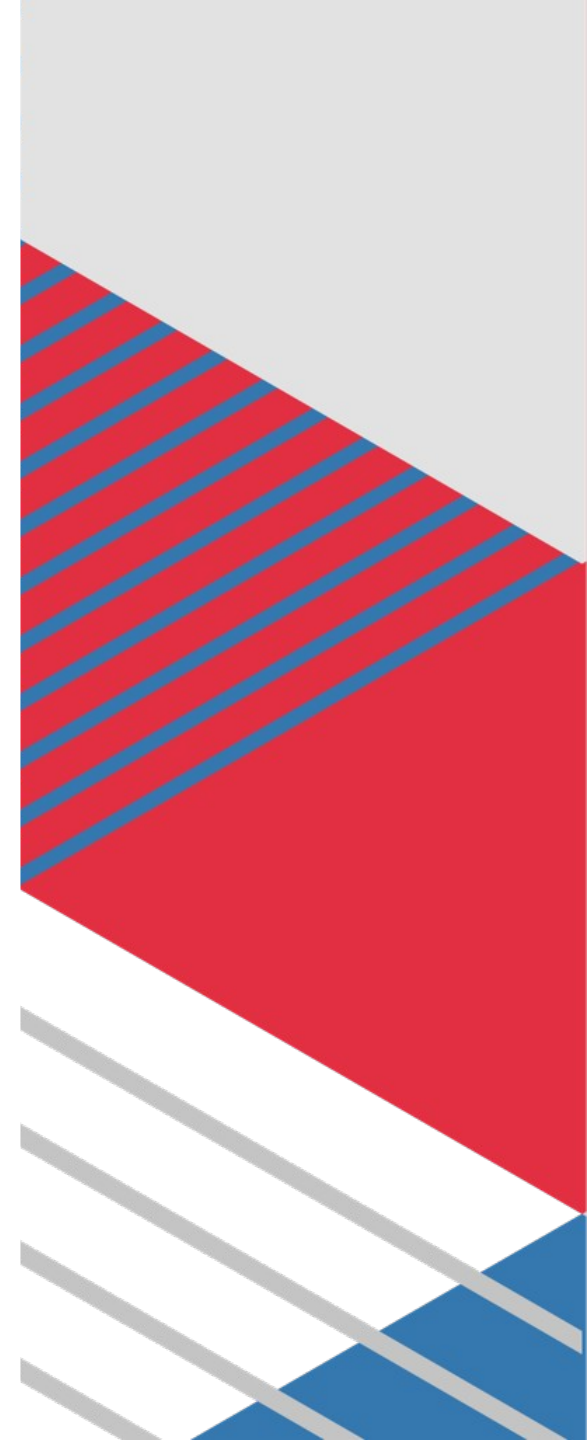
# Svantaggi

- Progetto di interesse accademico
- Piccole dimensioni
- Aggiornamenti non costanti

- Ottenere C<sub>D</sub>uce
- Compatibilità
- Formattazione dell'output
- Reperire informazioni



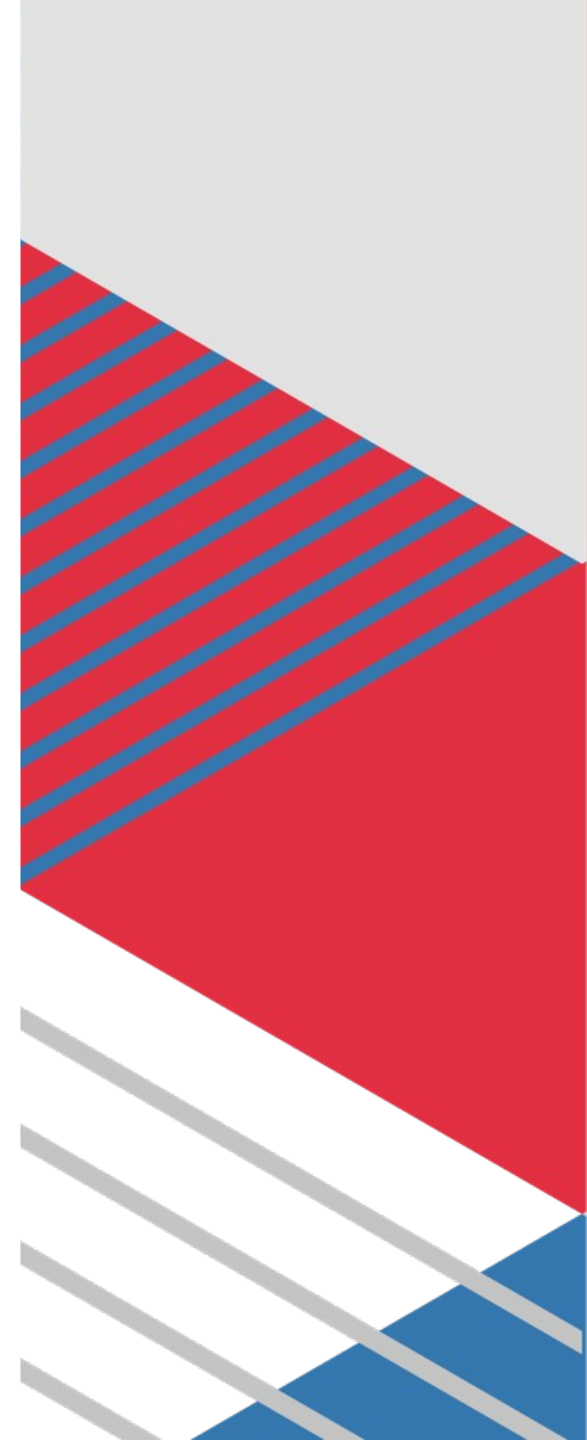
- Difficoltà nella scrittura del codice





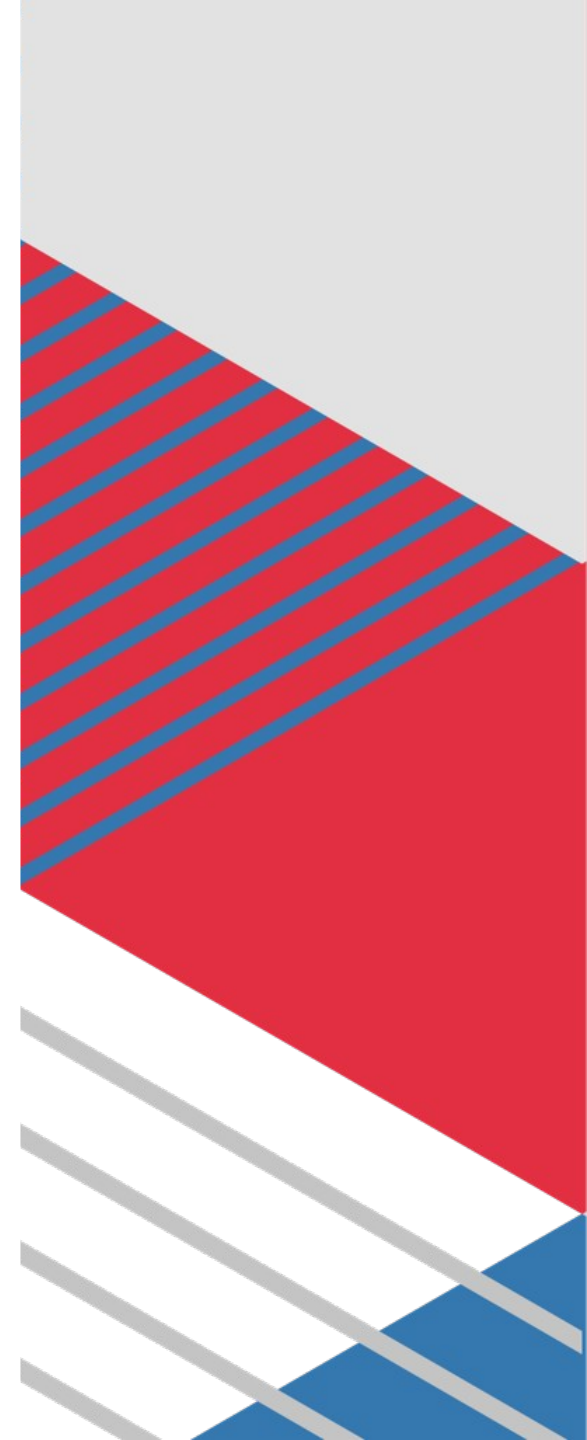
# Vantaggi

- Codice sintetico e leggibile
- Sistema di tipi
- Funzioni di ordine superiore
- Trasformazioni potenti e corrette



# Sviluppi futuri

- Confronto con esperti del settore
- Creazione di strumenti generali
- Sviluppo di interfaccia grafica



# Grazie per l'attenzione

Domande?



# UNIVERSITÀ DI TORINO