Ontologias (III)

Linguagens de Anotação de Documentos

Ontologias

- Uma ontologia modela os conceitos e as relações de um domínio
- Juntamente com os indivíduos, formam uma base de conhecimento
- O RDF permite codificar informação sobre recursos através de triplos
- O RDFS permite criar ontologias básicas
- Com o OWL podemos criar ontologias avançadas

Recursos

- Os triplos relacionam um sujeito com um objeto através de uma predicado
- Estes serão partilhados e processados por outros serviços
- Mas como garantimos todos os serviços, independentes entre si, se referem aos mesmos recursos?

Recursos Unicamente Identificados

- Atualmente a informação já não se encontra em “silos”
- Os dados não só estão partilhados, como estão ligados entre si
- Já não se pode basear em nomes ou atributos para identificar recursos
- Tem que se usar identificadores únicos, para garantir que todas as partes se referem exatamente ao mesmo recurso
- Na web são naturalmente usados endereços URI, do domínio das partes
- Não quer dizer que exista realmente algo no endereço

Recursos Unicamente Identificados

```
http://di.uminho.pt/lad#Aluno1 rdf:type http://di.uminho.pt/lad#Aluno .
```

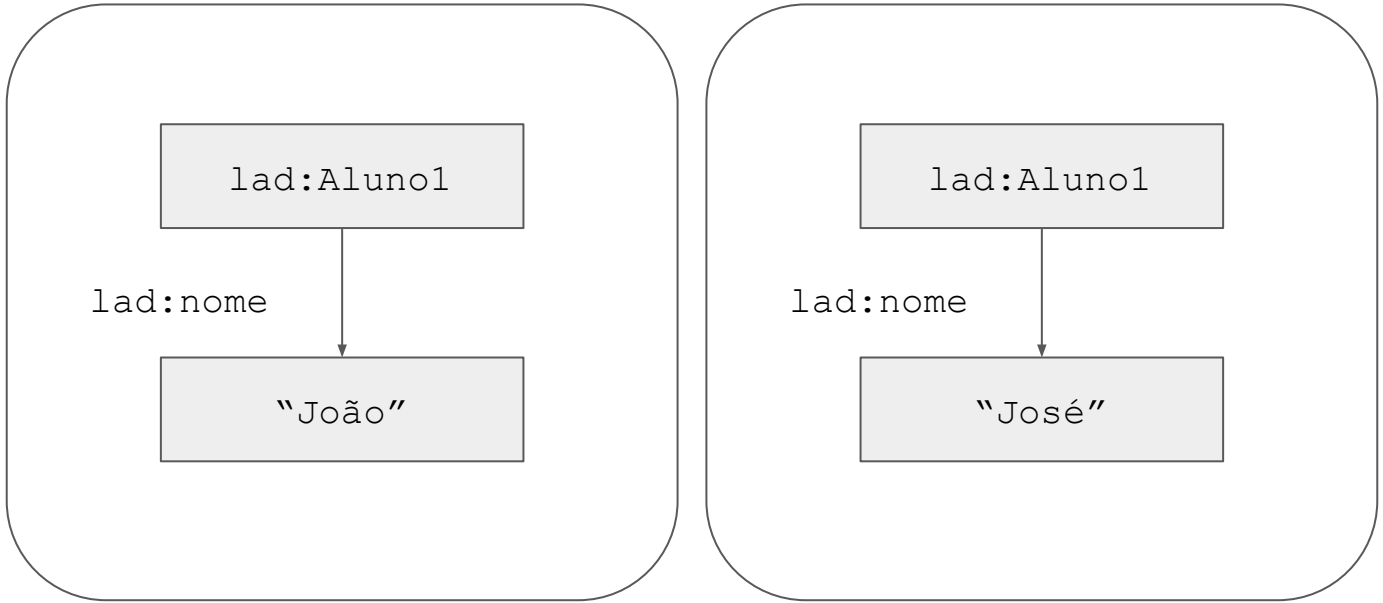
- `rdf` é apenas o *namespace* `http://www.w3.org/1999/02/22-rdf-syntax-ns#` abreviado
- Se definirmos o *namespace* `lad` podemos abreviar para

```
lad:Aluno1 rdf:type lad:Aluno .
```

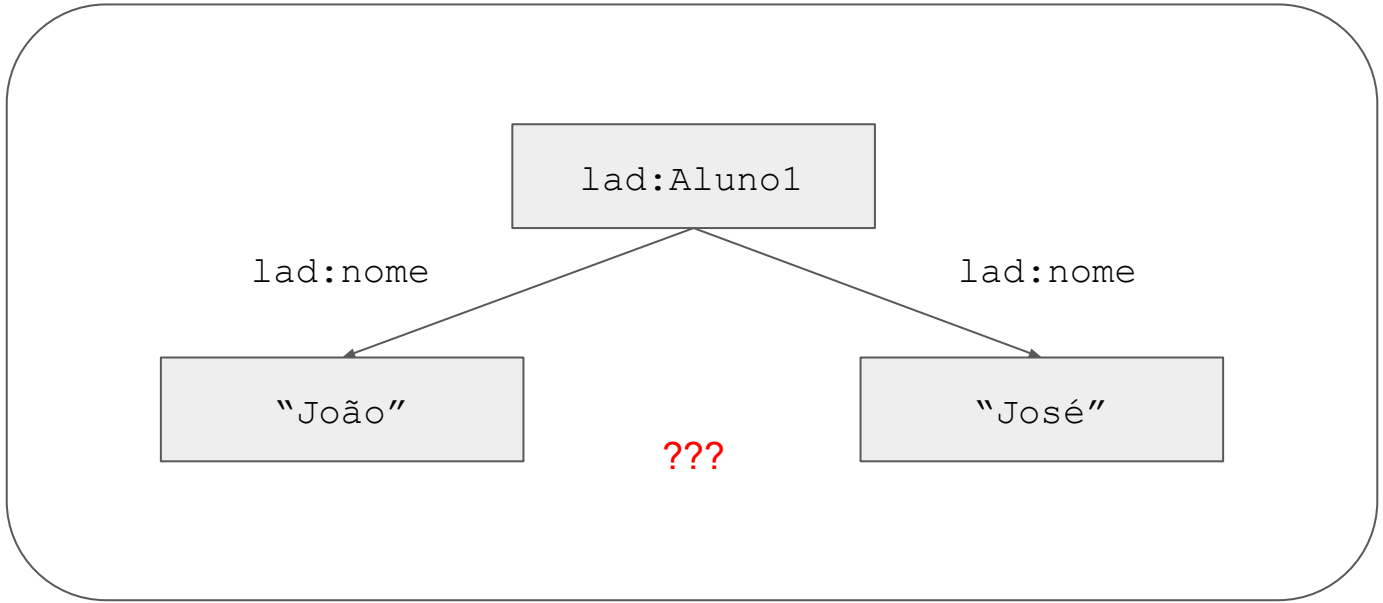
Valores Literais

- Os sujeitos e as propriedades dos triplos são sempre recursos unicamente identificados
- Os objetos, no entanto, podem ser recursos ou valores *literais*
- Literais são por exemplo texto livre ou números
- Literais são considerados o mesmo recurso se o seu conteúdo (e tipo) forem exatamente o mesmo
- Os literais não são ligados a outros elementos no grafo da ontologia, ao contrário dos recursos identificados unicamente

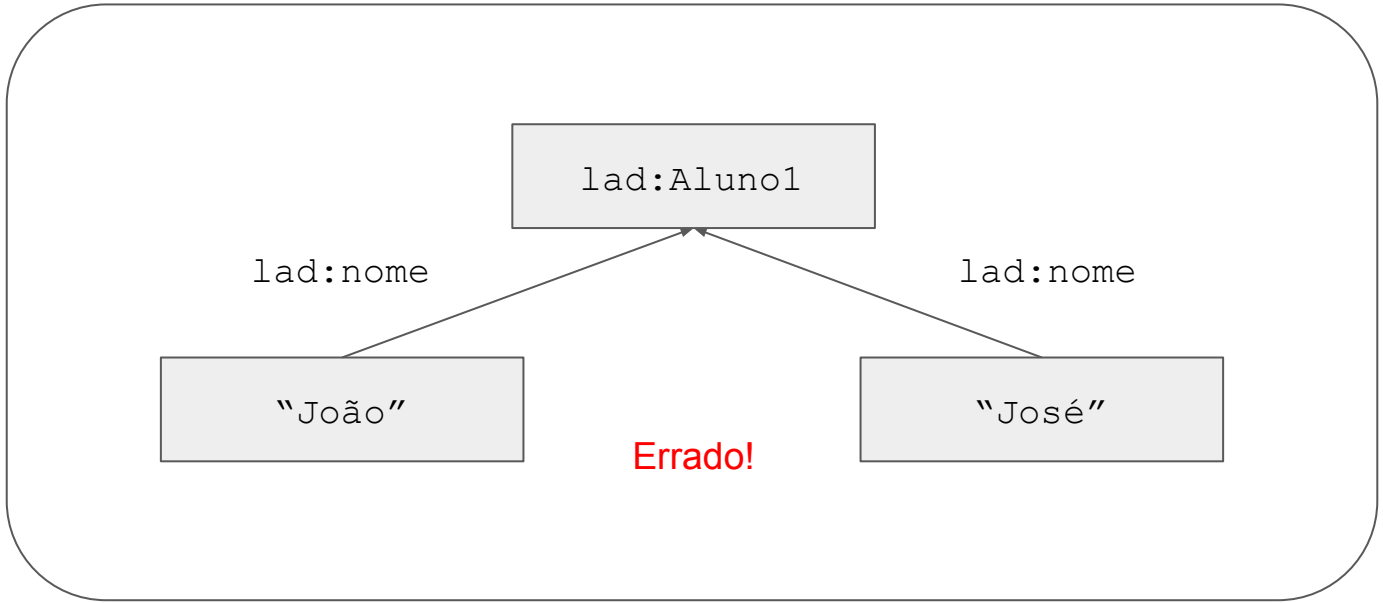
Recursos em RDF



Recursos em RDF



Recursos em RDF



Recursos em OWL DL

- Focamo-nos no OWL DL, mais bem comportado que o OWL Full
- Todos os indivíduos são do tipo `owl:Thing`
- Todas as outras classes (definidas por nós) são subclasses de `owl:Thing`
- Todos os valores literais são do tipo `rdfs:Literal`
- Todos os os tipos específicos de literais (números, etc.) são subclasses de `rdfs:Literal`

Exemplos

```
lad:Person rdfs:subClassof owl:Thing .
lad:Docente rdfs:subClassof lad:Person .
lad:Aluno rdfs:subClassof lad:Person .

lad:Aluno1 rdf:type owl:Thing .
lad:Aluno1 rdf:type lad:Person .
lad:Aluno1 rdf:type lad:Aluno .
```

Propriedades em OWL DL

- Em OWL DL, estes propriedades para indivíduos e para valores literais pertencem a duas classes diferentes de propriedades
- *Object Properties*: propriedades cujo objeto são indivíduos
- *Datatype Properties*: propriedades cujo objeto são valores literais
- Nestas últimas o tipo pode simplesmente ser `rdfs:literal` (texto arbitrário) ou subclasses mais concretas
  - `xsd:integer` (números inteiros), `xsd:date` (datas), ...

Propriedades em OWL DL

- Tal como as classes são todas subclasses the `owl:Thing`
- Todas as *datatype properties* são subclasse de `owl:topObjectProperty`
- Todas as *object properties* sao subclasse de `owl:topDataProperty`

Exemplos

```
lad:pertence rdfs:subPropertyOf owl:topObjectProperty  .
lad:nome rdfs:subPropertyOf owl:topDataProperty  .

lad:Aluno1 owl:topObjectProperty lad:Grupo1 .
lad:Aluno1 lad:pertence lad:Grupo1 .

lad:Aluno1 owl:topDataProperty “João” .
lad:Aluno1 lad:nome “João” .
```

Inferências

- Com ontologias os recursos passam a ter semântica associada
- Isto permite interpretar os dados formalmente, com regras matemáticas
- Isto permite às máquinas fazer inferências lógicas a partir dos dados conhecidos
- Inferências construídas a partir de regras "**Se** *… alguns triplos existem …* **então** *… outros triplos existem*"

Mundo Aberto

- Quando se assume um *mundo fechado*, toda a informação é conhecida: o que não está declarado é considerado falso
- O OWL assume um *mundo aberto*, informação desconhecida não quer dizer que não seja verdadeira
- Os sistemas só podem tirar conclusões baseados na informação conhecida atualmente

Inferências RDF

- O RDF permite fazer inferências apenas a partir dos triplos
- Se `a rdf:type b`
- Então `b rdf:type rdf:Class` (`b` é uma classe)

- Se `a p b`
- Então `p rdf:type rdf:Property` (`p` é uma propriedade)

Inferências RDFS

- O RDFS permite fazer inferências básicas
- A partir da hierarquia de classes
  - subclasses herdam todas as propriedades das super-classes
- A partir do domínio/contra-domínio das propriedades
  - determinam o tipo dos indivíduos relacionados por elas
- Ferramentas para a criação de ontologias fazem estas inferências automaticamente (mas não a versão *web* simplista que usamos!)

Inferências RDFS: Herança

- Hierarquia e herança
- Se
  - `a rdfs:subClassOf b`
  - `x rdf:type a`
- Então
  - `x rdf:type b`

- Transitiva! Se
  - `b rdfs:subClassOf c`
- Então
  - `a rdfs:subClassOf c`
  - `x rdf:type c`

Inferências RDFS: Domínios

- Domínios e contra-domínios
- Se
  - `p rdfs:domain a`
  - `p rdfs:range b`
  - `x p y`
- Então
  - `x rdf:type a`
  - `y rdf:type b`

- E se definirmos vários domínios/contra-domínios?

Inferências RDFS: Herança de propriedades

- Subpropriedades funcionam tal como as subclasses
- Se
  - `p rdfs:subPropertyOf q`
  - `x p y`
- Então
  - `x q y`

Inferências RDFS: Exemplos

```
lad:Person rdfs:subClassof owl:Thing .
lad:Docente rdfs:subClassof lad:Person .
lad:Aluno rdfs:subClassof lad:Person .
lad:pertence rdfs:domain lad:Person .
lad:pertence rdfs:range lad:Grupo .
lad:Aluno1 lad:pertence lad:Grupo1 .
```

Inferências RDFS: Exemplos

```
lad:Person rdfs:subClassof owl:Thing .
lad:Docente rdfs:subClassof lad:Person .
lad:Aluno rdfs:subClassof lad:Person .
lad:pertence rdfs:domain lad:Person .
lad:pertence rdfs:range lad:Grupo .
lad:Aluno1 lad:pertence lad:Grupo1 .

lad:Aluno1 rdf:type lad:Aluno .
lad:Aluno1 rdf:type lad:Person .
lad:Grupo1 rdf:type lad:Grupo .
```

Inferências OWL DL

- Já vimos que o OWL é construído sobre o RDFS, adicionando mais propriedades predefinidas
- Estas permitem construir ontologias mais ricas, e consequentemente, fazer inferências mais avançadas
- Estamos interessados apenas no conjunto bem comportado, OWL DL

Inferências OWL DL: Inversos

- As propriedades de uma ontologia são direcionadas (“aluno pertence a grupo”, e não “grupo pertence a aluno”)
- Por vezes é útil exprimir relações em ambas as direções (“grupo contém alunos”, o inverso de “pertence”)
- Como forçar que os elementos sejam os mesmos mas invertidos?

Inferências OWL DL: Inversos

- O OWL tem a propriedade `owl:inverseOf`
- Infere o inverso de todos os triplos sobre a propriedade
- Se
  - `lad:pertence owl:inverseOf lad:contem`
  - `lad:Aluno1 lad:pertence lad:Grupo1`
- Então
  - `lad:contem owl:inverseOf lad:pertence`
  - `lad:Grupo1 lad:contem lad:Aluno1`

Inferências OWL DL: Simetria

- Outras vezes simplesmente não queremos que as propriedades sejam direcionadas (se “aluno1 é colega de aluno2”, “aluno2 é colega de aluno1”)
- Como forçar uma propriedade a ter o mesmo significado em ambas as direções?

Inferências OWL DL: Simetria

- O OWL tem a class `owl:SymmetricProperty`
- Se dois indivíduos estão relacionados, o seu inverso também está
- Se
  - `lad:colega owl:subPropertyOf owl:SymmetricProperty`
  - `lad:Aluno1 lad:colega lad:Aluno2`
- Então
  - `lad:Aluno2 lad:colega lad:Aluno1`

Inferências OWL DL: Transitividade

- Também podemos forçar transitividade com a class `owl:TransitiveProperty`
- Dois indivíduos ficam relacionados através de terceiros
- Se
  - `lad:colega owl:subPropertyOf owl:TransitiveProperty`
  - `lad:Aluno1 lad:colega lad:Aluno2`
  - `lad:Aluno2 lad:colega lad:Aluno3`
- Então
  - `lad:Aluno1 lad:colega lad:Aluno3`

Inferências OWL DL: Funcionais

- Outras propriedades úteis limitam a cardinalidade das nossas propriedades
- Por exemplo, em propriedades funcionais, um indivíduo só pode estar relacionado com um outro
- E em inversamente funcionais, o oposto, dois sujeitos diferenes não podem apontar para o mesmo objeto

Inferências OWL DL: Funcionais

- Propriedades funcionais `owl:FunctionalProperty` (e inversamente com `owl:InverseFunctionalProperty`)
- Um indivíduo só pode estar relacionado com um outro pela propriedade
- Se:
  - `lad:nota owl:subpropretyOf owl:FunctionalProperty`
  - `lad:Aluno1 lad:nota “10”`
  - `lad:Aluno1 lad:nota “20”`
- Então: Erro!

Inferências OWL DL: Funcionais

- Então e se
  - `lad:pertence owl:subpropretyOf owl:FunctionalProperty`
  - `lad:Aluno1 lad:pertence lad:Grupo1`
  - `lad:Aluno1 lad:pertence lad:Grupo2`
- De facto, em OWL não se assume que os *nomes são únicos*
- Ou seja, a não ser que seja dito algo, não se sabe se dois indivíduos são ou não a mesma coisa
- A ontologia em cima à partida não está errada

Unicidade de Nomes

- Como lidar com a não unicidade de nomes?
- Igualdade entre indivíduos forçada com `owl:sameAs`
- Se
  - `lad:Aluno1 lad:pertence lad:Grupo1`
  - `lad:Aluno2 lad:nota “10”`
  - `lad:Aluno1 owl:sameAs lad:Aluno2`
- Então
  - `lad:Aluno1 lad:nota “10”`
  - `lad:Aluno2 lad:pertence lad:Grupo1`

Inferências OWL DL: Igualdade

- De volta às propriedades funcionais
- Se
  - `lad:pertence owl:subpropretyOf owl:FunctionalProperty`
  - `lad:Aluno1 lad:pertence lad:Grupo1`
  - `lad:Aluno1 lad:pertence lad:Grupo2`
- Então
  - `lad:Grupo1 owl:sameAs lad:Grupo2`

Inferências OWL DL: Desigualdade

- Desigualdade entre indivíduos forçada com `owl:differentFrom`
- Se
  - `lad:pertence owl:subpropretyOf owl:FunctionalProperty`
  - `lad:Aluno1 lad:pertence lad:Grupo1`
  - `lad:Aluno1 lad:pertence lad:Grupo2`
  - `lad:Grupo1 owl:differentFrom lad:Grupo2`
- Então: Erro!

Inferências OWL DL: Classes disjuntas

- Classes podem ser facilmente forçadas a não ter elementos em comum com `owl:disjointWith`
- Se
  - `lad:Aluno owl:disjointWith lad:Docente`
  - `lad:Aluno1 rdf:type lad:Aluno`
  - `lad:Aluno1 rdf:type lad:Docente`
- Então: Erro!

Inferências OWL DL: Classes equivalentes

- Para forçar classes a terem exatamente os mesmos indivíduos podemos definir uma equivalência entre elas com `owl:equivalentClass`
- Se
  - `lad:Estudante owl:equivalentClass lad:Aluno`
  - `lad:Aluno1 rdf:type lad:Aluno`
- Então:
  - `lad:Aluno1 rdf:type lad:Estudante`
- O mesmo processo para propriedades com `owl:equivalentProperty`

Inferências OWL DL: Classes equivalentes

- Mais interessante quando se constrói uma classe combinando outras
- Podemos declarar uma classe como equivalente à
  - Interseção ou reunião de duas outras
  - Uma outra classe com um filtro que remove alguns indíviduos
  - ...

Consultas

- Além de inferências, outra grande vantagem é a possibilidade de fazer consultas avançadas
- Para ontologias baseadas em RDF, isto é feito com a linguagem SPARQL
- Exemplos (básicos) em texto corrido (não na sintaxe real)
  - `lad:Aluno1 lad:pertence ?grupo`
  - `lad:Aluno1 lad:nota ?nota`
  - `?aluno lad:pertence lad:Grupo1`

Consultas

- Podem ser bastante complexas
- Suportadas pelos editores de ontologias (mas não pela versão *web* simplista que estamos a usar!)

Take-home Lesson

- A vantagem principal de definir ontologias em OWL é a semântica que é atribuída aos dados que partilhamos
- Isto permite fazer inferências e consultas avançadas
- Como a semântica está formalizada, é fácil de partilhar e combinar dados na *web* semântica usando ontologias
- Os dados serão interpretados de maneira consistente por todas as partes interessadas