Existential restrictions

Henriette Harmse

Prerequisites

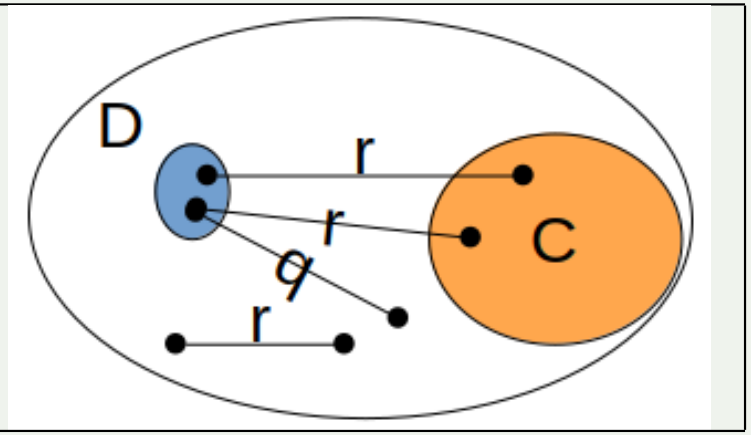
Before doing this tutorial you need to have the following knowledge:

1. Building blocks of OWL and Description Logics.
2. `SubClassOf` vs `EquivalentTo`

Qualified existential restrictions

Syntax

OWL	DL	

`ObjectProperty: r`
`Class: D`
`EquivalentTo:`
`r some C`
`Class: C`
	$D \equiv \exists r.C$	

Semantics

- $(\exists r.C)^{\mathcal{I}} = \{x \in \triangle^{\mathcal{I}} | \text{there is an } y \in \triangle^{\mathcal{I}} \text{ such that } (x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
- `r some C` $(\exists r.C)$ is the set of individuals such that for each individual x there is at least 1 individual y of type C that is linked to x via the object property (role) r.

Qualified existential restrictions

Example using EquivalentTo

```
ObjectProperty:  owns
Class:  PetOwner
  EquivalentTo:  owns some Pet
Class:  Pet
```

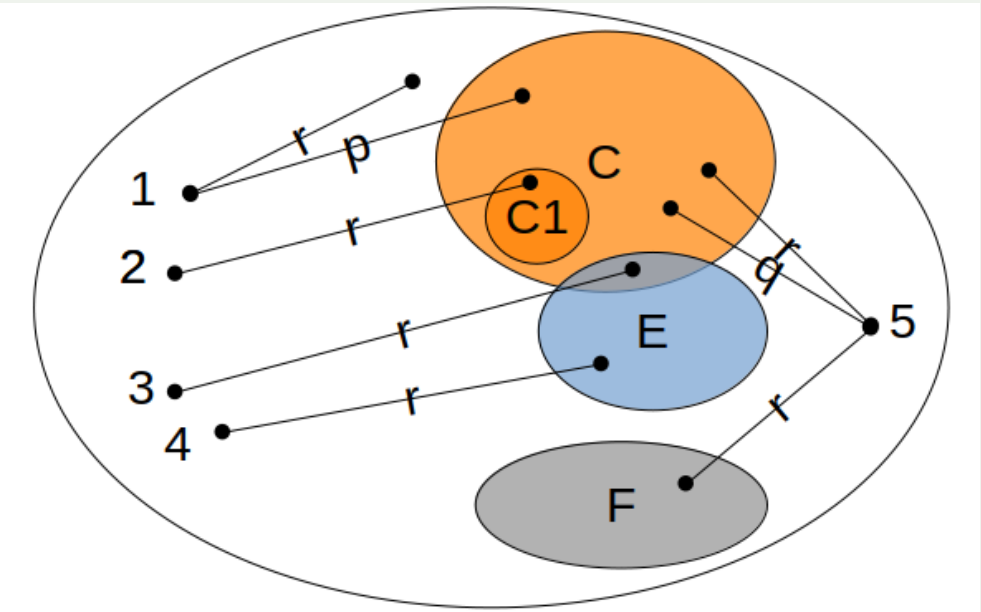
Example using SubClassOf

```
ObjectProperty:  owns
Class:  DogOwner
  SubClassOf:  owns some Pet
Class:  Pet
```

Qualified existential restrictions

Examples

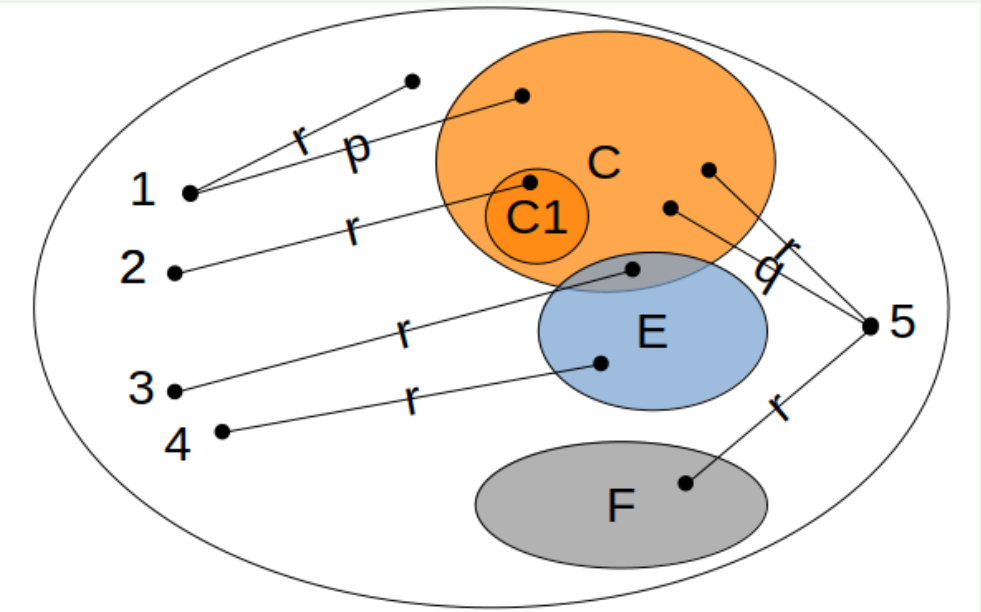
Which of these individuals will be in `r some C` and therefore as well in `D`?



Qualified existential restrictions

Examples

Which of these individuals will be in `r some C` and therefore as well in `D`?



Answer

Individuals 2, 3, 5

Variations on existential restrictions

Syntax

Name	OWL	DL

Unqualified existential restrictions	`ObjectProperty: owns`
	`Class: Owner`
	`  EquivalentTo:`
	`  owns some owl:Thing`
		$Owner \equiv \exists owns.\top$ or
		$Owner \equiv \exists owns$

Value restrictions	`ObjectProperty: citizenOf`
	`Class: UKCitizen`
	`  EquivalentTo:`
	`  citizenOf hasValue UK`
	`Individual: UK`
		$UKCitizen \equiv \exists citizenOf.\{UK\}$

Existential restriction on data property	`DataProperty: name`
	`Class: Person`
	`  SubClassOf:`
	`  name some xsd:string`
		$Person \sqsubseteq \exists name.\texttt{xsd:string}$

Using existential restrictions with SubClassOf vs EquivalentTo

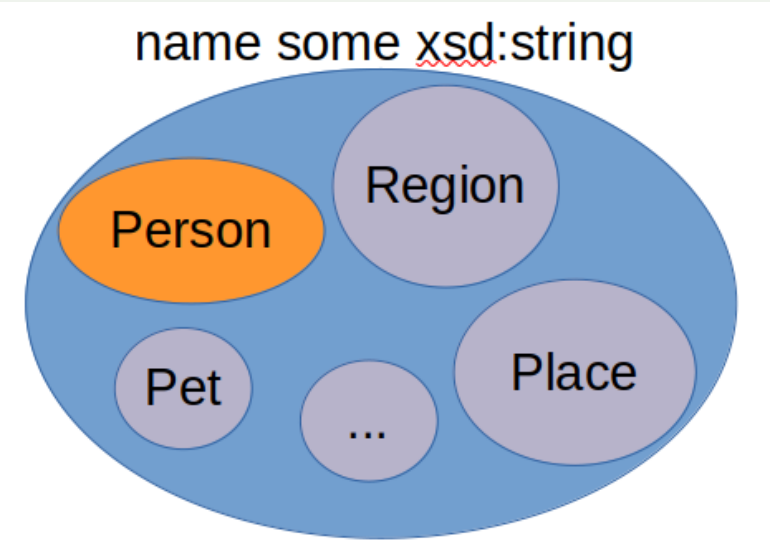
A Person have 1 or more name

Why did we use `SubClassOf` rather than `EquivalentTo`?

Using existential restrictions with SubClassOf vs EquivalentTo

A Person have 1 or more name

Why did we use `SubClassOf` rather than `EquivalentTo`?



Using existential restrictions with SubClassOf vs EquivalentTo

A DogOwner is a Person that owns a Dog

```
ObjectProperty:  owns
Class:  Dog
Class:  Person
Class:  DogOwner
  EquivalentTo:
    Person and owns some Dog
```