SubClass,Equivalence,Existential Restrictions

Henriette Harmse

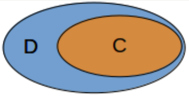
Building Blocks of DLs and OWL

OWL	**DL**	**Semantics**	**Example**
instance or individual	instance or individual	A member of a set.	A person called `Mary` or a dog called `Fido`.
class	concept	A set of individuals.	The `Person` class (concept) consisting of persons or the `Dog` class (concept) consisting of dogs.
object property	role	A set of pairs of individuals.	The `owns` object property (role) can link a pet and its owner: `Mary owns Fido` (in DL $owns(Mary, Fido)$).
data property	concrete role	A set of pairs where each pair consists of an individual linked to a data value.	The data property (concrete role) `hasAge` can link a number representing an age to an individual: `hasAge(Mary, 10) (in DL` $hasAge(Mary, 10)$)

The semantics of `SubClassOf`

Syntax

OWL	DL	Semantics

`Class: C`
 `SubClassOf: D`
 `Class: D`
	$C \sqsubseteq D$	

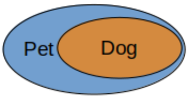
Semantics

The set C is a subset of the set D. This means every individual of C is necessarily an individual of D, but not every individual of D is necessarily an individual of C.

The semantics of `SubClassOf`

Example

OWL	**DL**	**Semantics**

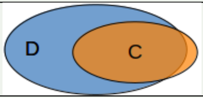
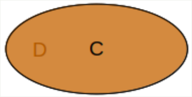
`Class: Dog`
`SubClassOf: Pet`
`Class: Pet`
	$Dog \sqsubseteq Pet$	

The semantics of SubClassOf

Guidance - When to use

`SubClassOf` is used when you want to define a hierarchy from the most general to the most specific. I.e., it is typically what you see in taxonomies.

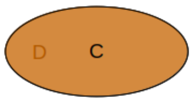
Guidance - When **not** to use

When not to use	**Venn diagram**
When there is an individual of C that is not an individual of D.	
When every individual of D is also an individual of C and every individual of C is also and individual of D, then prefer using `EquivalentTo`.	

The semantics of EquivalentTo

Syntax

OWL	DL	Semantics

`Class: C`
`  EquivalentTo: D`
`Class: D`
which can be seen as shorthand for:
`Class: C`
`  SubClassOf: D`
`Class: D`
`  SubClassOf: C`
	$C \equiv D$
	which can be seen as shorthand for
	$C \sqsubseteq D$
	$D \sqsubseteq C$
		

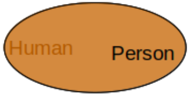
Semantics

Every individual of C is an individual of D, **and** every individual of D is an individual of C.

The semantics of `EquivalentTo`

Example

OWL	DL	Semantics

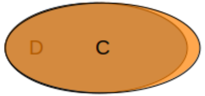
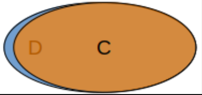
`Class: Person`
`EquivalentTo: Human`
`Class: Human`
	$Person \sqsubseteq Human$	

The semantics of `EquivalentTo`

Guidance - when to use

`EquivalentTo` is used for definitions. That is when you want to state the necessary and sufficient conditions for a concept.

Guidance - When **not** to use

When not to use	Venn diagram
When there is an individual of C that is not in D.	
When there is an individual of D that is not in C.	

In summary - Using `EquivalentTo` versus `SubClassOf`

`SubClassOf`

`SubClassOf` is used when you want to define a hierarchy from the most general to the most specific. I.e., it is typically what you see in taxonomies

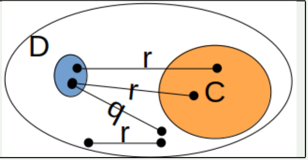
`EquivalentTo`

`EquivalentTo` is used for definitions. That is when you want to state the necessary and sufficient conditions for a concept.

Qualified existential restrictions

Syntax

OWL	DL	

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

Semantics

- $(\exists r.C)^{\mathcal{I}} = \{x \in \triangle^{\mathcal{I}} \mid \text{there is an } y \in \triangle^{\mathcal{I}} \text{ such that } (x, y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
- `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.
- C is called the **filler** of r, or more succinctly the r-filler.

Qualified existential restrictions

Example using `EquivalentTo`

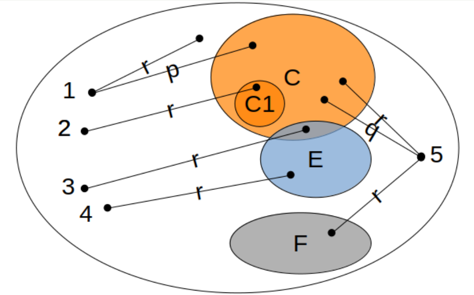
```
ObjectProperty: owns
Class: PetOwner
  EquivalentTo: owns some Pet
Class: Pet
```

Example using `SubClassOf`

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

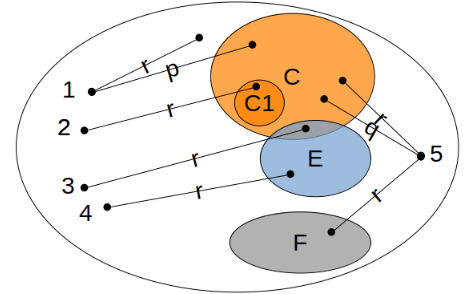
Examples

Assuming we have `D EquivalentTo: r some C`, which of these individuals will be in `r some C` and therefore as well in `D`?



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`
		$Person \sqsubseteq \exists name.\mathtt{xsd:string}$

Using existential restrictions with SubClassOf vs EquivalentTo

A Person have 1 or more name

```
DataProperty: name
Class: Person
  SubClassOf:
    name some xsd:string
```

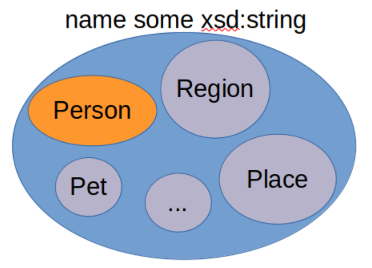
Why did we use `SubClassOf` rather than `EquivalentTo`?

Using existential restrictions with SubClassOf vs EquivalentTo

A Person have 1 or more name

```
DataProperty: name
Class: Person
  SubClassOf:
    name some xsd:string
```

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
```