INF3580 – Semantic Technologies – Spring 2011

Lecture 11: OWL 2

Martin G. Skjæveland

5th April 2011

Department of Informatics

University of Oslo

Outline

$\mathcal{ALC}$ Semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each atomic concept A, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each role R, and $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual a.

Interpretation of concept descriptions

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b \text{, if } \langle a, b\rangle \in R^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \text{there is a } b \text{ where } \langle a, b\rangle \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}
\end{array}
$$

Interpretation of Axioms

- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $\mathcal{I} \models C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I} \models R(a, b)$ if $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$.

TBox, ABox

- The TBox
  - is for *terminological knowledge*,
  - is independent of any actual instance data, and
  - for $\mathcal{ALC}$, it is a set of $\sqsubseteq$ axioms and $\equiv$ axioms.
  - Example TBox axioms:
    - $\mathit{TwoCV} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$
    - $\mathit{FrontDrivenCar} \equiv \mathit{Car} \sqcap \forall \mathit{driveAxle}.\mathit{FrontAxle}$.
- The ABox
  - is for *assertional knowledge*,
  - contains facts about individuals a, b, c,
  - a set of concept membership assertions $C(a)$,
  - and role assertions $R(b, c)$.
  - Example ABox axioms:
    - $\mathit{driveAxle}(\mathit{myCar}, \mathit{axle})$
    - $(\mathit{FrontAxle} \sqcup \mathit{RearAxle})(\mathit{axle})$.

Modelling patterns

So, what can we say with $\mathcal{ALC}$?

- ✓ Every person has a mother.
- ✓ Penguins eats only fish. Horses eats only chocolate.
- ✗ Every nuclear family has two parents, at least two children and a dog.
- ✓ No smoker is a non-smoker (and vice versa).
- ✗ Everybody loves Mary.
- ✗ Adam is not Eve (and vice versa).
- ✓ Everything is black or white.
- ✓ There is no such thing as a free lunch.
- ✗ Brothers of fathers are uncles.
- ✗ My friend's friends are also my friends.
- ✗ If Homer is married to Marge, then Marge is married to Homer.
- ✗ If Homer is a parent of Bart, then Bart is a child of Homer.

Today we'll learn how to say more.

Outline

$\mathcal{SHOIN}(\mathcal{D})$ and OWL 2

- OWL 2 is based on the DL $\mathcal{SHOIN}(\mathcal{D})$:
  - $\mathcal{S}$ for $\mathcal{ALC}$[1] plus role transitivity,
  - $\mathcal{H}$ for roles hierarchies,
  - $\mathcal{O}$ for closed classes,
  - $\mathcal{I}$ for inverse roles,
  - $\mathcal{N}$ for cardinality restrictions, and
  - $\mathcal{D}$ for datatypes.
- So, today we'll see:
  - new concept and role builders,
  - new TBox axioms,
  - new ABox axioms,
  - new RBox and axioms, and
  - datatypes!

Focus!

[1] Attributive Concept Language with Complements

OWL 2 and its profiles

- OWL 2 has various *profiles* that correspond to different DLs.
- OWL 2 DL is the "normal" OWL 2 (sublanguage): "maximum" expressivity while keeping reasoning problems decidable—but still very expensive.
- (Other) profiles are tailored for specific ends, e.g.,
  - OWL 2 QL:
    - Specifically designed for efficient database integration.
  - OWL 2 EL:
    - A lightweight language with polynomial time reasoning.
  - OWL 2 RL:
    - Designed for compatibility with rule-based inference tools.
- OWL Full: Anything goes: classes, relations, individuals, ... like in RDFS, are not kept apart. Highly expressive, not decidable. But we want OWL's reasoning capabilities, so stay away if you can—and you almost always can.

OWL 2 Validator: `http://owl.cs.manchester.ac.uk/validator/`

Outline

Individual identity

- New ABox axioms.
- Express equality and non-equality between individuals.
- Syntax:
  - DL: $a = b$, $a \neq b$;
  - RDF/OWL: `:a owl:sameAs :b`, `:a owl:differentFrom :b`,
  - Manchester: `SameAs`, `DifferentFrom`.
- Semantics:
  - $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$
  - $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$
- Examples:
  - `sim:Bart owl:sameAs dbpedia:Bart_Simpson`,
  - `sim:Bart owl:differentFrom sim:Homer`.
- Remember:
  - Non unique name assumption (NUNA) in Sem. Web,
  - must use $=$ and $\neq$ to get expected results.

Creating concepts using individuals

- New concept builder.
- Create (anonymous) concepts by explicitly giving all members.
- Called *closed classes* in OWL.
- Syntax:
  - DL: $\{a, b, \ldots\}$
  - RDF/OWL: `oneOf` + `rdf:List`++
  - Manchester: `{a, b, ...}`
- Example:
  - $\mathit{SimpsonFamilyMember} \equiv \{\mathit{Homer}, \mathit{Marge}, \mathit{Bart}, \mathit{Lisa}, \mathit{Maggie}\}$
- Note:
  - The individuals does not necessarily represent different objects,
  - we still need $=$ and $\neq$ to say that members are the same/different.
  - "Closed classes of data values" are *datatypes*.

Axioms involving individuals: Closed classes

- Using closed classes we can exclude individuals from classes.
- Example: $\{NedFlanders\} \sqsubseteq \neg SimpsonFamilyMember$.
  - Ned Flanders is not a family member of the Simpsons.
  - (or better: $FlandersFamilyMember \equiv \{NedFlanders, \ldots\}$ and $FlandersFamilyMember \sqsubseteq \neg SimpsonFamilyMember$.)
- *Closed properties* does not exist in OWL
- (can be done with closed classes),
- but there is *negated role assignment* to exclude relationships from relations/roles (on next slide):

Axioms involving individuals: Negative Property Assertions

- New ABox axiom.
- Syntax:
  - DL: $\neg R(a, b)$,
  - RDF/OWL: `NegativePropertyAssertion`,
  - Manchester: `a not R b`.
- Semantics:
  - $\mathcal{I} \models \neg R(a, b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \notin R^{\mathcal{I}}$,
- Notes:
  - Works both for object properties and datatype properties.
- Examples:
  - `:Bart not :hasFather :NedFlanders`
  - `:Bart not :hasAge ‘‘2’’`

Outline

Recap of existential and universal restrictions

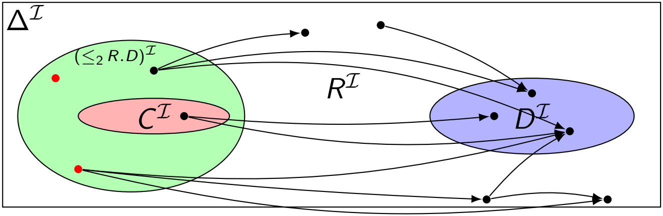
- Existential restrictions
  - have the form $\exists R.D$,
  - typically used to connect classes,
  - $C \sqsubseteq \exists R.D$: A C is R-related to (at least) *some* D:
    - Example: A person has a female parent: $Person \sqsubseteq \exists hasParent.Woman$.
  - Note that C-objects can be R-related to other things:
    - A person may have other parents who are not women—but there must be one who's a woman.
- Universal restrictions
  - have the form $\forall R.D$,
  - restrict the things a type of object can be connected to,
  - $C \sqsubseteq \forall R.C$: C is R-related to D's *only*:
    - Example: A horse eats only chocolate: $Horse \sqsubseteq \forall eats.Chocolate$.
  - Note that C-objects may not be R-related to anything at all:
    - A horse does not have to eat anything—but if it does it must be chocolate.

Cardinality restrictions

- New concept builder.
- Syntax:
  - DL: $\leq_n R.D$ and $\geq_n R.D$ (and $=_n R.D$).
  - RDF/OWL: `minCardinality`, `maxCardinality`, `cardinality`.
  - Manchester: `min`, `max`, `exactly`.
- Semantics:
  - $(\leq_n R.D)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \{b : \langle a, b\rangle \in R^\mathcal{I} \wedge b \in D^\mathcal{I}\}^\# \leq n\}$
  - $(\geq_n R.D)^\mathcal{I} = \{a \in \Delta^\mathcal{I} \mid \{b : \langle a, b\rangle \in R^\mathcal{I} \wedge b \in D^\mathcal{I}\}^\# \geq n\}$
- Restricts the number of relations a type of object can/must have.
- TBox axioms read:
  - $C \sqsubseteq \Box_n R.D$: "A C is R-related to n number of D's."
    - $\leq$: *at least*
    - $\geq$: *at most*
    - $=$: *exactly*

Example cardinality restriction

- $C \sqsubseteq \leq_2 R.D$



- Examples:
  - $Car \sqsubseteq \leq_2 driveAxle.\top$
    - "A car has at least two drive axles."
  - $RangeRover \sqsubseteq =_1 driveAxle.FrontAxle \sqcap =_1 driveAxle.RearAxle$
    - "A Range Rover has one front axle as drive axle and one rear axle as drive axle".

One more value restriction

- Existential and Universal restrictions are called *value restrictions*.
- Restrictions of the form $\forall R.D$, $\exists R.D$, $\leq_n R.D$, $\geq_n R.D$ are called *qualified* when D is not $\top$.
- Qualified: the restriction require R-relations to “hit” D’s.
- We can also qualify with a closed class.
- Syntax:
  - RDF/OWL: `hasValue`,
  - DL, Manchester: just use: $\{\ldots\}$.
- Example:
  - $Bieberette \equiv Girl \sqcap \exists loves.\{J.Bieber\}$

Self restriction

- New construct builder.
- Local reflexivity restriction. Restricts to objects which are related to themselves.
- Syntax:
  - DL: $\exists R.\mathit{Self}$
  - RDF/OWL: `owl:hasSelf`,
  - Manchester: `Self`
- Semantics:
  - $(\exists R.\mathit{Self})^{\mathcal{I}} = \{x \mid \langle x, x \rangle \in R^{\mathcal{I}}\}$
- Examples:
  - $\mathit{AutoregulatingProcess} \sqsubseteq \exists \mathit{regulate}.\mathit{Self}$
  - $\exists \mathit{hate}.\mathit{Self} \sqsubseteq \mathit{UnhappyPerson}$

Outline

Restrictions, non-unique names and open worlds

Restrictions + the OWA and the NUNA can be tricky, consider:

TBox:

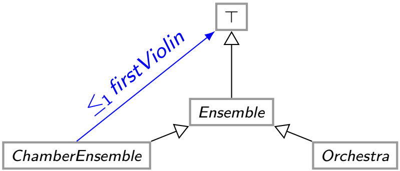
$$Orchestra \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq Ensemble$$

$$ChamberEnsemble \sqsubseteq \leq_1 firstViolin.\top$$

ABox:

```
Ensemble(oslo)
firstViolin(oslo, skolem)
firstViolin(oslo, lie)
```



- Orchestras and Chamber ensembles are Ensembles.
- Chamber ensembles have only one instrument on each voice,
- in particular, only one first violin.
- `oslo` has two first violins; is `oslo` an Orchestra?

Unexpected (non-)results

It does not follow from TBox + ABox that `oslo` is an *Orchestra*:

- An ensemble need neither be an orchestra nor a chamber ensemble, its "just" an ensemble.
- Add "covering axiom" $\mathit{Ensemble} \sqsubseteq \mathit{Orchestra} \sqcup \mathit{ChamberEnsemble}$:
  - An ensemble is an orchestra or a chamber ensemble.

It still does not follow that `oslo` is an *Orchestra*:

- This is due to the NUNA.
- We cannot assume that `skolem` and `lie` are distinct.
- The statement `skolem owl:differentFrom lie`, i.e., `skolem` $\neq$ `lie`, makes `oslo` an orchestra.

If we remove `firstViolin(oslo, lie)`, is `oslo` a *ChamberEnsemble*?

- it does not follow that `oslo` is a *ChamberEnsemble*.
- This is due to the OWA:
- `oslo` may have other first violinists.

Protégé demo of previous slide

- Make class Ensemble.
- Make subclass Orchestra.
- Make subclass ChamberEnsemble.
- Make object property firstViolin.
- Make `firstViolin max 1` superclass of ChamberEnsemble.
- Make an Ensemble oslo
- Make a Thing skolem
- Make a Thing lie
- Add firstViolin skolem to oslo
- Add firstViolin lie to oslo
- Classify! Nothing happens.
- Add covering axiom: `Orchestra or ChamberEnsemble` superclass of Ensemble.
- Classify! Nothing happens.
- `skolem` is different from `lie`
- Classify! Bingo! `oslo` is an Orchestra!

A tempting mistake?

Cardinality restrictions cannot be used to reason with

- intervals or any kind of sequence
- and it cannot be used for arithmetic.
- Example of incorrect modelling:
  - Scotch whisky is casked for (a duration of) more than three years:
  - $\mathit{Scotch} \sqsubseteq \mathit{Whisky} \sqcap \geq_3 \mathit{casked}.\mathit{Years}$ $(*)$

  Why incorrect?

  - The class *Years* is just a set of objects,
  - so the axiom $(*)$ reads "Scotch is Whisky which is casked in at least three (different) years."
  - These years may be unrelated (other then by type), e.g: 1996, 1999, 2010.
  - $\geq_{12} \mathit{casked}.\mathit{Years}$ is not *longer* than $\geq_3 \mathit{casked}.\mathit{Years}$

Outline

Roles and RBoxes

- Just as we have TBoxes and ABoxes for axioms concerning concepts and individual respectively,
- there is an RBox for axioms on roles.
- In the RBox we find
  - role relationships axioms and
  - role characteristics axioms.
- Consider these boxes convenient for bookkeeping,
- and they are used in literature.

Boxes!

Role characteristics and relationships

- A role can be:
  - atomic,
  - the universal role, the empty role,
  - the inverse of a role, or
  - a chain of roles. (The two latter are role builders).
- A role can have the characteristics (axioms):
  - reflexive, irreflexive,
  - symmetric, asymmetric,
  - transitive, or/and[2]
  - functional, inverse functional.
- Role axioms: Let R and S be roles, then we can assert
  - subsumption: $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$,
  - equivalence: $R^{\mathcal{I}} = S^{\mathcal{I}}$,
  - disjointness: $R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$,
  - key: R is a *key* for concept C.

OWL keys!

[2] Restrictions apply

New roles

- The universal role, and the empty role—for both object values and data values.
- Syntax:
  - (DL: U (universal object role), mcD (universal data value role))
  - RDF/OWL, Manchester: `owl:topObjectProperty`, `owl:topDataProperty`, `owl:bottomObjectProperty`, `owl:bottomDataProperty`
- Semantics:
  - $U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\mathcal{D}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Lambda$
- Reads:
  - all pairs of individuals are connected by `owl:topObjectProperty`,
  - no individuals are connected by `owl:bottomObjectProperty`.
  - all possible individuals are connected with all literals by `owl:topDataProperty`,
  - no individual is connected by `owl:bottomDataProperty` to a literal.

Corresponding mathematical properties and operations

A relation R over a set X ($R \subseteq X \times X$) is

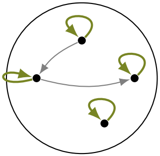
Reflexive:	if $\langle a, a\rangle \in R$ for all $a \in X$
Irreflexive:	if $\langle a, a\rangle \notin R$ for all $a \in X$
Symmetric:	if $\langle a, b\rangle \in R$ implies $\langle b, a\rangle \in R$
Asymmetric:	if $\langle a, b\rangle \in R$ implies $\langle b, a\rangle \notin R$
Transitive:	if $\langle a, b\rangle, \langle b, c\rangle \in R$ implies $\langle a, c\rangle \in R$
Functional:	if $\langle a, b\rangle, \langle a, c\rangle \in R$ implies $b = c$
Inverse functional:	if $\langle a, b\rangle, \langle c, b\rangle \in R$ implies $a = c$

If R and S are binary relations on X then

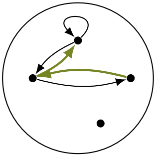
$\langle a, c\rangle \in R \circ S$:	if $\langle a, b\rangle \in R$ and $\langle b, c\rangle \in S$ for some $b \in X$
$\langle b, a\rangle \in R^{-}$:	if $\langle a, b\rangle \in R$.

The syntax for the corresponding axioms is similar, and their semantics should be clear from this slide.

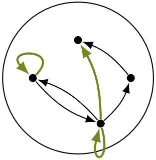
Role characteristics and operations illustrated



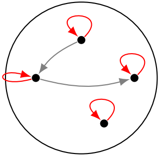
Reflexive



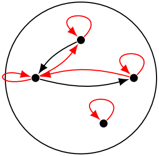
Symmetric



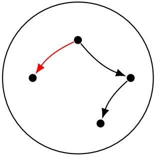
Transitive



Irreflexive

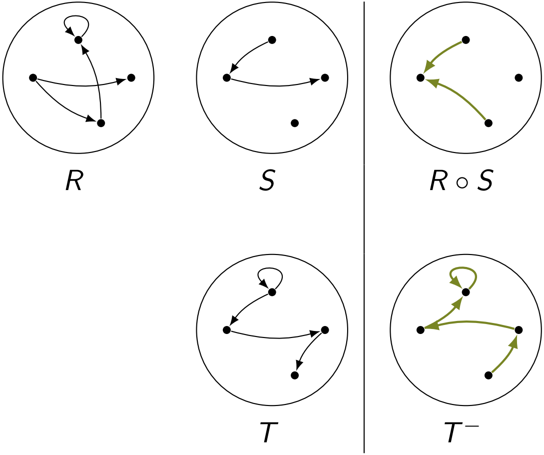


Asymmetric



Functional

Role chaining and inverses illustrated



Properties in OWL

Remember: three kinds of *mutually disjoint* properties in OWL:

1. `owl:DatatypeProperty`
   - link individuals to data values, e.g., `xsd:string`.
   - Examples: `:hasAge`, `:hasSurname`.
2. `owl:ObjectProperty`
   - link individuals to individuals.
   - Example: `:hasFather`, `:driveAxle`.
3. `owl:AnnotationProperty`
   - has no logical implication, ignored by reasoners.
   - Examples: `rdfs:label`, `dc:creator`.

Drive axle!

Characteristics of OWL properties

- Object properties link individuals to individuals, so all characteristics and operations are defined for them.
- Datatype properties link individuals to data values, so they cannot be
  - reflexive—or they would not be datatype properties,
  - transitive—since no property takes data values in 1. position,
  - symmetric—as above,
  - inverses—as above,
  - inverse functional—for computational reasons,
  - part of chains—as above,
  - so, what remains is: functionality,
  - (and subsumption, equivalence and disjointness).
- (Annotation properties have no logical implication, so nothing can be said about them.)

Some relations from ordinary language

- Symmetric relations:
  - *hasSibling*
  - *differentFrom*
- *Non*-symmetric relations:
  - *hasBrother*
- Asymmetric relations:
  - *olderThan*
  - *memberOf*
- Transitive relations:
  - *olderThan*
  - *hasSibling*
- Functional relations:
  - *hasBiologicalMother*
- Inverse functional relations:
  - *gaveBirthTo*

Brother!

Examples inverses and chains

Some inverses:

- $hasParent \equiv hasChild^{-}$
- $hasBiologicalMother \equiv gaveBirthTo^{-}$
- $olderThan \equiv youngerThan^{-}$

Some role chains:

- $hasParent \circ hasParent \sqsubseteq hasGrandParent$
- $isLocatedIn \circ isPartOf \sqsubseteq isLocatedIn$

Grandparents!

Quirks

Role modelling in OWL 2 can get excessively complicated.

- For instance:
  - transitive roles cannot be irreflexive or asymmetric,
  - role inclusions are not allowed to cycle, i.e. not

$$\mathtt{hasParent} \circ \mathtt{hasHusband} \sqsubseteq \mathtt{hasFather}$$
$$\mathtt{hasFather} \sqsubseteq \mathtt{hasParent}.$$

  - transitive roles R and S cannot be declared disjoint
- Note:
  - these restrictions can be hard to keep track of
  - the reason they exist are computational, not logical
- Fortunately:
  - There are also *simple* patterns
  - that are quite useful.

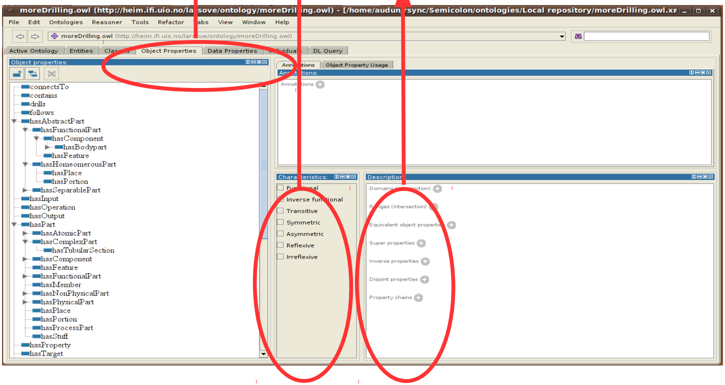
Quirk!

Managing roles in Protege

Object/datatype property tabs

Role characteristics

Domain/range, role relationships



OWL keys

- The OWL equivalent of a database primary key, but not completely ...
- Inverse functional properties apply to instances whose existence may only be implied.
- For inverse datatype properties reasoning is impossible in practise.
- OWL Keys apply only to *named instances*, i.e., it's computationally feasible.
- Works for object properties and datatype properties.
- Example: `Course hasKey {hasCode, hasSemester, hasYear}`:
  - e.g., *this* course is identifies by the values ("INF3580", Spring, "2011").
  - if two courses share the same values, they are the same course.

Outline

Creating datatypes

- Many predefined datatypes are available in OWL:
  - all common XSD datatypes: `xsd:string`, `xsd:int`, ...
  - a few from RDF: `rdf:PlainLiteral`,
  - and a few of their own: `owl:real` and `owl:rational`.
- New datatypes can be defined by boolean operations: $\neg$, $\sqcap$, $\sqcup$:
  - `owl:datatypeComplementOf`, `owl:intersectionOf`, `owl:unionOf`.
- Datatypes may be restricted with *constraining facets*, borrowed from XML Schema.
  - For numeric datatypes: `xsd:minInclusive`, `xsd:maxInclusive`
  - For string datatypes: `xsd:minLenght`, `xsd:maxLenght`, `xsd:pattern`.
- Example:
  - Teenager is equivalent to: (Manchester)
    `Person and (age some positiveInteger[>= 13, <= 19])`
  - "A teenager is a person of age 13 to 19."

Modelling patterns

So, what can we say now?

- ✓ A person has a mother.
- ✓ A penguin eats only fish. A horse eats only chocolate.
- ✓ A nuclear family has two parents, at least two children and a dog.
- ✓ A smoker is not a non-smoker (and vice versa).
- ✓ Everybody loves Mary. ????
- ✓ Adam is not Eve (and vice versa).
- ✓ Everything is black or white.
- ✓ The brother of my father is my uncle.
- ✓ My friend's friends are also my friends.
- ✓ If Homer is married to Marge, then Marge is married to Homer.
- ✓ If Homer is a parent of Bart, then Bart is a child of Homer.

... and more!

Next week

- Recaps.
- More modelling with OWL/OWL 2.
- What cannot be expressed in OWL/OWL 2?

Cap!