Web Ontology Languages

A Troumpoukis, A Charalambidis, S Konstantopoulos

January 2020

What is an ontology

Based on Tom Gruber's ideas, and subsequent refinements,
An ontology is a formal, explicit specification of a shared conceptualization.

- Ontologies introduce terminology and place terms in a taxonomic hierarchy. This is the most conservative definition of what an ontology is.
  - This can already be achieved with RDFS class subsumption or the SKOS vocabulary
- Ontologies are the *specification* of conceptualization: They state axioms that constrain the possible interpretations of the terms they define, but do not necessarily encode the complete meaning of each term.
  - This specification is explicit and formal, so that machines can share it and process it.

Frames

Minsky (1974) already proposed organizing knowledge in *frames*: blocks of *slots* (property-value pairs) that formed the context within which to process content in vision and NLP.

Teaching event	
has-participant	instructor
has-participant	student
has-location	room-name
has-time	date-time
is-part-of	course

Concepts

KL-ONE (Brachman and Schmolze, 1985) organizes frames in a subsumption hierarchy, where more specific frames inherit restrictions and add to them.

Meeting		Teaching event	
has-participant	person	has-participant	instructor
has-participant	person	has-participant	student
has-location	location		room-name
has-time	date-time		
		is-part-of	course

Base and defined Concepts

Base concepts are defined only by their position in the terminology. Are partially specified by *necessary* conditions, but the *sufficient* conditions are not machine-readable and membership must be explicitly asserted.

Defined concepts have *necessary and sufficient* membership conditions explicitly asserted, and membership is automatically deducible.

Base and defined Concepts

'meeting' is a base concept:

- Other events will also have participant, place, and time slots and will not necessarily be meetings.
- But if we know something is a meeting, then there are some specifications about what properties it must have.

$$\begin{aligned} \text{meeting}(X) \quad \rightarrow \quad & \text{has-participant}(X, P) \wedge \text{person}(P) \wedge \\ & \text{has-location}(X, L) \wedge \text{location}(L) \wedge \\ & \text{has-time}(X, T) \wedge \text{date-time}(T) \end{aligned}$$

Notice how only two variables are really needed. We will revisit this point.

Base and defined Concepts

'teaching event' is a defined sub-concept of a meeting:

- But if we know something is a meeting, then there are such properties that if it has them, it must be a teaching event.

$$\begin{aligned}\text{teaching-event}(X) \quad \leftrightarrow \quad & \text{meeting}(X) \wedge \\ & \text{has-participant}(X, I) \wedge \text{instructor}(I) \wedge \\ & \text{has-participant}(X, S) \wedge \text{student}(S) \wedge \\ & \text{has-location}(X, L) \wedge \text{room}(L) \wedge \\ & \text{is-part-of}(X, C) \wedge \text{course}(C)\end{aligned}$$

It is a bit less straightforward, but again only two variables are really needed.

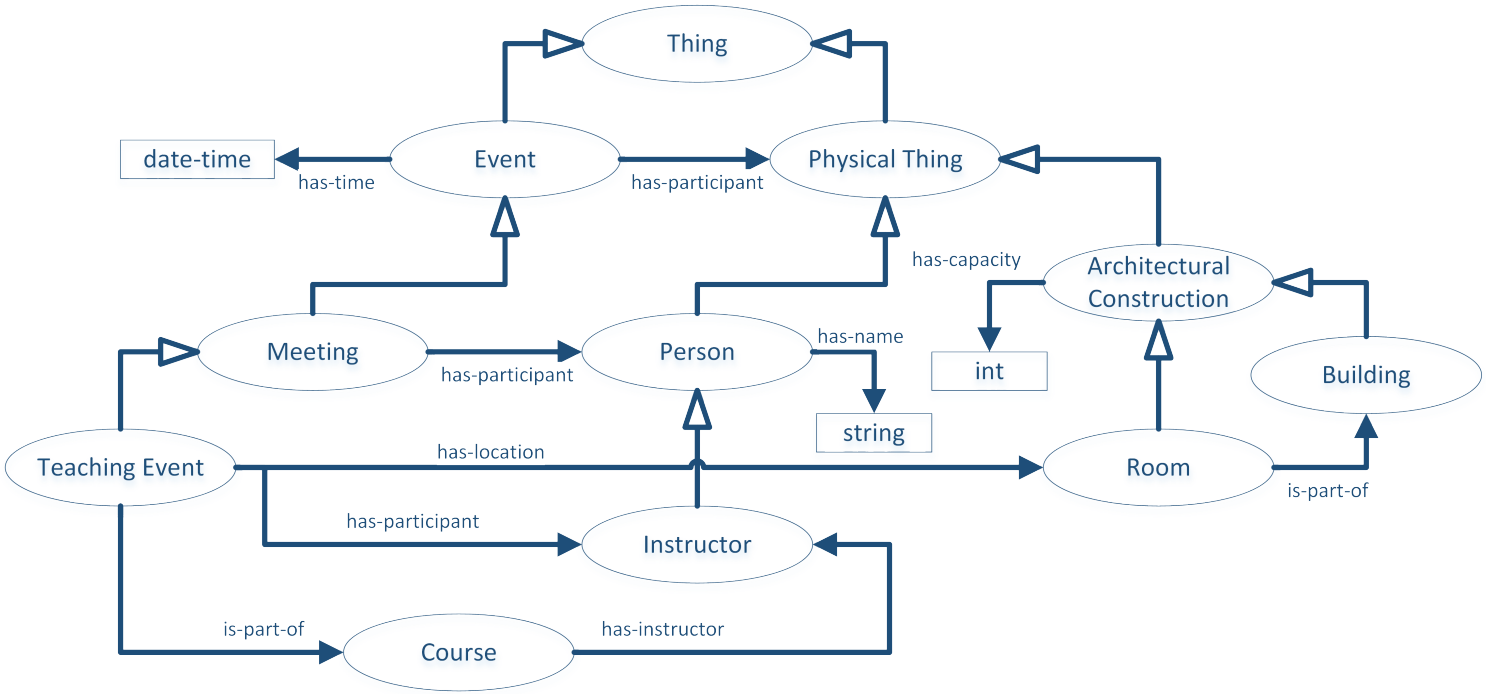
Semantic Networks

The restrictions on slot fillers imply a relationship between frames, besides the subsumption hierarchy. These relationships organize concepts into *semantic networks* that link concepts both taxonomically and non-taxonomically.

Teaching event		Instructor	
has-participant	Instructor	has-name	string
has-participant	Student	teaches	Course
has-location	Room		
is-part-of	Course		
Room		**Course**	
has-name	string	has-name	string
has-capacity	int	is-in-semester	int
has-beamer	boolean	has-instructor	Instructor

Semantic Networks

The restrictions on slot fillers imply a relationship between frames, besides the subsumption hierarchy. These relationships organize concepts into *semantic networks* that link concepts both taxonomically and non-taxonomically.



Formalizing frames into modern ontology languages

- Throughout the 1980s, frames and semantic networks were used to express knowledge for expert systems
- *Description Logics (DL)* start to get studied as the subset of FOPL needed to formally express entailment over the same kind of knowledge as frames and semantic networks
- KL-ONE (1985) is the first KRR system using DL to formalize its rules language
  - Since 1990: new inference algorithms, usable DL reasoners
  - Since 2000: DL reasoners see productive use (Racer, Pellet)
  - Multiple initiatives start developing ontology engineering systems based on DL (DAML, OIL, DAML+OIL).
- In 2004 *OWL DL* is defined as one of the semantic extensions of RDF graphs officially specified by the W3C
  - OWL is mostly based on DAML+OIL
  - Three inference profiles: RDFS, OWL Lite, OWL DL

Dyadic FOPL

Description Logics is a family of different subsets of dyadic (with a minor exception) FOPL

Dyadic FOPL (and exception) are decidable fragments of FOPL.
Navigator for complexity results:
`http://www.cs.man.ac.uk/~ezolin/dl`

DLs are a different, overlapping, fragment than Horn clauses:
DLs have existential quantification and negation, but are restricted to two variables and constants of arity zero.

Proof theory

Simpler DLs have a finite fix-point, and forward-chaining can be applied.

For more complex DLs, we use goal-driven *tableaux*:

1. Start from P
2. Derive new propositions, branching off on disjunctions.
3. If a branch reaches a contradiction, it is closed.
4. If all branches are closed, P is inconsistent.

Proof theory: propositional case

To prove the validity of A w.r.t. $\mathcal{O}$, we need to prove $\neg A$ inconsistent. Let $\mathcal{O}$ define r as $p \vee q \leftrightarrow r$

$\mathcal{O}$:	$p \rightarrow r \quad q \rightarrow r \quad r \rightarrow p \vee q$
$\neg P$:	$\neg(r \wedge (p \rightarrow w) \rightarrow (w \vee q))$
1 (from P) :	$r \wedge (p \rightarrow w)$
2 (from P) :	$\neg w \wedge \neg q$
3 (from 1) :	r
4 (from 1) :	$\neg p \vee w$
5 (from $\mathcal{O}_3$) :	$\neg r \vee p \vee q$
6a (from 4) :	w
7a (from 2, 6a) :	closed
6b (from 4) :	$\neg p$
7b (from 5,3,6b,2) :	closed

Proof theory: universal quantifiers

Universal quantifiers get grounded to existing ground terms

$$\begin{aligned} \mathcal{O}: &\quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \\ P: &\quad \text{teaches}(a, b) \end{aligned}$$

$$\begin{aligned} \mathcal{O}_1: &\quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \wedge \text{course}(Y) \\ \mathcal{O}_2: &\quad \forall X.\text{instructor}(X) \rightarrow \text{person}(X) \\ P: &\quad \text{teaches}(c, c) \end{aligned}$$

$$\begin{aligned} \mathcal{O}_1: &\quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \wedge \text{course}(Y) \\ \mathcal{O}_2: &\quad \forall X.\text{instructor}(X) \rightarrow \text{person}(X) \\ \mathcal{O}_3: &\quad \forall X.((\text{person}(X) \rightarrow \neg\text{course}(X)) \wedge (\text{course}(X) \rightarrow \neg\text{person}(X))) \\ P: &\quad \text{teaches}(c, c) \end{aligned}$$

Proof theory: skolemization

Existential quantifiers introduce new ground terms

$$\begin{aligned}
\mathcal{O}_1 &: \quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \\
\mathcal{O}_2 &: \quad \forall X.\text{instructor}(X) \rightarrow \text{person}(X) \\
P &: \quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{person}(X)
\end{aligned}$$

$$\begin{aligned}
\mathcal{O}_1 &: \quad \forall X.\text{instructor}(X) \rightarrow \exists Y.\text{teaches}(X, Y) \\
\mathcal{O}_1 &: \quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \wedge \text{course}(Y) \\
\mathcal{O}_3 &: \quad \text{instructor}(a) \\
P &: \quad \neg\exists Y.\text{teaches}(a, Y)
\end{aligned}$$

$$\begin{aligned}
\mathcal{O}_1 &: \quad \forall X.\text{instructor}(X) \rightarrow \exists Y.\text{teaches}(X, Y) \\
\mathcal{O}_1 &: \quad \forall X, Y.\text{teaches}(X, Y) \rightarrow \text{instructor}(X) \wedge \text{course}(Y) \\
\mathcal{O}_3 &: \quad \neg\text{instructor}(a) \\
P &: \quad \neg\exists Y.\text{teaches}(a, Y)
\end{aligned}$$

OWL, OWL DL, OWL 2...

- The Web Ontology Language (OWL) is a suite of W3C recommendations on Web semantics.
- Independent of, but compatible with, RDF.
- In the original recommendation suite (2004): OWL Lite, OWL DL, OWL Full.
- In the current (2009, 2012) recommendation suite:
  - OWL 2 QL: Most efficient profile, minimal additions to RDFS
  - OWL 2 RL: Middle ground, including all features that can be efficiently implemented as rules
  - OWL 2 EL: Richest OWL profile, including all features in DL literature used in actual practice

Subsumption and equivalence

DL syntax

$C \sqsubseteq D$ $\quad\quad$ $C \equiv D$

Functional syntax

SubClassOf(C D) $\quad\quad$ EquivalentClasses(C D)

Manchester syntax

C SubClassOf D $\quad\quad$ C EquivalentTo D

RDF

```
ex:C rdfs:subClassOf ex:D .
ex:C owl:equivalentClass ex:D .
```

FOPL

$\forall X.C(X) \rightarrow D(X)$ $\quad\quad$ $\forall X.C(X) \leftrightarrow D(X)$

Disjointness

DL syntax

$\mathrm{C} \sqsubseteq \neg \mathrm{D}$ $\quad\quad$ $\mathrm{D} \sqsubseteq \neg \mathrm{C}$

Functional syntax

DisjointClasses(C D)

Manchester syntax

C DisjointWith D

RDF

```
ex:C owl:disjointWith ex:D .
```

FOPL

$\forall X.C(X) \rightarrow \neg D(X)$
$\forall X.D(X) \rightarrow \neg C(X)$

Local CWA

DL syntax

$$\text{Instructor} \equiv \{\text{stasinos angelos antonis}\}$$

Functional syntax

```
EquivalentClasses( Instructor
                   ObjectOneOf(stasinos angelos antonis) )
```

Manchester syntax

Instructor EquivalentTo { stasinos, angelos, antonis }

RDF

```
ex:Instructor owl:equivalentClass _:x owl:oneOf _:1 .
_:1 rdf:first ex:stasinos ; rdf:rest _:2 .
_:2 rdf:first ex:angelos ; rdf:rest _:3 .
_:3 rdf:first ex:antonis ; rdf:rest rdf:nil .
```

Filler constraints

DL syntax

$\text{Instructor} \sqsubseteq \forall \text{teaches}.\text{Course}$

Functional syntax

SubClassOf(Instructor ObjectAllValuesFrom(teaches Course))

RDF

```
ex:Instructor rdfs:subClassOf _:x .
_:x rdf:type owl:Restriction ;
    owl:onProperty ex:teaches ;
    owl:allValuesFrom ex:Course .
```

FOPL

$\forall X, Y.\text{Instructor}(X) \wedge \text{teaches}(X, Y) \rightarrow \text{Course}(Y)$

Inverse and functional properties

DL syntax

$\text{hasInstructor} \equiv \text{teaches}^-$

Functional syntax

FunctionalObjectProperty(hasInstructor)
InverseObjectProperties(hasInstructor teaches)

Manchester syntax

Functional: hasInstructor
hasInstructor InverseOf teaches

FOPL

$\forall X, Y.\text{hasInstructor}(X, Y) \leftrightarrow \text{teaches}(Y, X)$

Existential constraints

DL syntax

$$\text{Instructor} \equiv \text{Person} \sqcap \exists \text{teaches}.\text{Course}$$

Functional syntax

```
EquivalentClasses(
        Instructor
        ObjectIntersectionOf(
                Person
                ObjectSomeValuesFrom(teaches Course)))
```

Manchester syntax

Instructor EquivalentTo Person and (teaches some Course)

FOPL

$$\forall X \exists Y.\text{Instructor}(X) \leftrightarrow \text{Person}(X) \wedge \text{teaches}(X, Y) \wedge \text{Course}(Y)$$

Existential constraints

DL syntax

$$\text{Instructor} \equiv \text{Person} \sqcap \exists \text{teaches}.\text{Course}$$

RDF

```
ex:Instructor owl:EquivalentClass _:1 .
_:1  ObjectIntersectionOf _:2 .
_:2 rdf:first ex:Person ;
    rdf:rest _:3 .
_:3 rdf:first _:x ;
    rdf:rest rdf:nil .
_:x rdf:type owl:Restriction ;
    owl:onProperty ex:teaches ;
    owl:someValuesFrom ex:Course .
```

Property transitivity

DL syntax

$\text{partOf}^{+} \sqsubseteq \text{partOf}$
$\text{ArchitecturalConstr} \sqsubseteq \forall \text{partOf}^{+}.\text{ArchitecturalConstr}$

Functional syntax

TransitiveObjectProperty(partOf)

Manchester syntax

Transitive: partOf

RDF

ex:partOf rdf:type owl:TransitiveProperty .

Property transitivity

DL syntax

$\text{partOf}^+ \sqsubseteq \text{partOf}$
$\text{ArchitecturalConstr} \sqsubseteq \forall \text{partOf}^+.\text{ArchitecturalConstr}$

FOPL

$\forall X, Y, Z.\text{partOf}(X, Y) \wedge \text{partOf}(Y, Z) \rightarrow \text{partOf}(X, Z)$
Three variables, but the works of Fischer and Ladner (1979) and Horrocks et al. (2000) tell us it is still decidable.

Cardinality restrictions

DL syntax

$\text{Course} \sqsubseteq \leq 4\ \text{hasParticipant.Instructor}$

$\text{Course} \sqsubseteq \geq 5\ \text{hasParticipant.Student}$

Functional syntax

SubClassOf(Course ObjectMaxCardinality(4 hasPartic Instructor))

SubClassOf(Course ObjectMinCardinality(5 hasPartic Student))

Manchester syntax

Course SubClassOf hasParticipant max 4 Instructor

Course SubClassOf hasParticipant min 5 Student

Cardinality restrictions

RDF

```
ex:Instructor rdfs:subClassOf _:x owl:Restriction .
_:x owl:maxQualifiedCardinality "4"^^xsd:nonNegInt;
    owl:onProperty ex:hasParticxipant ;
    owl:onClass ex:Instructor .
```

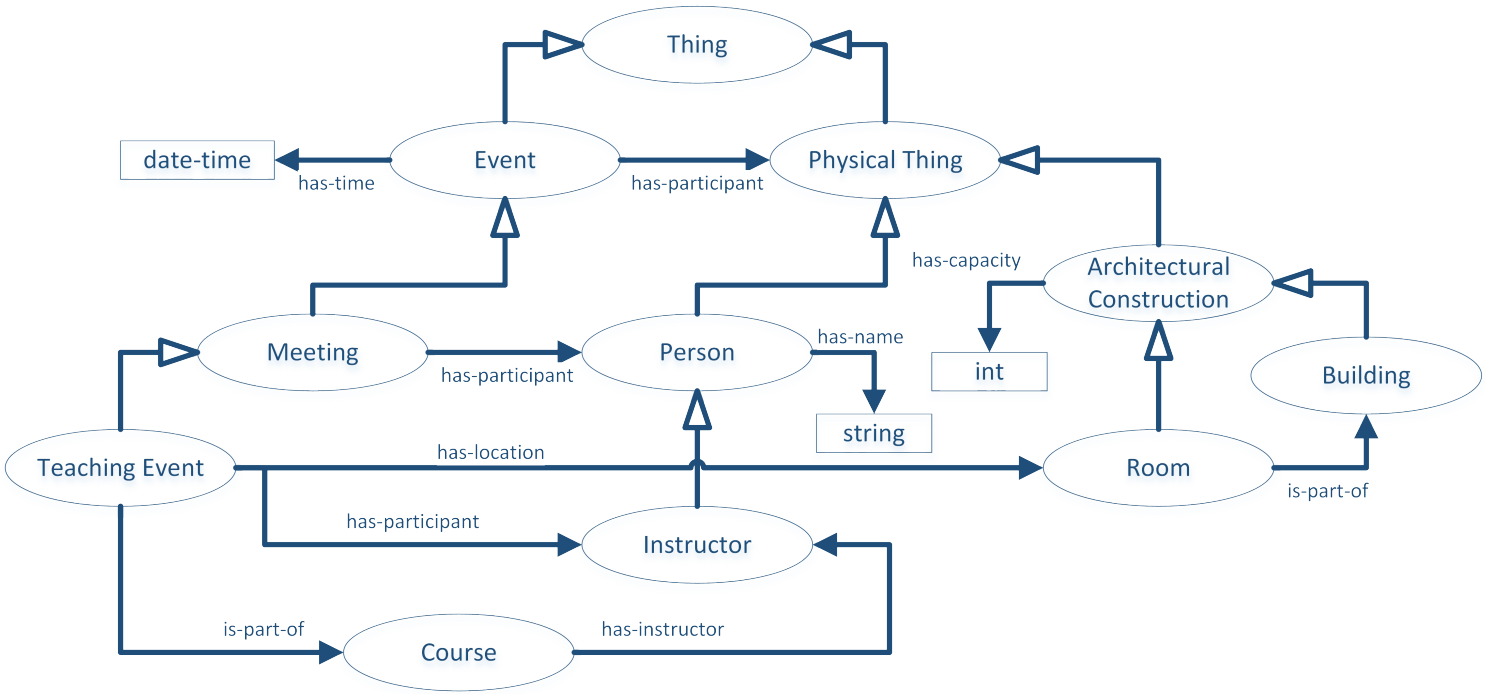
FOPL

More than two variables are needed, but still decidable (Grädel et al., 1997):

$$\forall X, Y_{i=1..5}$$
$$\text{Course}(X) \wedge \text{hasParticipant}(X, Y_i) \wedge_{i=1..5} \text{Instructor}(Y_i)$$
$$\rightarrow \neg \left(\wedge_{i,j=1..5, i \neq j} Y_i \neq Y_j\right)$$

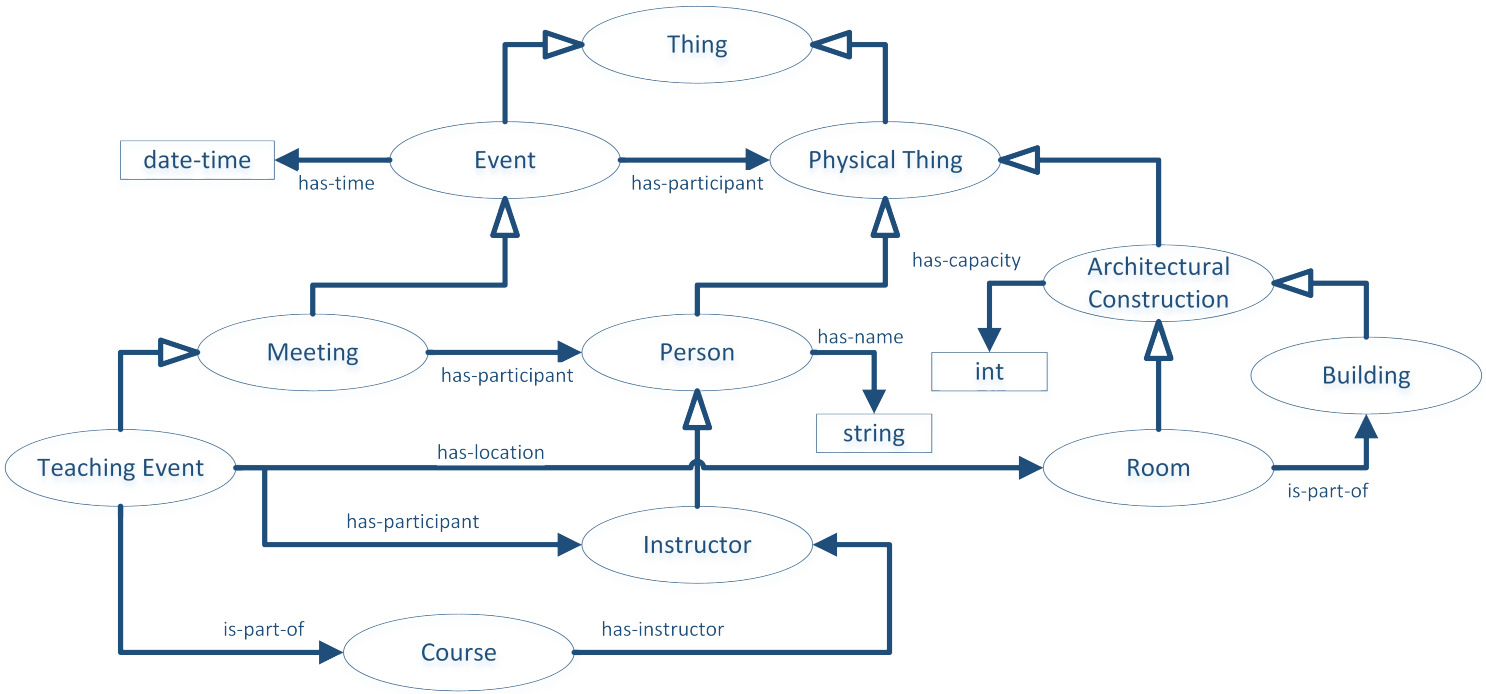
What am I missing?

- Arithmetic: the capacity of a construction cannot be axiomatized to be the sum of the capacities of its parts, even under local CWA.



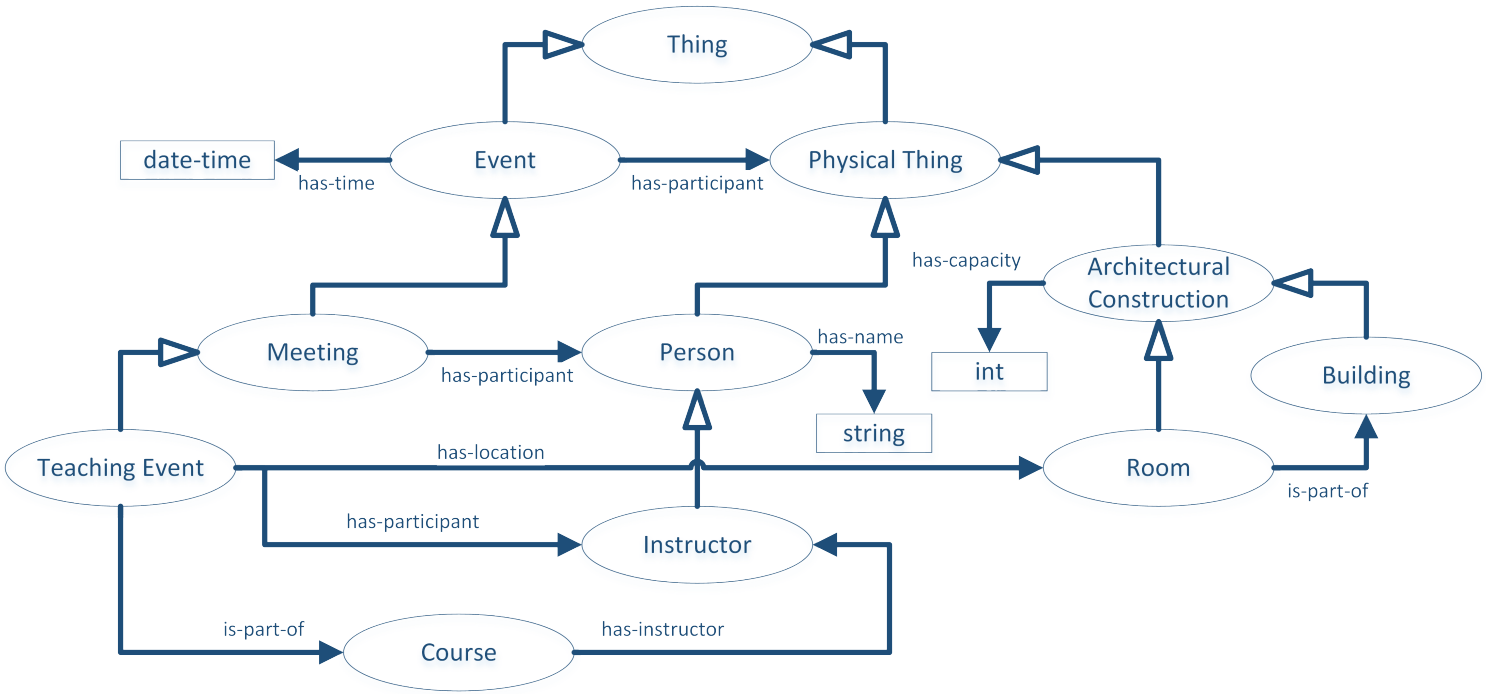
What am I missing?

- Variable cardinality restrictions: the number of students cannot be axiomatized to be higher than the number of instructors.



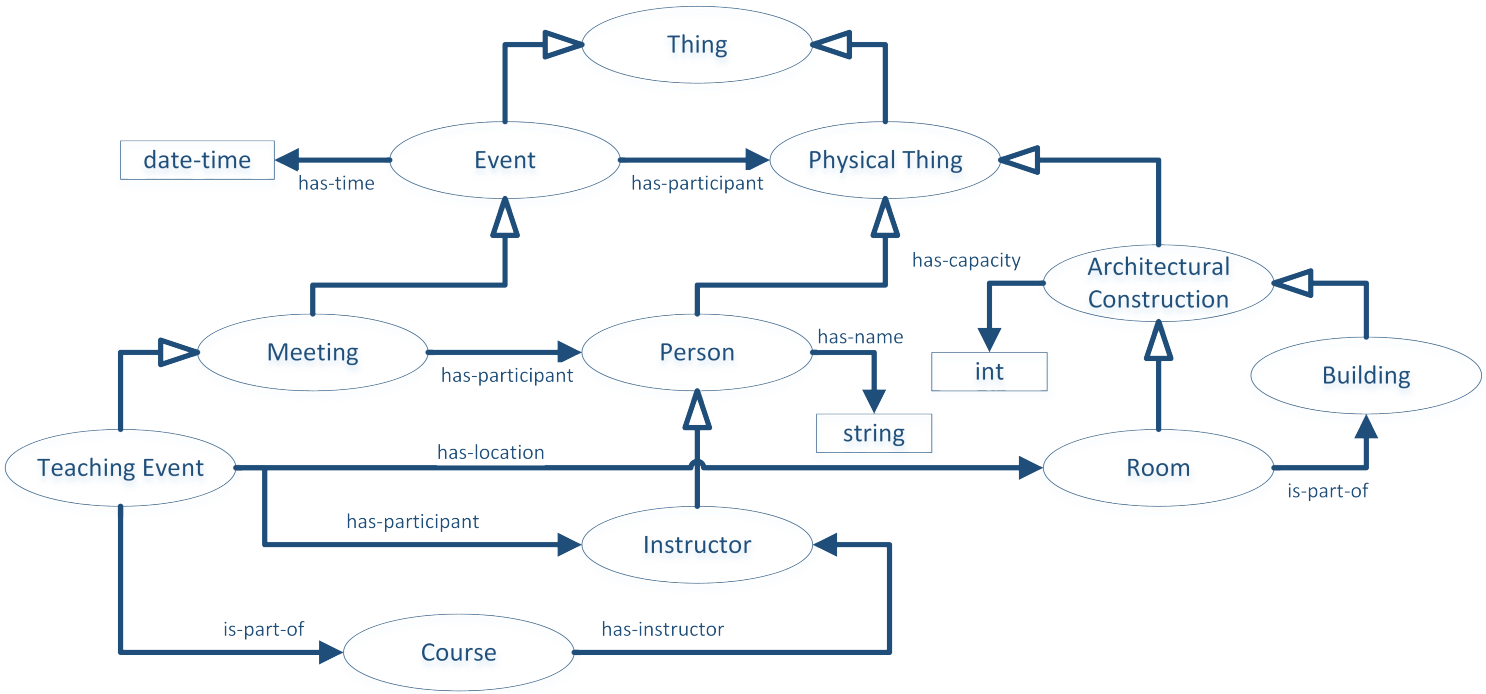
What am I missing?

- Expressivity on property fillers: cannot infer the course instructor to be a participant the course's events.



What am I missing?

- Defaults: cannot infer the course instructor to be a participant the course's events unless another instructor is present; or all rooms of Buidling 42 to have a capacity of 16 unless known otherwise.



References

Textbooks and standards:

W3C OWL Working Group, *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation, 11 December 2012. `http://www.w3.org/TR/owl2-overview`

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (eds), *The Description Logic Handbook: Theory, Implementation and Applications.* 2nd Edition, Cambridge University Press, May 2010.

Protege ontology editor: `https://protege.stanford.edu`
Strongly recommended for the exercises.

Original works cited here:

(Minsky, 1974): Marvin Minsky, A Framework for Representing Knowledge. *MIT Report 306*, June 1974.

(Brachman and Schmolze, 1985): RJ Brachman and JG Schmolze, An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2). doi: 10.1207/s15516709cog0902_1

(Fischer and Ladner, 1979): Michael J. Fischer and Richard E. Ladner, Propositional Dynamic Logic of regular programs. *Journal of Computer and System Science* 18(2), 1979.

(Horrocks et al., 2000): Ian Horrocks, Ulrike Sattler, and Stephan Tobies, Practical reasoning for very expressive Description Logics. *Logic Journal of the IGPL* 8(3), 2000.

(Grädel et al., 1997): E. Grädel, M. Otto, and E. Rosen, Two-variable logic with counting is decidable. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, 1997.