Mappings between Description Logics syntax – OWL syntax – FOPL syntax

Stasinos Konstantopoulos

January 2024

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.
  - The formal definition gives a set-theoretic semantics to the Functional syntax
  - Equivalent RDF code is provided as a recommendation on how to represent OWL ontologies that describe RDF data

OWL, OWL DL, OWL 2...

The original recommendation suite (2004) defined three variants:

- OWL Lite: Most efficient, can use backward chaining (Goal-driven resolution) to answer queries
- OWL DL: Middle ground, including all features that DL reasoners implemented at the time
- OWL Full: Undecidable, including second-order construct, only means as an expressive formalism for knowledge exchange without inference

The current recommendation suite (2009, 2012) makes a distinction at the low end and abandons OWL Full:

- OWL 2 QL: Most efficient profile, minimal additions to RDFS
- OWL 2 RL: Very efficient, including all features that are compatible with backward chaining
- OWL 2 EL: Richest OWL profile, including all features in DL literature implemented in current reasoners.

Subsumption and equivalence

DL syntax

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

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

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

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

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

Functional syntax

FunctionalObjectProperty(hasInstructor)
InverseObjectProperties(hasInstructor teaches)

Manchester syntax

Functional: hasInstructor
hasInstructor InverseOf teaches

FOPL

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

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 .
```

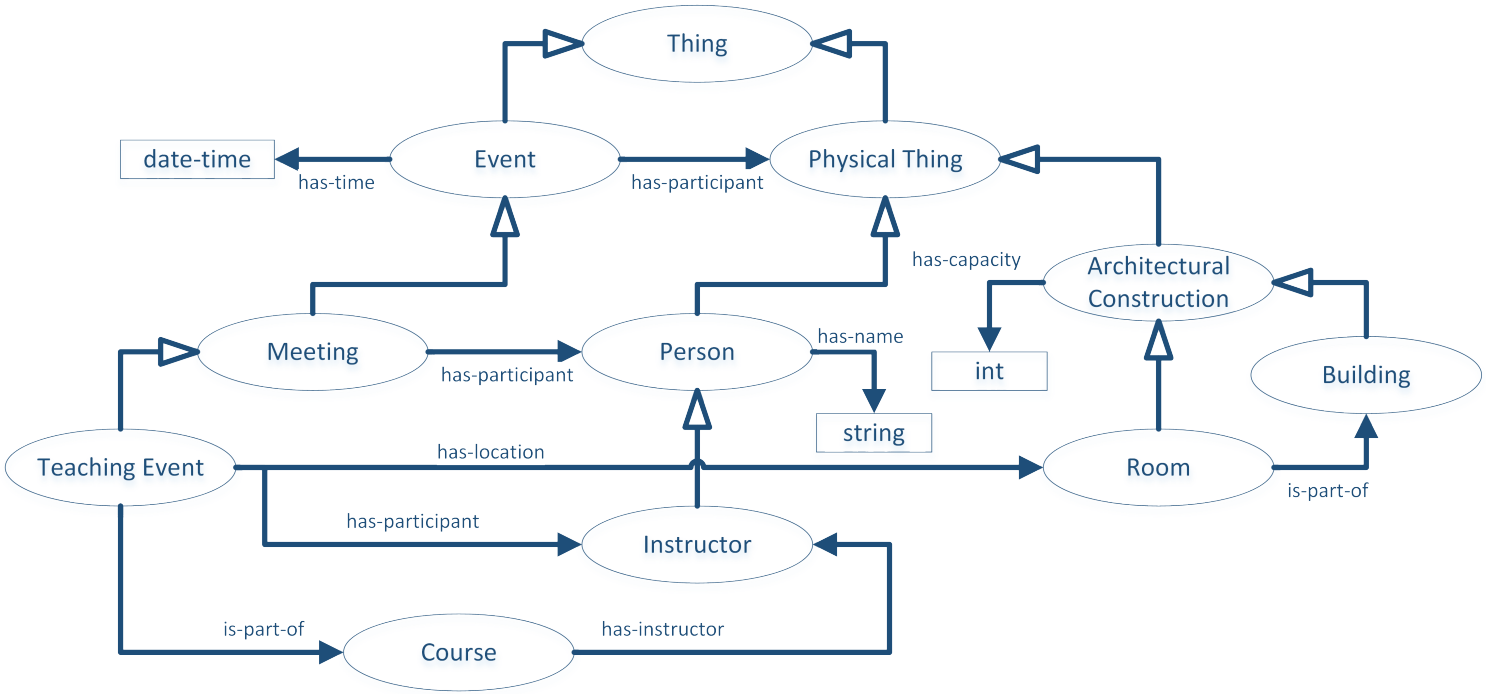
FOPL

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

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

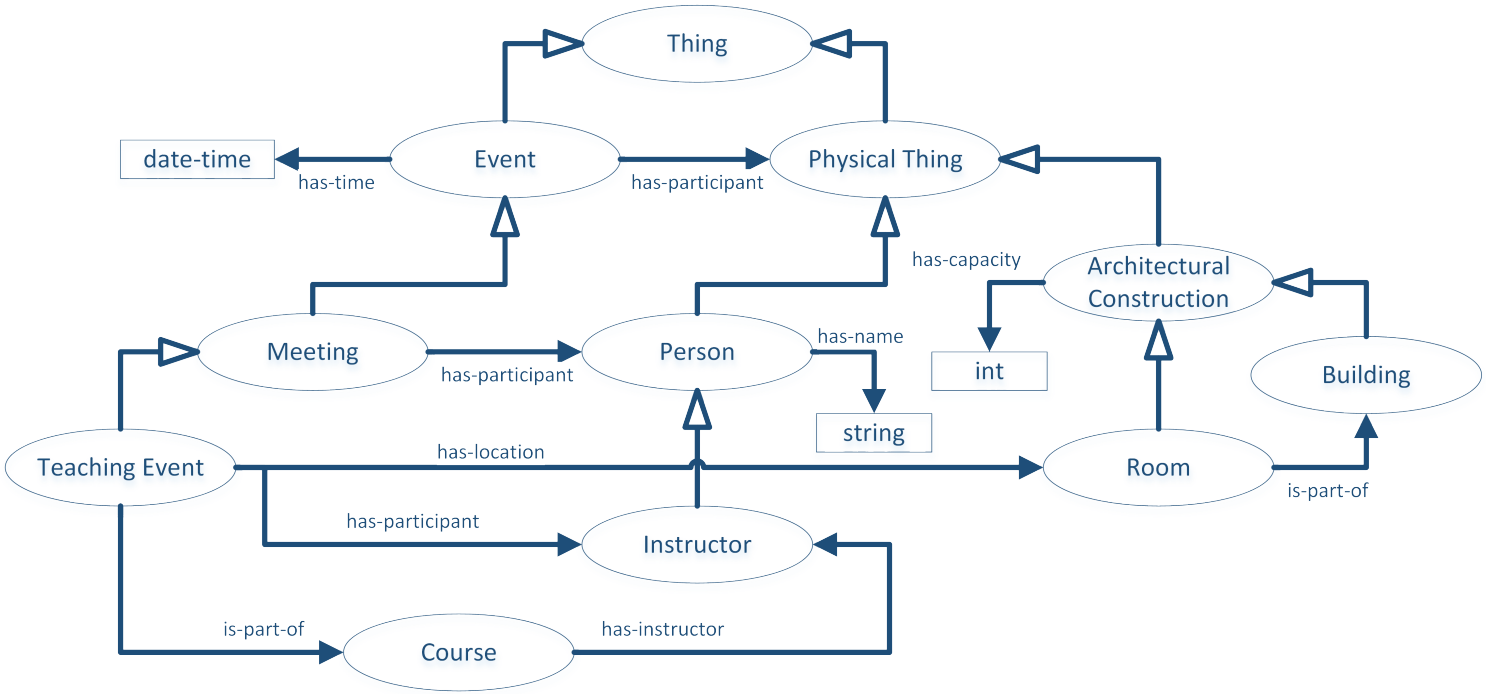
What am I missing?

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



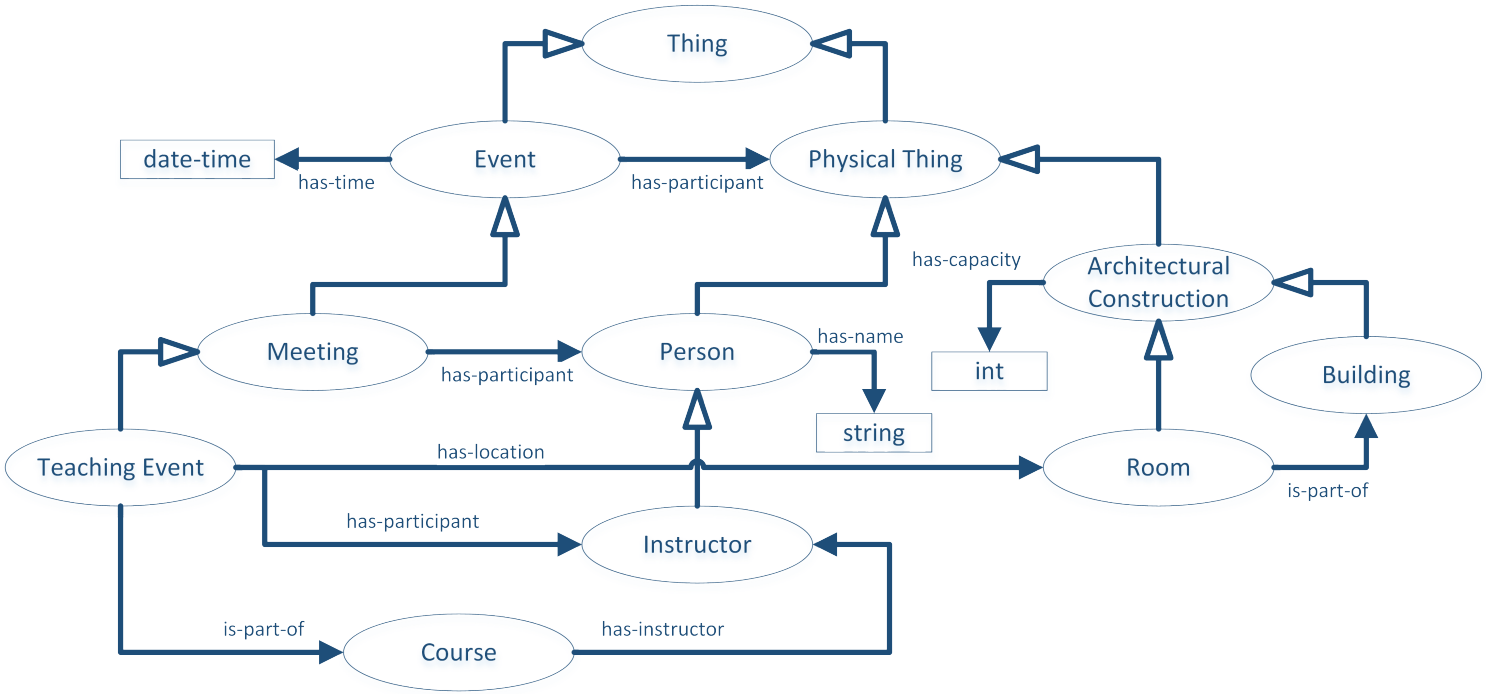
What am I missing?

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



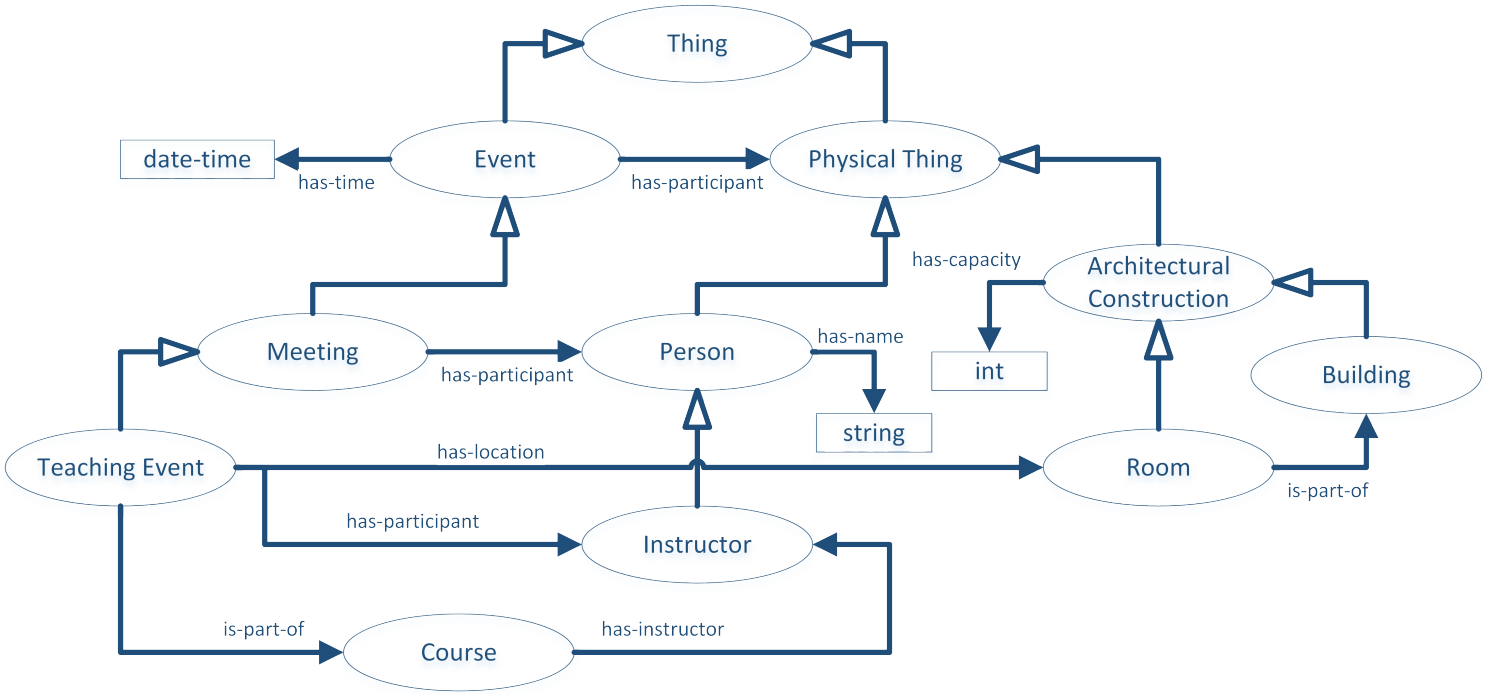
What am I missing?

- 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.



Further Reading

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.

(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.