Chapter 7
Ontologies and the Web Ontology Language – OWL

- *vocabularies* can be defined by RDFS
  - not so much stronger than the ER Model or UML
    (even weaker: no cardinalities)
  - not only a conceptual model, but a “real language” with a close connection to the data level (RDF)
  - *incremental* world-wide approach
  - “global” vocabulary can be defined by autonomous partners
- but: still restricted when *describing* the vocabulary.

Ontologies/ontology languages further extend the expressiveness:

- Description Logics
- Topic Maps (in SGML) since early 90s, XTM (XML Topic Maps)
- Ontolingua – non-XML approach from the Knowledge Representation area
- OIL (Ontology Inference Layer): initiative funded by the EU programme for Information Society Technologies (project: On-To-Knowledge, 1.2000-10.2002); based on RDF/RDFS
- DAML (Darpa Agent Markup Language; 2000) ... first ideas for a Semantic Web language
- DAML+OIL (Jan. 2001)
- developed into OWL (1st version March 02, finalized Feb. 04)

Three Variants of OWL

Several expressiveness/complexity/decidability levels:

- OWL Full: extension of RDF/RDFS
  - classes can also be regarded as individuals (have properties, classes of classes etc.)
- OWL DL
  - fragment of OWL that fits into the Description Logics Framework:
    - ⋆ the sets of classes, properties, individuals and literals are disjoint
    - ⇒ only individuals can have arbitrary user-specified properties;
      classes and properties have only properties from the predefined RDFS and OWL vocabularies.
  - decidable reasoning
  - OWL 1.0 (2004), OWL 2.0 (2009)
- OWL Lite
  - subset of OWL DL
  - easier migration from frame-based tools (note: F-Logic is a frame-based framework)
  - easier reasoning (translation to Datalog)

7.1 Description Logics

- Focus on the description of *concepts*, not of instances
- Terminological Reasoning
- Origin of DLs: Semantic Networks (graphical formalism)

Notions

- Concepts (= classes),
  note: literal datatypes (string, integer etc.) are not classes in DL and OWL, but *data ranges*
  (cf. XML Schema: distinction between simpleTypes and complexTypes)
- Roles (= relationships),
- A Description Logic alphabet consists of a finite set of concept names (e.g. Person, Cat, LivingBeing, Male, Female, . . .) and a finite set of role names (e.g., hasChild, marriedTo, . . .),
- constructors for derived concepts and roles,
- axioms for asserting facts about concepts and roles.

Comparison with other Logics

Syntax and semantics defined different but similar from first-order logic

- formulas over an alphabet and a small set of additional symbols and combinators
- semantics defined via *interpretations* of the combinators
- set-oriented, no instance variables
  (FOL: instance-oriented with domain quantifiers)
- family of languages depending on what combinators are allowed.

The base: $\mathcal{AL}$

The usual starting point is $\mathcal{AL}$:

- "attributive language"
- Manfred Schmidt-Schauss and Gert Smolka: *Attributive Concept Descriptions with Complements*. In *Artificial Intelligence* 48(1), 1991, pp. 1–26.
- extensions (see later: $\mathcal{ALC}$, $\mathcal{ALCQ}$, $\mathcal{ALCQ}(D)$, $\mathcal{ALCQI}$, $\mathcal{ALCN}$ etc.)

Atomic, Named Concepts

- atomic concepts, e.g., Person, Male, Female
- the "universal concept" $\top$ (often called "Thing" – everything is an instance of Thing)
- the empty concept $\bot$ ("Nothing"). There is no thing that is an instance of $\bot$.

Concept Expressions using Set Operators

- intersection of concepts: $A \sqcap B$
  Adult ⊓ Male
- negation: $\neg A$
  ¬Italian , Person ⊓ ¬Italian
- union (disjunctive concept): $A \sqcup B$
  Cat ⊓ Dog – things where it is known that they are cats or dogs, but not necessarily which one.

Concept Expressions using Roles

Concepts (as an intensional characterization of sets of instances) can be described implicitly by their properties (wrt. *roles*).

Let R be a role, C a concept. Then, the expressions $\exists R.C$ and $\forall R.C$ also describe concepts (intensionally defined concepts) by constraining the roles:

- Existential quantification: $\exists R.C$ – all things that have a *filler* for the role R that is in C.
  ∃hasChild.Male means "all things that have a male child".
  Syntax: the whole expression is the "concept expression", i.e., ∃hasChild.Male$(john)$ stands for (∃hasChild.Male)$(john)$.
- Range constraints: $\forall R.C$
  ∀hasChild.Male means "all things that have only male children (including those that have no children at all)".
- Note that $\bot$ can be used to express non-existence: $\forall R.\bot$: all things where all fillers of role R are of the concept $\bot$ (= Nothing) – i.e., all things that do not have a filler for the role R.
  ∀hasChild.$\bot$ means "all things that have no children".

Semantics of Concept Constructors

As usual: by interpretations.

An interpretation $\mathcal{I} = (\mathcal{I}, \mathcal{D})$ consists of the following:

- a domain $\mathcal{D}$,
- for every concept name C: $I(C) \subseteq \mathcal{D}$ is a subset of the domain,
- for every role name R: $I(R) \subseteq \mathcal{D} \times \mathcal{D}$ is a binary relation over the domain.

Structural Induction

- $I(A \sqcup B) = I(A) \cup I(B)$
- $I(A \sqcap B) = I(A) \cap I(B)$
- $I(\neg A) = \mathcal{D} \setminus I(A)$
- $I(\exists R.C) = \{x \mid$ there is an y such that $(x, y) \in I(R)$ and $y \in I(C)\}$
- $I(\forall R.C) = I(\neg\exists R.(\neg C)) = \{x \mid$ for all y such that $(x, y) \in I(R)$, $y \in I(C)\}$

Example

Male ⊓ ∀hasChild.Male is the set of all men who have only sons.

Structure of a DL Knowledge Base

DL Knowledge Base

TBox (schema)

Statements/Axioms about concepts

Man $\equiv$ Human $\sqcap$ Male

Parent $\equiv$ Human $\sqcap$ ($\exists \geq 1$ hasChild.$\top$)

ParentOfSons $\equiv$ Human $\sqcap$ ($\exists \geq 1$ hasChild.Male)

ParentOfOnlySons $\equiv$ Human $\sqcap$ ($\forall$ hasChild.Male)

ABox (data)

Statements/Facts about individuals

Person(John), (Adult $\sqcap$ Male)(John), ($\neg$Italian)(John)

hasChild(John,Alice), age(Alice,10), Female(Alice)

hasChild(John,Bob), age(Bob,8), Male(Bob)

$\forall$hasChild.$\bot$ (Alice), $\neg\exists$hasChild.$\top$ (Bob)

The TBox: Terminological Axioms

Definitions and assertions (not to be understood as constraints) about concepts:

- concept subsumption: $C \sqsubseteq D$; defining a concept hierarchy.
  $\mathcal{I} \models C \sqsubseteq D \;\;:\Leftrightarrow\;\; I(C) \subseteq I(D)$.
- concept equivalence: $C \equiv D$; often used for defining the left-hand side concept.
  Semantics: $\mathcal{I} \models C \equiv D \;\;:\Leftrightarrow\;\; C \sqsubseteq D$ and $D \sqsubseteq C$.

TBox Reasoning

- is a concept C satisfiable?
- is $C \sqsubseteq D$ implied by a TBox
- given the definition of a new concept D, classify it wrt. the given concept hierarchy.

The ABox: Assertional Axioms

- contains the facts about instances (using names for the instances) in terms of the basic concepts and roles:
  Person(John), Male(John), hasChild(John,Alice)
- contains also knowledge in terms of intensional concepts, e.g., ∃hasChild.Male(John)

TBox + ABox Reasoning

- check consistency between ABox and a given TBox
- ask whether a given instance satisfies a concept C
- ask for all instances that have a given property
- ask for the most specific concepts that an instance satisfies

Note: instances are allowed only in the ABox, not in the TBox.

If instances should be used in the definition of concepts (e.g., "European Country" or "Italian City"), *Nominals* must be used (see later).

Family of DL Languages up to $\mathcal{ALC}$

- $\mathcal{AL}$: intersection, negation of *atomic* concepts
- $\mathcal{AL}$: restricted existential quantification: $\exists R.\top$
  ∃hasChild.$\top$ means "all things that have a child (... that belongs to the concept "Thing")".
- $\mathcal{AL}$ has no "branching" (no union, or any kind of disjunction); so proofs in $\mathcal{AL}$ are linear.
- $\mathcal{U}$: "union"; e.g. Parent $\equiv$ Father $\sqcup$ Mother.
- $\mathcal{C}$: negation ("complement") of non-atomic concepts.
  Person $\sqcap$ ¬∃hasChild.$\top$ characterizes the set of persons who have no children (note: open-world semantics of negation!)

  Note: the FOL equivalent would be expressed via variables:
  $\forall x(\text{Childless}(x) \leftrightarrow (\text{Person}(x) \wedge \neg\exists y(\text{hasChild}(x, y))))$
- $\mathcal{U}$ and $\mathcal{E}$ can be expressed by $\mathcal{C}$.
- $\mathcal{ALC}$ is the "smallest" Description Logic that is closed wrt. the set operations.
- A frequently used restriction of $\mathcal{AL}$ is called $\mathcal{FL}^-$ (for "Frame-Language"), which is obtained by disallowing negation completely (i.e., having only positive knowledge).

FAMILY OF DL LANGUAGES: EXTENSIONS TO $\mathcal{ALC}$

- $\mathcal{E}$: (unrestricted) existential quantification of the form $\exists R.C$ (recall that $\mathcal{AL}$ allows only $\exists R.\top$).
  $\exists$hasChild.Male , for persons who have at least one male child,
  $\exists$hasChild.hasChild.$\top$ for grandparents.
  Note: the FOL equivalent uses variables:
  $p(x) \leftrightarrow \exists y($hasChild$(x, y) \wedge$ Male$(y))$,
  $p(x) \leftrightarrow \exists y($hasChild$(x, y) \wedge \exists x :$ hasChild$(x, y))$.
- Exercise: show why unrestricted existential quantification $\exists R.C$ in contrast to $\exists R.\top$ leads to branching.
- $\mathcal{N}$: (unqualified) cardinalities of roles ("number restrictions").
  $(\geq 3$ hasChild.$\top)$ for persons who have at least 3 children.
- $\mathcal{Q}$: qualified role restrictions:
  $(\leq 2$ hasChild.Male$)$
  $\mathcal{F}$: like $\mathcal{Q}$, but restricted to cardinalites 0, 1 and "arbitrary".

COMPLEXITY AND DECIDABILITY: OVERVIEW

- Logic $\mathcal{L}^2$, i.e., FOL with only two (reusable) variable symbols is decidable.
- Full FOL is undecidable.
- DLs: incremental, modular set of semantical notions.
- only part of FOL is required for concept reasoning.
- $\mathcal{ALC}$ can be *expressed* by FOL, but then, the inherent semantics is lost $\rightarrow$ full FOL reasoner required.
- Actually, $\mathcal{ALC}$ can be encoded in FOL by only using two variables $\rightarrow$ $\mathcal{ALC}$ is decidable.
- Consistency checking of $\mathcal{ALC}$-TBoxes and -ABoxes is PSPACE-complete (proof by reduction to *Propositional Dynamic Logic* which is in turn a special case of propositional multimodal logics).
  There are algorithms that are efficient in the average case.
- $\mathcal{ALCN}$ goes beyond $\mathcal{L}^2$ and PSPACE. Reduction to $\mathcal{C}^2$ (including "counting" quantifiers) yields decidability, but now in NEXPTIME. There are algorithms for $\mathcal{ALCN}$ and even $\mathcal{ALCQ}$ in PSPACE.

Further Extensions

- Role hierarchy ($\mathcal{H}$; role subsumption and role equivalence, union/intersection of roles):
  hasSon $\sqsubseteq$ hasChild , hasChild $\equiv$ hasSon $\sqcup$ hasDaughter
- *Role Constructors* similar to regular expressions:
  concatenation (hasGrandchild $\equiv$ hasChild $\circ$ hasChild), transitive closure (hasDescendant $\equiv$ hasChild$^+$) (indicated by e.g. $\mathcal{ALCH}_{reg}$), and inverse (isChildOf $\equiv$ hasChild$^-$) ($\mathcal{I}$).
- *Data types* (indicated by "(D)"), e.g. integers.
  Adult $\equiv$ Person $\sqcap$ $\exists$age. ≥ 18.
- *Nominals* ($\mathcal{O}$) allow to use individuals from the ABox also in the TBox.
  Enumeration Concepts: BeNeLux $\equiv$ {Belgium, Netherlands, Luxemburg},
  HasValue-Concepts: GermanCity $\equiv$ $\exists$inCountry.Germany.
- Role-Value-Maps:
  Equality Role-Value-Map: $(R_1 \equiv R_2)(x) \;\Leftrightarrow\; \forall y : R_1(x,y) \leftrightarrow R_2(x,y)$.
  Containment Role-Value-Map: $(R_1 \sqsubseteq R_2)(x) \equiv \forall y : R_1(x,y) \rightarrow R_2(x,y)$.
  (knows $\sqsubseteq$ likes) describes the set of people who like all people they know;
  i.e., (knows $\sqsubseteq$ likes)(John) denotes that John likes all people he knows.

Formal Semantics of Expressions

- $I(\geq nR.C) = \{x \mid \#\{y \mid (x,y) \in I(R)$ and $y \in I(C)\} \geq n\}$,
- $I(\leq nR.C) = \{x \mid \#\{y \mid (x,y) \in I(R)$ and $y \in I(C)\} \leq n\}$,
- $I(nR.C) = \{x \mid \#\{y \mid (x,y) \in I(R)$ and $y \in I(C)\} = n\}$,
- $I(R \sqcup S) = I(R) \cup I(S),\;\; I(R \sqcap S) = I(R) \cap I(S)$,
- $I(R \circ S) = \{(x,z) \mid \exists y : (x,y) \in I(R)$ and $(y,z) \in I(S)\}$,
- $I(R^-) = \{(y,x) \mid (x,y) \in I(R)\}$,
- $I(R^+) = (I(R))^+$.
- If nominals are used, $\mathcal{I}$ also assigns an element $I(x) \in \mathcal{D}$ to each nominal symbol x (similar to constant symbols in FOL). With this,
  $I(\{x_1,\ldots,x_n\}) = \{I(x_1),\ldots,I(x_n)\}$, and
  $I(R.y) = \{x \mid \{z \mid (x,z) \in I(R)\} = \{y\}\}$,
- $I(R_1 \equiv R_2) = \{x \mid \forall y : R_1(x,y) \leftrightarrow R_2(x,y)\}$,
  $I(R_1 \sqsubseteq R_2) = \{x \mid \forall y : R_1(x,y) \rightarrow R_2(x,y)\}$.

Complexity of Extensions

- Role constructors: $\mathcal{ALC}_{reg}$, including transitivity, composition and union is ExpTime-complete; this stays the same when inverse roles and even cardinalities for *atomic* roles are added ($\mathcal{ALCQI}_{reg}$).
  Recall that inverse and transitive closure are important for ontologies.
- Combining such *composite* roles with cardinalities becomes undecidable (encoding in FOL requires 3 variables).
- Encoding of Role-Value Maps with composite roles in FOL is undecidable (encoding in FOL requires 3 variables; the logic loses the *tree model property*).
- $\mathcal{ALCQI}_{reg}$ with role-value maps restricted to boolean compositions of *basic* roles remains decidable. Decidability is also preserved when role-value-maps are restricted to functional roles.

Description Logic Model Theory

The definition is the same as in FOL:

- an interpretation is a model of an ABox A if
  - for every atomic concept C and individual x such that $C(x) \in A$, $I(x) \in I(C)$, and
  - for every atomic role R and individuals x, y such that $R(x, y) \in A$, $(I(x), I(y)) \in I(R)$.
- note: the interpretation of the non-atomic concepts and roles is given as before,
- all axioms ϕ of the TBox are satisfied, i.e., $\mathcal{I} \models \phi$.

Based on this, DL entailment is also defined as before:

- a set Φ of formulas entails another formula Ψ (denoted by $\Phi \models \psi$), if $\mathcal{I}(\Psi) =$ true in all models $\mathcal{I}$ of Φ.

Decidability, Complexity, and Algorithms

Many DLs are decidable, but in high complexity classes.

- decidability is due to the fact that often *local* properties are considered, and the verification proceeds tree-like through the graph without connections between the branches.
- This locality does not hold for cardinalities over composite roles, and for role-value maps – these lead to undecidability.
- Reasoning algorithms for $\mathcal{ALC}$ and many extensions are based on tableau algorithms, some use model checking (finite models), others use tree automata.

Three types of Algorithms

- restricted (to polynomial languages) and complete
- expressive logics with complete, worst-case EXPTIME algorithms that solve realistic problems in "reasonable" time. (Fact, Racer, Pellet)
- more expressive logics with incomplete reasoning.

Example

- Given facts: Person $\equiv$ Male $\sqcup$ Female and Person(unknownPerson).
- Query ?-Male(X) yields an empty answer
- Query ?-Female(X) yields an empty answer
- Query ?-(Male $\sqcup$ Female)(X) yields unknownPerson as an answer
- for query answering, *all* models of the TBox+ABox are considered.
- in some models, the unknownPerson is Male, in the others it is female.
- in all models it is in (Male $\sqcup$ Female).

Summary and Comparison with FOL

Base Data (DL atomic concepts and atomic roles $\sim$ RDF)

- unary predicates (concepts/classes): Person(John),
- binary predicates (roles/properties): hasChild(John,Alice)

Expressions

Concept/Role Expressions act as unary/binary predicates:

- ($\exists$ hasChild.Male)(John), (Adult $\sqcap$ Parent)(John),
- (hasChild $\circ$ hasChild)(Jack,Alice), (neighbor*)(Portugal,Germany)

$\Rightarrow$ disjunction, conjunction and quantifiers *only* in the restricted contexts of expressions

$\Rightarrow$ implications *only* in the restricted contexts of TBox Axioms:

- $C_1 \sqsubseteq C_2$ Parent $\sqsubseteq$ Person
- $C_1 \equiv C_2$ Parent $\equiv$ $\exists$hasChild.$\top$
- $R_1 \sqsubseteq R_2$ capital $\sqsubseteq$ hasCity
- $R_1 \equiv R_2$ neighbor $\equiv$ (neighbor $\sqcup$ neighbor$^-$)

$\Rightarrow$ ABox/TBox (=database) is a conjunctive set of atoms.

$\Rightarrow$ No formulas!