OWL Formal Semantics

Chapter 5.

Mads Opheim & Nicolay Hvidsten

NTNU
Norwegian University of
Science and Technology

OWL description logics

Description logics are usually fragments of first-order predicate logic, and its notation is called **ALC**.

ALC consists of *classes*, *roles*, and *individuals*.

rainer(ashKetchum)

hasAffiliation(ashKetchum, palletTown)

hasAffiliation is an abstract role.

Pikachu $\sqsubseteq$ Pokémon (subclass)

Pocket Monster $\equiv$ Pokémon (equivalence)

Charizard $\sqsubseteq$ (Pokémon$\sqcap$FireType)$\sqcup$(Pokémon$\sqcap$FlyingType)

is the same as

```
<owl:Class rdf:about="Charizard">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="Pokémon"/>
          <owl:Class rdf:about="FireType"/>
        </owl:intersectionOf>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="Pokémon"/>
          <owl:Class rdf:about="FlyingType"/>
          </owl:intersectionOf>
      </owl:unionOf>
    </owl:Class>
  </owl:subClassOf>
</owl:Class>
```

$Gym \sqsubseteq \forall hasTrainer.GymTrainer$ states that all trainers in a gym must be gym trainers, and corresponds to using owl:allValuesFrom.

$Gym \sqsubseteq \exists hasTrainer.GymTrainer$ states that every gym must have at least one trainer who is a gym trainer, and corresponds to using owl:someValuesFrom

We can use ALC to model parts of OWL

$\bot \equiv C \sqcap \neg C$ represents the empty class owl:Nothing

$\top \equiv C \sqcup \neg C$ represents owl:Thing

and so on.

NTNU
Norwegian University of
Science and Technology

ALC syntax

Statements in ALC are divided into two groups: TBox and ABox statements.

TBox: schema knowledge - $C \equiv D$ etc.

ABox: assertional knowledge about instances / individuals - C(a), R(a,b) (same as RDFS)

Not all OWL DL can be expressed in ALC, thus we need to extend ALC to the description logic SHOIN(D) which encompasses ALC and also further expressive means.

NTNU
Norwegian University of
Science and Technology

SHOIN(D) - cardinality

SHOIN(D) provides cardinality restrictions

Trainer $\sqsubseteq \leq 8$hasBadge

states that each trainer has at most eight badges, which corresponds to owl:maxCardinality

NTNU
Norwegian University of
Science and Technology

SHOIN(D) - relationships between individuals

$\{a\} \equiv \{b\}$ - equivalence

$\{a\} \sqcap \{b\} \sqsubseteq \bot$ - inequal

roles

Using role inclusion axioms, $R \sqsubseteq S$ means that R is a subrole of S.

datatypes

SHOIN(D) allows the usage of data values in the second argument of concrete roles and closed classes - also concrete domains, but this is not part of the OWL standard.

NTNU
Norwegian University of
Science and Technology

In addition to ALC SHOIN(D) provides

ALC = Attributive Langugae with Complement

S = ALC plus role transitivity

H = role hierarchies

O = nominals

I = inverse roles

N = cardinality restrictions

D = datatypes

. . .

(F = functionality)

(Q = qualifie cardinality restrictions)

(R=generalized role inclusion)

(E = existential role restrictions)

OWL DL corresponds to SHOIN(D)

NTNU
Norwegian University of
Science and Technology

F

Role functionality can be expressed through cardinality restrictions.

Q

$\leq$nR.C as opposed to $\leq$nR - which is a generalization of cardinality restrictions already present in SHOIN(D). This makes for more concise logical statements, and this is why description logic literature usually concerns itself with SHIQ rather than SHIN.

R R1 $\circ$ ... $\circ$ Rn $\sqsubseteq$ R means that the concatenation of R1, ..., Rn is a subrole of R

E Sub-Boolean description logics contained in ALC

NTNU
Norwegian University of
Science and Technology

Correspondence between OWL variants and description logics

OWL Full - is not a description logic

OWL DL - SHOIN(D)

OWL Lite - SHIF(D)

OWL 2 Full - is not a description logic

OWL 2 DL - SROIQ(D)

OWL 2 EL - EL++

OWL 2 QL - DL-Lite

OWL 2 RL - DLP

NTNU
Norwegian University of
Science and Technology

SROIQ

RBox: roles

TBox: schema

ABox: assertional

RBox Based on a set **R** of atomic roles which contains all role names, inverse of role names and the universal role **U**. U is a super role of all other roles and inverse roles.

NTNU
Norwegian University of
Science and Technology

Generalized role inclusion axiom: $S1 \circ ... \circ S_n \sqsubseteq R$ A set of such axioms is called a generalized role hierarchy. This hierarchy is regular if there exists a strict partial order ($\prec$) on R such that $S \prec R$ if and only if $S_- \prec R$

Not regular

$\text{hasParent} \circ \text{hasHusband} \sqsubseteq \text{hasFather}$ $\text{hasFather} \sqsubseteq \text{hasParent}$

This is not regular because regularity would force that hasParent $\prec$ hasFather, as well as hasFather $\prec$ asParent - which is impossible because $\prec$ must be strict, i.e. hasParent and hasFather cannot be equal.

As such, regular role hierarchies cannot have equal roles.

NTNU
Norwegian University of
Science and Technology

Reflexivity: Everything is related to itself by this role: isIdenticalTo

Antisymmetry: Whenever a is related to b via R then b is not related to a via R: isParent

Disjointness: The roles do not share any pair of instances: hasParent and hasChild are disjoint, while hasParent and hasFather are not.

The **RBox** itself is a set of role characteristics and a role hierarchy.

NTNU
Norwegian University of
Science and Technology

Knowledge base

The classes within the RBox contain the exact same language constructs as SHOIN(D), with the exception of $\exists$S.Self - an individual a is an instance of this expression if a is related to itself via the S role.

PersonCommittingSuicide $\sqsubseteq \exists$kills.Self

TBox: a set of class inclusion axioms on the form $C \sqsubseteq D$ where C and D are class expressions.

ABox: a set of individual assignments in the predicate-logic form: C(a), R(a,b) etc. Where C is a class expression and R is a role expression.

NTNU
Norwegian University of
Science and Technology

SROIQ allows for negated role assignment as well:

$\neg$hasTrainer(Ash, Raichu)

means that Ash is not the trainer of Raichu.

The **knowledge base** itself is the union of a regular RBox **R**, an ABox and a TBox for **R**.

NTNU
Norwegian University of
Science and Technology

SHIQ

This particular description logic is of particular importance for research around OWL, even though it is no more complicated than ALC. It's also close to encompass OWL DL, as it only requires nominals to accomplish this. However, SHOIN (which is essentially OWL DL) is a lot more complex.

For this reason when research into reasoning issues around OWL, methods and algorithms are often first developed for ALC and then lifted to SHIQ before even attempting to modifiy them for SHOIQ.

NTNU
Norwegian University of
Science and Technology

SHIQ is defined by restricting SROIQ - SHIQ RBoxes are RIOQ Rboxes restricted to axioms of the form R $\circ$ R $\sqsubseteq$ R (Trans(R)), R$^-$ $\sqsubseteq$ R (Sym(R)) and S $\sqsubseteq$ R, and as such regularity does not need to be imposed for SHIQ.

This simplifies the definition for roles: a role is simple unless it is transitive, its inverse is transitive, it has a transitive subrole, or its inverse has a transitive subrole.

TBoxes are like SROIQ TBoxes where Self and nominals such as $\{a\}$ are not allowed.

ABoxes are like SROIQ Aboxes where $\neg$ is not allowed and where inequality of individuals may be explicitly stated (a $\neq$ b)

NTNU
Norwegian University of
Science and Technology

Model-theoretic semantics of OWL

For OWL 2 DL (and everything else using SROIQ)

Mainly similar to that of RDF(S).

Interpretation

- set I of symbols for individuals
- set C of symbols for class names
- set R of symbols for roles

Map individuals to elements of the domain, classnames to subsets and roles to binary domain relations.

The sets are mutually disjoint.

Functions

Elementary set functions you already know ($\emptyset$, $\neg$, and, or, exists, all)

Quantifiers are a bit more interesting:

$\exists R.C$ - those things connected via R to something in C

You get the idea of $\forall R.C$ from that (every y connected with x via R is in C)

NTNU
Norwegian University of
Science and Technology

Example of an interpretation and mapping

$$\Delta = a, r$$

$$I_I(Student) = r$$

$$I_I(Friend) = a$$

$$I_C(Professor) = \{r\}$$

$$I_C(FacultyMember) = \{r\}$$

$$I_R(areFriends) = \{(r, a)\}$$

NTNU
Norwegian University of
Science and Technology

What is a model?

"Models capture the structure of a knowledge base in the sense that they give a truthful representation of the axioms in terms of sets"

An interpretation of a knowledge base is a model of it if
$C(a) \epsilon K \rightarrow a^I \epsilon C^I$
same with $R(a, b) \rightarrow (a^I, b^I) \epsilon R^I$ - and negated
$C \sqsubseteq D \rightarrow C^I \subseteq D^I$ and a few more.

NB: you see that we have other types of axioms than just triples (e.g. Student(Mads)). Remember RDF(S)...

NTNU
Norwegian University of
Science and Technology

Logical consequences

One knowledge base can have several models. Each of these is a different view on or realisation of the KB. But it can also add additional non-general relationships.

The logical consequences of a KB are that common to all its possible models.

Definitions

Satisfiable/consistent

versus

unsatisfiable/contradictory/inconsistent

- both for KB and class expressions.

First-order predicate logic - with equality

Same shit, different wrapping.

FOPL extended with $=, T, \bot$

A SROIQ-KB can be translated to FOPLWE through a series of simple translation rules. See page 180, and you will get it. E.g.
$\pi_x(\neg C) = \neg\pi_x(C)$ or $\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$

Hence - and this is fancy - SROIQ is a fragment (subset) of FOPL

NTNU
Norwegian University of
Science and Technology

Automated reasoning with OWL

Our logical consequences-notion is computationally horrible. But we can just take algorithms from predicate logic and adapt them to our setting.

Tableaux algorithms is state-of-the-art for this.

Refresh from chapter 4

Interesting inference types in the OWL-world:

— Subsumption - C subclass of D?

— Class equivalence - $C \equiv D$?

— Class disjointness - $C \sqcap D \sqsubseteq \bot$?

— Global consistency - show that it has a model.

NTNU
Norwegian University of
Science and Technology

Refresh from chapter 4 - continued

— Class consistency - $C \sqsubseteq \bot$ is not a logical consequence of the KB

— Instance checking - Individual a belongs to C if C(a) is a logical cons. of KB

— Instance retrieval - check each instance: do you belong to C?

Many types, one solution

These can be reduced to each other, we reduce them to *does the KB have at least one model?*[1] which is grat for tableaux.

You can consider this conversion as a challenge (or see page 184) - it's really just simple rewriting.

E.g. instance checking iff $K \cup \{\neg C(a)\}$ is unsatisfiable.

Note that this statement is outside the ABox limitations we talked about earlier, but this can be solved through a technique called ABox reduction.

[1] We recall that this is satisfiability reformulated

NTNU
Norwegian University of
Science and Technology

So easy?

Well, no... We could now convert into predicate logic and do automated reasoning with predicate logic techniques, but this is inefficient (obviously?).

SROIQ and its description logic is decidable, FOPL is not. Thus, DL reasoning termination cannot be guaranteed with tecniques for FOPL-reasoning.

The book and we decide to ignore the first problem. We will address the second through *blocking* later on.

Negation normal form

NNF is a rewriting of the KB. It is equivalent with FOPL.

Think DeMorgan.

$KB \equiv NNF(KB)$

Tableaux algorithms

Determines if a KB is satisfiable.

A tableau is a structured way of deriving and representing logical consequences of a KB.

A TA tries to construct a generic representation of a model. Success or failure in this determines satisfiability.

TA for ALC - content

What does a TA consist of?

- A set of name-labeled nodes
- Directed edges between two nodes
- For each node, a set of class memberships - L(a)
- For each node pair, a set of role names

TA for ALC - initial tableau

Start with a trivial tableau:

$L(a, b) = \emptyset$

$C(a)\ \epsilon\ K \Rightarrow L(a) \leftarrow C$ (or $\emptyset$, if no C)

$R(a, b)\ \epsilon\ K \Rightarrow L(a, b) \leftarrow R$

TA for ALC - algorithm

Apply rules from a defined transformation rule set until a contradiction is found or no rules can be applied.

The rule set transforms from $\cap, \cup, \exists$ and $\forall$ The algorithm chooses rule direction arbitrarily (and thus nondeterministically). The $\cup$-function also chooses nondeterministically - is a a member of C or D? (Especially if also $C \not\sqcap D$ Hence, to know whether all choices leads to contradiction (or success), we will have to backtrack to each such choice and choose the other.

TBox-rule: If C is a TBox statement not in L(x), $(Lx) \leftarrow C$.

NTNU
Norwegian University of
Science and Technology

Blocking

A node x is directly blocked by node y if x is a variable, y is one of its ancestors and $L(x) \subseteq L(y)$.

TA will now only use transformation rules if the node is not blocked.

TA for SHIQ

The central ideas same as for SROIQ.

Consists of same as TA-ALC, plus two (symmetrical) relations between nodes $\approx$ *and* $\not\approx$ Also, role names can be inversed.

Same initial tableau, plus $a \neq b \rightarrow a \not\approx b$ and $\approx$ as the empty relation.

Blocking for SHIQ

Extends direct blocking to consider pairs, not single nodes: (x,x1) repeats (y,y1).

Indirectly blocked if an ancestor is blocked or there is no node y with $L(x, y) \neq \emptyset$

TA for SHIQ

Same rules as for ALC, plus some more:

transitive subroles

choose-rule

$\geq$-rule

$\leq$-rule

$\leq$-root-rule

Complexity

. . . are worst-case ugly, average-case quite ok.