Part V

The Semantic Web

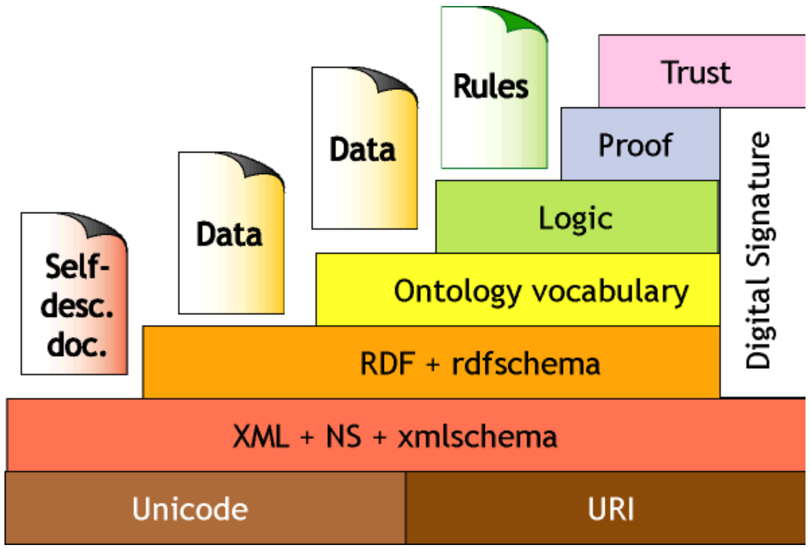
Contents

Semantic Technologies

- Semantics = meaning
- Semantic Technologies = technologies building on (formalized) meaning
- Declarative Programming as a semantic technology
  - A procedure definition describes its intended meaning
    - e.g. `intersect(L1, L2) :- member(X, L1), member(X, L2).`
      Lists `L1` and `L2` intersect
      if there exists an `X`, which is a member of both `L1` and `L2`.
  - The execution of a program can be viewed as a process of deduction
- The main goal of the Semantic Web (SW) approach:
  - make the information on the web processable by computers
  - machines should be able to understand the web, not only read it
- Achieving the vision of the Semantic Web
  - Add (computer processable) meta-information to the web
  - Formalize background knowledge – build so called ontologies
  - Develop reasoning algorithms and tools

The vision of the Semantic Web

- The Semantic Web layer cake – Tim Berners-Lee



The Semantic Web

- The goal: making the information on the web processable by computers
- Achieving the vision of the Semantic Web
  - Add meta-information to web pages, e.g.
    (*AIT* hasLocation *Budapest*)
    (*AIT* hasTrack *Track:Foundational-courses*)
    (*Track:Foundational-courses* hasCourse *Semantic-and-declarative...*)
  - Formalise background knowledge – build so called terminologies
    - hierarchies of notions, e.g.
      a *University* is a (subconcept of) *Inst-of-higher-education*,
      the hasFather relationship is a special case of hasParent
    - definitions and axioms, e.g.
      a *Father* is a *Male Person* having at least one child
  - Develop reasoning algorithms and tools
- Main topics
  - Description Logic, the maths behind the Semantic Web is the basis of Web Ontology Languages OWL 1 & 2 (W3C standards)
  - A glimpse at reasoning algorithms for Description Logic

Contents

First Order Logic (recap)

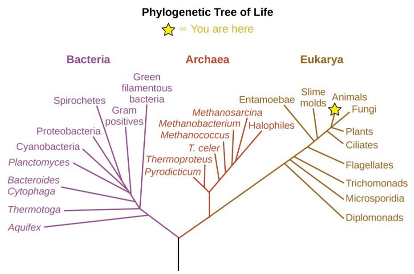
- Syntax:
  - non-logical ("user-defined") symbols: predicates and functions, including constants (function symbols with 0 arguments)
  - terms (refer to individual elements of the universe, or interpretation), e.g. *fatherOf*(*Susan*)
  - formulas (that hold or do not hold in a given interpretation), e.g. $\varphi = \forall x.(\mathit{Optimist}(\mathit{fatherOf}(x)) \rightarrow \mathit{Optimist}(x))$
- Semantics:
  - determines if a closed formula φ is true in an interpretation $\mathcal{I}$: $\mathcal{I} \models \varphi$ (also read as: $\mathcal{I}$ is a model of φ)
  - an interpretation $\mathcal{I}$ consists of a domain Δ and a mapping from non-logical symbols (e.g. *Optimist*, *fatherOf*, *Susan*) to their meaning
  - semantic consequence: $S \models \alpha$ means: if an interpretation is a model of all formulas in the set S, then it is also a model of α (note that the symbol $\models$ is overloaded)
- Deductive system (also called proof procedure): an algorithm to deduce a consequence α of a set of formulas S: $S \vdash \alpha$
  - example: resolution

Soundness, completeness and decidability (recap)

- A deductive system is **sound** if $S \vdash \alpha \Rightarrow S \models \alpha$ (deduces only truths).
- A deductive system is **complete** if $S \models \alpha \Rightarrow S \vdash \alpha$ (deduces all truths).
- Resolution is a sound and complete deductive system for FOL
- Kurt Gödel was first to show such a system: Gödel's completeness theorem: there is a sound and complete deductive system for FOL
- FOL is not decidable: no decision procedure for the question "does S imply α ($S \vdash \alpha$)?" (Gödel's completeness theorem ensures that if the answer is "yes", then there exists a proof of α from S; but if the answer is "no", we have no guarantees – this is called semi-decidability)
- Developers of the Semantic Web strive for using decidable languages
  - for languages with a sound and complete proof procedure
- Semantic Web languages are based on Description Logics, which are decidable sublanguages of FOL, i.e. there is an algorithm that delivers a yes or no answer to the question "does S imply α"

Ontologies

- **Ontology**: computer processable description of knowledge
- Early ontologies include classification system (biology, medicine, books)



- Entities in the Web Ontology Language (OWL):
  - classes – describe sets of objects (e.g. optimists)
  - properties (attributes, slots) – describe binary relationships (e.g. has parent)
  - objects – correspond to real life objects (e.g. people, such as Susan, her parents, etc.)

Knowledge Representation

- **Natural Language**:
  1. Someone having a non-optimist friend is bound to be an optimist.
  2. Susan has herself as a friend.
- **First order Logic** (unary predicate, binary predicate, constant):
  1. $\forall x.(\exists y.(\mathrm{hasFriend}(x,y) \land \neg \mathrm{opt}(y)) \rightarrow \mathrm{opt}(x))$
  2. $\mathrm{hasFriend}(\mathrm{Susan}, \mathrm{Susan})$
- **Description Logics** (concept, role, individual):
  1. $(\exists \mathrm{hasFriend}.\neg\,\mathrm{Opt}) \sqsubseteq \mathrm{Opt}$ (GCI – Gen. Concept Inclusion axiom)
  2. $\mathrm{hasFriend}(\mathrm{Susan}, \mathrm{Susan})$ (role assertion)
- **Web Ontology Language** (Manchester syntax)[5] (class, property, object):
  1. `(hasFriend some (not Opt)) SubClassOf: Opt`
     Those having `some not` Opt friends must be Opt
     (GCI – Gen. Class Inclusion axiom)
  2. `hasFriend(Susan,Susan)` (object property assertion)

[5] protegeproject.github.io/protege/class-expression-syntax

A sample ontology to be entered into Protégé

1. There is a class of Animals, some of which are Male, some are Female.
2. No one can be both Male and Female.
3. There are Animals that are Human.
4. There are Humans who are Optimists.
5. There is a relationship hasP meaning "has parent". Relations hasFather and hasMother are sub-relations (special cases) of hasP.
6. Let's define the class C1 as those who have an optimistic parent.
7. State that everyone belonging to C1 is Optimistic.
8. State directly that anyone having an Optimistic parent is Optimistic.
9. There is a relation hasF, denoting "has friend". State that someone having a non-Optimistic friend must be Optimistic.
10. There are individuals: Susan, and her parents Mother and Father.
11. Mother has Father as her friend.

The sample ontology in Description Logic and OWL/Protégé

	English	Description Logic	OWL (Manchester syntax)
1	Male is a subclass of Animal.	$\mathrm{Male} \sqsubseteq \mathrm{Animal}$	Male `SubClassOf:` Animal
	Female is a subclass of Animal.	$\mathrm{Female} \sqsubseteq \mathrm{Animal}$	Female `SubClassOf:` Animal
2	Male and Female are disjoint.	$\mathrm{Male} \sqsubseteq \neg\,\mathrm{Female}$	Male `DisjointWith:` Female
3	Human is a subclass of Animal.	$\mathrm{Human} \sqsubseteq \mathrm{Animal}$	Human `SubClassOf:` Animal
4	Optimist is a subclass of Human.	$\mathrm{Opt} \sqsubseteq \mathrm{Human}$	Opt `SubClassOf:` Human
5	hasFather is a subprop. of hasP.	$\mathrm{hasFather} \sqsubseteq \mathrm{hasP}$	hasFather `SubPropertyOf:` hasP
	hasMother is a subprop. of hasP.	$\mathrm{hasMother} \sqsubseteq \mathrm{hasP}$	hasMother `SubPropertyOf:` hasP
6	C1 = those having an Opt parent.	$\mathrm{C1} \equiv \exists\, \mathrm{hasP} . \mathrm{Opt}$	C1 `EquivalentTo:` hasP `some` Opt
7	Everyone in C1 is Opt.	$\mathrm{C1} \sqsubseteq \mathrm{Opt}$	C1 `SubClassOf:` Opt
8	Children of Opt parents are Opt.	$\exists\, \mathrm{hasP} . \mathrm{Opt} \sqsubseteq \mathrm{Opt}$	hasP `some` Opt `SubClassOf:` Opt
9	Those with a non-Opt friend are Opt.	$\exists\, \mathrm{hasF} . \neg\mathrm{Opt} \sqsubseteq \mathrm{Opt}$	hasF `some not` Opt `SubClassOf:` Opt
10	Susan has parents Mother and Father.	hasP(Susan, Mother) hasP(Susan, Father)	hasP(Susan, Mother) hasP(Susan, Father)
11	Mother has Father as a friend.	hasF(Mother, Father)	hasF(Mother, Father)

(In Protégé, select the "save as" format as "Latex syntax" to obtain DL notation.)

Contents

Description Logic (DLs) – overview

DL, a subset of FOL, is the mathematical background of OWL

- Signature – relation and function symbols allowed in DL
  - concept name (A) – unary predicate symbol (cf. OWL class)
  - role name (R) – binary predicate symbol (cf. OWL property)
  - individual name ($a, \ldots$) – constant symbol (cf. OWL object)
  - No non-constant function symbols, no preds of arity > 2, no vars
- Concept names and concept expressions represent sets, e.g. ∃hasParent.Optimist – the set of those who have an optimist parent
- Terminological axioms (TBox) – stating background knowledge
  - A simple axiom using the DL language $\mathcal{ALE}$: ∃hasParent.Optimist ⊑ Optimist – the set of those who have an optimist parent is a subset of the set of optimists
  - Translation to FOL: $\forall x.(\exists y.(hasP(x, y) \wedge Opt(y)) \rightarrow Opt(x))$
- Assertions (ABox) – stating facts about individual names
  - Example: Optimist(JACOB), hasParent(JOSEPH, JACOB)
- A consequence of these TBox and ABox axioms is: Optimist(JOSEPH)
- DLs behind OWL 1 and OWL 2 are decidable: there are bounded time algorithms for checking if a set of axioms implies a statement.

Some further examples of terminological axioms

(1) A Mother is a Person, who is a Female and who has(a)Child.

$$\text{Mother} \equiv \text{Person} \sqcap \text{Female} \sqcap \exists \text{hasChild}.\top$$

(2) A Tiger is a Mammal.

$$\text{Tiger} \sqsubseteq \text{Mammal}$$

(3) Children of an Optimist Person are Optimists, too.

$$\text{Optimist} \sqcap \text{Person} \sqsubseteq \forall \text{hasChild}.\text{Optimist}$$

(4) Childless people are Happy.

$$\forall \text{hasChild}.\bot \sqcap \text{Person} \sqsubseteq \text{Happy}$$

(5) Those in the relation hasChild are also in the relation hasDescendant.

$$\text{hasChild} \sqsubseteq \text{hasDescendant}$$

(6) The relation hasParent is the inverse of the relation hasChild.

$$\text{hasParent} \equiv \text{hasChild}^{-}$$

(7) The hasDescendant relationship is transitive.

$$\textbf{Trans}(\text{hasDescendant})$$

Description Logics – why the plural?

- These logic variants were progressively developed in the last two decades
- As new constructs were proved to be "safe", i.e. keeping the logic decidable, these were added
- We will start with the very simple language $\mathcal{AL}$, extend it to $\mathcal{ALE}$, $\mathcal{ALU}$ and $\mathcal{ALC}$
- As a side branch we then define $\mathcal{ALCN}$
- We then go back to $\mathcal{ALC}$ and extend it to languages $\mathcal{S}$, $\mathcal{SH}$, $\mathcal{SHI}$ and $\mathcal{SHIQ}$ (which encompasses $\mathcal{ALCN}$)
- We briefly tackle further extensions $\mathcal{O}$, (**D**) and $\mathcal{R}$
- OWL 1, published in 2004, corresponds to $\mathcal{SHOIN}$(**D**)
- OWL 2, published in 2012, corresponds to $\mathcal{SROIQ}$(**D**)

Contents

Overview of the $\mathcal{ALCN}$ language

- In $\mathcal{ALCN}$ a statement (axiom) can be
  - a subsumption (inclusion), e.g. Tiger $\sqsubseteq$ Mammal, or
  - an equivalence, e.g. Woman $\equiv$ Female $\sqcap$ Person, Mother $\equiv$ Woman $\sqcap\ \exists$hasChild.$\top$
- In general, an $\mathcal{ALCN}$ axiom can take these two forms:
  - subsumption: $C \sqsubseteq D$
  - equivalence: $C \equiv D$, where C and D are concept expressions
- A concept expression C denotes a set of objects (a subset of the Δ universe of the interpretation), and can be:
  - an atomic concept (or concept name), e.g. Tiger, Female, Person
  - a composite concept, e.g. Female $\sqcap$ Person, $\exists$hasChild.Female
  - composite concepts are built from atomic concepts and atomic roles (also called role names) using some constructors (e.g. $\sqcap$, $\sqcup$, $\exists$, etc.)
- We first introduce language $\mathcal{AL}$, that allows a minimal set of constructors (all examples on this page are valid $\mathcal{AL}$ concept expressions)
- Next, we discuss richer extensions named $\mathcal{U}$, $\mathcal{E}$, $\mathcal{C}$, $\mathcal{N}$

The syntax of the $\mathcal{AL}$ language

Language $\mathcal{AL}$ (Attributive Language) allows the following concept expressions, also called concepts, for short:

A is an atomic concept, C, D are arbitrary (possibly composite) concepts
R is an atomic role

DL concept	OWL class	Name	Informal definition
A	A (class name)	atomic concept	those in A
$\top$	`owl:Thing`	top	the set of all objects
$\bot$	`owl:Nothing`	bottom	the empty set
$\neg A$	`not` A	atomic negation	those not in A
$C \sqcap D$	C `and` D	intersection	those in both C and D
$\forall R.C$	R `only` C	value restriction	those whose all Rs belong to C
$\exists R.\top$	R `some owl:Thing`	limited exist. restr.	those having at least one R

Examples of $\mathcal{AL}$ concept expressions:

Person $\sqcap \neg$Female	Person `and not` Female
Person $\sqcap\ \forall$hasChild.Female	Person `and` (hasChild `only` Female)
Person $\sqcap\ \exists$hasChild.$\top$	Person `and` (hasChild `some owl:Thing`)

The semantics of the $\mathcal{AL}$ language (as a special case of FOL)

- An interpretation $\mathcal{I}$ is a mapping:
  - $\Delta^{\mathcal{I}} = \Delta$ is the universe, the **nonempty** set of all individuals/objects
  - for each concept/class name A, $A^{\mathcal{I}}$ is a (possibly empty) subset of Δ
  - for each role/property name R, $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ is a binary relation on Δ
- The semantics of $\mathcal{AL}$ extends $\mathcal{I}$ to composite concept expressions, i.e. describes how to "calculate" the meaning of arbitrary concept exprs:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta | \forall b.(\langle a, b \rangle \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}})\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta | \exists b.\langle a, b \rangle \in R^{\mathcal{I}}\}
\end{aligned}$$

- Finally we define how to obtain the truth value of an axiom:

$$\mathcal{I} \models C \sqsubseteq D \quad \text{iff} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$
$$\mathcal{I} \models C \equiv D \quad \text{iff} \quad C^{\mathcal{I}} = D^{\mathcal{I}}$$

The $\mathcal{AL}$ language: limitations

Recall the elements of the language $\mathcal{AL}$:

DL concept	OWL class	Name	Informal definition
A	A (class name)	atomic concept	those in A
$\top$	`owl:Thing`	top	the set of all objects
$\bot$	`owl:Nothing`	bottom	the empty set
$\neg A$	`not` A	atomic negation	those not in A
$C \sqcap D$	C `and` D	intersection	those in both C and D
$\forall R.C$	R `only` C	value restriction	those whose all Rs belong to C
$\exists R.\top$	R `some` `owl:Thing`	limited exist. restr.	those having at least one R

What is missing from $\mathcal{AL}$?

- We can specify the intersection of two concepts, but not the union, e.g. those who are either blue-eyed or tall.
- $\exists R.\top$ – we cannot describe e.g. those having a female child. Remedy: allow for full exist. restr., e.g. $\exists$hasCh.*Female*
- $\neg A$ – negation can be applied to atomic concepts only. Remedy: full negation, $\neg C$, where C can be non-atomic, e.g. $\neg(U \sqcap V)$

The $\mathcal{ALCN}$ language family: extensions $\mathcal{U}$, $\mathcal{E}$, $\mathcal{C}$, $\mathcal{N}$

Further concept constructors, OWL equivalents shown in [`square brackets`]:

- Union: $C \sqcup D$, [C `or` D] – those in either C or D
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \qquad (\mathcal{U})$$
- Full existential restriction: $\exists R.C$, [R `some` C] – those who have at least one R belonging to C
$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.\langle a, b\rangle \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \qquad (\mathcal{E})$$
- (Full) negation: $\neg C$, [`not` C] – those who do not belong to C
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (\mathcal{C})$$
- Number restrictions (unqualified): $(\geqslant nR)$, [R `min` n `owl:Thing`] and $(\leqslant nR)$, [R `max` n `owl:Thing`] – those who have at least n R-s, or have at most n R-s
$$(\geqslant n\,R)^{\mathcal{I}} = \{\, a \in \Delta^{\mathcal{I}} \mid \ |\{b \mid \langle a, b\rangle \in R^{\mathcal{I}}\}| \geq n \,\} \qquad (\mathcal{N})$$
$$(\leqslant n\,R)^{\mathcal{I}} = \{\, a \in \Delta^{\mathcal{I}} \mid \ |\{b \mid \langle a, b\rangle \in R^{\mathcal{I}}\}| \leq n \,\}$$
Note that qualified number restrictions, such as $(\geqslant n\,R.C)$ (e.g., those having at least 3 blue-eyed children) are not covered by this extension
- E.g.: Person $\sqcap$ (($\leqslant$ 1 hasCh) $\sqcup$ ($\geqslant$ 3 hasCh)) $\sqcap$ $\exists$hasCh.Female
Person `and` (hasCh `max 1 or` hasCh `min 3`) `and` (hasCh `some` Female)

Summary table of the $\mathcal{ALCUEN}$ language

DL	OWL	Name	Informal definition	
A	A	atomic concept	those in A	$\mathcal{AL}$
$\top$	`owl:Thing`	top	the set of all objects	$\mathcal{AL}$
$\bot$	`owl:Nothing`	bottom	the empty set	$\mathcal{AL}$
$C \sqcap D$	C `and` D	intersection	those in both C and D	$\mathcal{AL}$
$\forall R.C$	R `only` C	value restriction	those whose all Rs belong to C	$\mathcal{AL}$
$\neg C$	`not` C	full negation	those not in C	$\mathcal{C}$
$C \sqcup D$	C `or` D	union	those in either C or D	$\mathcal{U}$
$\exists R.C$	R `some` C	existential restr.	those with an R belonging to C	$\mathcal{E}$
$(\leqslant nR)$	R `max` n `o:T`	unq. numb. restr.	those having at most n Rs	$\mathcal{N}$
$(\geqslant nR)$	R `min` n `o:T`	unq. numb. restr.	those having at least n Rs	$\mathcal{N}$

Rewriting $\mathcal{ALCN}$ to first order logic

- Concept expressions map to predicates with one argument, e.g.
Tiger $\Longrightarrow$ Tiger(x) Mammal $\Longrightarrow$ Mammal(x)
Person $\Longrightarrow$ Person(x) Female $\Longrightarrow$ Female(x)
- Simple connectives $\sqcap$, $\sqcup$, $\neg$ map to boolean operations $\wedge$, $\vee$, $\neg$, e.g.
Person $\sqcap$ Female $\Longrightarrow$ Person(x) $\wedge$ *Female*(x)
Person $\sqcup$ $\neg$Mammal $\Longrightarrow$ Person(x) $\vee$ $\neg$Mammal(x)
- An axiom $C \sqsubseteq D$ is rewritten as $\forall x.(C(x) \rightarrow D(x))$, e.g.
Tiger $\sqsubseteq$ Mammal $\Longrightarrow$ $\forall x.$(Tiger(x) $\rightarrow$ Mammal(x))
- An axiom $C \equiv D$ is rewritten as $\forall x.(C(x) \leftrightarrow D(x))$, e.g.
Woman $\equiv$ Person $\sqcap$ Female $\Longrightarrow$ $\forall x.(\mathit{Woman}(x) \leftrightarrow \mathit{Person}(x) \wedge \mathit{Female}(x))$
- Concept constructors involving a quantifier $\exists$ or $\forall$ are rewritten to an appropriate quantified formula, where a role name is mapped to a binary predicate (a predicate with two arguments), e.g.

$\exists$hasParent.Opt$\sqsubseteq$Opt $\Longrightarrow$ $\forall x.(\exists y.$(hasParent(x, y) $\wedge$ Opt(y)) $\rightarrow$ Opt(x))

Rewriting $\mathcal{ALCN}$ to first order logic, example

- Consider $C = \text{Person} \sqcap ((\leqslant 1\, \text{hasCh}) \sqcup (\geqslant 3\, \text{hasCh})) \sqcap \exists \text{hasCh}.\text{Female}$
- Let's outline a predicate $C(x)$ which is true when x belongs to concept C:

$$C(x) \leftrightarrow \quad Person(x) \wedge (hasAtMost1Child(x) \vee hasAtLeast3Children(x)) \wedge hasFemaleChild(x)$$

- Class practice:
  - Define the FOL predicates $hasAtMost1Child(x)$, $hasAtLeast3Children(x)$, $hasFemaleChild(x)$
  - Additionally, define the following FOL predicates:
    - $hasOnlyFemaleChildren(x)$, corresponding to the concept $\forall \text{hasCh}.\text{Female}$
    - $hasAtMost2Children(x)$, corresponding to the concept $(\leqslant 2\, \text{hasCh})$

General rewrite rules $\mathcal{ALCN} \rightarrow$ FOL

Each concept expression can be mapped to a FOL formula:

- Each concept expression C is mapped to a formula $\Phi_C(x)$ (expressing that x belongs to C).
- Atomic concepts (A) and roles (R) are mapped to unary and binary predicates $A(x)$, $R(x, y)$.
- $\sqcap$, $\sqcup$, and $\neg$ are transformed to their counterpart in FOL ($\wedge, \vee, \neg$), e.g. $\Phi_{C \sqcap D}(x) = \Phi_C(x) \wedge \Phi_D(x)$
- Mapping further concept constructors:

$$\begin{aligned}
\Phi_{\exists R.C}(x) &= \exists y.\, (R(x, y) \wedge \Phi_C(y)) \\
\Phi_{\forall R.C}(x) &= \forall y.\, (R(x, y) \rightarrow \Phi_C(y)) \\
\Phi_{\geqslant n\, R}(x) &= \exists y_1, \ldots, y_n.\, \left(R(x, y_1) \wedge \cdots \wedge R(x, y_n) \wedge \bigwedge_{i<j} y_i \neq y_j \right) \\
\Phi_{\leqslant n\, R}(x) &= \forall y_1, \ldots, y_{n+1}.\, \left(R(x, y_1) \wedge \cdots \wedge R(x, y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j \right)
\end{aligned}$$

Equivalent languages in the $\mathcal{ALCN}$ family

- Language $\mathcal{AL}$ can be extended by arbitrarily choosing whether to add each of $\mathcal{UECN}$, resulting in $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{C}][\mathcal{N}]$.
  Do these $2^4 = 16$ languages have different expressive power?
  Two concept expressions are said to be equivalent, if they have the same meaning, in all interpretations.
  Languages $\mathcal{L}_1$ and $\mathcal{L}_2$ have the same expressive power ($\mathcal{L}_1 \stackrel{e}{=} \mathcal{L}_2$), if any expression of $\mathcal{L}_1$ can be mapped into an equivalent expression of $\mathcal{L}_2$, and vice versa.
- As a preparation for discussing the above let us recall that these axioms hold in all models, for arbitrary concepts C and D and role R:

$$\begin{aligned}
C \sqcup D &\equiv \neg(\neg C \sqcap \neg D) \\
\exists R.C &\equiv \neg \forall R.\neg C
\end{aligned}$$

$$\begin{aligned}
\neg\neg C &\equiv C \\
\neg \top &\equiv \bot \\
\neg \bot &\equiv \top \\
\neg(C \sqcap D) &\equiv \neg C \sqcup \neg D \\
\neg \exists R.\top &\equiv \forall R.\bot \\
\neg \forall R.C &\equiv \exists R.\neg C
\end{aligned}$$

Equivalent languages in the $\mathcal{ALCN}$ family

Let us show that $\mathcal{ALUE}$ and $\mathcal{ALC}$ are equivalent:

- As $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.C \equiv \neg \forall R.\neg C$, union and full existential restriction can be eliminated by using (full) negation. That is, to each $\mathcal{ALUE}$ concept expression there exists an equivalent $\mathcal{ALC}$ expression.
- The other way, each $\mathcal{ALC}$ concept can be transformed to an equivalent $\mathcal{ALUE}$ expression, by moving negation inwards, until before atomic concepts, and removing double negation; using the axioms from the right hand column on the previous slide
- Thus $\mathcal{ALUE}$ and $\mathcal{ALC}$ have the same expressive power, and so have the intermediate languages:
  $\mathcal{ALC}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCU}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCE}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALCUE}(\mathcal{N}) \stackrel{e}{=} \mathcal{ALUE}(\mathcal{N})$.

Further remarks:

- As $\mathcal{U}$ and $\mathcal{E}$ is subsumed by $\mathcal{C}$, we will use $\mathcal{ALC}$ to denote the language allowing $\mathcal{U}$, $\mathcal{E}$ and $\mathcal{C}$
- It can be shown that any two of $\mathcal{AL}, \mathcal{ALU}, \mathcal{ALE}, \mathcal{ALC}, \mathcal{ALN}, \mathcal{ALUN}, \mathcal{ALEN}, \mathcal{ALCN}$ have different expressive power

Contents

A special case of ontology: definitional TBox

- $\mathcal{T}_{fam}$: a sample definitional TBox for family relationships

$$
\begin{aligned}
\text{Woman} &\equiv \text{Person} \sqcap \text{Female}\\
\text{Man} &\equiv \text{Person} \sqcap \neg\text{Woman}\\
\text{Mother} &\equiv \text{Woman} \sqcap \exists\text{hasChild}.\text{Person}\\
\text{Father} &\equiv \text{Man} \sqcap \exists\text{hasChild}.\text{Person}\\
\text{Parent} &\equiv \text{Father} \sqcup \text{Mother}\\
\text{Grandmother} &\equiv \text{Woman} \sqcap \exists\text{hasChild}.\text{Parent}
\end{aligned}
$$

- A definitional TBox consists of equivalence axioms only, the left hand sides being distinct concept names (atomic concepts)
- The concepts on the left hand sides are called name symbols
- The remaining atomic concepts are called base symbols, e.g. in our example the two base symbols are Person and Female.
- In a definitional TBox the meanings of name symbols can be obtained by evaluating the right hand side of their definition

Interpretations and semantic consequence

Recall the definition of assigning a truth value to TBox axioms in an interpretation $\mathcal{I}$:

$$
\begin{aligned}
\mathcal{I} \models C \sqsubseteq D \quad &\text{iff} \quad C^{\mathcal{I}} \subseteq D^{\mathcal{I}}\\
\mathcal{I} \models C \equiv D \quad &\text{iff} \quad C^{\mathcal{I}} = D^{\mathcal{I}}
\end{aligned}
$$

Based on this we introduce the notion of "semantic consequence" exactly in the same way as for FOL

- We can naturally extend the above $\mathcal{I} \models \alpha$ notation – where α is either $C \sqsubseteq D$ or $C \equiv D$ – to a TBox (i.e. a set of α axioms) $\mathcal{T}$
  - $\mathcal{I} \models \mathcal{T}$ ($\mathcal{I}$ satisfies $\mathcal{T}$, $\mathcal{I}$ is a model of $\mathcal{T}$) iff for each $\alpha \in \mathcal{T}$, $\mathcal{I} \models \alpha$, i.e. $\mathcal{I}$ is a model of α
- We now overload even further the " $\models$ " symbol: $\mathcal{T} \models \alpha$ (read axiom α is a semantic consequence of the TBox $\mathcal{T}$) iff
  - all models of $\mathcal{T}$ are also models of α, i.e.
  - for all interpretations $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$ holds, then $\mathcal{I} \models \alpha$ also holds

TBox reasoning tasks

Reasoning tasks on TBoxes only (i.e. no ABoxes involved)

- A base assumption: the TBox is **consistent** (does not contain a contradiction), i.e. it has a model
- **Subsumption**: concept C is subsumed by concept D wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \sqsubseteq D)$, i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all $\mathcal{I}$ models of $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$)
  e.g. $\mathcal{T}_{fam} \models$ (Grandmother $\sqsubseteq$ Parent) (recall that $\mathcal{T}_{fam}$ is the family TBox)
- **Equivalence**: concepts C and D are equivalent wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \equiv D)$, i.e. $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds in all $\mathcal{I}$ models of $\mathcal{T}$ ($C \equiv_{\mathcal{T}} D$).
  e.g. $\mathcal{T}_{fam} \models$ (Parent $\equiv$ Person $\sqcap \exists$hasChild.Person)
- **Disjointness**: concepts C and D are disjoint wrt. a TBox $\mathcal{T}$, iff $\mathcal{T} \models (C \sqcap D \equiv \bot)$, i.e. $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ holds in all $\mathcal{I}$ models of $\mathcal{T}$.
  e.g. $\mathcal{T}_{fam} \models$ (Woman $\sqcap$ Man) $\equiv \bot$
- Note that all these tasks involve two concepts, C and D

Reducing reasoning tasks to testing satisfiability

- We now introduce a simpler, but somewhat artificial reasoning task: checking the satisfiability of a concept
- **Satisfiability**: a concept C is satisfiable wrt. TBox $\mathcal{T}$, iff there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is non-empty
  (hence C is non-satisfiable wrt. $\mathcal{T}$ iff in all $\mathcal{I}$ models of $\mathcal{T}$ $C^{\mathcal{I}}$ is empty)
- We will reduce each of the earlier tasks to checking non-satisfiability
- E.g. to prove: Woman $\sqsubseteq$ Person, let's construct a concept C that contains all counter-examples to this statement: $C =$ Woman $\sqcap \neg$Person
- If we can prove that C has to be empty, i.e. there are no counter-examples, then we have proven the subsumption
- Assume we have a method for checking satisfiability.
  Other tasks can be reduced to this method (usable in $\mathcal{ALC}$ and above):
  - C is subsumed by $D \iff C \sqcap \neg D$ is not satisfiable
  - C and D are equivalent $\iff (C \sqcap \neg D) \sqcup (D \sqcap \neg C)$ is not satisfiable
  - C and D are disjoint $\iff C \sqcap D$ is not satisfiable
- In simpler languages, not supporting full negation, such as $\mathcal{ALN}$, all reasoning tasks can be reduced to subsumption