Description Logics and First-Order Logic; Outlook

Till Mossakowski

WiSe 2009/10
A *DL-signature* $\Sigma = (C, R, I)$ consists of

- a set $C$ of concept names,
- a set $R$ of role names,
- a set $I$ of individual names,
For a signature $\Sigma = (C, R, I)$ the set of $\mathcal{ALC}$-concepts over $\Sigma$ is defined by the following grammar:

$$C ::= A \text{ for } A \in C$$

$$\vdash, \bot$$

$$\neg C$$

$$C \sqcap C$$

$$C \sqcup C$$

$$\exists R.C \text{ for } R \in R$$

$$\forall R.C \text{ for } R \in R$$

(Hets) Manchester syntax

a concept name

Thing

Nothing

not C

C and C

C or C

R some C

R only C

$\mathcal{ALC}$ stands for “attributive language with complement”
The set of $\mathcal{ALC}$-Sentences over $\Sigma$ ($\text{Sen}(\Sigma)$) is defined as

- $C \sqsubseteq D$, where $C$ and $D$ are $\mathcal{ALC}$-concepts over $\Sigma$.
  
  Class: $C$ SubclassOf: $D$

- $a : C$, where $a \in I$ and $C$ is a $\mathcal{ALC}$-concept over $\Sigma$.
  
  Individual: $a$ Types: $C$

- $R(a_1, a_2)$, where $R \in R$ and $a_1, a_2 \in I$.
  
  Individual: $a_1$ Facts: $R$ $a_2$
Given $\Sigma = (C, R, I)$, a $\Sigma$-model is of form $I = (\Delta^I, \cdot^I)$, where

- $\Delta^I$ is a non-empty set
- $A^I \subseteq \Delta^I$ for each $A \in C$
- $R^I \subseteq \Delta^I \times \Delta^I$ for each $R \in R$
- $a^I \in \Delta^I$ for each $a \in I$
We can extend \( \mathcal{I} \) to all concepts as follows:

\[
\begin{align*}
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
\neg C^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
(C \cap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \cup D)^\mathcal{I} &= C^\mathcal{I} \cup D^\mathcal{I} \\
\exists R.C)^\mathcal{I} &= \{ x \in \Delta^\mathcal{I} | \exists y \in \Delta^\mathcal{I}. (x, y) \in R^\mathcal{I}, y \in C^\mathcal{I} \} \\
(\forall R.C)^\mathcal{I} &= \{ x \in \Delta^\mathcal{I} | \forall y \in \Delta^\mathcal{I}. (x, y) \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I} \}
\end{align*}
\]
$\mathcal{I} \models C \sqsubseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$.

$\mathcal{I} \models a : C$ iff $a^\mathcal{I} \in C^\mathcal{I}$.

$\mathcal{I} \models R(a_1, a_2)$ iff $(a_1^\mathcal{I}, a_2^\mathcal{I}) \in R^\mathcal{I}$.
$\phi((C, R, I)) = (F, P)$ with

- $S = \{\text{Thing}\}$ (one sort = single-sorted)
- $F = \{a : \text{Thing} | a \in I\}$ (constants)
- $P = \{A : \text{Thing} | A \in C\} \cup \{R : \text{Thing} \times \text{Thing} | R \in R\}$ (predicate symbols)
Translating ALC to FOL: Concepts

- $\alpha_x(A) = A(x : \text{Thing})$
- $\alpha_x(\neg C) = \neg \alpha_x(C)$
- $\alpha_x(C \cap D) = \alpha_x(C) \land \alpha_x(D)$
- $\alpha_x(C \cup D) = \alpha_x(C) \lor \alpha_x(D)$
- $\alpha_x(\exists R.C) = \exists y : \text{Thing}.(R(x, y) \land \alpha_y(C))$
- $\alpha_x(\forall R.C) = \forall y : \text{Thing}.(R(x, y) \rightarrow \alpha_y(C))$
Sentence translation

- $\alpha_{\Sigma}(C \subseteq D) = \forall x : \text{Thing}. (\alpha_x(C) \rightarrow \alpha_x(D))$
- $\alpha_{\Sigma}(a : C) = \alpha_x(C)[a/x]^1$
- $\alpha_{\Sigma}(R(a, b)) = R(a, b)$

Model translation (FOL-models are translated to $\mathcal{ALC}$-models!)

- For $M' \in \text{Mod}^{\text{FOL}}(\phi_{\Sigma})$ define $\beta_{\Sigma}(M') := (\Delta, \cdot^l)$ with $\Delta = M'_{\text{Thing}}$ and $A^l = M'_A$, $a^l = M'_a$, $R^l = M'_R$.  

^1Replace $x$ by $a$. 

Translating ALC to FOL: Correctness

**Theorem 1:** \( C^I = \{ m \in M'_{\text{Thing}} | M' + \{ x \mapsto m \} \models \alpha_x(C) \} \)

**Proof:** By Induction over the structure of \( C \).

- \( A^I = M'_A = \{ m \in M'_{\text{Thing}} | M' + \{ x \mapsto m \} \models A(x) \} \)
- \( (\neg C)^I = \Delta \setminus C^I \)
  \[ = \text{I.H.} \quad \Delta \setminus \{ m \in M'_{\text{Thing}} | M' + \{ x \mapsto m \} \models \alpha_x(C) \} \]
  \[ = \{ m \in M'_{\text{Thing}} | M' + \{ x \mapsto m \} \models \neg \alpha_x(C) \} \]

**Theorem 2:** (Satisfaction condition)

\[ \beta(M) \models \varphi \text{ iff } M \models \alpha(\varphi) \]

**Theorem 3:** (Logical consequence coincides)

\[ \Gamma \models \varphi \text{ (in } ALC) \text{ iff } \alpha(\Gamma) \models \alpha(\varphi) \text{ (in } FOL) \]
Outlook
Beyond first-order logic

- **many-sorted logic** (variables, constants, predicates and functions have types)
  E.g.: $\forall n : \text{Nat} \ \forall l : \text{List} \ \text{head}(\text{cons}(n, l)) = n$

- **partial function logic**: $D(f(x))$ ("$f(x)$ is defined")

- **higher-order logic**: $\forall f : s \to t \ldots, \forall p : \text{Pred}(t) \ldots$
  $\forall u \forall v (\text{Path}(u, v) \leftrightarrow$
  $\forall R \ [\forall x \forall y \forall z (R(x, y) \land R(y, z) \rightarrow R(x, z))$
  $\land \forall x \forall y (\text{DirectWay}(x, y) \rightarrow R(x, y))]$
  $\rightarrow R(u, v)])$
Beyond first-order logic

- **many-sorted logic** (variables, constants, predicates and functions have types)
  E.g.: \( \forall n : \text{Nat} \; \forall l : \text{List} \; \text{head}(\text{cons}(n, l)) = n \)

- **partial function logic**: \( D(f(x)) \) ("\( f(x) \) is defined")

- **higher-order logic**: \( \forall f : s \to t \ldots, \forall p : \text{Pred}(t) \ldots \)
  \( \forall u \forall v (\text{Path}(u, v) \leftrightarrow \forall R \; \{[\forall x \forall y \forall z (R(x, y) \land R(y, z) \rightarrow R(x, z)) \land \forall x \forall y (\text{DirectWay}(x, y) \rightarrow R(x, y))] \rightarrow R(u, v)\}) \)
Beyond first-order logic

- **many-sorted logic** (variables, constants, predicates and functions have types)
  E.g.: $\forall n : \text{Nat} \; \forall l : \text{List} \; \text{head(cons}(n, l)) = n$

- **partial function logic**: $D(f(x))$ (“$f(x)$ is defined”)

- **higher-order logic**: $\forall f : s \to t \ldots, \forall p : \text{Pred}(t) \ldots$
  $\forall u \forall v (\text{Path}(u, v) \iff$
  $\forall R \quad \{ [\forall x \forall y \forall z (R(x, y) \land R(y, z) \to R(x, z))$
  $\land \forall x \forall y (\text{DirectWay}(x, y) \to R(x, y))]$
  $\to R(u, v)] \}$
modal logic:

\(\square P\) ("necessarily \(P\)"") and \(\Diamond P\) ("possibly \(P\)"")

Other readings of \(\square P\):
- It ought to be that \(P\)
- It is known that \(P\)
- It is provable that \(P\)
- Always \(P\) (temporal logic)
temporal logic: \( \square P \) ("always in the future, \( P \)"), \( \Diamond P \) ("sometimes in the future, \( P \)"), and \( P \) ("in the next step, \( P \)"

e.g. \( \square bank\_account > 0 \) (very unrealistic)
Further modal and temporal logics

- **temporal logic of actions (TLA):** $\Box [\text{state}' = f(\text{state})]_{\text{state}}$
  read: always in the future, either the state does not change, or the next state is $f$ applied to the previous state

- **dynamic logic:**
  $[p]P$ ("after every run of program $p$, $P$ holds")
  $\langle p \rangle P$ ("after some run of program $p$, $P$ holds")
Further modal and temporal logics

- **temporal logic of actions (TLA):** $\square[\text{state} = f(\text{state})]_{\text{state}}$
  read: always in the future, either the state does not change, or the next state is $f$ applied to the previous state

- **dynamic logic:**
  $[p]P$ ("after every run of program $p$, $P$ holds")
  $<p> P$ ("after some run of program $p$, $P$ holds")
More exotic modal logics

- **agent logics**, e.g. ATL: agents $A$ and $B$ have the possibility to make a telephone call, if they cooperate
- **logics for security**, e.g. ABLP: $A$ controls $P$ (“agent $A$ has the permission to perform action $P$”)
More exotic modal logics

- *agent logics*, e.g. ATL: agents $A$ and $B$ have the possibility to make a telephone call, if they cooperate
- *logics for security*, e.g. ABLP: $A$ *controls* $P$ ("agent $A$ has the permission to perform action $P$")
description logics, e.g. $\mathcal{ALC}$:

$\text{Elephant} \sqsubseteq \text{Mammal} \sqcap \exists \text{bodypart}. \text{Trunk} \sqcap \forall \text{color}. \text{Grey}$

abbreviates

$\forall x [\text{Elephant}(x) \leftrightarrow$

$(\text{Mammal}(x) \land \exists y (\text{bodypart}(x, y) \land \text{Trunk}(y))$

$\land \forall z (\text{color}(x, z) \rightarrow \text{Grey}(z))]$
Multi-valued logics

- three-valued logics: truth values are true, false, and undefined
- object constraint logic (OCL)
  (for UML — the unified modeling language)

  -- Managers get a higher salary than employees
  inv Branch2:
    self.employee->forall(e | e <> self.manager
    implies self.manager.salary > e.salary)
- **three-valued logics**: truth values are true, false, and undefined
- **object constraint logic (OCL)**
  (for UML — the unified modeling language)

  -- Managers get a higher salary than employees
  inv Branch2:
  self.employee->forall(e | e <> self.manager
  implies self.manager.salary > e.salary)
- **fuzzy logic**: truth values in the interval $[0, 1]$ correspond to different degrees of truth (e.g. Peter is quite tall, is tall, is very tall)
Even more exotic logics

- **paraconsistent logics**
  for databases, local inconsistency is o.k. and should not lead to global inconsistency

- **non-monotonic logics**
  new facts make previous arguments invalid, e.g.
  
  \[
  \begin{align*}
  &\text{Bird}(x) \vdash \text{CanFly}(x) \\
  &\{\text{Bird}(x), \text{Penguin}(x)\} \not\vdash \text{CanFly}(x)
  \end{align*}
  \]

- **linear logic** (resource-bounded logic)
  \[A \otimes A \vdash B\]
  (we can prove $B$ when we are allowed to use $A$ twice)
Even more exotic logics

- **paraconsistent logics**
  for databases, local inconsistency is o.k. and should not lead to global inconsistency

- **non-monotonic logics**
  new facts make previous arguments invalid, e.g.
  \[ Bird(x) \vdash CanFly(x) \]
  \[ \{ Bird(x), Penguin(x) \} \not\vdash CanFly(x) \]

- **linear logic** (resource-bounded logic)
  \[ A \otimes A \vdash B \]
  (we can prove \( B \) when we are allowed to use \( A \) twice)
Even more exotic logics

- **paraconsistent logics**
  for databases, local inconsistency is o.k. and should not lead to global inconsistency

- **non-monotonic logics**
  new facts make previous arguments invalid, e.g.
  \[ \text{Bird}(x) \vdash \text{CanFly}(x) \]
  \[ \{ \text{Bird}(x), \text{Penguin}(x) \} \not\vdash \text{CanFly}(x) \]

- **linear logic** (resource-bounded logic)
  \[ A \otimes A \vdash B \]
  (we can prove \( B \) when we are allowed to use \( A \) twice)
Why do we need so many logics?

- different aspects of the complex world / of software systems
- one “big” logic covering everything would be too clumsy
- good news: most of the logics are based on propositional or first-order logics
- most of the logics have central notions in common
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A *calculus* $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A *calculus* $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A *calculus* $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A calculus $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A calculus $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to (most of) these logics?

- A notion of *language* (or vocabulary of symbols, or signature)
- A syntax for *sentences*
- A notion of *model*
- A notion of *satisfaction*, i.e. $M \models P$ (read: “$M$ satisfies $P$”, or “$P$ holds in $M$”)
- A *calculus* $\mathcal{T} \vdash P$ (read “$P$ is provable from $\mathcal{T}$”)
What is common to all these logics? (cont’d)

- **logical consequence**: \( \mathcal{T} \models P \) iff for all models \( M \) with \( M \models \mathcal{T} \), also \( M \models P \)
- **logical validity**: \( \models P \) iff for all models \( M \), also \( M \models P \)
- **satisfiability**: \( \mathcal{T} \) is satisfiable iff there is some \( M \) with \( M \models \mathcal{T} \)
- **formal consistency**: \( \mathcal{T} \) is formally consistent iff \( \mathcal{T} \not\models P \) for some \( P \)
- **soundness of the calculus**: \( \mathcal{T} \vdash P \) implies \( \mathcal{T} \models P \)
- (sometimes) **completeness**: \( \mathcal{T} \models P \) implies \( \mathcal{T} \vdash P \)
What is common to all these logics? (cont’d)

- **logical consequence**: $\mathcal{T} \models P$ iff for all models $M$ with $M \models \mathcal{T}$, also $M \models P$
- **logical validity**: $\models P$ iff for all models $M$, also $M \models P$
- **satisfiability**: $\mathcal{T}$ is satisfiable iff there is some $M$ with $M \models \mathcal{T}$
- **formal consistency**: $\mathcal{T}$ is formally consistent iff $\mathcal{T} \not\vdash P$ for some $P$
- **soundness of the calculus**: $\mathcal{T} \vdash P$ implies $\mathcal{T} \models P$
- **(sometimes) completeness**: $\mathcal{T} \models P$ implies $\mathcal{T} \vdash P$
What is common to all these logics? (cont’d)

- **logical consequence**: \( \mathcal{T} \models P \) iff for all models \( M \) with \( M \models \mathcal{T} \), also \( M \models P \)
- **logical validity**: \( \models P \) iff for all models \( M \), also \( M \models P \)
- **satisfiability**: \( \mathcal{T} \) is satisfiable iff there is some \( M \) with \( M \models \mathcal{T} \)
- **formal consistency**: \( \mathcal{T} \) is formally consistent iff \( \mathcal{T} \not \vdash P \) for some \( P \)
- **soundness of the calculus**: \( \mathcal{T} \vdash P \) implies \( \mathcal{T} \models P \)
- **(sometimes) completeness**: \( \mathcal{T} \models P \) implies \( \mathcal{T} \vdash P \)
What is common to all these logics? (cont’d)

- **logical consequence:** $\mathcal{T} \models P$ iff for all models $M$ with $M \models \mathcal{T}$, also $M \models P$

- **logical validity:** $\models P$ iff for all models $M$, also $M \models P$

- **satisfiability:** $\mathcal{T}$ is satisfiable iff there is some $M$ with $M \models \mathcal{T}$

- **formal consistency:** $\mathcal{T}$ is formally consistent iff $\mathcal{T} \not\vdash P$ for some $P$

- **soundness of the calculus:** $\mathcal{T} \vdash P$ implies $\mathcal{T} \models P$

- **(sometimes) completeness:** $\mathcal{T} \models P$ implies $\mathcal{T} \vdash P$
What is common to all these logics? (cont’d)

- **Logical consequence**: $\mathcal{T} \models P$ iff for all models $M$ with $M \models \mathcal{T}$, also $M \models P$
- **Logical validity**: $\models P$ iff for all models $M$, also $M \models P$
- **Satisfiability**: $\mathcal{T}$ is satisfiable iff there is some $M$ with $M \models \mathcal{T}$
- **Formal consistency**: $\mathcal{T}$ is formally consistent iff $\mathcal{T} \not\vdash P$ for some $P$
- **Soundness of the calculus**: $\mathcal{T} \vdash P$ implies $\mathcal{T} \models P$
- **(Sometimes) completeness**: $\mathcal{T} \models P$ implies $\mathcal{T} \vdash P$
What is common to all these logics? (cont’d)

- **logical consequence**: \( \mathcal{T} \models P \) iff for all models \( M \) with \( M \models \mathcal{T} \), also \( M \models P \)
- **logical validity**: \( \models P \) iff for all models \( M \), also \( M \models P \)
- **satisfiability**: \( \mathcal{T} \) is satisfiable iff there is some \( M \) with \( M \models \mathcal{T} \)
- **formal consistency**: \( \mathcal{T} \) is formally consistent iff \( \mathcal{T} \not\models P \) for some \( P \)
- **soundness** of the calculus: \( \mathcal{T} \vdash P \) implies \( \mathcal{T} \models P \)
- (sometimes) **completeness**: \( \mathcal{T} \models P \) implies \( \mathcal{T} \vdash P \)
The central notions common to all logics can be axiomatized. Based on this meta-notion, multi-logic systems can be defined. In Bremen, we also develop multi-logic tools.
Next semester

CASL for software specification
Evaluation of this course

Please (anonymously) fill out the questionnaire and return it to me! (either now, or MZH 6. Ebene, Postfach 99)