Logik für Informatiker
Logic for computer scientists

Description Logics and First-Order Logic; Outlook

Till Mossakowski

WiSe 2009/10

A *DL-signature* $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ consists of

- a set $\mathbf{C}$ of concept names,
- a set $\mathbf{R}$ of role names,
- a set $\mathbf{I}$ of individual names,

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

		(Hets) Manchester syntax
$C ::=$	A for $A \in \mathbf{C}$	a concept name
	$\mid \top$	`Thing`
	$\mid \bot$	`Nothing`
	$\mid \neg C$	`not C`
	$\mid C \sqcap C$	`C and C`
	$\mid C \sqcup C$	`C or C`
	$\mid \exists R.C$ for $R \in \mathbf{R}$	`R some C`
	$\mid \forall R.C$ for $R \in \mathbf{R}$	`R only C`

$\mathcal{ALC}$ stands for "attributive language with complement"

Description Logic: Sentences

The set of $\mathcal{ALC}$-Sentences over Σ (Sen(Σ)) is defined as

- $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-concepts over Σ.

  ```
  Class: C SubclassOf: D
  ```

- $a : C$, where $a \in \mathbf{I}$ and C is a $\mathcal{ALC}$-concept over Σ.

  ```
  Individual: a Types: C
  ```

- $R(a_1, a_2)$, where $R \in \mathbf{R}$ and $a_1, a_2 \in \mathbf{I}$.

  ```
  Individual: a1 Facts: R a2
  ```

Description Logic: Models

Given $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$, a Σ-*model* is of form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}}$ is a non-empty set
- $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for each $A \in \mathbf{C}$
- $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for each $R \in \mathbf{R}$
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each $a \in \mathbf{I}$

Description Logic: semantics of concepts

We can extend $\cdot^{\mathcal{I}}$ to all concepts as follows:

$$
\begin{array}{lcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \exists y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} | \forall y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}
\end{array}
$$

Description Logic: Satisfaction of sentences in a model

$$
\begin{array}{lll}
\mathcal{I} \models C \sqsubseteq D & \text{iff} & C^{\mathcal{I}} \subseteq D^{\mathcal{I}}. \\
\mathcal{I} \models a : C & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}}. \\
\mathcal{I} \models R(a_1, a_2) & \text{iff} & (a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}.
\end{array}
$$

Translating ALC to FOL: Signatures

$\phi((\mathbf{C}, \mathbf{R}, \mathbf{I})) = (F, P)$ with

- $S = \{\mathtt{Thing}\}$ (one sort = single-sorted)
- $F = \{a : \mathtt{Thing} | a \in \mathcal{I}\}$ (constants)
- $P = \{A : \mathtt{Thing} | A \in \mathbf{C}\} \cup \{R : \mathtt{Thing} \times \mathtt{Thing} | R \in \mathbf{R}\}$ (predicate symbols)

Translating ALC to FOL: Concepts

- $\alpha_x(A) = A(x : \texttt{Thing})$
- $\alpha_x(\neg C) = \neg\alpha_x(C)$
- $\alpha_x(C \sqcap D) = \alpha_x(C) \wedge \alpha_x(D)$
- $\alpha_x(C \sqcup D) = \alpha_x(C) \vee \alpha_x(D)$
- $\alpha_x(\exists R.C) = \exists y : \texttt{Thing}.(R(x, y) \wedge \alpha_y(C))$
- $\alpha_x(\forall R.C) = \forall y : \texttt{Thing}.(R(x, y) \rightarrow \alpha_y(C))$

Translating ALC to FOL: Sentences and Models

Sentence translation

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

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

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

[1] Replace x by a.

Translating ALC to FOL: Correctness

Theorem 1: $C^{\mathcal{I}} = \left\{m \in M'_{\mathtt{Thing}} | M' + \{x \mapsto m\} \models \alpha_x(C)\right\}$
Proof: By Induction over the structure of C.

- $A^{\mathcal{I}} = M'_A = \left\{m \in M'_{\mathtt{Thing}} | M' + \{x \mapsto m\} \models A(x)\right\}$
- $(\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$
  $=^{I.H.} \Delta \setminus \{m \in M'_{\mathtt{Thing}} | M' + \{x \mapsto m\} \models \alpha_x(C)\}$
  $= \{m \in M'_{\mathtt{Thing}} | M' + \{x \mapsto m\} \models \neg\alpha_x(C)\}$

Theorem 2: (Satisfaction condition)

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

Theorem 3: (Logical consequence coincides)

$$\Gamma \models \varphi \text{ (in } \mathcal{ALC}\text{) iff } \alpha(\Gamma) \models \alpha(\varphi) \text{ (in FOL)}$$

Outlook

Beyond first-order logic

- *many-sorted logic* (variables, constants, predicates and functions have types)
  E.g.: $\forall n : Nat\ \forall l : List\ \ head(cons(n, l)) = n$
- *partial function logic*: $D(f(x))$ ("$f(x)$ is defined")
- *higher-order logic*: $\forall f : s \rightarrow t \ldots, \forall p : Pred(t) \ldots$

$$\begin{aligned}&\forall u \forall v (Path(u, v) \leftrightarrow \\ &\quad \forall R\ \ \{[\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z)) \\ &\qquad\quad \wedge \forall x \forall y (DirectWay(x, y) \rightarrow R(x, y))] \\ &\qquad\qquad\qquad \rightarrow R(u, v)\})\end{aligned}$$

Beyond first-order logic

- *many-sorted logic* (variables, constants, predicates and functions have types)
  E.g.: $\forall n : Nat\ \forall l : List\ head(cons(n, l)) = n$
- *partial function logic*: $D(f(x))$ ("$f(x)$ is defined")
- *higher-order logic*: $\forall f : s \rightarrow t \ldots, \forall p : Pred(t) \ldots$
  $$\forall u \forall v (Path(u, v) \leftrightarrow \forall R\ \{[\forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z)) \wedge \forall x \forall y (DirectWay(x, y) \rightarrow R(x, y))] \rightarrow R(u, v)\})$$

Beyond first-order logic

- *many-sorted logic* (variables, constants, predicates and functions have types)
  E.g.: $\forall n : Nat\ \forall l : List\ \ head(cons(n, l)) = n$
- *partial function logic*: $D(f(x))$ ("$f(x)$ is defined")
- *higher-order logic*: $\forall f : s \to t \ldots, \forall p : Pred(t) \ldots$

$$\begin{aligned}
&\forall u \forall v (Path(u, v) \leftrightarrow \\
&\quad \forall R\ \ \{[\forall x \forall y \forall z (R(x, y) \land R(y, z) \to R(x, z)) \\
&\qquad\qquad \land \forall x \forall y (DirectWay(x, y) \to R(x, y))] \\
&\qquad\qquad\qquad\qquad \to R(u, v)\})
\end{aligned}$$

Modal and temporal logics

- *modal logic*:
  $\Box P$ ("necessarily P") and $\Diamond P$ ("possibly P")
  Other readings of $\Box P$:
  It ought to be that P
  It is known that P
  It is provable that P
  Always P (temporal logic)

- *temporal logic*: $\Box P$ ("always in the future, P"), $\Diamond P$ ("sometimes in the future, P"), and P ("in the next step, P")
  e.g. $\Box bank\_account > 0$ (very unrealistic)

Further modal and temporal logics

- *temporal logic of actions (TLA)*: $\Box[state' = f(state)]_{state}$
  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")

Further modal and temporal logics

- *temporal logic of actions (TLA)*: $\Box[state' = f(state)]_{state}$
  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. $\mathcal{ALC}$:
  $Elephant \doteq Mammal \sqcap \exists bodypart.Trunk \sqcap \forall color.Grey$
  abbreviates
  $$\forall x[Elephant(x) \leftrightarrow (Mammal(x) \wedge \exists y(bodypart(x,y) \wedge Trunk(y)) \wedge \forall z(color(x,z) \rightarrow Grey(z)))]$$

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)
```

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)
```

Multi-valued logics (cont'd)

- *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.
  $Bird(x) \vdash CanFly(x)$
  $\{Bird(x), Penguin(x)\} \nvdash CanFly(x)$
- *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.
  $Bird(x) \vdash CanFly(x)$
  $\{Bird(x), Penguin(x)\} \not\vdash CanFly(x)$
- *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.
  $Bird(x) \vdash CanFly(x)$
  $\{Bird(x), Penguin(x)\} \not\vdash CanFly(x)$
- *linear logic* (resource-bounded logic)
  $A \otimes A \vdash B$
  (we can prove B when we are allowed to use A twice)

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
  $\mathcal{T} \nvdash P$ 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
  $\mathcal{T} \nvdash P$ 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
  $\mathcal{T} \nvdash P$ 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
  $\mathcal{T} \nvdash P$ 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
  $\mathcal{T} \nvdash P$ 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 $\mathcal{T} \vdash P$

Multi logic systems

- 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 questionaire and return it to me!
(either now, or MZH 6. Ebene, Postfach 99)