

4 Beschreibungslogiken (DLs) – Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL (durch Studer)
- 4.7 Ontologien (durch Studer)

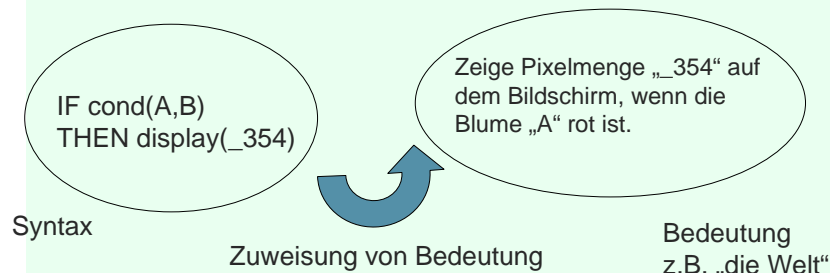
4.0 Allgemeine Motivation

- Intelligente Informationssysteme benötigen (u.A.)
 - Wissensrepräsentation und -verarbeitung (Künstliche Intelligenz)
 - IT Wissensmanagement
 - Automatische Deduktion (Logik)
- Aktuelles Buzzword: **Semantik**
Semantische Technologien
Semantic Web

Was ist das eigentlich, Semantik?

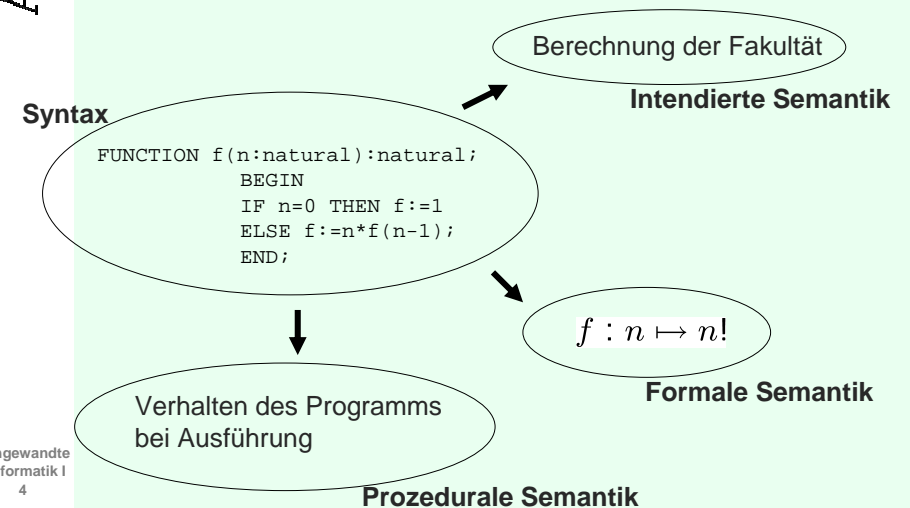
4.0 Syntax und Semantik

Syntax: Bedeutungsleere Zeichenfolgen
Semantik: Bedeutung der Syntax

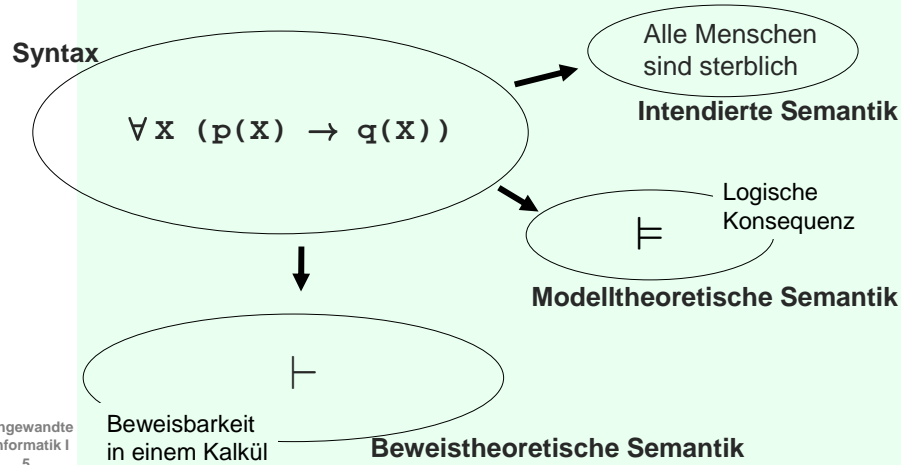


Wie kann man *Semantik* für maschinelle Verarbeitung formalisieren?

4.0 Was ist Semantik? Beispiel Programmiersprache



4.0 Semantik von Logik/Wissensrepräsentationssprachen



4.0 Beschreibungslogiken (DLs)

Beschreibungslogiken sind Formalismen zur Wissensrepräsentation

- Logikbasiert
 - logikbasierte Semantik
 - automatische Deduktion
- Ausdrucksstark für komplexes Wissen
- Schlank genug für Anwendbarkeit
- Grundlage für Ontologiesprache OWL (Web Ontology Language, W3C Standard April 2004)
- Kerntechnologie für das Semantic Web
- Grundlage für Semantisches Wissensmanagement in Unternehmen

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

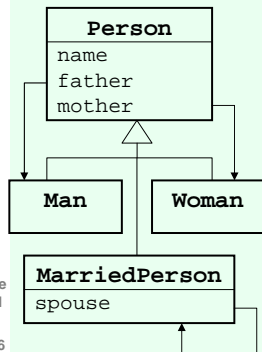
- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
 - 4.1.1 Vorteile von Modellen mit formaler Semantik
 - 4.1.2 Architektur von DL-basierten Applikationen
 - 4.1.3 Fokus und Zielsetzung
 - 4.1.4 Historie
- 4.2 DL Syntax und Semantik
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.1 Einführung in DLs

- Beschreibungslogiken (Description Logics, kurz DLs) sind eine **Familie von Formalismen** für explizite und implizite **Repräsentation** von **strukturiertem Wissen**.
- Beschreibungslogiken **vereinheitlichen** und **reichern** u.a. die traditionellen Frame-basierten, netzwerkartigen und Objekt-orientierten Modellierungssprachen **mit formaler Semantik an**.

4.1 Einführung in DLs

- DLs sind **nicht Diagramm-basiert** sondern **Logik-basiert** und haben deshalb **logikbasierte formale Semantik**
 - Disambiguierung der Bedeutung!



Was bedeutet es, dass die Klasse `MarriedPerson` als `spouse` ebenfalls eine `MarriedPerson` hat?
D.h., was ist die Bedeutung von Quantoren?

- \exists Für jede `MarriedPerson` existiert **irgendeine** `MarriedPerson` die ein Ehegatte ist
- \forall **Jeder** Ehegatte einer `MarriedPerson` ist ebenfalls `MarriedPerson`
- C **Nur eine** `MarriedPerson` **kann** einen Ehegatten haben, der ebenfalls `MarriedPerson` sein muss.

4.1.1 Vorteile von Modellen mit formaler Semantik

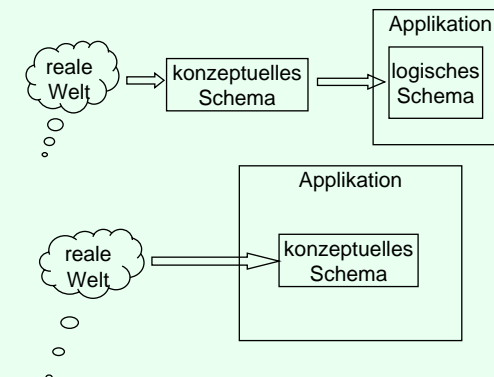
- Disambiguierte Bedeutung
 - Interoperabilität
 - Austausch von Wissen zwischen Partnern
 - Zusammenführen von Wissensbasen einfach
 - Eindeutige Schlussfolgerungen
 - Bedeutung impliziten Wissens eindeutig
- Nachweis von Eigenschaften ermöglicht durch mathematische Formalisierung
 - Algorithmische Komplexität des Schlußfolgerns
 - Korrektheit von Datenkonversionen
 - Verifikation von Algorithmen

4.1.1 Vorteile von Modellen mit formaler Semantik

- Automatisches Schließen
 - Wie kann man **implizites Wissen** repräsentieren?
 - Aus X Vater von Y und Z Bruder von X möchte man automatisch schließen können, dass Z Onkel von Y ist.
 - Ist das Modell **konsistent**?
 - gibt es eine nichtleere Menge von Individuen, die die im Modell beschriebenen Bedingungen erfüllen.
 - Beispiel: Bezeichnen wir Personen unter 12 Jahre als Kinder und nehmen an, dass alle Studenten erwachsen sind, dann führt „Student ist Unterklasse von Kind“ zu einem inkonsistenten Modell, weil die Klasse Student keine Objekte haben kann.

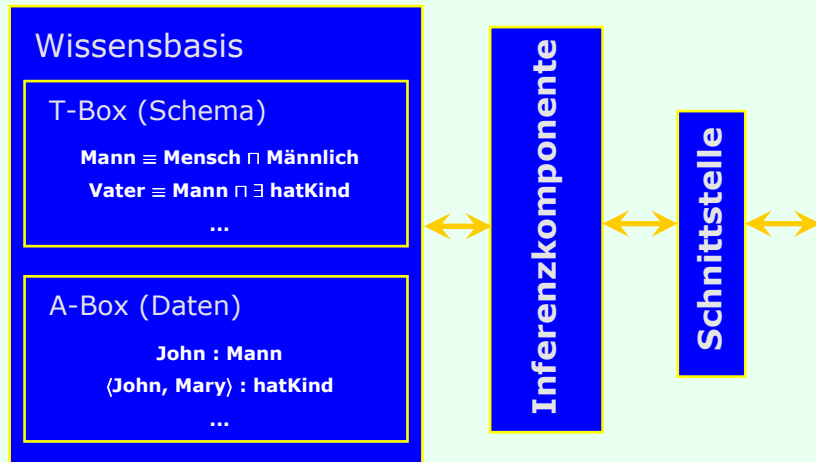
4.1.1 Vorteile von Modellen mit formaler Semantik

- Der Übersetzungsschritt vom konzeptionellen Modell zum ausführbaren Modell ist nicht mehr nötig.



- Stichwort: *Deklarativität*

4.1.2 Architektur von DL-basierten Applikationen



4.1.3 Fokus und Zielsetzung

- Je **ausdrucksmächtiger** die Logik **desto schwieriger** wird das automatische Schließen.
 - Komplexität steigt
 - Manche Probleme sind sogar unentscheidbar
- **Ziel** ist ein **gesundes Gleichgewicht** zu finden. D.h. eine möglichst **ausdrucksmächtige** Logik, die für wichtige Probleme **entscheidbares** und möglichst **effizientes** automatisches Schließen anbietet.

4.1.4 Historie

- 70er Jahre: die ersten Formalismen für Wissensrepräsentation. Grob zwei Arten von Formalismen
 - Logik-basierte und nicht Logik-basierte
- Logik-basierte Formalismen waren von Anfang an sehr mächtig und universal benutzbar, da sie auf Logik erster Stufe basierten.
- Nicht Logik-basierte Formalismen wurden zwar für spezielle Anwendungen entwickelt, evolvierten aber zu universal einsetzbaren Formalismen -> Semantische Netzwerke und Frames.

4.1.4 Historie

- Nicht Logik-basierte Formalismen waren für die Praxis zunächst attraktiv, führten aber dazu, dass die darauf basierenden Systeme sich für dieselben Strukturen unterschiedlich verhielten -> Notwendigkeit von formaler Semantik für die Strukturen.
- 1979: FOL Semantik von Frames [Hayes]
- 1985: Die Erkenntnis, dass Frames and semantische Netze als Fragmente von FOL betrachtet werden können [Brachman, Levesque].
- 1985: Beginn von Forschung im Bereich Beschreibungslogiken erst unter dem Namen „terminologische Systeme“ und dann „Konzeptsprachen“.
- Ende 90er: Erste praktikable Algorithmen zum Schließen über Beschreibungslogiken.
- ca. 2000: Beschreibungslogiken finden Anwendung in Wissensmanagement, Künstliche Intelligenz, Semantic Web.
- April 2004: W3C Standard OWL DL (= *SHOIN(D)*)

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: *ALC*
 - 4.2.1 *ALC* TBox Syntax
 - 4.2.2 *ALC* TBox Semantik
 - 4.2.3 *ALC* ABox Syntax und Semantik
 - 4.2.4 Erweiterungen von *ALC*
 - 4.2.5 Modellierungsbeispiele
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.2 DL Syntax und Semantik

- Beschreibungslogiken sind eine Familie von Logiken.
 - Es gibt nicht *die* Beschreibungslogik, sondern **viele verschiedene Beschreibungslogiken**.
- In DL werden mittels sog. **Konstruktoren** aus einfachen Modellen komplexere Modelle gebaut.
- Verschiedene Beschreibungslogiken **unterscheiden sich in der Menge der Konstruktor** (**Ausdrucksächtigkeit**), die sie anbieten.

Konstruktoren

- Konstruktor ermöglichen den **Aufbau** von **komplexeren Konzepten aus** weniger komplexeren bzw. **atomaren Konzepten**.
- Welche Arten von Konstruktor es gibt, hängt von der konkreten DL ab.
- Die meisten DLs bieten jedoch
 - Konjunktion (\sqcap), Disjunktion (\sqcup), Negation (\neg)
 - Existenzquantor (\exists) und Allquantor (\forall)

4.2.1 *ALC* TBox Syntax

- *ALC* ist die **einfachste** Beschreibungslogik
- Atomare Typen
 - Konzept- (oder Klassen-) name A, B, ... und folgende zwei spezielle Konzepte
 - \perp Bottom Konzept
 - \top Top oder universelles Konzept
 - Rollennamen R, S, ...
- Konstruktor
 - $\neg C$ (Negation)
 - $C \sqcap D$ (Konjunktion)
 - $C \sqcup D$ (Disjunktion)
 - $\exists R.C$ (Existenzquantor)
 - $\forall R.C$ (Allquantor)

4.2.1 *ALC* Atomare Typen

- Konzeptname / Klassenname
 - Beispiele: Student, Mitarbeiter, Professor, Vorlesung, Studiengang
 - vergleichbar mit einer Klasse in UML und einem Entitätstyp in ER.
- Rollenname
 - zweistellige Prädikate (verbinden zwei Klassen oder Individuen)
 - Beispiele: Student *besucht* Vorlesung, Mitarbeiter *betreut* Vorlesung, Professor *hält* Vorlesung
 - vergleichbar mit Assoziation in UML und Beziehungstyp in ER

4.2.1 *ALC* Klassenbeziehungen

- Klassen können auf zwei Arten miteinander in Beziehung gesetzt werden:
 - Inklusion
 $C \sqsubseteq D$
 - z.B. Man \sqsubseteq Human
 - Gleichheit
 $C \equiv D$
 - z.B. Frau \equiv Woman
 - $C \equiv D$ ist gleichbedeutend damit, dass *beide* Aussagen $C \sqsubseteq D$ und $D \sqsubseteq C$ gelten

4.2.1 *ALC* Konstruktoren

- Negation von C bedeutet intuitiv „alles ausser C“
 - Mann $\equiv \neg$ Frau
- Konjunktion von C und D bedeutet intuitiv „sowohl C als auch D“
 - Touchscreen = Eingabegerät \sqcap Ausgabegerät
- Disjunktion von C und D bedeutet intuitiv „C oder D“
 - UniAngestellte = Mitarbeiter \sqcup Professor

4.2.1 *ALC* Konstruktoren

- $\exists R.C$ (Existenzquantor) bedeutet intuitiv „alle x, die eine Beziehung vom Typ R mit einem y vom Typ C haben“
 - \exists besucht.Vorlesung
- $\forall R.C$ (Allquantor) bedeutet intuitiv „alle x, für die alle Beziehungen vom Typ R mit einem y vom Typ C sind“
 - \forall hatElternteil.Mensch
 - hat ein Individuum gar keine Beziehung vom Typ R, erfüllt es immer die Bedingung $\forall R.C$.

4.2.1 \mathcal{ALC} Klassenbeziehungen

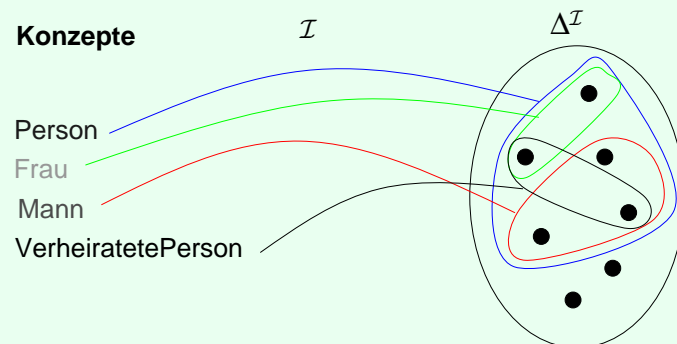
- Klassenbeziehungen können für zusammengesetzte Klassen verwendet werden:
 - Inklusion
 $C \sqsubseteq D$
 - z.B. $\text{Man} \sqsubseteq \text{Human} \sqcap \text{Male}$
 - Gleichheit
 $C \equiv D$
 - z.B. $\text{Frau} \equiv \text{Human} \sqcap \text{Female} \sqcap \text{Adult}$
 - z.B. $\text{Orphan} \equiv \text{Human} \sqcap \neg \text{hasParent} \sqcap \text{Dead}$
- Terminologisches Wissen (sog. **TBox**) besteht aus einer Menge solcher Klassenbeziehungen (sog. *Axiome*).

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
 - 4.2.1 \mathcal{ALC} TBox Syntax
 - 4.2.2 \mathcal{ALC} TBox Semantik
 - 4.2.3 \mathcal{ALC} ABox Syntax und Semantik
 - 4.2.4. Erweiterungen von \mathcal{ALC}
 - 4.2.5. Modellierungsbeispiele
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.2.2 \mathcal{ALC} TBox Semantik

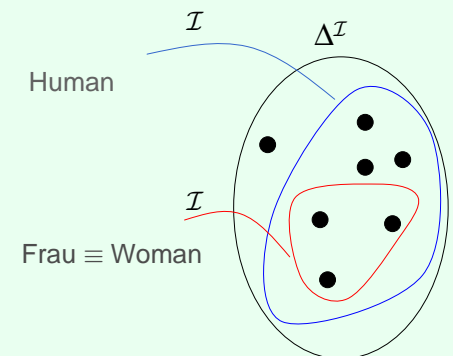
- Semantik basiert auf Interpretationen $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, wobei $\Delta^{\mathcal{I}}$ eine nichtleere Menge ist (die Domäne) und $\cdot^{\mathcal{I}}$ jeden Konzeptnamen A auf eine Teilmenge $A^{\mathcal{I}}$ von $\Delta^{\mathcal{I}}$ abbildet. Es gilt $\perp^{\mathcal{I}} = \emptyset$ und $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$.



4.2.2 \mathcal{ALC} TBox Semantik

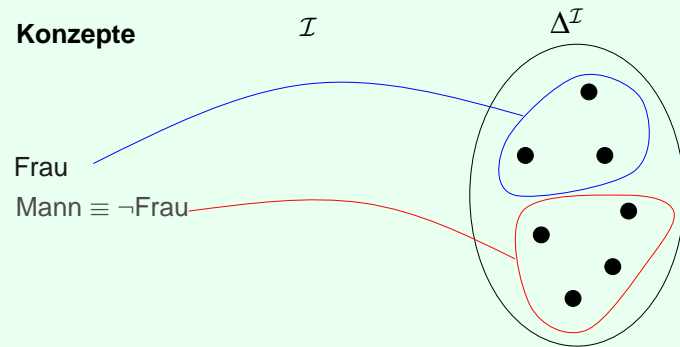
Klassenbeziehungen:

- Gleichheit: $A \equiv C$
- Inklusion: $C \sqsubseteq D$
- Eine Interpretation \mathcal{I} erfüllt
 - $C \equiv D$ gdw. $C^{\mathcal{I}} = D^{\mathcal{I}}$
 - $C \sqsubseteq D$ gdw. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
 - eine TBox \mathcal{T} gdw. \mathcal{I} jedes Axiom in \mathcal{T} erfüllt



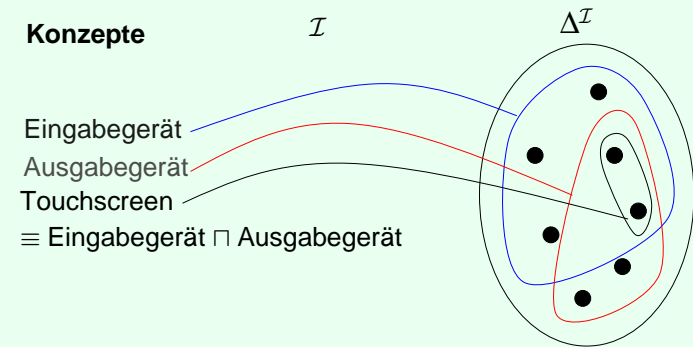
4.2.2 \mathcal{ALC} TBox Semantik

- Semantik der Negation $\neg(C)^I = \Delta^I \setminus C^I$



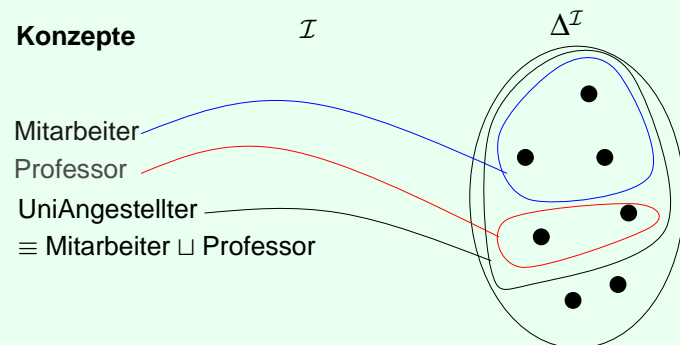
4.2.2 \mathcal{ALC} TBox Semantik

- Semantik der Konjunktion $(C \sqcap D)^I = C^I \cap D^I$



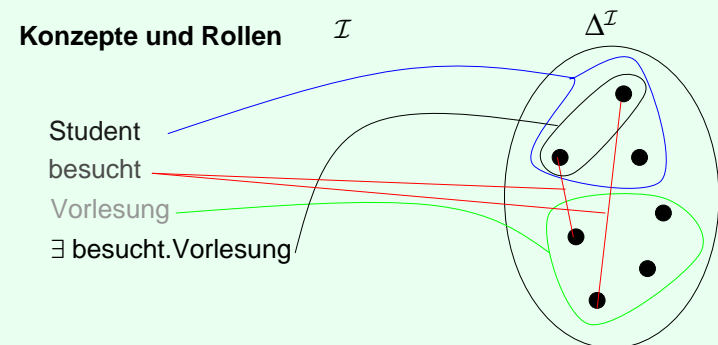
4.2.2 \mathcal{ALC} TBox Semantik

- Semantik der Disjunktion $(C \sqcup D)^I = C^I \cup D^I$



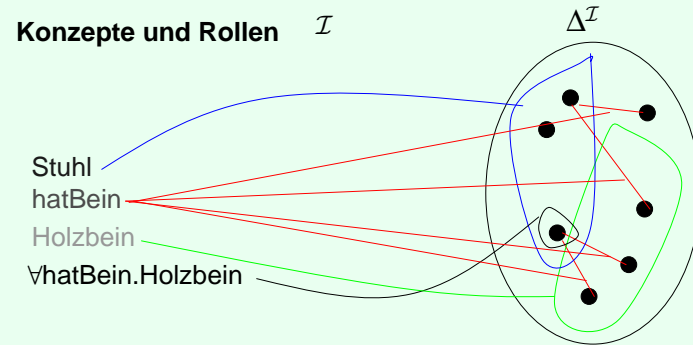
4.2.2 \mathcal{ALC} TBox Semantik

- Semantik des Existenzquantors
 - $(\exists R.C)^I = \{d \in \Delta^I \mid \exists e \in \Delta^I \text{ mit } (d,e) \in R^I \wedge e \in C^I\}$



4.2.2 \mathcal{ALC} TBox Semantik

- Semantik des Allquantors
 - $(\forall R.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}} (d,e) \in R^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$



4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

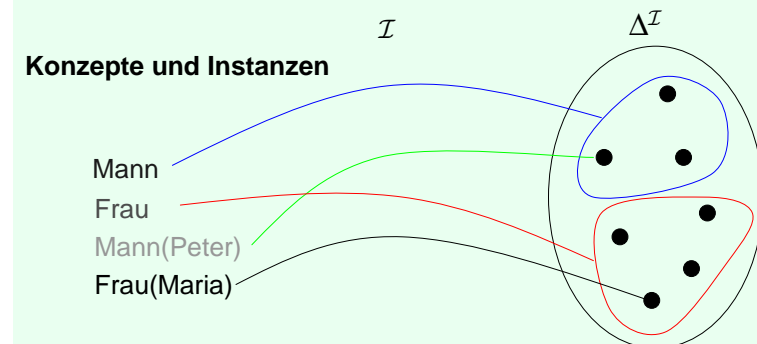
- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
 - 4.2.1 \mathcal{ALC} TBox Syntax
 - 4.2.2 \mathcal{ALC} TBox Semantik
 - 4.2.3 \mathcal{ALC} ABox Syntax und Semantik
 - 4.2.4. Erweiterungen von \mathcal{ALC}
 - 4.2.5. Modellierungsbeispiele
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.2.3 ABox Syntax

- Konzept Assertion** $C(a)$
 - Beispiel: Student(Peter), Vorlesung(AngInfol)
 - Vergleichbar mit Objekten in UML und Entitäten in ER.
- Rolle Assertion** $R(a, b)$
 - Beispiel: besucht(Peter, AngInfol)
 - Vergleichbar mit Assoziationen in UML und Beziehungen in ER.
- Eine **ABox** ist eine endliche Menge von solchen Axiomen (Konzept Assertion und Rolle Assertion).
- Die in ABox Axiomen benutzten Konzepte **können aber müssen nicht** in TBox definiert sein.

4.2.3 ABox Semantik

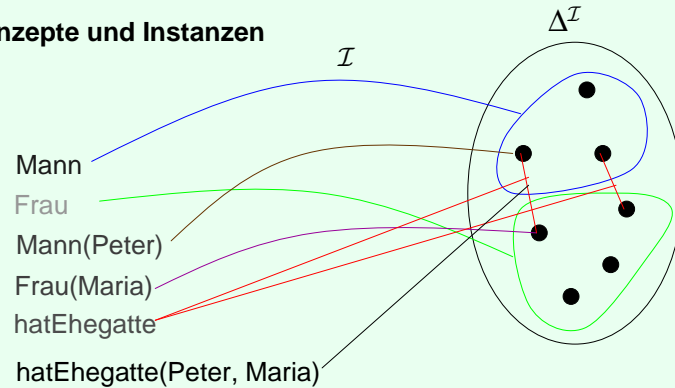
- Eine Interpretation \mathcal{I} erfüllt
 - $C(a)$ gdw. $a^{\mathcal{I}} \in C^{\mathcal{I}}$



4.2.3 ABox Semantik

- Eine Interpretation \mathcal{I} erfüllt
 - $R(a, b)$ gdw. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Konzepte und Instanzen



4.2.3 ABox Semantik

- Eine Interpretation \mathcal{I} erfüllt
 - eine ABox gdw. \mathcal{I} jedes Axiom in \mathcal{A} erfüllt ($\mathcal{I} \models \mathcal{A}$)
 - ein Axiom α bzgl. einer TBox \mathcal{T} wenn $\mathcal{I} \models \alpha$ und $\mathcal{I} \models \mathcal{T}$

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
 - 4.2.1 \mathcal{ALC} TBox Syntax
 - 4.2.2 \mathcal{ALC} TBox Semantik
 - 4.2.3 \mathcal{ALC} ABox Syntax und Semantik
 - 4.2.4. Erweiterungen von \mathcal{ALC}
 - 4.2.5. Modellierungsbeispiele
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.2.4 Erweiterung von \mathcal{ALC}

Mit der Zeit sind verschiedene DL Konstruktoren eingeführt worden, um \mathcal{ALC} zu erweitern. Z.B.

- Zahlenrestriktionen:
 - Syntax:
 - $\geq n R$
 - $\leq n R$
 - Semantik:
 - $\{x \mid |\{(x,y) \in R^{\mathcal{I}}\}| \geq n\}$
 - $\{x \mid |\{(x,y) \in R^{\mathcal{I}}\}| \leq n\}$
 - Beispiele:
 - ≥ 3 hatKind bedeutet alle Individuen, die mindestens drei Kinder haben
 - ≤ 1 hatMutter bedeutet alle Individuen, die höchstens eine Mutter haben.

4.2.4 Erweiterung von \mathcal{ALC}

- qualifizierte Zahlenrestriktionen: wie Zahlenrestriktion nur mit dem Unterschied, dass der Wertebereich festgelegt ist
 - Syntax: $\geq n$ R.C bzw. $\leq n$ R.C
 - Semantik: $\{x \mid |\{(x,y) \in R^I \wedge y \in C^I\}| \geq n\}$ bzw. $\{x \mid |\{(x,y) \in R^I \wedge y \in C^I\}| \leq n\}$
 - Beispiele:
 - ≥ 2 hatKind.Frau bedeutet alle Individuen, die mindestens 2 weibliche Kinder (Töchter) haben.
 - ≤ 1 hatElternteil.Mann bedeutet alle Individuen, die höchstens einen männlichen Elternteil (Vater) haben.

4.2.4 Erweiterung von \mathcal{ALC}

- inverse Rollen
 - Syntax: R^-
 - Semantik: $\{\langle x,y \rangle \mid \langle y,x \rangle \in R^I\}$
 - Beispiel: hatKind $^-$ (hatElternteil) bedeutet, dass hatElternteil eine inverse Rolle von hatKind ist. D.h. wenn a Kind von b ist, ist b Elternteil von a.
- transitive Rollen
 - Syntax : R^*
 - Semantik: $\{R^I\}^*$
 - Beispiel: hatNachkommen* bedeutet, dass die Nachkommen von Nachkommen auch Nachkommen sind. D.h. wenn b Nachkommen von a ist und c Nachkommen von b, dann ist c Nachkommen von a.

4.2.4 Erweiterung von \mathcal{ALC}

- Nominal: Damit kann man vorgeben welche Individuen Instanz von einem Konzept sein dürfen.
 - Syntax: $\{a_1, \dots, a_n\}$
 - Semantik: $\{a_1, \dots, a_n\}^I = \{a_1^I, \dots, a_n^I\}$
 - Beispiel: EU-Mitglied \equiv {Deutschland, Italien, ...}
- Konkrete Rollen: Rollen, die als zweite Argumente Datentypen wie z.B. Zahlen oder Strings haben.
 - z.B. hasAge(Markus,25)
 - dabei ist die Semantik der Datentypen festgelegt, d.h. 25^I ist die Zahl 25.

4.2.4 Erweiterungen von \mathcal{ALC}

Concepts	
Atomic	A, B
Not	$\neg C$
And	$C \sqcap D$
Or	$C \sqcup D$
Exists	$\exists R.C$
For all	$\forall R.C$
At least	$\geq n$ R.C ($\geq n$ R)
At most	$\leq n$ R.C ($\leq n$ R)
Nominal	$\{i_1, \dots, i_n\}$

Roles	
Atomic	R
Inverse	R^-

Concept Axioms (TBox)	
Subclass	$C \sqsubseteq D$
Equivalent	$C \equiv D$

Role Axioms (RBox)	
Subrole	$R \sqsubseteq S$
Transitivity	Trans(S)

Assertional Axioms (ABox)	
Instance	$C(a)$
Role	$R(a,b)$
Same	$a = b$
Different	$a \neq b$

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: *ALC*
 - 4.2.1 *ALC* TBox Syntax
 - 4.2.2 *ALC* TBox Semantik
 - 4.2.3 *ALC* ABox Syntax und Semantik
 - 4.2.4. Erweiterungen von *ALC*
 - 4.2.5. Modellierungsbeispiele
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

Wissensmodellierung: Beispiele

$\text{bus_driver} \equiv \text{person} \sqcap \exists \text{drives.bus}$

$\text{bus} \sqsubseteq \text{vehicle}$

$\text{driver} \equiv \text{person} \sqcap \exists \text{drives.vehicle}$

- A bus driver is a person that drives a bus.
- A bus is a vehicle.
- A driver is a person that drives a vehicle.

Schlussfolgerung: $\text{bus_driver} \sqsubseteq \text{driver}$

- A bus driver drives a vehicle, so must be a driver.

Wissensmodellierung: Beispiele

$\text{driver} \equiv \text{person} \sqcap \exists \text{drives.vehicle}$

$\text{driver} \sqsubseteq \text{adult}$

$\text{grownup} \equiv \text{adult} \sqcap \text{person}$

- Drivers are persons that drive cars.
- All drivers are adults.
- Grown-ups are adult persons.

Schlussfolgerung: $\text{driver} \sqsubseteq \text{grownup}$

- So all drivers are adult persons.

Wissensmodellierung: Beispiele

$\text{cow} \sqsubseteq \text{vegetarian}$

$\text{vegetarian} \equiv \forall \text{eats}.\neg \exists \text{partof.animal} \sqcap \forall \text{eats}.\neg \text{animal}$

$\text{madcow} \equiv \text{cow} \sqcap \exists \text{eats}.\left(\exists \text{partof.sheep} \sqcap \text{brain}\right)$

$\text{sheep} \sqsubseteq \text{animal}$

- Cows are vegetarians.
- Vegetarians eat neither animal parts nor animals.
- A mad cow is one that has been eating sheep brains.
- Sheep are animals.

Schlussfolgerung: Wissensbasis ist inkonsistent!

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einführung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.3 DL und FOL

(FOL = Prädikatenlogik 1. Stufe)

\mathcal{ALC} (ebenso andere DLs) ist ein Fragment von FOL

- jede Wissensbasis kann semantikerhaltend nach FOL übersetzt werden
- Instrumentarium der Logik (aus Mathematik, Philosophie, Informatik, Künstlicher Intelligenz) kann verwendet werden.
(über 2000 Jahre Entwicklung seit Aristoteles)

4.3 DL und FOL

- Umformungen nach FOL

$$C \equiv D \quad \Leftrightarrow \quad (\forall x) (C(x) \leftrightarrow D(x))$$

$$C \sqsubseteq D \quad \Leftrightarrow \quad (\forall x) (C(x) \rightarrow D(x))$$

$$C \sqcap D \quad \Leftrightarrow \quad C(x) \wedge D(x)$$

$$C \sqcup D \quad \Leftrightarrow \quad C(x) \vee D(x)$$

$$\neg C \quad \Leftrightarrow \quad \neg C(x)$$

$$\forall R.C \quad \Leftrightarrow \quad (\forall y) (R(x,y) \rightarrow C(y))$$

$$\exists R.C \quad \Leftrightarrow \quad (\exists y) (R(x,y) \rightarrow C(y))$$

4.3 DL und FOL

- Der Begriff der logischen Konsequenz (Schlussfolgerung) kann aus der FOL übernommen werden.
- Deduktionsalgorithmen können aus FOL übernommen werden.
- \mathcal{ALC} (auch OWL DL = $\mathcal{SHOIN}(\mathcal{D})$) sind entscheidbar, d.h.
 - Es existieren korrekte, vollständige und stets terminierende Algorithmen zur Berechnung von Schlussfolgerungen.
- Komplexität von \mathcal{ALC} ist PSPACE.
- Komplexität von OWL DL ist NExptime.

4.3 Closed World (CW) vs. Open World Semantik (OW)

- FOL (also DL) ist OW.
- Expertensysteme und Datenbanken sind idR CW.
- Der **Unterschied** liegt in der **Interpretation von fehlender Information**
 - CW: **vorhandene Information** wird als **vollständig** betrachtet, d.h. was nicht bewiesen werden kann, wird **als falsch** interpretiert.
 - OW: Es wird in Betracht gezogen, dass Informationen evt. unvollständig sind.
 - Wenn etwas **nicht bewiesen** werden kann, wird das **nicht als falsch** interpretiert.
 - klare **Unterscheidung** zwischen **Bekanntem** und **Unbekanntem**

4.3 Closed World vs. Open World Semantik

- Die Unterscheidung hat Wirkung auf die Antworten einer Anfrage.
- Beispiel:
 - Gegeben sei die A-Box: Mann(James), Mann(Harry), hasChild(James, Harry).
 - Die Anfrage „sind alle Kinder von James männlich?“ wird unter CW mit „ja“ und unter OW mit „unbekannt“ beantwortet, da die Information, dass Harry das einzige Kind von James ist, fehlt.

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einfung in DLs
- 4.2 DL Syntax und Semantik: *ALC*
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
 - 4.4.1 T-Box Inferenzprobleme
 - 4.4.2 A-Box Inferenzprobleme
 - 4.4.3 Grundidee automatischen Schließens:
Beispiel Tableauxverfahren
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.4 Automatisches Schließen

- Analog zu Wissensbasis, werden die Inferenzprobleme auch in zwei Kategorien geteilt:
 - T-Box Inferenzprobleme
 - A-Box Inferenzprobleme

4.4.1 TBox-Inferenzprobleme

- **Erfüllbarkeit:** Ein Konzept C ist bzgl. \mathcal{T} erfüllbar, wenn es ein Modell \mathcal{I} von \mathcal{T} gibt so dass die Interpretation von C eine nichtleere Menge ist.
 - existiert ein Modell \mathcal{I} für C so dass $C^{\mathcal{I}} \neq \emptyset$?

- **Subsumption:** Ein Konzept C ist subsumiert durch ein Konzept D bzgl. \mathcal{T} , wenn für jedes Modell \mathcal{I} von \mathcal{T} die Interpretation von C eine Teilmenge von der Interpretation von D ist
 - gilt $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ für jedes Modell \mathcal{I} von \mathcal{T} ?

4.4.1 TBox-Inferenzprobleme

- **Äquivalenz:** Zwei Konzepte C und D sind äquivalent bzgl. \mathcal{T} wenn für jedes Modell \mathcal{I} von \mathcal{T} die Interpretation von C gleich der Interpretation von D ist.
 - gilt $C^{\mathcal{I}} = D^{\mathcal{I}}$ für jedes Modell \mathcal{I} von \mathcal{T} ?

- **Disjunktheit:** Zwei Konzepte C und D sind disjunkt bzgl. \mathcal{T} wenn für jedes Modell \mathcal{I} von \mathcal{T} die Interpretationen von C und D disjunkt sind
 - gilt $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ für jedes Modell \mathcal{I} von \mathcal{T} ?

4.4.1 Reduktion auf Unerfüllbarkeit

- Subsumption:
 - C ist subsumiert durch $D \Leftrightarrow C \sqcap \neg D$ ist unerfüllbar

- Äquivalenz:
 - C und D sind äquivalent \Leftrightarrow sowohl $(C \sqcap \neg D)$ als auch $(\neg C \sqcap D)$ ist unerfüllbar

- Disjunktheit:
 - C und D sind disjunkt $\Leftrightarrow C \sqcap D$ ist unerfüllbar

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einfügung in DLs
- 4.2 DL Syntax und Semantik: *ALC*
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
 - 4.4.1 T-Box Inferenzprobleme
 - 4.4.2 A-Box Inferenzprobleme
 - 4.4.3 Grundidee automatischen Schließens: Beispiel Tableauxverfahren
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.4.2 A-Box Inferenzprobleme

- **Konsistenz:**
 - Eine ABox \mathcal{A} ist konsistent bzgl. einer TBox \mathcal{T} , wenn es eine Interpretation gibt, die Modell sowohl von \mathcal{A} als auch von \mathcal{T} ist.
 - Eine ABox ist konsistent wenn sie konsistent bzgl. der leeren TBox ist.

- **Instanzüberprüfung:** Gegeben sei eine Instanz a und ein Konzept C . Gilt $a^I \in C^I$ für alle Modelle von \mathcal{A} und \mathcal{T} . D.h. $\mathcal{A}, \mathcal{T} \models C(a)$?

4.4.2 Reduktion auf Unerfüllbarkeit

- Instanzüberprüfung:
 - Instanz a gehört zum Konzept C wenn $\neg C(a)$ unerfüllbar ist.

Für Konsistenzprüfung kann z.B. ein Modell konstruiert werden.

4 Beschreibungslogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einfügung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
 - 4.4.1 T-Box Inferenzprobleme
 - 4.4.2 A-Box Inferenzprobleme
 - 4.4.3 Grundidee automatischen Schließens: Beispiel Tableauverfahren
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.4.3 Automatisierung: Beispiel Tableaux

TBox: $\text{human} \sqsubseteq \exists \text{parentOf}.\top$
 $\text{orphan} \equiv \text{human} \sqcap \neg \exists \text{parentOf}.\text{alive}$

Übersetzung nach FOL:

$(\forall X) (\text{human}(X) \rightarrow (\exists Y) \text{parentOf}(Y,X))$

$(\forall X) (\text{orphan}(X) \leftrightarrow$

$(\text{human}(X) \wedge \neg(\exists Y) (\text{parentOf}(Y,X) \wedge \text{alive}(Y)))$

ABox: $\text{orphan}(\text{harrypotter})$

$\text{parentOf}(\text{jamespotter}, \text{harrypotter})$

Können wir folgern: $\neg \text{alive}(\text{jamespotter})$?

4.4.3 Automatisierung: Beispiel Tableaux

Zu zeigen:

$$((\forall X) (\text{human}(X) \rightarrow (\exists Y) \text{parent_of}(Y,X))$$

$$\wedge (\forall X) (\text{orphan}(X) \leftrightarrow$$

$$\quad (\text{human}(X) \wedge \neg(\exists Y) (\text{parent_of}(Y,X) \wedge \text{alive}(Y)))$$

$$\wedge \text{orphan}(\text{harrypotter})$$

$$\wedge \text{parent_of}(\text{jamespotter}, \text{harrypotter})$$

$$) \rightarrow \neg \text{alive}(\text{jamespotter}))$$

ist allgemeingültig.

4.4.3 Automatisierung: Beispiel Tableaux

Zu zeigen:

$$\neg((\forall X) (\text{human}(X) \rightarrow (\exists Y) \text{parent_of}(Y,X))$$

$$\wedge (\forall X) (\text{orphan}(X) \leftrightarrow$$

$$\quad (\text{human}(X) \wedge \neg(\exists Y) (\text{parent_of}(Y,X) \wedge \text{alive}(Y)))$$

$$\wedge \text{orphan}(\text{harrypotter})$$

$$\wedge \text{parent_of}(\text{jamespotter}, \text{harrypotter})$$

$$) \rightarrow \neg \text{alive}(\text{jamespotter}))$$

ist unerfüllbar.

4.4.3 Automatisierung: Beispiel Tableaux

Vorgehensweise:

- Zeige Allgemeingültigkeit von Formel F.
- Eliminiere \rightarrow und \leftrightarrow durch äquivalente Formeln.
- Negation der Formel: $\neg F$
- Konstruiere Tableau für $\neg F$.
Kann das Tableau *abgeschlossen* werden, dann ist F allgemeingültig.

4.4.3 Automatisierung: Beispiel Tableaux

Regeln für Tableauerstellung:

$$\frac{\neg\neg F}{F} \quad \frac{F_1 \wedge F_2}{\begin{array}{c} F_1 \\ F_2 \end{array}} \quad \frac{F_1 \vee F_2}{F_1 | F_2} \quad \frac{\neg(F_1 \vee F_2)}{\begin{array}{c} \neg F_1 \\ \neg F_2 \end{array}} \quad \frac{\neg(F_1 \wedge F_2)}{\neg F_1 | \neg F_2}$$

$$\frac{(\forall X)F}{F\{X \mapsto Y\}} \quad \frac{\neg(\exists X)F}{\neg F\{X \mapsto Y\}} \quad \text{mit } Y \text{ als neue freie Variable}$$

$$\frac{(\exists X)F}{F\{X \mapsto f(X_1, \dots, X_n)\}} \quad \frac{\neg(\forall X)F}{\neg F\{X \mapsto f(X_1, \dots, X_n)\}}$$

mit f als neuem Funktionszeichen und
 X_1, \dots, X_n als alle in F frei vorkommenden Variablen

4.4.3 Automatisierung: Beispiel Tableaux

$$\neg(\neg((\forall X) (\neg\text{orphan}(X) \vee$$

$$(\text{human}(X) \wedge \neg(\exists Y) (\text{parent_of}(Y,X) \wedge \text{alive}(Y))))))$$

$$\wedge \text{orphan}(\text{harrypotter})$$

$$\wedge \text{parent_of}(\text{jamespotter}, \text{harrypotter})$$

$$\vee \neg\text{alive}(\text{jamespotter}))$$

Kürzer:

$$\neg(\neg((\forall X) (\text{o}(X) \rightarrow (\text{h}(X) \wedge \neg(\exists Y) (\text{pof}(Y,X) \wedge \text{al}(Y))))))$$

$$\wedge \text{o}(\text{harry}) \wedge \text{pof}(\text{james}, \text{harry}))$$

$$\vee \neg\text{al}(\text{james}))$$

Beispieltableau auf nächster Folie!

- $\neg(\neg((\forall X) (\neg\text{o}(X) \vee (\text{h}(X) \wedge \neg(\exists Y) (\text{pof}(Y,X) \wedge \text{al}(Y))))))$
 $\wedge \text{o}(\text{harry}) \wedge \text{pof}(\text{james}, \text{harry}) \vee \neg\text{al}(\text{james}))$
- $\neg(\neg((\forall X) (\neg\text{o}(X) \vee (\text{h}(X) \wedge \neg(\exists Y) (\text{pof}(Y,X) \wedge \text{al}(Y))))))$
 $\wedge \text{o}(\text{harry}) \wedge \text{pof}(\text{james}, \text{harry}))$
- $\neg\text{al}(\text{james})$
- $(\forall X) (\neg\text{o}(X) \vee (\text{h}(X) \wedge \neg(\exists Y) (\text{pof}(Y,X) \wedge \text{al}(Y))))$
 $\wedge \text{o}(\text{harry}) \wedge \text{pof}(\text{james}, \text{harry}))$
- $\text{al}(\text{james})$
- $(\forall X) (\neg\text{o}(X) \vee (\text{h}(X) \wedge \neg(\exists Y) (\text{pof}(Y,X) \wedge \text{al}(Y))))$
- $\text{o}(\text{harry})$
- $\text{pof}(\text{james}, \text{harry})$
- $\neg\text{o}(Z) \vee (\text{h}(Z) \wedge \neg(\exists Y) (\text{pof}(Y,Z) \wedge \text{al}(Y)))$

- $\neg\text{o}(Z)$ mit $[Z/\text{harry}]$

globale Bindung!

- $\text{h}(Z) \wedge \neg(\exists Y) (\text{pof}(Y,Z) \wedge \text{al}(Y))$
- $\text{h}(Z)$
- $\neg(\exists Y) (\text{pof}(Y,Z) \wedge \text{al}(Y))$
- $\neg(\text{pof}(W,Z) \wedge \text{al}(W))$

globale Bindung!

- $\neg\text{pof}(W,Z)$
- $\neg\text{al}(W)$

mit $[W/\text{james}]$

4 Beschreibungsllogiken (DLs) - Inhaltsverzeichnis

- 4.0 Allgemeine Motivation
- 4.1 Einfung in DLs
- 4.2 DL Syntax und Semantik: \mathcal{ALC}
- 4.3 DL und Prädikatenlogik 1. Stufe
- 4.4 Automatisches Schließen
 - 4.4.1 T-Box Inferenzprobleme
 - 4.4.2 A-Box Inferenzprobleme
 - 4.4.3 Grundidee automatischen Schließens: Beispiel Tableauxverfahren
- 4.5 Modellierung des begleitenden Beispiels
- 4.6 Vergleich UML / (E)ER / DL
- 4.7 Ontologien

4.5 Modellierung des begleitenden Beispiels mit DL

- Das Unternehmen ist hierarchisch gegliedert. Es besteht aus verschiedenen Unternehmensbereichen (Elektro, KFZ, ...), die auf Betriebe verteilt sind, die sich an verschiedenen Standorten befinden. An einem Standort befindet sich nur jeweils ein Betrieb. Zu jedem Betrieb soll der Betriebsleiter und das Personalbudget, sowie die zum Betrieb gehörenden Gebäude festgehalten werden.

Unternehmen $\sqsubseteq \exists$ bestehtAus.Unternehmensbereich
 Betrieb $\sqsubseteq =1$ hatUB.Unternehmensbereich \sqcap
 $=1$ befindetSichAm.Standort \sqcap
 $=1$ hatBetriebsleiter.Betriebsleiter \sqcap
 $=1$ hatPersonalBudget.PersonalBudget \sqcap
 \exists hatGebaeude.Gebaeude
 Gebaeude $\sqsubseteq =1$ gehoertZu.Betrieb

4.5 Modellierung des begleitenden Beispiels mit DL

- Zu jedem Standort sollten sowohl die Postanschrift, als auch die geografischen Koordinaten gespeichert werden. Letztere dienen der Tourenplanung mittels eines GPS (Global Positioning System). Die Abteilungen des Unternehmens sind jeweils geschlossen in einem Gebäude untergebracht, um kurze Wege zu gewährleisten.

```
Standort ⊑ =1 hatPostAnschrift.Postanschrift ⊐
           =1 hatGeoKoord.GeoKoordinaten
Abteilung ⊑ =1 abtBefindetSichIm.Gebaeude
```

4.5 Modellierung des begleitenden Beispiels mit DL

- Zu den zur Verwaltung der Mitarbeiter notwendigen Informationen gehören neben Personal-Nummer, Name und Gehalt auch die Zugehörigkeit zu Unternehmensbereich, Betrieb und Abteilung, sowie das Gebäude in dem sich der Arbeitsplatz befindet.

```
Mitarbeiter ⊑ =1 hatPersonalNr.PersonalNummer ⊐
              =1 hatName.Name ⊐
              =1 hatGehalt.Gehalt ⊐
              =1 gehoertZuUB.Unternehmensbereich ⊐
              =1 gehoertZuBetrieb.Betrieb ⊐
              =1 gehoertZuAbt.Abteilung ⊐
              =1 hatArbeitsplatzIn.Gebaeude
```

4.5 Modellierung des begleitenden Beispiels mit DL

- Die Aufgaben der Firma sind stark projektbezogen. Deshalb sind für eine Projektverwaltung alle Projekte, Projektleiter und die zugehörigen Mitarbeiter zu speichern. Da manche Mitarbeiter parallel an verschiedenen Projekten arbeiten, ist auch der prozentuale Anteil der Arbeitszeit, mit dem an einem Projekt gearbeitet wird, von Bedeutung.

```
Projekt ⊑ =1 hatProjektleiter.Projektleiter ⊐
          =1 hatMitarbeiter.Mitarbeiter
ProzentualeMitarbeit ⊑ =1 hatMitarbeiter.Mitarbeiter ⊐
                      =1 hatProjekt.Projekt ⊐
                      =1 anteil.Anteil
gehoertZuAbt ◦ abtBefindetSichIm ◦ gehoertZu ≡ gehoertZuBetrieb
gehoertZuBetrieb ◦ hatUB ≡ gehoertZuUB
gehoertZuAbt ◦ abtBefindetSichIm ≡ hatArbeitsplatzIn
Betriebsleiter ⊑ Mitarbeiter
Projektleiter ⊑ Mitarbeiter
```