Karlsruhe Institute of Technology

Teil 9

Ontologien und Beschreibungslogik

Ontologien - Einführung

- Ontologie ....
  - ⇨ ist eine philosophische Disziplin, Zweig der Philosophie welcher sich mit der Natur und der Organisation von Wirklichkeit beschäftigt

- Lehre des Seins (Aristoteles, Metaphysik, IV, 1)

- Versucht folgende Fragen zu beantworten
  - ✧ Was ist Sein?
  - ✧ Was sind alle gemeinsamen Eigenschaften des Seins?

Ontologien in der Informatik

- An ontology is an explicit specification of a conceptualization. [Gruber, 93]
- An ontology is a shared understanding of some domain of interest. [Uschold, Gruninger, 96]
- Es gibt viele weitere Definitionen, in unserem Sinne ist – zunächst informell betrachtet - alles eine Ontologie was
  - ⇨ eine Konzeptualisierung eines Weltausschnitts darstellt,
  - ⇨ eine formale Spezifikation besitzt,
  - ⇨ Konsens einer Gruppe von Personen ist.

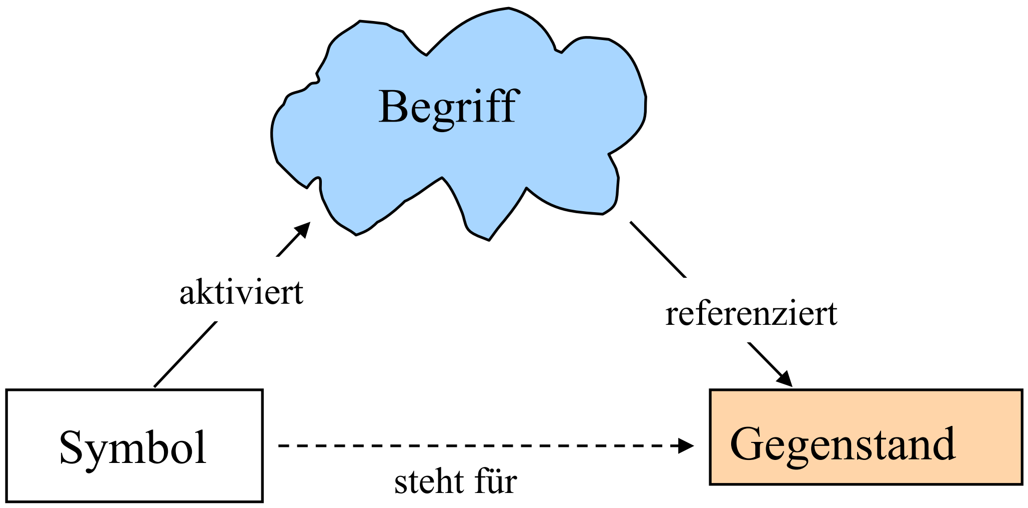
Gemeinsame Sprache

- "People can´t share knowledge if they don´t speak a common language„ [T. Davenport]

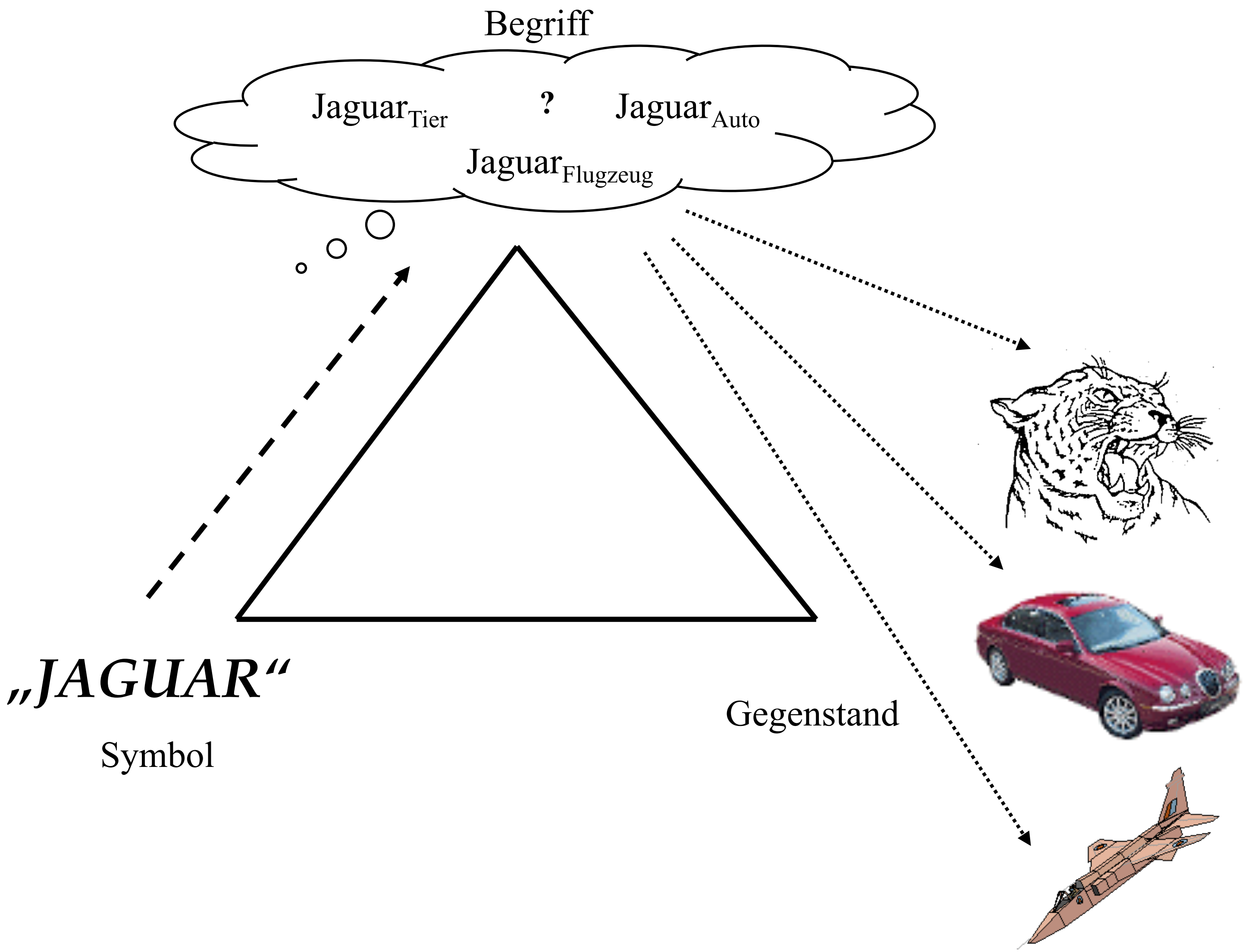
- "Gemeinsame Sprache" notwendig für funktionierende Kommunikation zwischen Personen bzw. Maschinen
  - ⇨ wohldefiniertes Vokabular an Symbolen (Lexemen, lexical entries)
  - ⇨ einheitliches Verständnis, welche Begriffe (concepts) und Beziehungen (relations) durch die Symbole referenziert werden

Definition von Ontologien

- Der allgemeine Zusammenhang wird von dem Bedeutungsdreieck beschrieben, welches die Interaktion zwischen **Symbolen, Begriffen** und **Gegenständen** der Welt definiert.

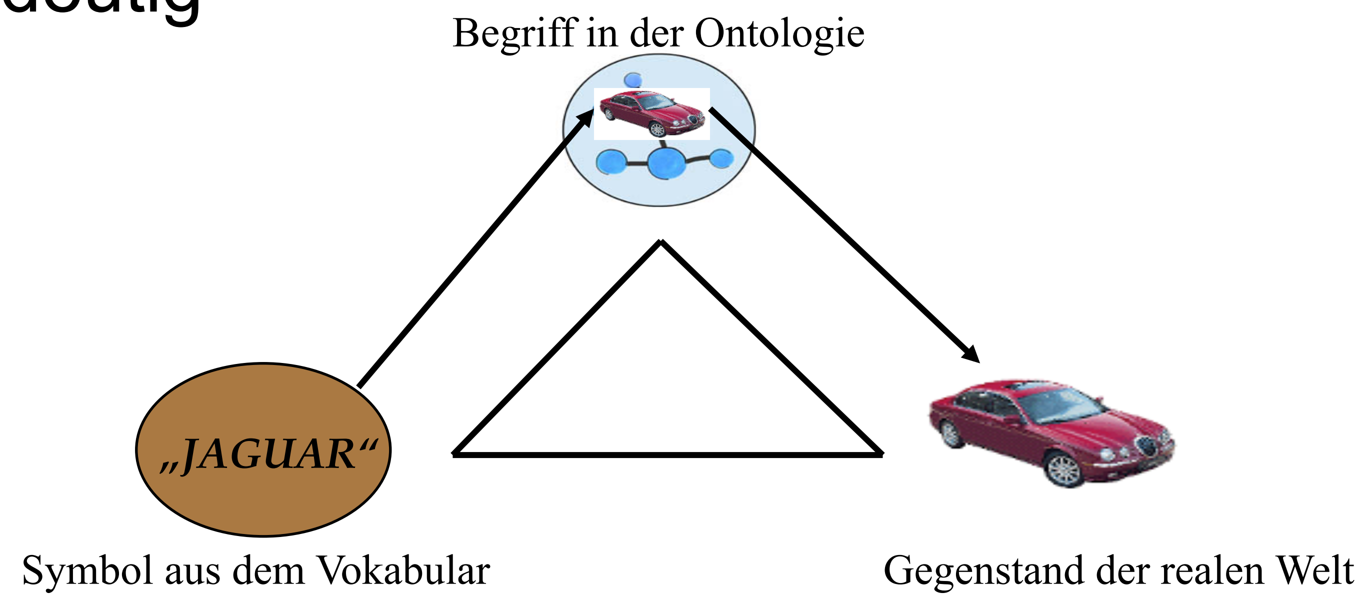


Symbole und Gegenstände



Ontologien für Kommunikation

- Die Ontologie reduziert die Anzahl von Abbildungen von Symbolen auf Gegenstände der realen Welt, im Idealfall ist die Abbildung eindeutig



Ontologiesprachen

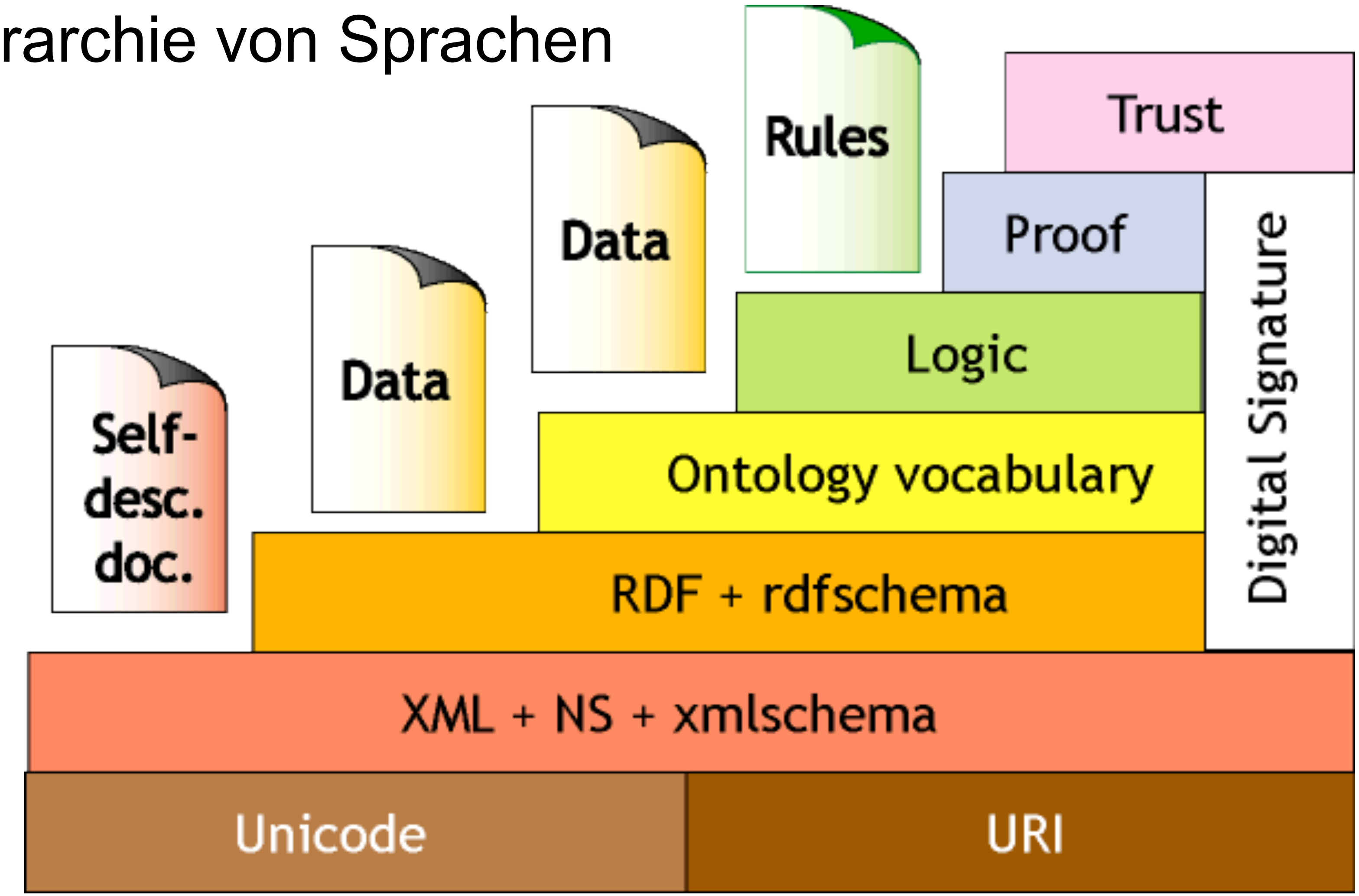
- Sprachen, mit denen man konzeptuell modellieren kann:
  - ⇨ Prädikatenlogik erster Stufe
  - ⇨ Frame-Logic
  - ⇨ RDF(S) und OWL (W3C Recommendations)
- Sprachen unterscheiden sich in ihrer Ausdrucksmächtigkeit
- Anforderung: Formale Semantik und Ausführbarkeit
- Ontologiesprachen sollten für das Semantic Web geeignet sein, siehe z.B. OWL

Semantic Web

- Das WWW ist zur Nutzung durch den Menschen bestimmt
- WWW basiert auf der Markupsprache HTML, die beschreibt:
  - ⇨ wie Informationen **dargestellt** werden sollen,
  - ⇨ wie Informationen miteinander **verknüpft** werden können,
  - ⇨ aber nicht, was diese Informationen **bedeuten**….
- Semantik Web ist das Web der nächste Generation, das zur Nutzung durch **den Menschen und Maschinen** bestimmt ist
  - ⇨ Die Herausforderung für die SW Sprache: die Bedeutung von Informationen (Web Seiten, Web Links) in maschinelesbarer Form darzustellen

Semantic Web

Hierarchie von Sprachen



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

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 framebasierten, netzwerkartigen und objektorientierten Modellierungssprachen **mit formaler Semantik an**.

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.

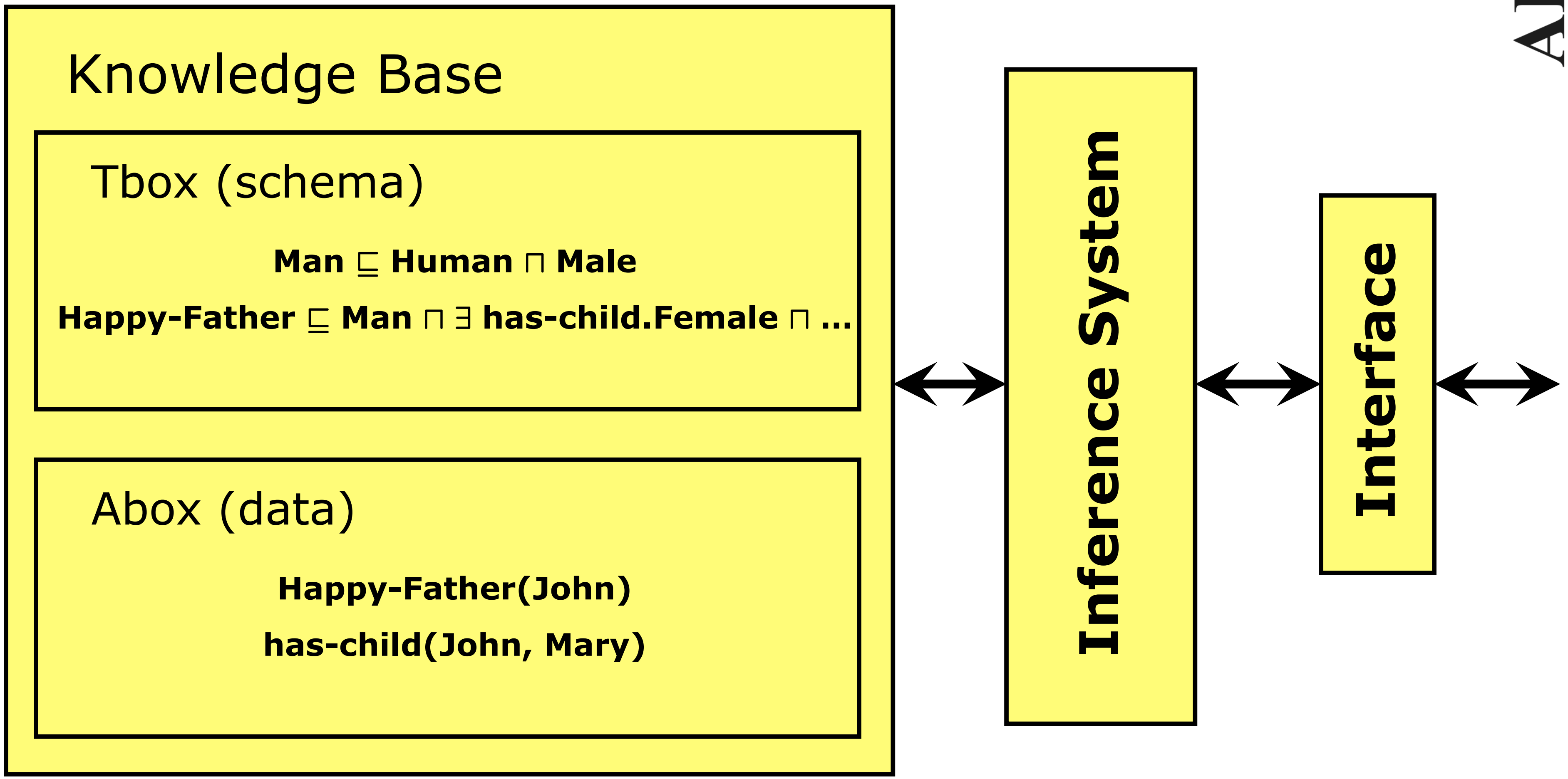
DL Syntax und Semantik

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

Konstruktoren

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

Allgemeine DL Architektur



Einfaches Beispiel

Terminologisches Wissen (*TBox*):

Axiome, die die Struktur der zu modellierenden Domäne beschreiben (konzeptionelles Schema)

$\text{Human} \sqsubseteq \exists \text{hasParent.Human}$

$\text{Orphan} \equiv \text{Human} \sqcap \neg\exists \text{hasParent.Alive}$

Wissen um Individuen (*ABox*):

Axiome, die konkrete Situationen (Daten) beschreiben

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)

$\mathcal{ALC}$ 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

ABox Syntax

- **Konzept Assertion** C(a)
  - Beispiel: Student(Peter), Vorlesung(AngInfoI)
  - Vergleichbar mit Objekten in UML und Entitäten in ER.
- **Rolle Assertion** R(a, b)
  - Beispiel: besucht(Peter, AngInfoI)
  - 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.

$\mathcal{ALC}$ TBox Syntax

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

$\mathcal{ALC}$ 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

$\mathcal{ALC}$ 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 $\equiv$ Eingabegerät $\sqcap$ Ausgabegerät
- Disjunktion von C und D bedeutet intuitiv “C oder D”
  - ⇨ UniAngestellte $\equiv$ Mitarbeiter $\sqcup$ Professor

$\mathcal{ALC}$ Konstruktoren

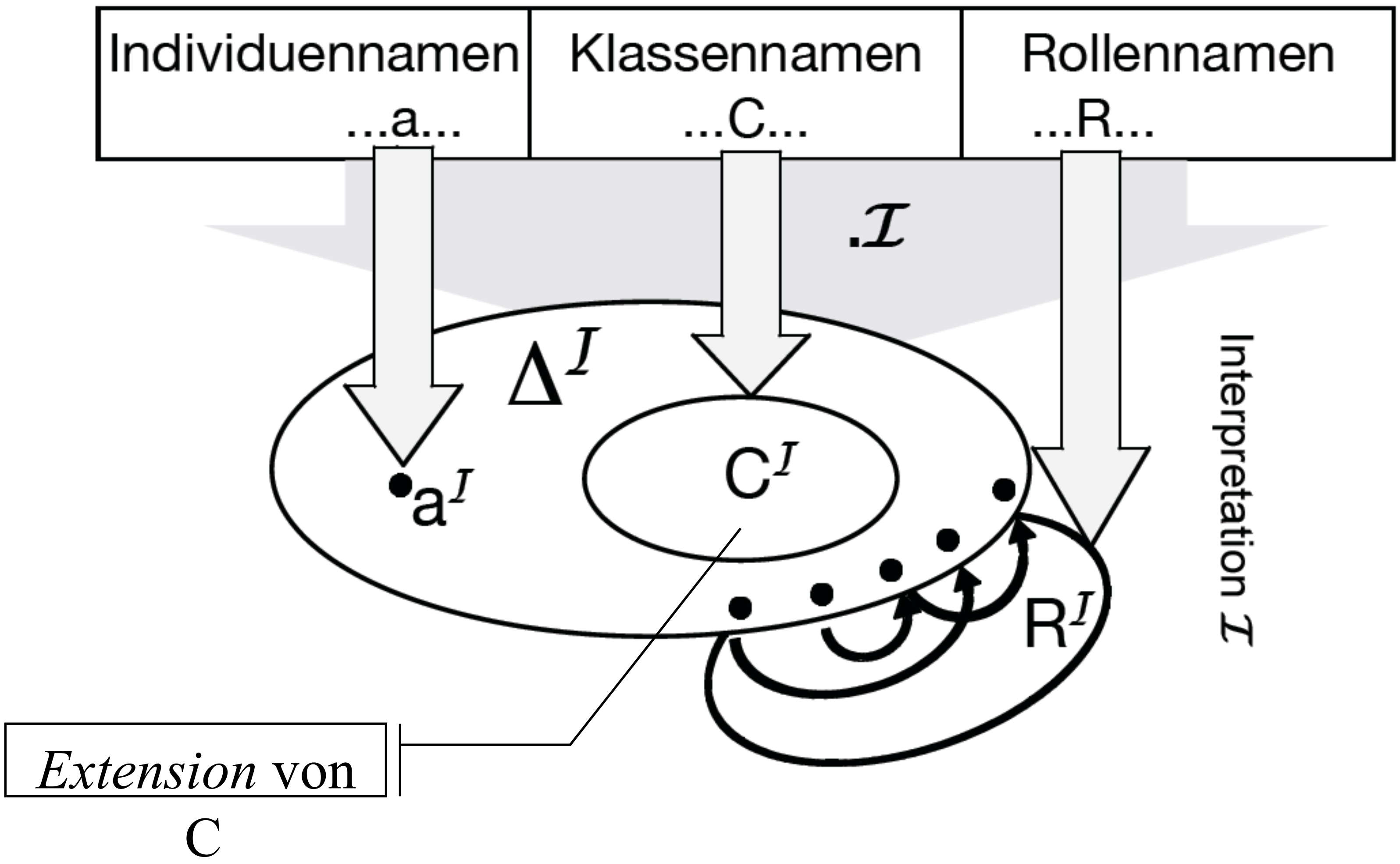
- ∃R.C (Existenzquantor) bedeutet intuitiv „alle x, die eine Beziehung vom Typ R mit einem y vom Typ C haben“
  - ⇨ ∃besucht.Vorlesung
- ∀R.C (Allquantor) bedeutet intuitiv „alle x, für die alle Beziehungen vom Typ R mit einem y vom Typ C sind“
  - ⇨ ∀hatElternteil.Mensch
  - ⇨ hat ein Individuum gar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

$\mathcal{ALC}$ Klassenbeziehungen

- Klassenbeziehungen können für zusammengesetzte Klassen verwendet werden:
  - Inklusion
    $C \sqsubseteq D$
    - z.B. Man $\sqsubseteq$ Human $\sqcap$ Male
  - Gleichheit
    $C \equiv D$
    - z.B. Frau $\equiv$ Human $\sqcap$ Female $\sqcap$ Adult
    - z.B. Orphan $\equiv$ Human $\sqcap$ $\forall$hasParent.Dead

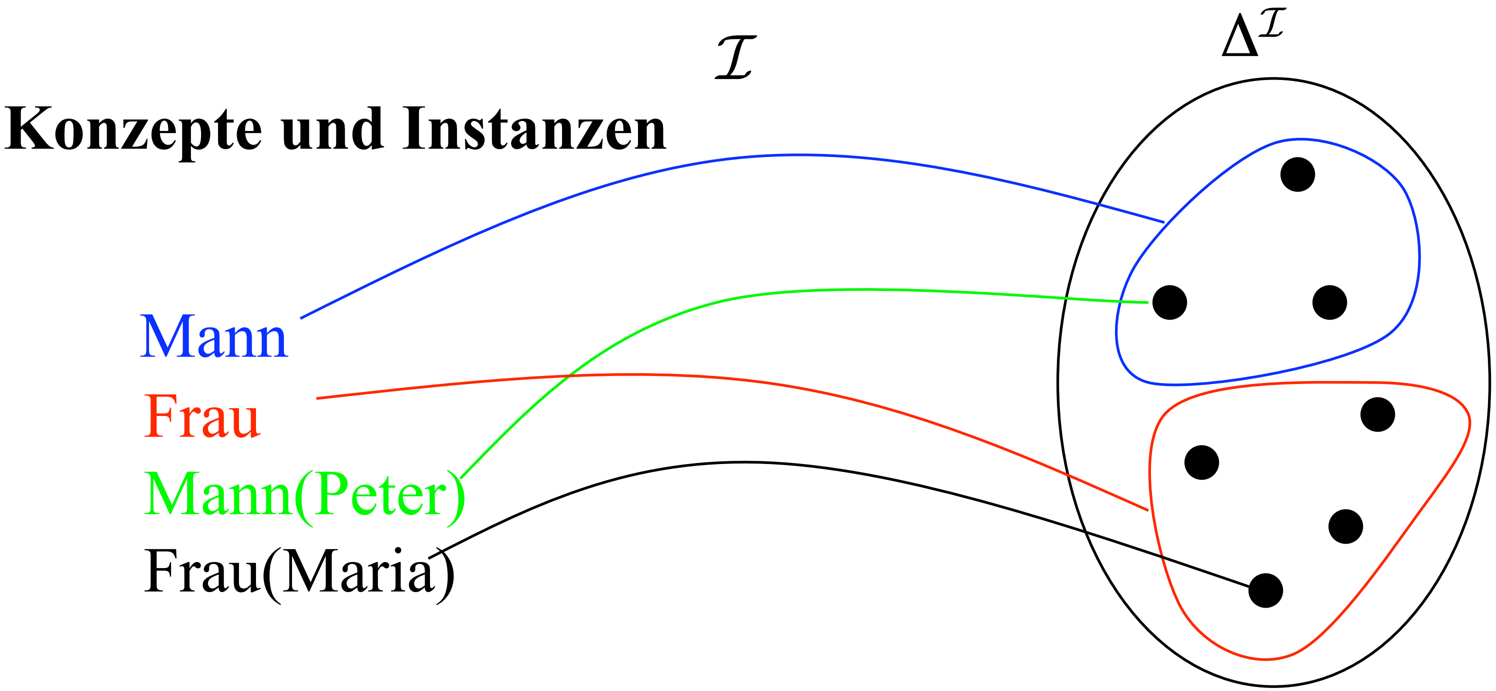
- Terminologisches Wissen (sog. **TBox**) besteht aus einer Menge solcher Klassenbeziehungen (sog. *Axiome*).

Interpretationen für $\mathcal{ALC}$



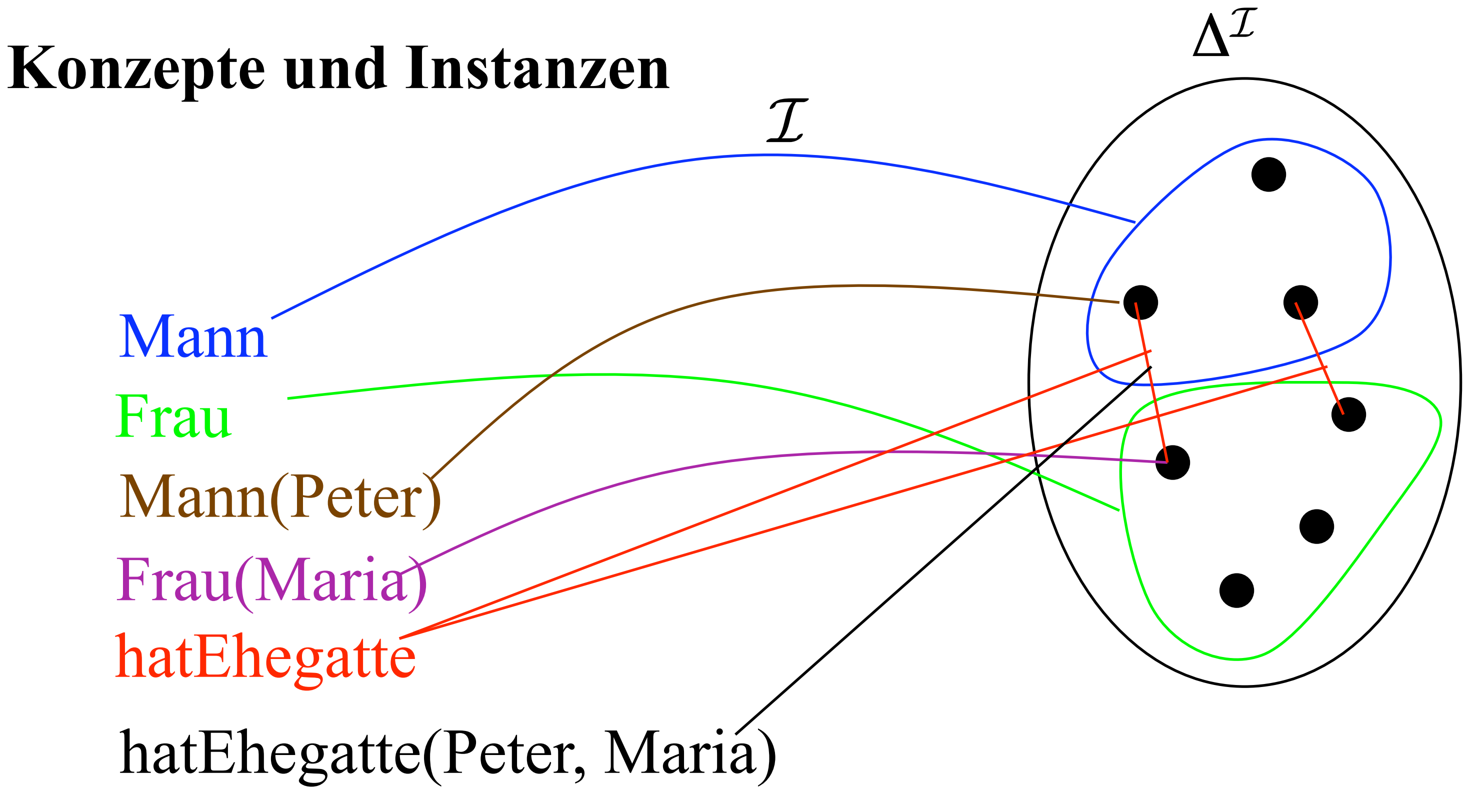
ABox Semantik

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



ABox Semantik

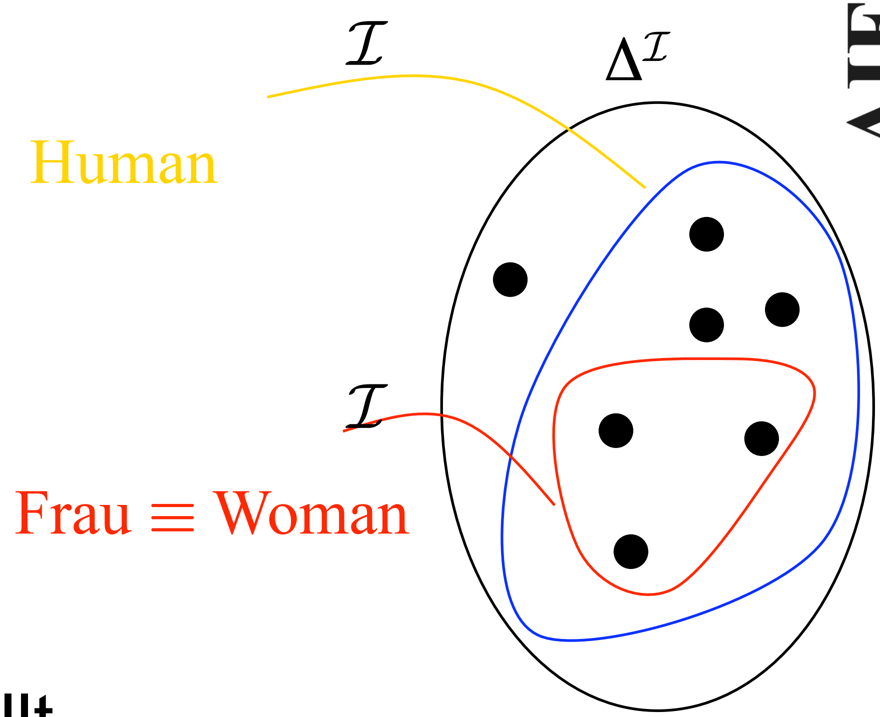
- Eine Interpretation $\mathcal{I}$ erfüllt
  - ⇨ R(a, b) gdw. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$



$\mathcal{ALC}$ TBox Semantik

Klassenbeziehungen:

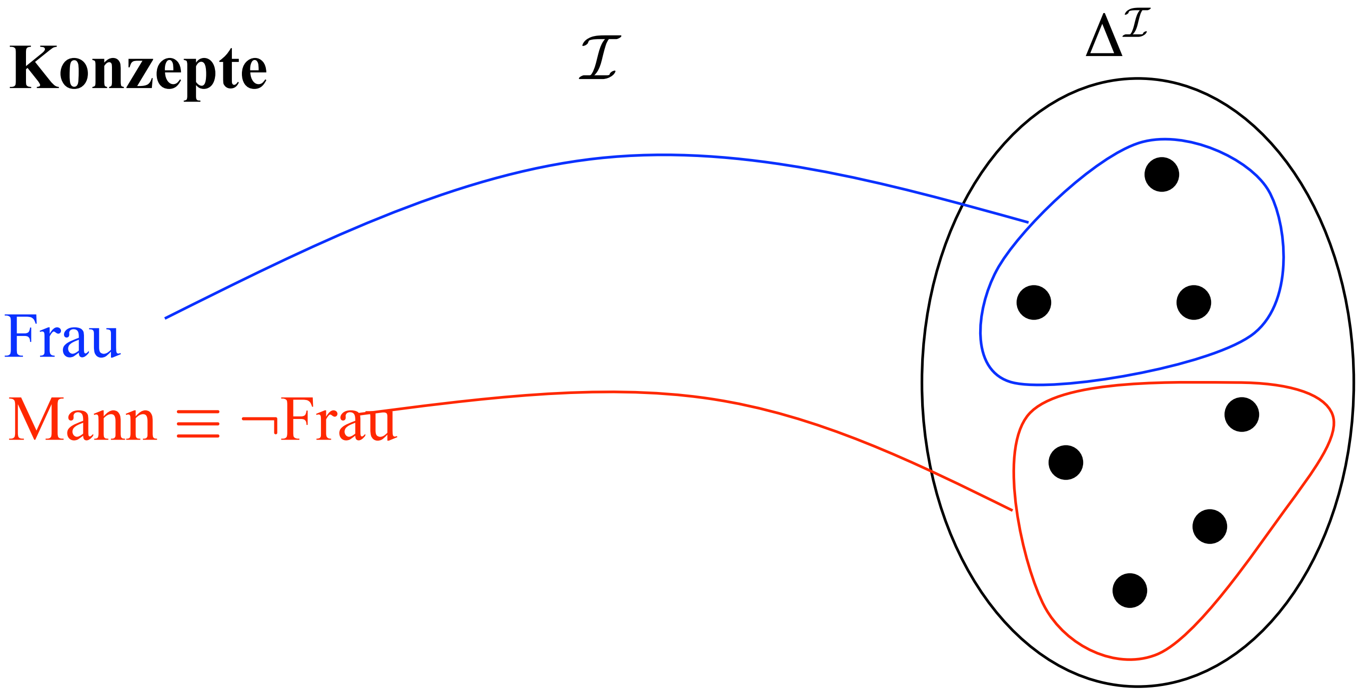
- Gleichheit: A $\equiv$ C
- Inklusion: C $\sqsubseteq$ D



- 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

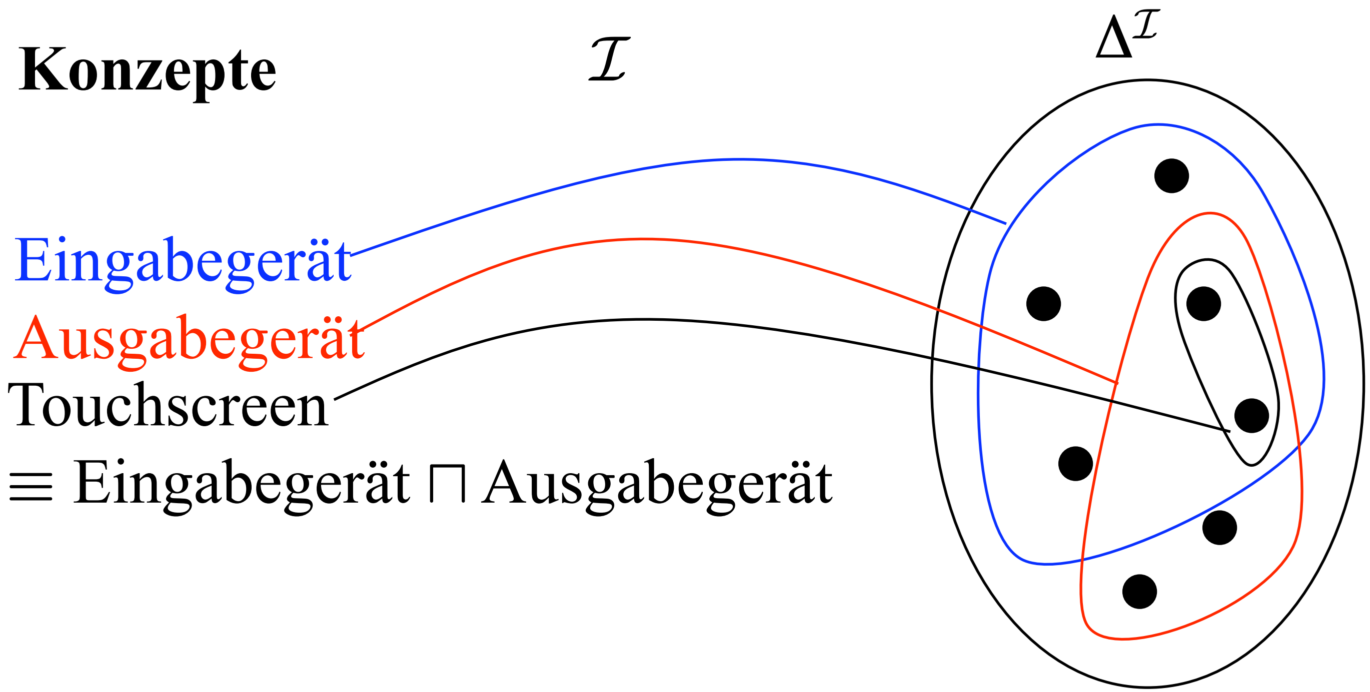
$\mathcal{ALC}$ TBox Semantik

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



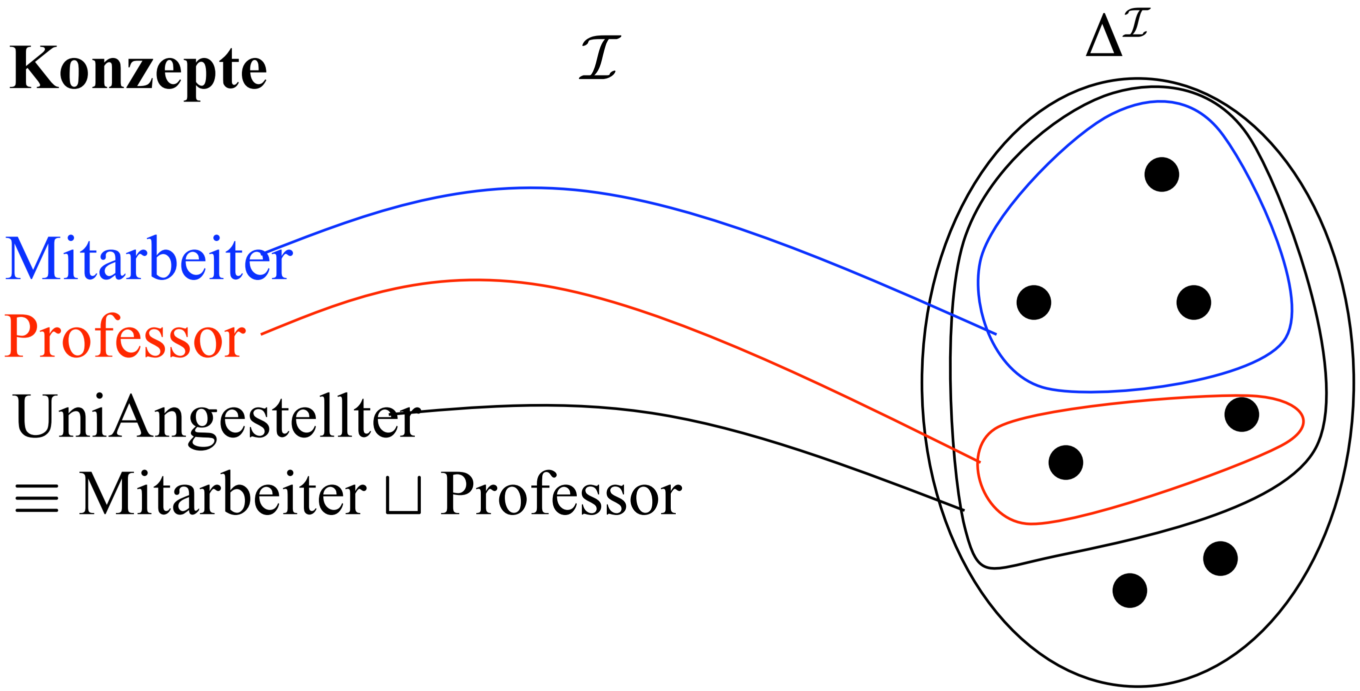
$\mathcal{ALC}$ TBox Semantik

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



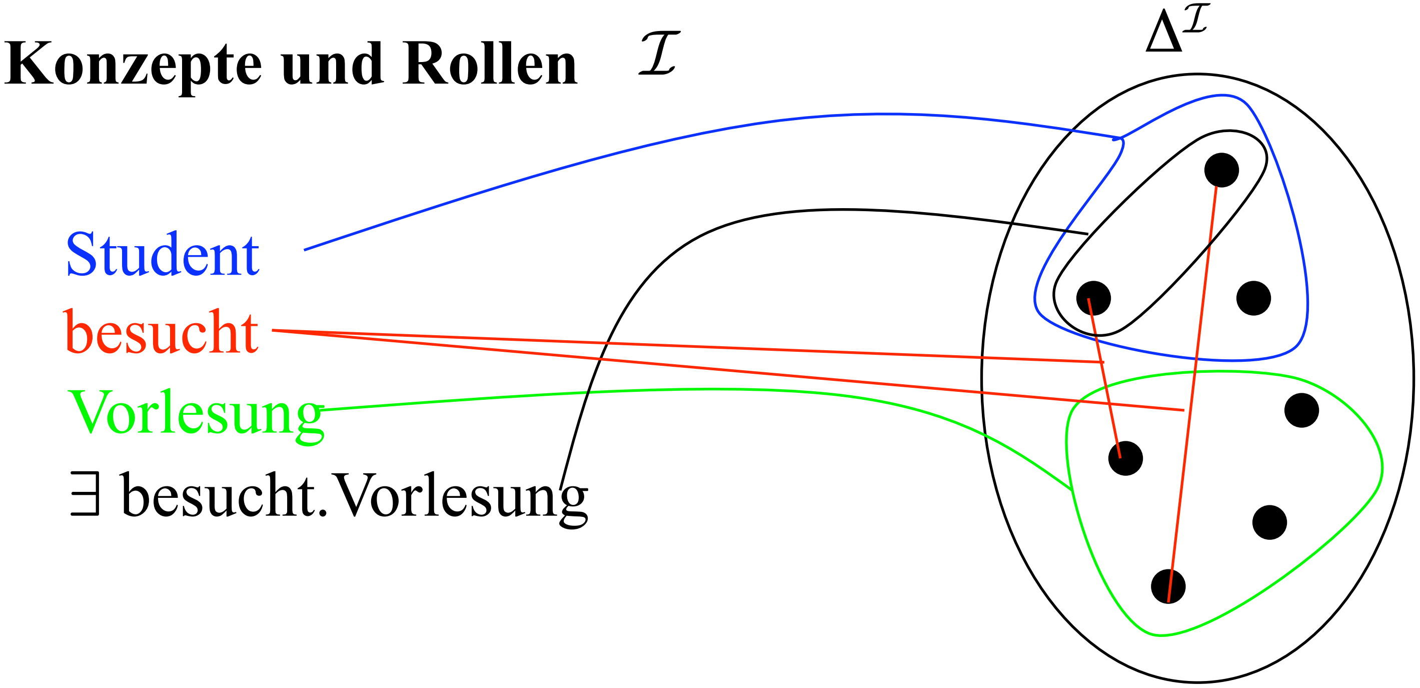
$\mathcal{ALC}$ TBox Semantik

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



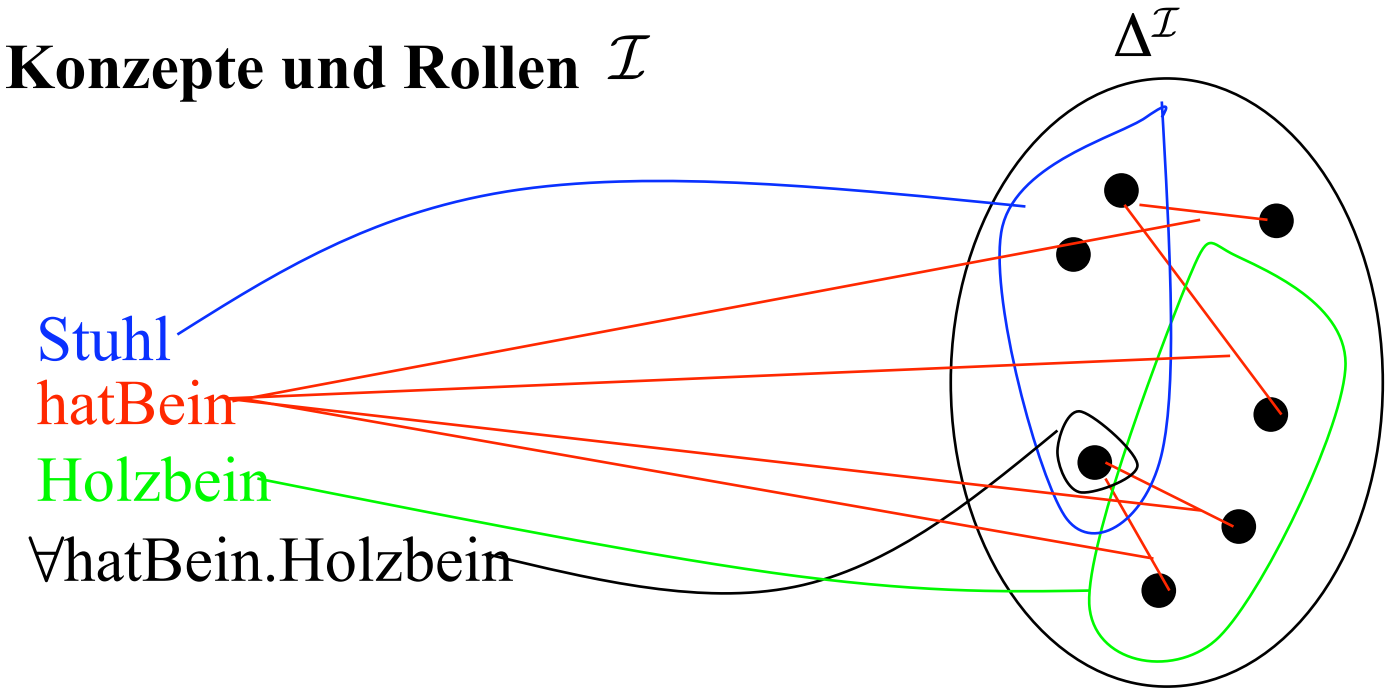
$\mathcal{ALC}$ TBox Semantik

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



$\mathcal{ALC}$ TBox Semantik

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



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

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: $\{x \mid |\{\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\}$ bzw. $\{x \mid |\{\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leq n\}$
  - Beispiele:
    - $\geq$2 hatKind.Frau bedeutet alle Individuen, die mindestens 2 weibliche Kinder (Töchter) haben.
    - $\leq$1 hatElterteil.Mann bedeutet alle Individuen, die höchstens einen männlichen Elternteil (Vater) haben.

Erweiterung von $\mathcal{ALC}$

- Subrollenbeziehung
  - Syntax: $R \sqsubseteq S$
  - Semantik: $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
  - Beispiel: istVaterVon $\sqsubseteq$ hatKind bedeutet, dass wann immer a Vater von b ist, a auch b als Kind hat.

- Rollenäquivalenz
  - Syntax: $R \equiv S$
  - Semantik: $R^{\mathcal{I}} = S^{\mathcal{I}}$
  - Beispiel: istVorfahrVon $\equiv$ hatNachfahren bedeutet, dass a Vorfahr von b genau dann ist, wenn a b als Nachfahren hat.

Erweiterung von $\mathcal{ALC}$

- inverse Rollen
  - Syntax: R^-
  - Semantik: $\{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$
  - Beispiel: $\text{hatKind}^- \equiv \text{hatElternteil}$ 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 : Trans(R)
  - Semantik: $R^{\mathcal{I}}=(R^{\mathcal{I}})^*$
  - Beispiel: Trans(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.

Erweiterung von $\mathcal{ALC}$

- Nominal: Damit kann man vorgeben welche Individuen Instanz von einem Konzept sein dürfen.
  - ⇨ Syntax: $\{a_1, ..., a_n\}$
  - ⇨ Semantik: $\{a_1, ..., a_n\}^\mathcal{I} = \{a_1^\mathcal{I}, ..., a_n^\mathcal{I}\}$
  - ⇨ 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. $25^\mathcal{I}$ ist die Zahl 25.

Erweiterungen von $\mathcal{ALC}$

	Concepts	
$\mathcal{ALC}$	Atomic	A, B
	Not	¬C
	And	C ⊓ D
	Or	C ⊔ D
	Exists	∃R.C
	For all	∀R.C
$\mathcal{Q}$ ($\mathcal{N}$)	At least	≥n R.C (≥n R)
	At most	≤n R.C (≤n R)
$\mathcal{O}$	Nominal	$\{i_1, \ldots, i_n\}$

	Roles	
$\mathcal{I}$	Atomic	R
	Inverse	R^-

Concept Axioms (TBox)	
Subclass	C ⊑ D
Equivalent	C ≡ D

	Role Axioms (RBox)	
$\mathcal{H}$	Subrole	R ⊑ S
$\mathcal{S}$	Transitivity	Trans(S)

Assertional Axioms (ABox)	
Instance	C(a)
Role	R(a,b)
Same	a = b
Different	a ≠ b

$\mathcal{S} = \mathcal{ALC}$ + Transitivity   OWL DL = $\mathcal{SHOIN}$(D) (D: Datentypen)

Wissensmodellierung (Bsp.)

$\mathbf{cow \sqsubseteq vegetarian}$
$\mathbf{vegetarian \equiv \forall eats.\neg\exists partof.animal \sqcap \forall eats.\neg animal}$
$\mathbf{madcow \equiv cow \sqcap \exists eats.(\exists partof.sheep \sqcap brain)}$
$\mathbf{sheep \sqsubseteq animal}$
$\mathbf{madcow(benita)}$

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

Schlussfolgerung: Wissensbasis ist inkonsistent!

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)

DL und FOL

- Umformungen nach FOL

$C \equiv D$	$\Leftrightarrow$	$(\forall x)\ (\ C(x) \leftrightarrow D(x)\ )$
$C \sqsubseteq D$	$\Leftrightarrow$	$(\forall x)\ (\ C(x) \rightarrow D(x)\ )$
$C \sqcap D$	$\Leftrightarrow$	$C(x) \land D(x)$
$C \sqcup D$	$\Leftrightarrow$	$C(x) \lor D(x)$
$\neg C$	$\Leftrightarrow$	$\neg C(x)$
$\forall R.C$	$\Leftrightarrow$	$(\forall y)\ (\ R(x,y) \rightarrow C(y)\ )$
$\exists R.C$	$\Leftrightarrow$	$(\exists y)\ (\ R(x,y) \land C(y)\ )$

Umformung nach FOL

cow $\sqsubseteq$ vegetarian
madcow $\equiv$ cow $\sqcap$ $\exists$eats.($\exists$partof.sheep $\sqcap$ brain)

wird zu

$$
\begin{aligned}
&(\forall x)\ (cow(x) \rightarrow vegetarian(x)) \\
&(\forall x)\ (madcow(x) \leftrightarrow \\
&\qquad (cow(x) \land (\exists y)(eats(x,y) \land \\
&\qquad\qquad (\exists z)(\ partof(y,z) \land (sheep(z) \land brain(z))\ ) \\
&\qquad\qquad) \\
&\qquad) \\
&\quad)
\end{aligned}
$$

Wichtige Inferenzprobleme

- Globale Konsistenz der Wissensbasis — $KB \models$ **false**?
  - ⇨ Ist Wissensbasis sinnvoll?
- Klassenkonsistenz — $C \equiv \bot$?
  - ⇨ Muss Klasse C leer sein?
- Klasseninklusion (Subsumption) — $C \sqsubseteq D$?
  - ⇨ Strukturierung der Wissensbasis
- Klassenäquivalenz — $C \equiv D$?
  - ⇨ Sind zwei Klassen eigentlich dieselbe?
- Klassendisjunktheit — $C \sqcap D = \bot$?
  - ⇨ Sind zwei Klassen disjunkt?
- Klassenzugehörigkeit — $C(a)$?
  - ⇨ Ist Individuum a in der Klasse C?
- Instanzgenerierung (Retrieval) — „alle X mit C(X) finden“
  - ⇨ Finde alle (bekannten!) Individuen zur Klasse C.

Rückführung auf Unerfüllbarkeit

- Klassenkonsistenz $C \equiv \bot$
  - ⇨ gdw. KB $\cup$ {C(a)} unerfüllbar (a neu)
- Klasseninklusion (Subsumption) $C \sqsubseteq D$
  - ⇨ gdw. KB $\cup$ {$(C \sqcap \neg D)(a)$} unerfüllbar (a neu)
- Klassenäquivalenz $C \equiv D$
  - ⇨ gdw. $C \sqsubseteq D$ und $D \sqsubseteq C$
- Klassendisjunktheit $C \sqcap D \sqsubseteq \bot$
  - ⇨ gdw. KB $\cup$ {$(C \sqcap D)(a)$} unerfüllbar (a neu)
- Klassenzugehörigkeit C(a)
  - ⇨ gdw. KB $\cup$ {$\neg$C(a)} unerfüllbar (a neu)
- Instanzgenerierung (Retrieval) alle C(X) finden
  - ⇨ prüfe Klassenzugehörigkeit für alle Individuen

Tableauverfahren

- Wir werden das Tableauverfahren behandeln

- Das Tableauverfahren versucht, auf "abstrakte" Weise ein Modell zu konstruieren.
  - ⇨ misslingt das, dann ist die Wissensbasis unerfüllbar
  - ⇨ gelingt es, ist sie erfüllbar

→ Rückführung der Inferenzprobleme auf das Zeigen der Unerfüllbarkeit der Wissensbasis!

Umformen in Negationsnormalform

Gegeben eine Wissensbasis W.

- Ersetze $C \equiv D$ durch $C \sqsubseteq D$ und $D \sqsubseteq C$.
- Ersetze $C \sqsubseteq D$ durch $\neg C \sqcup D$.
- Wende die Regeln auf der folgenden Folie an, bis es nicht mehr geht.

Resultierende Wissensbasis: NNF(W)

Negationsnormalform von W.

Negation steht nur noch direkt vor atomaren Klassen.

Umformen in Negationsnormalform

$\mathrm{NNF}(C) = C$, falls C atomar ist

$\mathrm{NNF}(\neg C) = \neg C$, falls C atomar ist

$\mathrm{NNF}(\neg\neg C) = \mathrm{NNF}(C)$

$\mathrm{NNF}(C \sqcup D) = \mathrm{NNF}(C) \sqcup \mathrm{NNF}(D)$

$\mathrm{NNF}(C \sqcap D) = \mathrm{NNF}(C) \sqcap \mathrm{NNF}(D)$

$\mathrm{NNF}(\neg(C \sqcup D)) = \mathrm{NNF}(\neg C) \sqcap \mathrm{NNF}(\neg D)$

$\mathrm{NNF}(\neg(C \sqcap D)) = \mathrm{NNF}(\neg C) \sqcup \mathrm{NNF}(\neg D)$

$\mathrm{NNF}(\forall R.C) = \forall R.\mathrm{NNF}(C)$

$\mathrm{NNF}(\exists R.C) = \exists R.\mathrm{NNF}(C)$

$\mathrm{NNF}(\neg\forall R.C) = \exists R.\mathrm{NNF}(\neg C)$

$\mathrm{NNF}(\neg\exists R.C) = \forall R.\mathrm{NNF}(\neg C)$

W und NNF(W) sind logisch äquivalent.

Naives Tableauverfahren

Test auf Unerfüllbarkeit

Idee:

- Gegeben Wissensbasis W.
- Versuch der Erzeugung eines Baumes (genannt *Tableau*), der ein Modell von W repräsentiert.
  (Eigentlich ein *Wald*.)
- Schlägt der Versuch fehl, ist W unerfüllbar.

Das Tableau

- Ein Tableau ist ein gerichteter beschrifteter Graph
- Knoten repräsentieren Elemente der Domäne $\Delta^{\mathcal{I}}$ des Modells
  - ⇨ Jeder Knoten x ist mit einer Menge L(x) von Klassenausdrücken beschriftet.
  - ⇨ $C \in L(x)$ steht für: "x ist in der Extension von C"
- Kanten stehen für Rollenverbindungen
  - ⇨ Jede Kante <x,y> ist mit einer Menge von Rollennamen L(<x,y>) beschriftet.
  - ⇨ $R \in L(<x,y>)$ steht für: "(x,y) ist in der Extension von R"

Initialisierung

- Input: K=TBox + ABox (in NNF)

- Ein Knoten für jedes Individuum in der ABox.
- Jeder Knoten beschriftet mit den entsprechenden Klassennamen aus der ABox.
- Eine mit R beschriftete Kante zwischen a und b falls R(a,b) in der ABox steht.

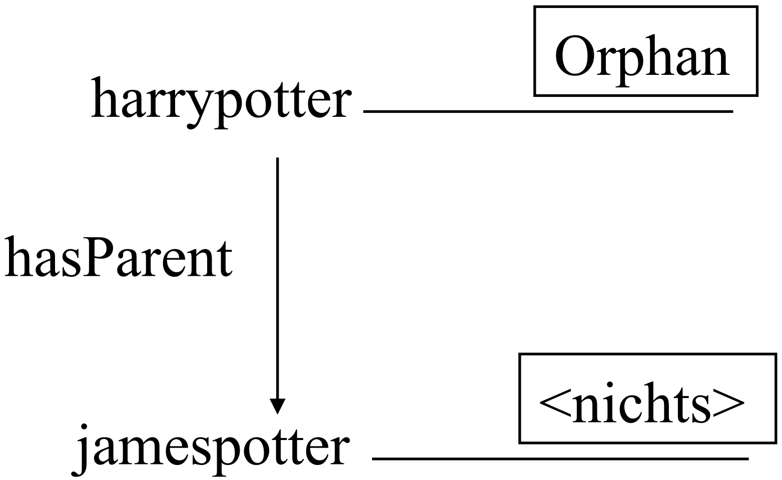
Beispiel: Initialisierung

$\text{Human} \sqsubseteq \exists \text{hasParent.Human}$

$\text{Orphan} \sqsubseteq \text{Human} \sqcap \neg \exists \text{hasParent.Alive}$

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



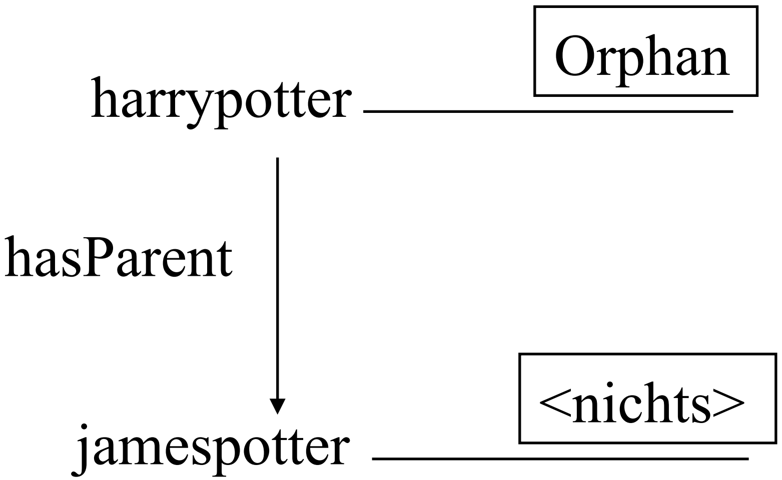
Negationsnormalform!

$\neg\text{Human} \sqcup \exists\text{hasParent.Human}$

$\neg\text{Orphan} \sqcup (\text{Human} \sqcap \forall\text{hasParent.}\neg\text{Alive})$

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)



Konstruktion des Tableaus

- Erweitere das Tableau nichtdeterministisch mit den Regeln auf der folgenden Seite.
- Terminiere, wenn
  - die Beschriftung eines Knotens einen Widerspruch enthält (d.h. Klassen C und ¬C enthält) oder
  - keine der Regeln mehr anwendbar ist
- Falls das Tableau keinen Widerspruch enthält, ist die Wissensbasis erfüllbar.
  (genauer: sind die Regeln durch geeignete Auswahl so anwendbar, dass kein Widerspruch entsteht und keine Regel mehr anwendbar ist, so ist die Wissensbasis erfüllbar)

Tableauregeln

$\sqcap$: Wenn $C \sqcap D \in L(x)$ und $\{C,D\} \not\subseteq L(x)$, dann setze

$L(x) = L(x) \cup \{C,D\}$

$\sqcup$: Wenn $C \sqcup D \in L(x)$ und $\{C,D\} \cap L(x)=\emptyset$, dann setze

$L(x) = L(x) \cup \{C\}$ oder $L(x) = L(x) \cup \{D\}$

$\exists$: Wenn $\exists R.C \in L(x)$ und x keinen R-Nachfolger y mit $C \in L(y)$ hat, dann

füge einen neuen Knoten y hinzu mit

$L(<x,y>) = \{R\}$ und $L(y) = C$

$\forall$: Wenn $\forall R.C \in L(x)$ und es gibt einen R-Nachfolger y von x mit $C \notin L(y)$, dann setze

$L(y) = L(y) \cup \{C\}$

TBox: Ist C in der TBox und $C \notin L(x)$, dann setze

$L(x) = L(x) \cup \{C\}$

Beispiel

¬Alive(jamespotter)
d.h. füge hinzu: Alive(jamespotter)
und suche Widerspruch

¬Human ⊔ ∃hasParent.Human

¬Orphan ⊔ (Human ⊓ ∀hasParent.¬Alive)

Orphan(harrypotter)

hasParent(harrypotter,jamespotter)

harrypotter

Orphan
¬Orphan ⊔ (Human ⊓ ∀hasParent.¬Alive)
1. ¬ Orphan (Widerspruch)
2. Human ⊓ ∀hasParent.¬Alive
   Human
   ∀hasParent.¬Alive

hasParent

jamespotter

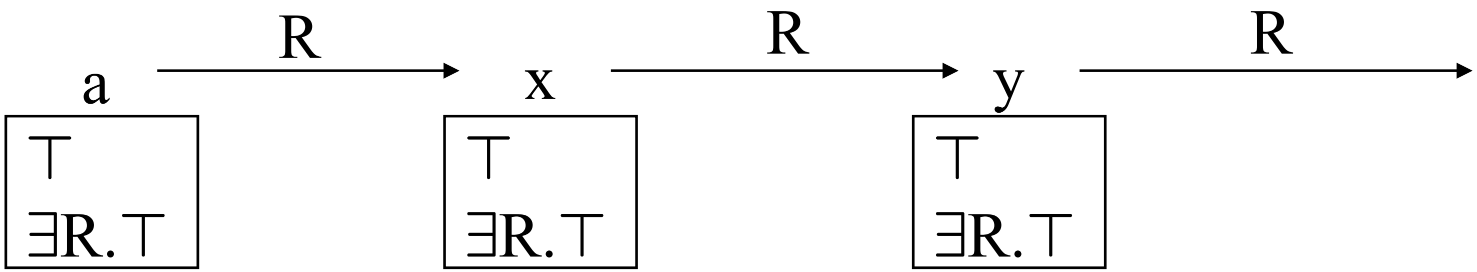
Alive
2. ¬Alive (Widerspruch)

Problem mit der Terminierung

TBox: $\exists R.\top$

ABox: $\top(a_1)$

- Ist offensichtlich erfüllbar:
  Modell M enthält Individuen $a_1^M, a_2^M, \ldots$
  und $R^M(a_i^M, a_{i+1}^M)$ für alle $i \geq 1$
- aber Tableauverfahren terminiert nicht!

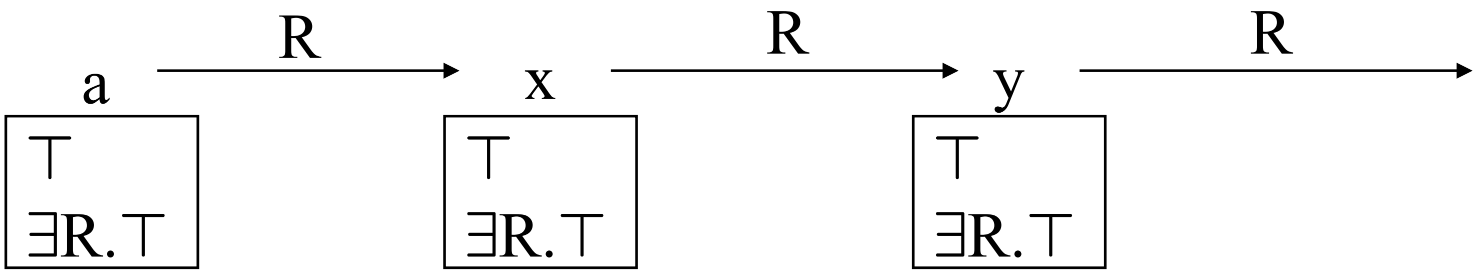


Lösung?

Eigentlich passiert ja nichts neues.

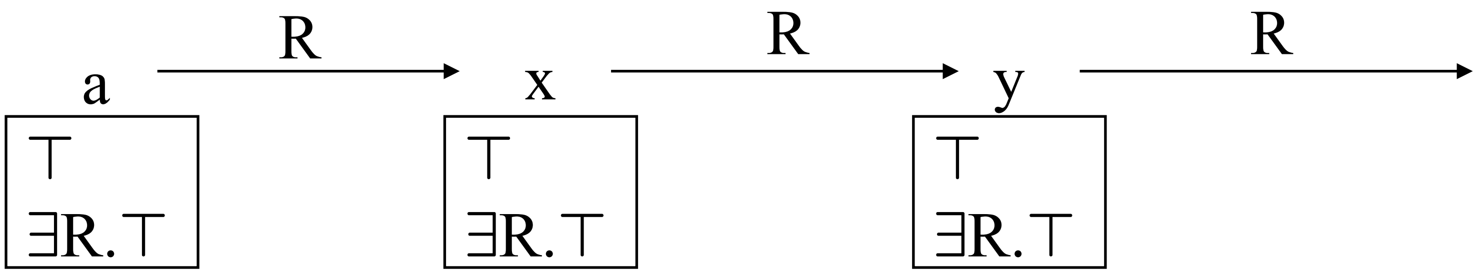
Idee: x muss nicht expandiert werden, da es eigentlich nur eine Kopie von a ist.

$\Rightarrow$ Blocking



Blocking

- x ist *geblockt* (duch y) falls
  - ⇨ y ein Vorgänger von x ist und $L(x) \subseteq L(y)$ ist
  - ⇨ oder ein Vorgänger von x geblockt ist.



x ist in diesem Beispiel durch a geblockt!

Tableau mit Blocking

- Erweitere das Tableau nichtdeterministisch mit den Regeln auf der folgenden Seite, aber wende eine Regel nur an, falls x nicht geblockt ist!
- Terminiere, wenn
  - die Beschriftung eines Knotens einen Widerspruch enthält (d.h. Klassen C und $\neg$ C enthält) oder
  - keine der Regeln mehr anwendbar ist
- Falls das Tableau keinen Widerspruch enthält, ist die Wissensbasis erfüllbar.
  (genauer: sind die Regeln durch geeignete Auswahl so anwendbar, dass kein Widerspruch entsteht und keine Regel mehr anwendbar ist, so ist die Wissensbasis erfüllbar)

Tableauregeln mit Blocking

$\sqcap$: Wenn $C \sqcap D \in L(x)$ und $\{C,D\} \not\subseteq L(x)$, dann setze
$L(x) = L(x) \cup \{C,D\}$

$\sqcup$: Wenn $C \sqcup D \in L(x)$ und $\{C,D\} \cap L(x)=\emptyset$, dann setze
$L(x) = L(x) \cup \{C\}$ oder $L(x) = L(x) \cup \{D\}$

$\exists$: Wenn $\exists R.C \in L(x)$ und x keinen R-Nachfolger y mit $C \in L(y)$ hat, dann
füge einen neuen Knoten y hinzu mit
$L(\langle x,y\rangle) = \{R\}$ und $L(y) = C$

$\forall$: Wenn $\forall R.C \in L(x)$ und es gibt einen R-Nachfolger y von x mit $C \notin L(y)$, dann setze
$L(y) = L(y) \cup \{C\}$

TBox: Ist C in der TBox und $C \notin L(x)$, dann setze
$L(x) = L(x) \cup \{C\}$

Regeln nur anwenden, falls x nicht geblockt ist!

Tableaux-Beweiser

- Fact
  - ⇨ http://www.cs.man.ac.uk/~horrocks/FaCT/
  - ⇨ SHIQ
- Fact++
  - ⇨ http://owl.man.ac.uk/factplusplus/
  - ⇨ SHOIQ(D)
- Pellet
  - ⇨ http://www.mindswap.org/2003/pellet/index.shtml
  - ⇨ SHOIN(D)
- RacerPro
  - ⇨ http://www.sts.tu-harburg.de/~r.f.moeller/racer/
  - ⇨ SHIQ(D)

Literatur

- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, York Sure. **Semantic Web – Grundlagen.** Springer 2008. (ISBN 9783540339939)
- Steffen Staab, Rudi Studer (Editors). **Handbook on Ontologies.** Springer 2003 (ISBN 3540408347).
- Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel -Schneider (eds.), **The Description Logic Handbook.** Cambridge University Press, 2007. (ISBN 9780521781763)