GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Logikbasierte Systeme der Wissensverarbeitung

Konzeptbeschreibungssprachen

Prof. Dr M. Schmidt-Schauß

SoSe 2018

Stand der Folien: 5. Juli 2018

Wissensverarbeitungssysteme

Wir betrachten die Grundlagen von
Wissenverarbeitungssystemen

Kernkomponenten:

- Wissenbasis zur Wissensrepräsentation
  - Darstellung von Fakten und Beziehungen
  - Maschinell verarbeitbar
  - Daten einfügen und löschen
- Inferenzsystem
  - Neue Schlüsse ziehen aus der Wissensbasis
  - Konsistenztest
  - Redundanztest

Semantische Netze, Frames, Beschreibungslogik

Wir betrachten im Wesentlichen:

Beschreibungslogik (Description Logics)

- „Vorgänger“: Semantische Netze und Frames
- Vorteil von DL: Eindeutige Syntax und Semantik
- Viele moderne Systeme basieren auf Beschreibungslogiken
- Z.B. OWL = Web Ontology Language basiert auf DL

Semantische Netze (1)

Semantische Netze: Gerichtete Graphen zur Darstellung von

- Konzepten (Begriffen) und
- Beziehungen zwischen den Konzepten
- im Wesentlichen:
  Unterbeziehungen, Enthaltenseinsbeziehungen

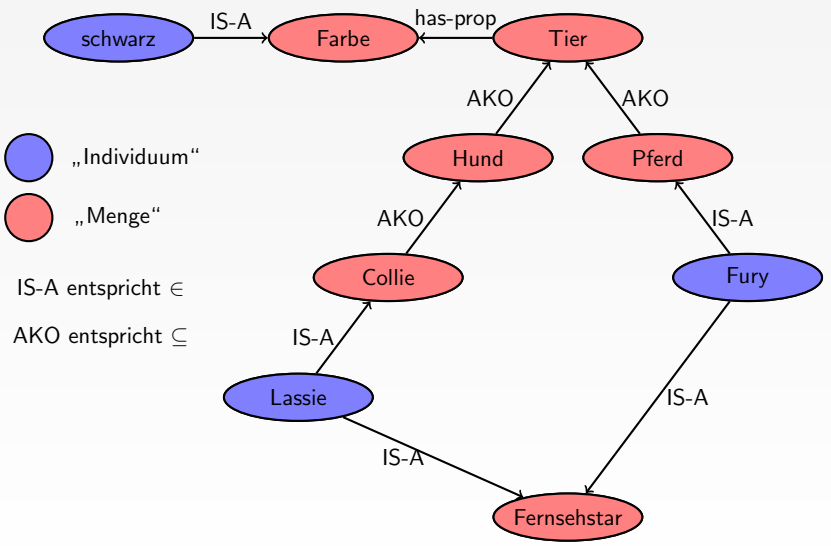
Semantische Netze (2)

Semantisches Netz: Gerichteter Graph mit

- **Knoten:** markiert mit Konzepten, Eigenschaften, Individuen
- **Gerichtete Kanten (Links):**
  geben Beziehungen (Relationen) zwischen Konzepten an.

  Wesentliche Kanten:
  - Instanzbeziehungen (IS-A)
  - Unterkonzeptbeziehungen (A-KIND-OF, AKO)
  - Eigenschaften (PROP),
  - PART-OF
  - ...

Beispiel: Semantisches Netz zu Tieren



Semantische Netze (3)

Es gibt viele verschiedene Notationen, Möglichkeiten
je nach Formalismus

Besonderheiten der Semantischen Netze:

- Repräsentieren von **Beziehungen** und **Eigenschaften** ist möglich.
- **Vererbung** von Eigenschaften über AKO-Links und über IS-A-Links

Operationen auf semantischen Netzen

- **Anfragen** der Form „Was kann fliegen?“
  können mittels **Matching von Teilgraphen** beantwortet werden
- Veränderung: Eintragung, Entfernen, Ändern von Kanten und Knoten.
- Operation: Suche nach Verbindungswegen im Netz.

Probleme der semantischen Netze

- Die **Semantik** war nur **ungenau** definiert.
- Jedes Programm arbeitete auf (s)einer **anderen** semantischen Basis.
- Z.B.: Was bedeuten jeweils zirkuläre Links?
- Darstellung als Graph wird sehr **unübersichtlich** für große Netze.

Frames

Frames (Marvin Minsky):

- Ansatz innerhalb von Repräsentationssprachen
- Analogie Frame-Sprachen und Objektorientierte Programmierung:
  - Frames entsprechen ungefähr Klassen und Objekten in objektorientierten Programmiersprachen
  - Aber: Anstatt Programme zu strukturieren, wird ein **Wissensbereich** strukturiert dargestellt
  - und: Frames haben keine Methoden nur Eigenschaften

Frames (2)

Frame: beschreibt Klasse oder Instanz

- Namen
- Oberklasse(e)
- Eigenschaften, Attribute: werden **Slots** genannt

Vererbung von Eigenschaften (Slot-Werten) von Oberklassen auf Unterklassen

Slot:

- Klasse (Wertebereich)
- Defaultwerte
- Bedingungen (z.B. Wertebereichseinschränkungen)
- Prozedurale Zusätze (z.B. Dämonen, die bei Eintragung eines Slotwertes aktiv werden)

Dies ergibt eine implizite Klassenhierarchie

Beispiel

Vogel	(Oberklasse: Wirbeltiere) (Farbe: Farbe , Gewicht: Zahl, kann-fliegen: Bool, . . . #Beine: 2)
grüne-Vögel	(Oberklasse: Vogel) (Farbe:grün)

Frames (4)

Probleme

- multiple Vererbung
- semantische Unterscheidung: Prototyp / individuelles Objekt
- logische Operatoren
- Semantik von Defaults und überschriebenen Defaults

Attributive Konzeptbeschreibungssprachen

- **Description Logics**
- Stellt eine Sprachfamilie dar, je nach Operatoren
- Kann als Nachfolger von KL-ONE gesehen werden
- KL-ONE: Wissensverarbeitungssystem basierend auf semantischen Netzen und Frames, zur Repräsentation und Verarbeitung der Semantik von natürlichsprachlichen Ausdrücken

Attributive Konzeptbeschreibungssprachen (2)

Literaturempfehlung:

Baader, Calvanese, McGuinness, Nardi, Patel-Schneider:

The Description Logic Handbook:
Theory, Implementation and Applications

Insbesondere: Kapitel 2: Baader, Nutt: Basic Description Logics

E-Book: Über UB-Lizenz (Link auf der Webseite)

Wissensverarbeitungssystem

Komponenten eines Wissenverarbeitungssystems basierend auf DL:

- **T-Box**: Legt die Terminologie des Anwendungsbereichs fest, das Vokabular
- **A-Box**: Annahmen (Assertions) über die Individuen

Analog zu Datenbanken:

- T-Box entspricht Datenbankschema
- A-Box entpricht den Daten

T-Box

T-Box besteht aus

- **Konzepten**: Repräsentieren Mengen von Individuen
- **Rollen**: Repräsentieren binäre Relationen zwischen Individuen

Atomar / Komplex

- **Atomare** Konzepte und Rollen: Nur Bezeichner / Namen
- **Komplexe** Beschreibungen von Konzepten und Rollen: Formeln, die als Atome atomare Konzepte und Rollen verwenden
- In der T-Box: Definition von Konzepten / Rollen aus atomaren Bezeichnern
  Daher: Definierte Namen und atomare Namen

Abwägungen

Viele / Mächtige Sprachkonstrukte

Vorteile:

- Wissen lässt sich einfacher ausdrücken
- Evtl. sogar: Mehr Wissen lässt sich ausdrücken

Nachteile:

- Effiziente Berechenbarkeit geht tlw. verloren
- Sogar die Entscheidbarkeit kann verloren gehen

Daher: Stets ein Kompromiss!

Typische Anfragen / Fragestellungen

Bzgl. der T-Box

- Sind die Beschreibungen erfüllbar (also nicht widersprüchlich)?
- Sind Beschreibungen in anderen Beschreibungen enthalten (Subsumption)?

Bzgl. der A-Box

- Sind die Daten konsistent? D.h. gibt es ein Modell, dass die Annahmen über die Individuen einhält.

Vorgehen im Folgenden

- Wir konzentrieren uns auf die Syntax und Semantik der DL
- T-Box und A-Box: lassen wir erstmal weg
- Stattdessen betrachten wir: Konstruktion von komplexen Beschreibungen

Danach: Inferenzen, T-Box, A-Box, etc.

Die Basissprache $\mathcal{AL}$

- $\mathcal{AL}$ = Attributive Language
- Artikel mit Gert Smolka, 1991
- Standard-Beschreibungs-Sprache von praktischem Interesse; und mit vollständigem Schlussfolgerungs-Verfahren

Notation im Folgenden:

- A, B atomare Konzepte
- C, D komplexe Konzepte
- R (atomare) Rollen

Vorher: Sprache $\mathcal{FL}$ (Levesque, Brachman, 1987.)

Syntax von $\mathcal{AL}$

Komplexe Konzeptbeschreibungen werden durch Konzept-Terme gebildet:

$C, D \in \mathcal{AL}$	$::=$	A	atomares Konzept
	$\mid$	$\top$	universelles Konzept
	$\mid$	$\bot$	leeres Konzept
	$\mid$	$\neg A$	atomares Komplement
	$\mid$	$C \sqcap D$	Schnitt
	$\mid$	$\forall R.C$	Wert-Einschränkung
	$\mid$	$\exists R.\top$	beschränkte existentielle Einschränkung

Beachte: $\exists R.C$ nicht erlaubt, nur $\exists R.\top$
Und: $\neg A$ aber nicht $\neg C$!

Beispiele

Seien **Person** und **Weiblich** atomare Konzepte, **hatKind** eine atomare Rolle.

- Weibliche Personen: **Person** $\sqcap$ **Weiblich**
- Nicht-weibliche Personen: **Person** $\sqcap \neg$**Weiblich**
- Alle Personen, die Kinder haben (Eltern):
  Person $\sqcap \exists$**hatKind**.$\top$
  Genauer:
  Person $\sqcap \exists$**hatKind**.$\top \sqcap \forall$**hatKind**.**Person**
- Alle Personen, die nur Töchter (oder keine Kinder) haben:
  Person $\sqcap \forall$**hatKind**.**Weiblich**
- Alle Personen, die keine Kinder haben:
  Person $\sqcap \forall$**hatKind**.$\bot$

Semantik von $\mathcal{AL}$

Eine Interpretation I einer $\mathcal{AL}$-Formel legt fest:

- Eine nichtleere **Menge** Δ von **Objekten**.
- Für jedes atomare Konzept A: $I(A)$ als **Teilmenge von** Δ, d.h. $I(A) \subseteq \Delta$.
- Für jede atomare Rolle R: $I(R)$ als **binäre Relation über** Δ, d.h. $I(R) \subseteq \Delta \times \Delta$.

Erweiterung von I auf komplexe Konzeptbeschreibungen:

$$
\begin{array}{lcl}
I(C_1 \sqcap C_2) & = & I(C_1) \cap I(C_2) \\
I(\bot) & = & \emptyset \\
I(\top) & = & \Delta \\
I(\neg A) & = & \Delta \setminus I(A) \\
I(\exists R.\top) & = & \{x \in \Delta \mid \exists y.(x,y) \in I(R)\} \\
I(\forall R.C) & = & \{x \in \Delta \mid \forall y.(x,y) \in I(R) \Rightarrow y \in I(C)\}
\end{array}
$$

C, D sind äquivalent ($C \equiv D$) gdw. $I(C) = I(D)$ für alle I.
D subsumiert C (Notation: $C \subseteq D$) gdw. $I(C) \subseteq I(D)$ für alle I

Beispiele: Interpretationen

Sei I:

- $\Delta = \{\text{Marie}, \text{Horst}, \text{Susi}, \text{Fritz}, \text{Lassie}\}$,
- $I(\textbf{Person}) = \Delta \setminus \{\text{Lassie}\}$
- $I(\textbf{Weiblich}) = \{\text{Marie}, \text{Susi}, \text{Lassie}\}$
- $I(\textbf{hatKind}) = \{(\text{Fritz}, \text{Susi}), (\text{Marie}, \text{Susi}), (\text{Fritz}, \text{Horst})\}$.

$$I(\textbf{Person} \sqcap \textbf{Weiblich}) = I(\textbf{Person}) \cap I(\textbf{Weiblich}) = \{\text{Marie}, \text{Susi}\}$$

$$\begin{aligned}
& I(\textbf{Person} \sqcap \exists \textbf{hatKind}.\top) \\
= \; & I(\textbf{Person}) \cap I(\exists \textbf{hatKind}.\top) \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \exists y.(x,y) \in I(\textbf{hatKind})\} \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \exists y.(x,y) \in \{(\text{Fritz}, \text{Susi}), (\text{Marie}, \text{Susi}), (\text{Fritz}, \text{Horst})\}\} \\
= \; & I(\textbf{Person}) \cap \{\text{Fritz}, \text{Marie}\} \\
= \; & \{\text{Fritz}, \text{Marie}\}
\end{aligned}$$

Beispiele: Interpretationen (2)

$$\begin{aligned}
& I(\textbf{Person} \sqcap \forall \textbf{hatKind}.\textbf{Weiblich}) \\
= \; & I(\textbf{Person}) \cap I(\forall \textbf{hatKind}.\textbf{Weiblich}) \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in I(\textbf{Weiblich})\} \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \forall y.(x,y) \in \{(\text{Fritz}, \text{Susi}), (\text{Marie}, \text{Susi}), (\text{Fritz}, \text{Horst})\} \\
& \qquad \Rightarrow y \in \{\text{Marie}, \text{Susi}, \text{Lassie}\}\} \\
= \; & I(\textbf{Person}) \cap \{\text{Marie}, \text{Horst}, \text{Susi}, \text{Lassie}\} \\
= \; & \{\text{Marie}, \text{Horst}, \text{Susi}\}
\end{aligned}$$

$$\begin{aligned}
& I(\textbf{Person} \sqcap \forall \textbf{hatKind}.\bot) \\
= \; & I(\textbf{Person}) \cap I(\forall \textbf{hatKind}.\bot) \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in I(\bot)\} \\
= \; & I(\textbf{Person}) \cap \{x \in \Delta \mid \forall y.(x,y) \in \{(\text{Fritz}, \text{Susi}), (\text{Marie}, \text{Susi}), (\text{Fritz}, \text{Horst})\} \Rightarrow y \in \emptyset\} \\
= \; & I(\textbf{Person}) \cap \{\text{Horst}, \text{Susi}, \text{Lassie}\} \\
= \; & \{\text{Horst}, \text{Susi}\}
\end{aligned}$$

Beispiel: Äquivalenz

Die Konzepte

- $(\forall \textbf{hatKind}.\textbf{Weiblich}) \sqcap (\forall \textbf{hatKind}.\textbf{Student})$ und
- $\forall \textbf{hatKind}.(\textbf{Weiblich} \sqcap \textbf{Student})$

(wobei **Student** ein neues atomares Konzept ist), sind äquivalente Konzepte:

$$\begin{aligned}
& I(\forall \textbf{hatKind}.\textbf{Weiblich} \sqcap \forall \textbf{hatKind}.\textbf{Student}) \\
= \; & I(\forall \textbf{hatKind}.\textbf{Weiblich}) \cap I(\forall \textbf{hatKind}.\textbf{Student}) \\
= \; & \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in I(\textbf{Weiblich})\} \\
& \cap \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in I(\textbf{Student})\} \\
= \; & \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow (y \in I(\textbf{Weiblich}) \wedge y \in I(\textbf{Student}))\} \\
= \; & \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in (I(\textbf{Weiblich}) \cap I(\textbf{Student}))\} \\
= \; & \{x \in \Delta \mid \forall y.(x,y) \in I(\textbf{hatKind}) \Rightarrow y \in (I(\textbf{Weiblich} \sqcap \textbf{Student}))\} \\
= \; & I(\forall \textbf{hatKind}.(\textbf{Weiblich} \sqcap \textbf{Student}))
\end{aligned}$$

Beweis ist unabhängig von den atomaren Konzepten und Rollen, allgemein gilt: $(\forall R.C) \sqcap (\forall R.D) \quad \equiv \quad \forall R.(C \sqcap D)$

Erweiterungen der Basisprache $\mathcal{AL}$ (1)

Atomare Funktionen

- weitere Bezeichnungen: Attribute, Features, Attributsbezeichner, funktionale Rollen
- Werden wie Rollen verwendet
- Aber: In der Semantik werden sie auf partielle **einstellige Funktionen** $f : \Delta \to \Delta$ abgebildet.
- D.h. semantisch sind es rechtseindeutige binäre Relationen (für alle $x, y, z \in \Delta$: Wenn $(x, y) \in f$ und $(x, z) \in f$ muss gelten $y = z$)
- Syntaktisch: Keine Änderung außer Trennung der Namen in: Konzepte, Rollen, Funktionen

Beispiele:

- **istMutterVon** (totale Funktion)
- **istEhepartner** (partielle Funktion, wenn Polygamie verboten)

Erweiterungen (2)

Mögliche weitere Konstrukte zur Konstruktion komplexer Konzeptbeschreibungen:

$C ::= \ldots$

$\mid \neg C$	Komplement
$\mid (C_1 \sqcup C_2)$	Vereinigung
$\mid (\exists R.C)$	existenzielle Einschränkung
$\mid (\leq n\ R), (\geq n\ R)$	Anzahlbeschränkung, number restrictions
$\mid (\leq n\ R.C), (\geq n\ R.C)$	qualifizierte Anzahlbeschränkung
$\mid (R_1; R_2; \ldots; R_n = R'_1; R'_2; \ldots; R'_m)$	Pfadgleichung, Attributsübereinstimmung

R, R_i dabei Rolle oder atomare Funktion

Erweiterungen (3)

Konstrukte, um **komplexe Rollen** zu konstruieren
(es gibt noch weitere):

(RESTRICT $R\ C$) Rolleneinschränkung

Symbole

In der Literatur werden teilweise andere Symbole verwendet, z.B.:

AND	entspricht	$\sqcap$
OR	entspricht	$\sqcup$
NOT	entspricht	$\neg$
SOME	entspricht	$\exists$
ALL	entspricht	$\forall$
$\ldots$		

Beispiele (1)

Atomar: **Mensch**, **Frau**, **Mann**, **hatKind** und **studiertFach**
Man hätte gerne als Axiome:

Frau $\sqcap$ **Mann** $= \bot$

Mensch $\equiv$ **Frau** $\sqcup$ **Mann**

Beispiele (2)

$$\textbf{Eltern} := \textbf{Mensch} \sqcap (\exists \textbf{hatKind}.\textbf{Mensch})$$

Genauer:

$$\textbf{Eltern} := \textbf{Mensch} \sqcap (\exists \textbf{hatKind}.\textbf{Mensch}) \sqcap (\forall \textbf{hatKind}.\textbf{Mensch})$$

Weitere Konzepte:

$\textbf{Mutter} := \textbf{Frau} \sqcap \textbf{Eltern}$

$\textbf{Studentin} := \textbf{Frau} \sqcap (\exists \textbf{studiertFach}.\top)$

Mögliche Inferenz: $\textbf{Mutter} \subseteq \textbf{Frau}$.

Beispiele (3)

Beispiel mit Anzahlbeschränkungen:

$$\begin{aligned}\textbf{Bigamist} \quad := \quad & \textbf{Mann} \sqcap (\geq\ 2\ \textbf{verheiratetMit}) \\ & \sqcap (\leq\ 2\ \textbf{verheiratetMit}) \sqcap (\forall \textbf{verheiratetMit}.\textbf{Frau})\end{aligned}$$

Beachte: Ist **verheiratetMit** eine atomare Funktion, dann liefert die Semantik $I(\textbf{Bigamist}) = \emptyset$ für jede Interpretation I.

Beispiele (4)

Beispiele mit Pfadgleichungen:

Rolle **isstGerne**, dann beschreibt:

$$\textbf{Mensch} \sqcap (\textbf{verheiratetMit}; \textbf{isstGerne} = \textbf{isstGerne})$$

alle Menschen deren Ehepartner das gleiche Essen mögen, wie sie selbst

Atomare Funktionen **hatNachnamen** und **hatMutter**, dann beschreibt:

$$\textbf{Mann} \sqcap (\textbf{hatNachnamen} = \textbf{hatMutter}; \textbf{hatNachnamen})$$

alle Männer, die denselben Nachnamen wie ihre Mutter haben

Semantik der erweiterten Sprachen (1)

Interpretation legt fest:

- Grundmenge Δ,
- Für jedes atomare Konzept $I(A) \subseteq \Delta$,
- Für jede atomare Rolle $I(R) \subseteq \Delta \times \Delta$
- Für jede atomare Funktion $I(R) : \Delta \to \Delta$ (einstellige partielle Funktion)

Da Funktionen auch Relationen sind, braucht man diese ansonsten nicht extra beachten.

Semantik der erweiterten Sprachen (2)

Erweiterung von I auf die neuen Konstrukte:

$$
\begin{aligned}
I(C_1 \sqcup C_2) &= I(C_1) \cup I(C_2) \\
I(\exists R.C) &= \{x \in \Delta \mid \exists y.(x,y) \in I(R) \text{ und } y \in I(C)\} \\
I(\leq\ n\ R) &= \{x \in \Delta \mid |\{y \in \Delta \mid (x,y) \in I(R)\}| \leq n\} \\
I(\geq\ n\ R) &= \{x \in \Delta \mid |\{y \in \Delta \mid (x,y) \in I(R)\}| \geq n\} \\
I(\leq\ n\ R.C) &= \{x \in \Delta \mid |\{y \in \Delta \mid (x,y) \in I(R) \wedge y \in I(C)\}| \leq n\} \\
I(\geq\ n\ R.C) &= \{x \in \Delta \mid |\{y \in \Delta \mid (x,y) \in I(R) \wedge y \in I(C)\}| \geq n\}
\end{aligned}
$$

Semantik der erweiterten Sprachen (3)

Erweiterung von I auf die neuen Konstrukte (Fortsetzung):

$$I(R_1; R_2; \ldots; R_n = R'_1; R'_2; \ldots; R'_m)$$

$$= \left\{ x \in \Delta \;\middle|\; \begin{array}{r} \{y \in \Delta \mid (x,y) \in (I(R_1) \circ \ldots \circ I(R_n))\} \\ = \{y \in \Delta \mid (x,y) \in (I(R'_1) \circ \ldots \circ I(R'_m))\} \end{array} \right\}$$

wobei $\circ$ die Komposition von Relationen ist
(D.h. $R_1 \circ R_2 = \{(x,z) \mid (x,y) \in R_1 \wedge (y,z) \in R_2\}$)

$$I(\text{RESTRICT } R\ C)$$

$$= \{(x,y) \in \Delta \times \Delta \mid (x,y) \in I(R) \text{ und } y \in I(C)\}$$

Beispiel

Zeige $(\exists R.C) \equiv (\exists(\text{RESTRICT } R\ C).\top)$

Linke Seite:

$$I(\exists R.C) = \{x \mid \exists y.(x,y) \in I(R) \wedge y \in I(C)\}$$

Rechte Seite:

$$
\begin{aligned}
& \{x \mid \exists y.(x,y) \in I(\text{RESTRICT } R\ C)\} \\
= \ & \{x \mid \exists y.(x,y) \in \{(a,b) \mid (a,b) \in I(R) \wedge b \in I(C)\} \\
= \ & \{x \mid \exists y.(x,y) \in \{(a,b) \in I(R) \mid b \in I(C)\} \\
= \ & \{x \mid \exists y.(x,y) \in I(R) \wedge y \in I(C)\}
\end{aligned}
$$

Das zeigt: $\exists R.C$ unnötig, wenn man RESTRICT und $\exists R.\top$ hat.

Es gilt auch umgekehrt, dass man andersherum auf RESTRICT verzichten kann.

Namensgebung der $\mathcal{AL}$-Sprachen

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}]$$

wobei

- $\mathcal{U}$ Union: Vereinigung ($\sqcup$)
- $\mathcal{E}$ Volle existenzielle Beschränkung ($\exists R.C$)
- $\mathcal{N}$ number restriction: Anzahlbeschränkung. $((\leq\ n\ R), (\geq\ n\ R))$
- $\mathcal{C}$ Volles Komplement ($\neg C$)

Z.B. $\mathcal{ALEN}$: $\mathcal{AL}$ + $\exists R.C$ + number restrictions.

Familie der $\mathcal{AL}$-Sprachen

- Nicht alle Sprachen sind (semantisch) verschieden.
- Z.B. $\mathcal{ALCE} = \mathcal{ALC}$.
  Eine Richtung ist einfach, denn es gilt:

$$\begin{array}{lcll} C \sqcup D & \equiv & \neg(\neg C \sqcap \neg D) & \text{und} \\ \exists R.C & \equiv & \neg(\forall R.\neg C). & \end{array}$$

Beweis für die letzte Gleichung:

$$\begin{aligned} I(\neg(\forall R.\neg C)) &= \Delta \setminus \{a \in \Delta \mid \forall b.(a,b) \in I(R) \Rightarrow y \in I(\neg C)\} \\ &= \Delta \setminus \{a \in \Delta \mid \forall b.(a,b) \in I(R) \Rightarrow b \in (\Delta \setminus I(C))\} \\ &= \{a \in \Delta \mid \neg(\forall b.(a,b) \in I(R) \Rightarrow b \in (\Delta \setminus I(C)))\} \\ &= \{a \in \Delta \mid \exists b.\neg((a,b) \in I(R) \Rightarrow b \in (\Delta \setminus I(C)))\} \\ &= \{a \in \Delta \mid \exists b.(a,b) \in I(R) \wedge \neg(b \in (\Delta \setminus I(C)))\} \\ &= \{a \in \Delta \mid \exists b.(a,b) \in I(R) \wedge (b \in I(C))\} \\ &= I(\exists R.C) \end{aligned}$$

$\mathcal{ALCE} = \mathcal{ALC}$

Gleichheit $\mathcal{ALCE} = \mathcal{ALC}$, andere Richtung:
Z.B. Stelle Komplemente in $\mathcal{ALCE}$ dar:

- Atomare Komplemente $\neg A$: sind bereits vorhanden.
- $\neg(C \sqcup D)$: dasselbe wie $\neg C \sqcap \neg D$, dann Induktion.
- $\neg(C \sqcap D)$: dasselbe wie $\neg C \sqcup \neg D$, dann Induktion.
- $\neg(\forall R.C)$ darstellbar als $\exists R.\neg C$, dann Induktion.
- $\neg(\exists R.C)$ darstellbar als $\forall R.\neg C$, dann Induktion.

Familie der $\mathcal{AL}$-Sprachen (2)

Insgesamt erhält man 8 verschiedene Sprachen:

$$\mathcal{AL}, \mathcal{ALC}, \mathcal{ALCN}, \mathcal{ALN}, \mathcal{ALU}, \mathcal{ALE}, \mathcal{ALUN}, \mathcal{ALEN}$$

Name	Konstrukte	implizite Konstrukte
$\mathcal{AL}$	$\sqcap, \bot, \top, \neg A, \forall R.C, \exists R.\top$	
$\mathcal{ALC}$	$\mathcal{AL}$, $\neg C$	$\sqcup$,$\exists R.C$
$\mathcal{ALCN}$	$\mathcal{AL}$, $\neg C$, $(\leq\ n\ R)$	$\sqcup$, $\exists R.C$, $(\geq\ n\ R)$
$\mathcal{ALN}$	$\mathcal{AL}$, $(\leq\ n\ R)$ $(\geq\ n\ R)$	
$\mathcal{ALU}$	$\mathcal{AL}$, $\sqcup$	
$\mathcal{ALE}$	$\mathcal{AL}$, $\exists R.C$	
$\mathcal{ALUN}$	$\mathcal{AL}$, $\sqcup$, $(\leq\ n\ R)$, $(\geq\ n\ R)$	
$\mathcal{ALEN}$	$\mathcal{AL}$, $\exists R.C$, $(\leq\ n\ R)$, $(\geq\ n\ R)$	

$\mathcal{ALC}$ und $\mathcal{ALCN}$ sind sehr allgemein!
Z.B. kann Aussagenlogik kodiert werden: $\neg \leftrightarrow \neg, \wedge \leftrightarrow \sqcap, \vee \leftrightarrow \sqcup$

Weitere Sprachen: Die $\mathcal{FL}$-Familie

$\mathcal{FL}$ = frame-based description language
Levesque, Brachman 1987: Expressiveness and tractability in knowledge representation and reasoning.
Drei Varianten:

Name	Konstrukte
$\mathcal{FL}_0$	$\sqcap, \forall R.C$
$\mathcal{FL}^-$	$\sqcap, \forall R.C, \exists R.\top$
$\mathcal{FL}$	$\sqcap, \forall R.C, \exists R.C$

Beziehungen zwischen $\mathcal{FL}$-Sprachen und $\mathcal{AL}$-Sprachen:

$$\begin{array}{lcl} \mathcal{AL} & \equiv & \mathcal{FL}^- \cup \{\neg A, \top\} \\ \mathcal{ALC} & \equiv & \mathcal{FL}^- \cup \{\neg C\} \\ \mathcal{ALC} & \equiv & \mathcal{FL} \cup \{\neg C\} \end{array}$$

Schnitt von Rollen

Erweiterung um Schnitt von Rollen (Buchstabe $\mathcal{R}$)

$$\mathcal{R}: \quad R_1 \sqcap R_2 \quad \text{Schnitt von Rollen}$$

Namensgebung dann:

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}][\mathcal{R}]$$

Ergibt 8 weitere Sprachen in Erweiterung der $\mathcal{ALUEN}$-Sprachen

Aber z.B. $\mathcal{ALUCR} = \mathcal{ALCR} = \mathcal{ALECR}$

Interessierende Eigenschaften

Definition

Seien C, D (evtl. komplexe) Konzepte

- Ein Konzept D **subsumiert** ein Konzept C (geschrieben als $C \sqsubseteq D$), gdw. für alle Interpretationen I gilt: $I(C) \subseteq I(D)$.
- Ein Konzept C ist **konsistent** gdw. es eine Interpretation I gibt, so dass gilt: $I(C) \neq \emptyset$. Gibt es keine solche Interpretation, so nennt man C **inkonsistent**.
- Zwei Konzepte C und D sind **disjunkt**, gdw. für alle Interpretationen I gilt: $I(C) \cap I(D) = \emptyset$.
- Zwei Konzepte C und D sind **äquivalent** ($C \equiv D$) gdw. für alle Interpretationen I gilt: $I(C) = I(D)$.

Beachte: Für Konsistenz genügt **eine** Interpretation.

Äquivalenz der Eigenschaften

Je nach Sprache kann gelten:

- C ist inkonsistent, gdw. $C \equiv \bot$.
- $C \equiv D$ gdw. $C \sqsubseteq D$ und $D \sqsubseteq C$.
- C ist disjunkt zu D gdw. $C \sqcap D$ inkonsistent.
- C inkonsistent, gdw. C wird von $\bot$ subsumiert ($C \sqsubseteq \bot$)
- Wenn allgemeine Komplemente erlaubt, dann gilt: $C \sqsubseteq D$ gdw. $C \sqcap \neg D$ inkonsistent ist.
- Wenn allgemeine Komplemente erlaubt, dann gilt: $C \sqsubseteq D$ gdw. C und $\neg D$ disjunkt sind.

Äquivalenz der Eigenschaften (2)

Satz

Wenn die Sprache allgemeine Komplemente erlaubt, dann sind Subsumtion, Disjunktheitstest und Konsistenztest äquivalent und haben auch gleiche Komplexität. Dies gilt z.B. in der Sprache $\mathcal{ALC}$.

Aber: In der Sprache $\mathcal{FL}$ sind das verschiedene Fragestellungen.

Anwendung der DL (Auswahl)

Ontologien der Life-Sciences (Medizin, Biologie)

- SNOMED CT (ein Akronym für Systematized Nomenclature of Medicine – Clinical Terms) ist eine medizinische Nomenklatur, die ca. 500.000 Konzepte umfasst
- The National Cancer Institute Thesaurus , 45.000 Konzepte
- GO: gene ontology: ca. 20.000 Konzepte
- „ GALEN was concerned with the computerisation of clinical terminologies.“

Anwendung der DL (Auswahl) (2)

Ausdrucksschwache Beschreibungslogik $\mathcal{EL}$:

$$C ::= \top \mid A \mid C \sqcap D \mid \exists R.C$$

Fragestellung:

- Überprüfung der Konsistenz einer Menge von Konzepten
- Überprüfung der Erweiterbarkeit einer Menge von Konzepten.

Die Subsumption in $\mathcal{EL}$ ist polynomiell und damit anwendungsgeeignet.
Beispiel (das so nicht gewollt war, aber stimmte):

„**Finger-Amputation** $\sqsubseteq$ **Arm-Amputation**“,

Terminologien

- Terminologie = Vereinbarung von Namen für Konzepte
- In der allgemeinsten Form in $\mathcal{AL}$:
  Menge von terminologischen Axiomen der Formen
  $C \sqsubseteq D$ oder $C \equiv D$ wobei C, D Konzepte sind.
- Wenn es Rollenterme gibt, dann auch Axiome der Form
  $R \sqsubseteq S$ oder $R \equiv S$ wobei R, D Rollen

Modelle

Definition

Eine Interpretation I **erfüllt** eine Menge T von terminologischen Axiomen wenn:

- für alle $C \sqsubseteq D$ (bzw. $R \sqsubseteq S$) $\in T$ gilt:
  $I(C) \subseteq I(D)$ (bzw. $I(R) \subseteq I(S)$)
- für alle $C \equiv D$ (bzw. $R \equiv S$) gilt:
  $I(C) = I(D)$ (bzw. $I(R) = I(S)$)

In diesem Fall sagen wir: I ist ein **Modell** für T.

T-Box

Menge von terminologischen Axiomen ist eine **Menge von Definitionen** wenn alle Axiome von der Form sind:

$$A = C \text{ (bzw. } (R = S)),$$

wobei A (bzw. R) atomares Konzept (atomare Rolle) ist.

T-Box

Eine Menge von Definitionen, wobei jeder Name höchstens einmal links vorkommt, nennt man **T-Box**

- Eine T-Box ist **zyklisch**, wenn Namen durch sich selbst definiert sind (evtl. implizit)
- Anderenfalls ist T-Box **azyklisch**

T-Box (2)

Bei **azyklischen T-Boxen** kann man alle definierten Namen durch Definitionseinsetzung eliminieren!
Evtl.: exponentielle Vergrößerung der Konzeptterme

Interpretation bei **azyklischen** T-Boxen:

- Starte mit I_0 der nicht-definierten Symbole
- Dann ist die Erweiterung I von I_0 auf die definierten Namen **eindeutig** und ein Modell

Bei **zyklischen Definitionen**: geht nur unter Einschränkungen!

Beispiel für eine T-Box

Frau	$\equiv$	**Person** $\sqcap$ **Weiblich**
Mann	$\equiv$	**Person** $\sqcap \neg$**Frau**
Mutter	$\equiv$	**Frau** $\sqcap \exists$**hatKind**.**Person**
Vater	$\equiv$	**Mann** $\sqcap \exists$**hatKind**.**Person**
Eltern	$\equiv$	**Vater** $\sqcup$ **Mutter**
Grossmutter	$\equiv$	**Mutter** $\sqcap \exists$**hatKind**.**Eltern**
MutterMitVielenKindern	$\equiv$	**Mutter** $\sqcap (\geq 3$ **hatKind**$)$
MutterMitTochter	$\equiv$	**Mutter** $\sqcap (\forall$**hatKind**.**Frau**$)$
Ehefrau	$\equiv$	**Frau** $\sqcap (\exists$**hatEhemann**.**Mann**$)$

Diese T-Box ist azyklisch

Nicht definierte Namen und definierte Namen

Beispiel: Interpretation

Basisinterpretation I_0 legt nur fest

Δ, I_0(**Person**), I_0(**Weiblich**), I_0(**hatKind**), I_0(**hatEhemann**)

Modell I für die T-Box erhält man nun durch:

$$\begin{aligned}
I(\textbf{Person}) &:= I_0(\textbf{Person}) \\
I(\textbf{Weiblich}) &:= I_0(\textbf{Weiblich} \\
I(\textbf{hatKind}) &:= I_0(\textbf{hatKind}) \\
I(\textbf{hatEhemann}) &:= I_0(\textbf{hatEhemann}) \\
I(\textbf{Frau}) &:= I_0(\textbf{Person}) \cap I_0(\textbf{Weiblich}) \\
I(\textbf{Mann}) &:= I_0(\textbf{Person}) \cap (\Delta \setminus (I_0(\textbf{Person}) \cap I_0(\textbf{Weiblich})))
\end{aligned}$$

usw.

Beispiele für zyklische T-Boxen

Menschen sind alle Tiere, deren Eltern nur Menschen sind:

$$\textbf{Mensch'} \equiv \textbf{Tier} \sqcap \forall \textbf{HatEltern}.\textbf{Mensch'}$$

Gibt es ein Modell?

- Sei $\Delta, I_0(\textbf{Tier}), I_0(\textbf{HatEltern})$ festgelegt
- Idee von vorher:
  $I(\textbf{Mensch'}) := I_0(\textbf{Tier}) \cap \{x \mid \forall y.(x,y) \in I_0(\textbf{HatEltern}) \implies y \in I(\textbf{Mensch'})\}$
- Das ist eine **rekursiv definierte Menge**!
- Man benötigt eine Interpretation (als Menge) $I(\textbf{Mensch'}) \subseteq \Delta$, die ein **Fixpunkt** der rekursiven Gleichung ist
- Solch ein Fixpunkt muss nicht existieren und ist nicht immer eindeutig!

Weitere Beispiele

„Mann, der nur Söhne hat, und für dessen Söhne das gleiche gilt“.

$$\textbf{MnurS} \equiv \textbf{Mann} \sqcap \forall \textbf{hatKind}.\textbf{MnurS}$$

Datenstruktur: Binärer Baum:

$$\textbf{BinBaum} \equiv \textbf{Baum} \sqcap (\leq\ 2\ \textbf{hatAst}) \sqcap \forall \textbf{hatAst}.\textbf{BinBaum}$$

Sinnvoll: **kleinster Fixpunkt**

Beispiel: Kein Fixpunkt

T-Box:

$$A \equiv \neg A$$

Es gibt keine Interpretation:

- $\Delta \neq \emptyset$ ist Voraussetzung der Semantik
- $I(A) = \Delta \setminus I(A)$ ist daher nie erfüllt.

Fixpunktsemantik für zyklische T-Boxen

Basisinterpretation I_B der nicht-definierten Namen.

Kleinster Fixpunkt:
Folge von Interpretationen $I_i, i = 0, 1, 2, \ldots$, die I_B auf die definierten Namen erweitert:

1. $I_0(A) = \emptyset$ für alle definierten Namen A.
2. $I_{i+1}(A) := I_i(C_A)$ für alle i und jede Definition $A \equiv C_A$.
3. Wenn die Folge monoton steigend ist:
   $$I_i(A) \subseteq I_{i+1}(A) \text{ für alle Namen } A,$$
   dann kann man $I_\infty(A) = \bigcup_i I_i(A)$ definieren.
   Das ergibt einen kleinsten Fixpunkt.

Fixpunktsemantik für zyklische T-Boxen (2)

Basisinterpretation I_B der nicht-definierten Namen.

Größter Fixpunkt:
Folge von Interpretationen $I_i, i = 0, 1, 2, \ldots$, die I_B auf die definierten Namen erweitert:

1. $I_0(A) = \Delta$ für alle definierten Namen A.
2. $I_{i+1}(A) := I_i(C_A)$ für alle i und jede Definition $A \equiv C_A$.
3. Wenn die Folge monoton fallend ist:

   $$I_{i+1}(A) \subseteq I_i(A) \text{ für alle Namen } A,$$

   dann kann man $I_\infty(A) = \bigcap_i I_i(A)$ definieren.

   Das ergibt einen größten Fixpunkt.

Beispiele

T-Box: $A \equiv \neg A$.
Kleinster Fixpunkt:

$$\begin{aligned} I_0(A) &= \emptyset \\ I_1(A) &= I_0(\neg A) = \Delta \setminus I_0(A) = \Delta \\ I_2(A) &= I_1(\neg A) = \Delta \setminus I_1(A) = \emptyset \end{aligned}$$

Nichtmonotonie: da $\Delta = I_1(A) \not\subseteq I_2(A) = \emptyset$.
Größter Fixpunkt:

$$\begin{aligned} I_0(A) &= \Delta \\ I_1(A) &= I_0(\neg A) = \Delta \setminus I_0(A) = \emptyset \\ I_2(A) &= I_0(\neg A) = \Delta \setminus I_1(A) = \Delta \end{aligned}$$

Da $I_2(A) \not\subseteq I_1(A)$ ist die Monotonie verletzt.

Beispiele (2)

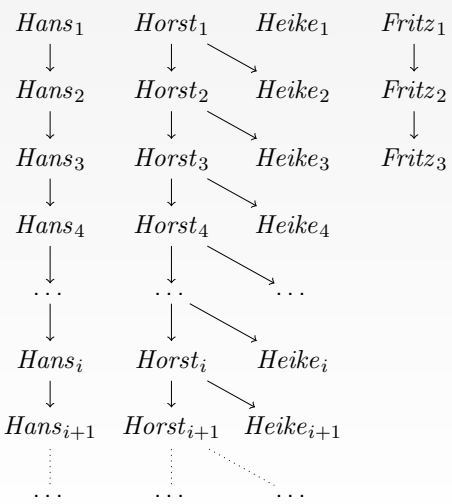
T-Box: **MnurS** $\equiv$ **Mann** $\sqcap \forall$**hatKind**.**MnurS**

I_B sei:

$$\begin{aligned} \Delta &= \{Hans_i \mid i \in \mathbb{N}\} \\ &\quad \cup \{Horst_i \mid i \in \mathbb{N}\} \\ &\quad \cup \{Heike_i \mid i \in \mathbb{N}\} \\ &\quad \cup \{Fritz_i \mid i \in \{1,2,3\}\} \\ I_B(\textbf{Mann}) &= \{Hans_i \mid i \in \mathbb{N}\} \\ &\quad \cup \{Horst_i \mid i \in \mathbb{N}\} \\ &\quad \cup \{Fritz_i \mid i \in \{1,2,3\}\} \\ I_B(\textbf{hatKind}) &= \{(Hans_i, Hans_{i+1}) \mid i \in \mathbb{N}\} \\ &\quad \cup \{(Horst_i, Horst_{i+1}) \mid i \in \mathbb{N}\} \\ &\quad \cup \{(Horst_i, Heike_{i+1}) \mid i \in \mathbb{N}\} \\ &\quad \cup \{(Fritz_i, Fritz_{i+1}) \mid i \in \{1,2\}\} \end{aligned}$$

Beispiele (3)

Als Graph (gerichtete Kante = „hatKind“)



Beispiele (4)

Kleinster Fixpunkt:

$I_0(\textbf{MnurS}) = \emptyset$

$$\begin{aligned} I_1(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_0(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i \mid i \in \mathbb{N}\} \cup \{Fritz_3\}) = \{Fritz_3\} \end{aligned}$$

$$\begin{aligned} I_2(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_1(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i \mid i \in \mathbb{N}\} \cup \{Fritz_2, Fritz_3\}) = \{Fritz_2, Fritz_3\} \end{aligned}$$

$$\begin{aligned} I_3(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_2(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i \mid i \in \mathbb{N}\} \cup \{Fritz_1, Fritz_2, Fritz_3\}) \\ &= \{Fritz_1, Fritz_2, Fritz_3\} \end{aligned}$$

$$\begin{aligned} I_4(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_3(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i \mid i \in \mathbb{N}\} \cup \{Fritz_1, Fritz_2, Fritz_3\}) \\ &= \{Fritz_1, Fritz_2, Fritz_3\} \end{aligned}$$

$I_j(\textbf{MnurS}) = \{Fritz_1, Fritz_2, Fritz_3\}$ für alle weiteren i

Das ergibt $\bigcup_i I_i(\textbf{MnurS}) = \{Fritz_1, Fritz_2, Fritz_3\}$.

Beispiele (5)

Größter Fixpunkt:

$I_0(\textbf{MnurS}) = \Delta$

$$\begin{aligned} I_1(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_0(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\Delta) \\ &= I_B(\textbf{Mann}) \end{aligned}$$

$$\begin{aligned} I_2(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_1(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_B(\textbf{Mann})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i, Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}) \\ &= \{Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}\} \end{aligned}$$

$$\begin{aligned} I_3(\textbf{MnurS}) &= I_B(\textbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x,y) \in I_B(hatKind) \Rightarrow y \in I_2(\textbf{MnurS})\} \\ &= I_B(\textbf{Mann}) \cap (\{Heike_i, Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}) \\ &= \{Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}\} \end{aligned}$$

$I_j(\textbf{MnurS}) = \{Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}\}$ für alle weiteren i

Das ergibt $\bigcap_i I_i(\textbf{MnurS}) = \{Hans_i \mid i \in \mathbb{N}\} \cup \{Fritz_i \mid i \in \{1,2,3\}\}$.

Fixpunktsemantik

Theorem

Ist die Terminologie ohne Komplemente definiert, dann kann man sowohl einen kleinsten als auch einen größten Fixpunkt der Interpretationen als Erweiterung einer Basisinterpretation definieren.

Begründung: Es gilt die Monotonie:

$$I \subseteq I' \Rightarrow I(C) \subseteq I'(C)$$

wobei $I \subseteq I'$: $\forall$atomare Konzepte $A : I(A) \subseteq I'(A)$.
Die Monotonie gilt, da $\sqcap, \sqcup, \forall R.C, \exists R.C, (\geq\ n\ R)$ alle monoton im Konzept-Argument sind.

Beachte: Wenn man über $\mathcal{ALCN}$ hinausgeht, ist das Konstrukt $(\geq\ n\ R.C)$ monoton, während $(\leq\ n\ R.C)$ nicht monoton ist.

Fixpunktsemantik (2)

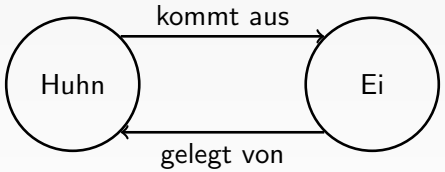
Theorem

Ist die Terminologie so definiert, dass jeder zyklische Pfad durch die Terme durch eine gerade Anzahl Negationen geht, denn kann man sowohl einen kleinsten als auch einen größten Fixpunkt der Interpretationen als Erweiterung einer Basisinterpretation definieren.

Begründung: Montonie, da „Doppelnegation“

Beispiel

„Jedes Huhn kommt aus einem Ei. Jedes Ei wurde von einem Huhn gelegt."



Beispiel (2)

T-Box dazu:

$$\begin{aligned} \textbf{Huhn} &\sqsubseteq (\exists \textbf{kommtAus}.\textbf{Ei}) \\ \textbf{Ei} &\sqsubseteq (\exists \textbf{gelegtVon}.\textbf{Huhn}) \end{aligned}$$

Versuche ein Modell zu finden, so dass $Clarissa \in I(\textbf{Huhn})$:

1. Annahme $Clarissa \in I(\textbf{Huhn})$, dann muss gelten:
   Es gibt Objekt $ClarissaEi$ mit $(Clarissa, ClarissaEi) \in I(\textbf{kommtAus})$.
2. Jetzt muss wiederum gelten $ClarissaEi \in I(\textbf{Ei})$, daher:
   Es gibt $ClarissaEi$ mit $(ClarissaEi, ClarissaMutter) \in I(\textbf{gelegtVon})$. Jetzt muss wiederum gelten $ClarissaMutter \in I(\textbf{Huhn})$ usw.

Fazit:

- In allen endlichen Modellen mit $I(\textbf{Huhn}) \neq \emptyset$ gibt es nur Hühner, die ihre eigene Vorfahren sind.
- Wenn das Modell unendlich sein darf, dann kann es auch Hühner geben, die nicht ihre eigenen Vorfahren sind.

T-Box mit Inklusionen

Erweiterung der T-Box:

Axiome der Form $A \sqsubseteq C$, mit A Konzeptname.

Zurückführen auf normale T-Boxen:

- Bedingung: Jeder Name kommt höchstens einmal links vor
- Zu jeder Inklusion $A \sqsubseteq C_A$ erfinde einen neuen Namen $\hat{A}$.
- Ersetze die Inklusion $A \sqsubseteq C_A$ durch die Definition $A \equiv \hat{A} \sqcap C_A$.

Satz: Die Modelle vorher und nachher sind die gleichen, bzgl. der gemeinsamen Namen

Beispiele

T-Box

$$\textbf{Mann} \sqsubseteq \textbf{Person}$$

wird zu

$$\textbf{Mann} \equiv \widehat{\textbf{Mann}} \sqcap \textbf{Person}$$

Bei:

$$\begin{aligned} \textbf{Mann} &\equiv \textbf{Person} \sqcap \neg\textbf{Frau} \\ \textbf{Mann} &\sqsubseteq \textbf{Tier} \end{aligned}$$

Genügt das nicht:

$$\begin{aligned} \textbf{Mann} &\equiv \textbf{Person} \sqcap \neg\textbf{Frau} \\ \textbf{Mann} &\equiv \widehat{\textbf{Mann}} \sqcap \textbf{Tier} \end{aligned}$$

A-Boxen

- A-Box = Assertions
- Konkrete Annahmen über Individuuen
- T-Box ≈ Datenbankschema
- A-Box ≈ Daten

Definition

Gegeben eine T-Box $\mathcal{T}$.
Eine **A-Box** $\mathcal{A}$ zu $\mathcal{T}$ ist definiert als Menge von Annahmen über Individuen (Objekte) der Formen

- $C(a)$ wobei C ein Konzeptterm ist und a ein Individuenname.
- $R(a, b)$ wobei R eine Rolle ist (evtl. ein Rollenterm) und a, b sind Individuennamen.

Beispiele: A-Box

MutterMitTochter(`Maria`)
Vater(`Peter`)
hatKind(`Maria`, `Paul`)
hatKind(`Maria`, `Peter`)
hatKind(`Peter`, `Harry`)

Dabei sind `Peter`, `Harry`, `Maria` und `Paul` Konstanten.

Ähnlichkeiten zu Datenbank

Eine A-Box ist ähnlich zu Fakten in Prolog.

Aber es ist mehr erlaubt, z.B.

$$(\exists \textbf{hatKind}.\textbf{Person})(\texttt{Michael})$$

Semantik der A-Box

Definition

Sei I eine Interpretation zur T-Box $\mathcal{T}$.
Die **Interpretation** kann **auf die A-Box $\mathcal{A}$ erweitert** werden:

- Jedem Individuennamen a wird ein Objekt $I(a) \in \Delta$ zugeordnet. Dabei wird die **unique names assumption** beachtet: Verschiedenen Individuennamen werden verschiedenen Objekte zugeordnet, d.h. $I(a) = I(b)$ gdw. $a = b$.
- Für $C(a) \in \mathcal{A}$, ist $I(C(a)) = 1$ gdw. $I(a) \in I(C)$.
- Für $R(a, b) \in \mathcal{A}$ ist $I(R(a, b)) = 1$ gdw. $(I(a), I(b)) \in I(R)$.

Terminologische Beschreibung

Eine **terminologische Beschreibung** besteht aus

- Menge von terminologischen Axiomen und
- Mengen von Annahmen über Existenz und Eigenschaft von Objekten (A-Box)

Beispiel

T-Box:

Motor	$\sqsubseteq$	**Komponente**
Lampe	$\sqsubseteq$	**Komponente** $\sqcap$ ($\neg$**Motor**)
Stecker	$\sqsubseteq$	**Komponente** $\sqcap$ ($\neg$**Lampe**) $\sqcap$ ($\neg$**Motor**)
Gerät	$\sqsubseteq$	($\forall$**hatTeil**.**Komponente**) $\sqcap$ ($\neg$**Komponente**)
ElektroGerät	$\equiv$	**Gerät** $\sqcap$ ($\exists$**hatTeil**.**Stecker**)

A-Box:

Motor(`Motor1234`)
Komponente(`Lichtmaschine320`)
hatTeil(`Motor1234`, `Lichtmaschine320`)
...

Konsistenz der A-Box

Definition

Gegeben eine T-Box $\mathcal{T}$ und eine A-Box $\mathcal{A}$.

- Dann ist $\mathcal{A}$ **konsistent**, wenn es ein Modell I von $\mathcal{T}$ gibt, das alle Einträge in der A-Box wahr macht.
- Wir schreiben $\mathcal{A} \models C(a)$ gdw. für alle Modelle I von $\mathcal{T}$, die alle Einträge der A-Box wahr machen, auch $I(C(a))$ gilt.

Inferenzen und Anfragen

- **Konsistenztest:** Sind definierte Konzepte konsistent?

  Z.B. **Motor** $\sqcap$ ($\neg$**Motor**) ist inkonsistent

  Damit kann man auch für die A-Box entscheiden: Enthält eine A-Box widersprüchliche Annahmen?

  Z.B. (**Motor** $\sqcap$ $\neg$**Motor**)(`Motor123`) ist nicht erfüllbar.

- **Subsumtionstest:** Ist ein Konzept Untermenge eines anderen?

- **Retrieval Problem** Finde alle Instanzen eines Konzepts, wobei nur auf die Konstanten in der A-Box zugegriffen wird

  Z.B. „Welche Motoren gibt es ?“

  Formal: Zu gegebenem Konzept C, finde alle a mit$\mathcal{A} \models C(a)$.

Inferenzen und Anfragen (2)

- **Pinpointing (bzw. Realization Problem)**: Einordnung von Objekten in Konzepte.

  Z.B. „Ist `Staubsauger1` ein **ElektroGerät**?“

  Formal: Gegeben ein Individuum a, finde das spezifischste Konzept C, so dass $\mathcal{A} \models C(a)$ gilt. D.h. finde das kleinste Konzept in der Subsumtionsordnung.

Zusammenhänge

Satz

- $\mathcal{A} \models C(a)$ gdw. $\mathcal{A} \cup \{\neg C(a)\}$ ist inkonsistent.
- C ist konsistent gdw. $C(a)$ konsistent ist für einen neuen Namen a.

Individuen in der T-Box

Erweiterung: Aufzählungskonzepte

$$A \equiv \{a_1, \ldots . a_n\} \quad \text{wobei } a_i \text{ Individuennamen sind}$$

Semantik:

$$I(\{a_1, \ldots, a_n\}) = \{I(a_1), \ldots, I(a_n)\}$$

Beispiel:

$$\textbf{Grundfarben} \equiv \{\texttt{rot}, \texttt{blau}, \texttt{gelb}\}$$

Open-World- vs. Closed-World-Semantik

Datenbanken: **Closed-World-Semantik**

- Nur das was in der DB steht ist wahr
- Alles andere ist falsch

DLs: **Open-World-Semantik**

- Was nicht in der A-Box steht, ist nicht automatisch falsch
- Fehlende Einträge = Unvollständiges Wissen
- Man kann neue Fakten hinzufügen, ohne dass alte Schlüsse ungültig werden, d.h. Schlussfolgern ist konservativ.

Beispiel

Datenbank / A-Box enthält nur:

$$\textbf{hatKind}(\texttt{Maria}, \texttt{Peter})$$

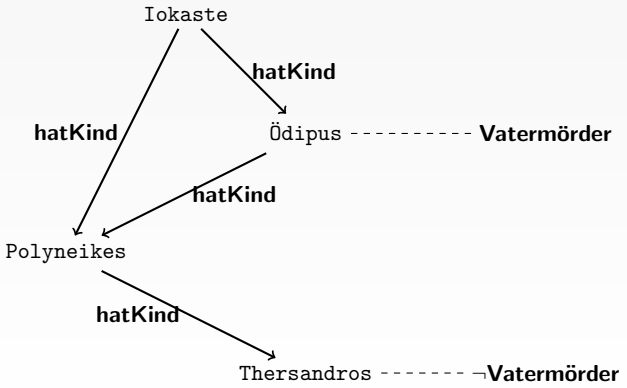
- Closed-World-Semantik: Es gilt: Peter hat keine Geschwister
- Open-World-Semantik: Unbekannt, ob Peter Geschwister hat

  Durch $(\leq\ 1\ \textbf{hatKind})(\texttt{Maria})$ kann sichergestellt werden, dass Peter keine Geschwister hat

Beispiel

A-Box $\mathcal{A}_{oed}$ zum Problem des Ödipus:

$\textbf{hatKind}(\texttt{Iokaste}, \texttt{Ödipus})$ $\quad$ $\textbf{hatKind}(\texttt{Iokaste}, \texttt{Polyneikes})$
$\textbf{hatKind}(\texttt{Ödipus}, \texttt{Polyneikes})$ $\quad$ $\textbf{hatKind}(\texttt{Polyneikes}, \texttt{Thersandros})$
$\textbf{Vatermörder}(\texttt{Ödipus})$ $\quad$ $\neg\textbf{Vatermörder}(\texttt{Thersandros})$



Beispiel (2)

$\textbf{hatKind}(\texttt{Iokaste}, \texttt{Ödipus})$ $\quad$ $\textbf{hatKind}(\texttt{Iokaste}, \texttt{Polyneikes})$
$\textbf{hatKind}(\texttt{Ödipus}, \texttt{Polyneikes})$ $\quad$ $\textbf{hatKind}(\texttt{Polyneikes}, \texttt{Thersandros})$
$\textbf{Vatermörder}(\texttt{Ödipus})$ $\quad$ $\neg\textbf{Vatermörder}(\texttt{Thersandros})$

Kann man aus dieser A-Box schließen?

$$\mathcal{A}_{oed} \models (\exists \textbf{hatKind}.(\textbf{Vatermörder} \sqcap (\exists \textbf{hatKind}.\neg\textbf{Vatermörder})))(\texttt{Iokaste})$$

- Iokaste hat zwei Kinder, Ödipus und Polyneikes.
- Ödipus erfüllt **Vatermörder**,
- kein Wissen über $\textbf{Vatermörder}(\texttt{Polyneikes})$.
- Trotzdem kann man schließen (wegen OWA): Fallunterscheidung über Polyneikes: entweder **Vatermörder** oder nicht.
  - Erster Fall: Polyneikes erfüllt **hatKind** von Iokaste
  - Zweiter Fall: Ödipus erfüllt **hatKind** von Iokaste

Inferenzen in Beschreibungslogiken: Subsumtion

Wir betrachten Verfahren für den Subsumtionstest, d.h. ob gilt:

$$C \sqsubseteq D$$

Wir betrachten nur Konzepte, keine T-Box (diese muss eliminiert werden)

Schwierigkeit des Problems: Hängt von der Sprache ab!

Erinnerung: $\mathcal{FL}$-Familie

$\mathcal{FL}$ = frame-based description language
Drei Varianten

Name	Konstrukte
$\mathcal{FL}_0$	$\sqcap, \forall R.C$
$\mathcal{FL}^-$	$\sqcap, \forall R.C, \exists R.\top$
$\mathcal{FL}$	$\sqcap, \forall R.C, \exists R.C$

Beziehungen zwischen $\mathcal{FL}$-Sprachen und $\mathcal{AL}$-Sprachen:

$$\begin{aligned} \mathcal{AL} &\equiv \mathcal{FL}^- \cup \{\neg A, \top\} \\ \mathcal{ALC} &\equiv \mathcal{FL}^- \cup \{\neg C\} \\ \mathcal{ALC} &\equiv \mathcal{FL} \cup \{\neg C\} \end{aligned}$$

Subsumtion und Konsistenz in $\mathcal{FL}_0$

$\mathcal{FL}_0$: $C \sqcap D$, $\forall R.C$

Lemma

Alle Konzepte in $\mathcal{FL}_0$ sind konsistent.

Beweis: Es gibt immer eine Interpretation mit $I(C) \neq \emptyset$:

$$\begin{aligned} I(A) &= \Delta \quad \text{für alle atomaren Konzepte } A \\ I(R) &= \Delta \times \Delta \quad \text{für alle Rollen } R \end{aligned}$$

Es gilt sogar für alle C: $I(C) = \Delta$.

Struktureller Subsumtionstest $\mathcal{FL}_0$ (2)

Konsistenz ist daher trivial, aber Subsumtion nicht, z.B.

Z.B. $A \sqcap B \sqsubseteq A$, aber $A \not\sqsubseteq A \sqcap B$

Struktureller Subsumtionstest

Allgemein (nicht nur in $\mathcal{FL}_0$): Test für $C \sqsubseteq D$

1. Bringe C und D in eine Normalform C' bzw. D'.
2. Vergleiche C' und D' syntaktisch.

Beide Teilalgorithmen variieren von Sprache zu Sprache.

Normalform in $\mathcal{FL}_0$

Normalform in $\mathcal{FL}_0$:

$$A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

wobei A_i atomare Konzepte, $A_i \neq A_j$ und C_i in Normalform sind

Normalformalgorithmus für $\mathcal{FL}_0$

- assoziativ ausklammern, dann umsortieren, und dann gleiche A_i in Konjunktionen eliminieren.
- $\forall R.C \sqcap \forall R.D \quad \rightarrow \quad \forall R.C \sqcap D$
- Rekursiv in den rechten Seiten C aller Ausdrücke $\forall R.C$ anwenden.

Begründung:

$$\begin{array}{ll} & I(\forall\ R.(C_1 \sqcap C_2)) \\ = & \{x \mid \forall y.xI(R)y \Rightarrow y \in I(C_1) \cap I(C_2)\} \\ = & \{x|\forall y.\neg(x\ I(R)\ y \vee y \in I(C_1) \cap I(C_2)\} \\ = & \{x|\forall y.(\neg(x\ I(R)\ y) \vee y \in I(C_1) \wedge (\neg(x\ I(R)\ y) \vee y \in I(C_2)\} \\ = & \{x|\forall y.(\neg(x\ I(R)\ y) \vee y \in I(C_1)) \wedge \forall y.\neg(x\ I(R)\ y) \vee y \in I(C_2)\} \\ = & I((\forall\ R.C_1) \sqcap\ (\forall\ R.C_2)) \end{array}$$

Vergleich der Normalformen in $\mathcal{FL}_0$

Sei

$$D \equiv A_1 \sqcap \ldots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n$$

und

$$D' \equiv A'_1 \sqcap \ldots \sqcap A'_{m'} \sqcap \forall R'_1.C'_1 \sqcap \ldots \sqcap \forall R'_{n'}.C'_{n'}$$

Dann ist $D \sqsubseteq D'$ gdw:

- Jedes atomare Konzept A'_i kommt unter den A_j vor und
- zu jedem Ausdruck $\forall R'_i.C'_i$ gibt es einen Ausdruck $\forall R_j.C_j$, so dass $R'_i = R_j$ und $C_j \sqsubseteq C'_i$. Dieser Test wird rekursiv durchgeführt.

Eigenschaften

Der strukturelle Subsumtionstest in $\mathcal{FL}_0$ ist:

- **korrekt**: Bei Antwort „ja“gilt $I(D) \subseteq I(D')$ in allen Modellen I
- **vollständig**: Bei Antwort „nein“ , gibt es ein Modell I mit $I(D) \not\subseteq I(D')$

Komplexität:

- Normalformherstellung: Sortieren $O(n * log(n))$
- Vergleichen: $O(n * log(n))$ oder sogar $O(n)$

Beispiel

Seien

$$\begin{aligned} C_1 &\equiv (\forall R_1.A_1) \sqcap A_2 \sqcap (\forall R_2.A_5) \sqcap (\forall R_2.\forall R_1.(A_2 \sqcap A_3 \sqcap A_4)) \\ C_2 &\equiv ((\forall R_2.\forall R_1.A_4) \sqcap A_2 \sqcap (\forall R_2.\forall R_1.(A_3 \sqcap A_4)) \end{aligned}$$

Prüfe ob $C_1 \sqsubseteq C_2$ und ob $C_2 \sqsubseteq C_1$.

Beispiel (2)

Normalformberechnung für C_1:

$$\begin{aligned} C_1 &\equiv (\forall R_1.A_1) \sqcap A_2 \sqcap (\forall R_2.A_5) \sqcap (\forall R_2.\forall R_1.(A_2 \sqcap A_3 \sqcap A_4)) \\ &\to A_2 \sqcap (\forall R_1.A_1) \sqcap (\forall R_2.A_5) \sqcap (\forall R_2.\forall R_1.(A_2 \sqcap A_3 \sqcap A_4)) \\ &\to A_2 \sqcap (\forall R_1.A_1) \sqcap (\forall R_2.A_5 \sqcap (\forall R_1.(A_2 \sqcap A_3 \sqcap A_4))) = NF(C_1) \end{aligned}$$

Normalformberechnung für C_2:

$$\begin{aligned} C_2 &\equiv ((\forall R_2.\forall R_1.A_4) \sqcap A_2 \sqcap (\forall R_2.\forall R_1.(A_3 \sqcap A_4)) \\ &\to A_2 \sqcap ((\forall R_2.\forall R_1.A_4) \sqcap (\forall R_2.\forall R_1.(A_3 \sqcap A_4)) \\ &\to A_2 \sqcap (\forall R_2.((\forall R_1.A_4) \sqcap \forall R_1.(A_3 \sqcap A_4))) \\ &\to A_2 \sqcap (\forall R_2.(\forall R_1.(A_4 \sqcap A_3 \sqcap A_4)) \\ &\to A_2 \sqcap (\forall R_2.(\forall R_1.(A_3 \sqcap A_4)) = NF(C_2) \end{aligned}$$

Beispiel (3)

Vergleich der Normalformen:

$$\begin{aligned} NF(C_1) &= A_2 \sqcap (\forall R_1.A_1) \sqcap (\forall R_2.A_5 \sqcap (\forall R_1.(A_2 \sqcap A_3 \sqcap A_4))) \\ NF(C_2) &= A_2 \sqcap (\forall R_2.(\forall R_1.(A_3 \sqcap A_4)) \end{aligned}$$

- Atomare Konzepte: beide gleich
- Werteinschränkungen: $\forall R_1 \ldots$. kommt nur in $NF(C_1)$ vor. Daraus folgt sofort: $C_2 \not\sqsubseteq C_1$.
- Für $C_1 \sqsubseteq C_2$: Für $\forall R_2 \ldots$. rekursiver Vergleich

$$A_5 \sqcap (\forall R_1.(A_2 \sqcap A_3 \sqcap A_4))) \sqsubseteq (\forall R_1.(A_3 \sqcap A_4))$$

- A_5 nur links: ok. Für $\forall R_1 \ldots$. rekursiver Vergleich

$$(A_2 \sqcap A_3 \sqcap A_4) \sqsubseteq (A_3 \sqcap A_4)$$

- Daher gilt: $C_1 \sqsubseteq C_2$.

Subsumtionstest für $\mathcal{FL}^-$

$\mathcal{FL}^-$: $\sqcap, \forall R.C, (\exists R.\top)$

Anpassung des Subsumtionsalgorithmus für $\mathcal{FL}^-$

1. Bringe die Konzeptterme in eine $\mathcal{FL}^-$-Normalform:

$$\begin{aligned} & A_1 \sqcap \ldots \sqcap A_m \\ \sqcap \quad & \forall R_1.C_1 \sqcap \ldots \sqcap \forall R_n.C_n \\ \sqcap \quad & \exists R'_1.\top \sqcap \ldots \sqcap \exists R'_k.\top \end{aligned}$$

2. Vergleich der Normalformen: Wie bei $\mathcal{FL}_0$ nur: Behandele $\exists R.\top$-Ausdrücke wie atomare Konzepte.

Auch der $\mathcal{FL}^-$-Subsumtions-Algorithmus ist korrekt und vollständig.
Der Zeitaufwand des Algorithmus ist $O(n * \log(n))$.

Struktureller Subsumtionstest für weitere DL-Sprachen

- $\mathcal{FL}_0 + \bot$: Geht immer noch, beachte $\bot \sqcap \ldots \to \bot$ und $\bot \sqsubseteq C$ für alle C.
- $\mathcal{FL}_0 + \bot + \neg A$: Geht auch noch:
  - bei NF-Berechnung: $A \sqcap \neg A \to \bot$
  - Vor dem Vergleich: $\neg A \to$ **NOTA**, wobei **NOTA** neuer Konzeptname

Subsumtions-Algorithmus für $\mathcal{AL}$

$\mathcal{AL}$: Neue Konzepte: $\top$ und $\exists R.\top$.
Insgesamt: $A, \top, \bot, \neg A, C \sqcap D, \forall R.C, \exists R.\top$

- NF-Berechnung:
  - $\top \sqcap C \to C$
  - $\forall R.\top \to \top$
  - $\neg\top \to \bot$
  - $\neg\bot \to \top$
- Vergleich: Beachte $C \sqsubseteq \top$ für alle C

Konflikte bei Anzahlbeschränkungen

Hat man Anzahlbeschränkungen, muss man mehr Konflikte beachten.
Z.B.

- $\forall R.\bot \sqcap (\geq\ 1\ R)$ ist äquivalent zu $\bot$.
- $(\geq\ n\ R) \sqsubseteq (\geq\ m\ R)$ gdw. $n \geq m$.

Weitere Sprachen mit strukturellem Subsumtionsalgorithmus

$\mathcal{ALE}$	$\mathcal{AL}$ erweitert um $\exists R.C$.
$\mathcal{ALU}$	$\mathcal{AL}$ erweitert um $\sqcup$.
$\mathcal{ALN}$	$\mathcal{AL}$ erweitert um Anzahlbeschränkungen $(\leq\ n\ R)$.

- $\mathcal{ALN}$ hat einen polynomiellen und strukturellen Subsumtionsalgorithmus.
- $\mathcal{FL}$ hat keinen strukturellen Subsumtionstest: das Subsumtionsproblem ist PSPACE-vollständig.

Grenzen des strukturellen Subsumtionsalgorithmus

Hat man **Disjunktion** (Vereinigung) oder **volle Negation**, dann funktioniert der strukturelle Subsumtionsalgorithmus **nicht**.

Z.B. gibt es für $\mathcal{ALC}$ keinen strukturellen Subsumtionsalgorithmus!

Subsumtion und Äquivalenzen in $\mathcal{ALC}$

$\mathcal{ALC}$: $\sqcap, \sqcup, \neg, \bot, \top, \forall R.C, \exists R.C$
In $\mathcal{ALC}$ gilt:

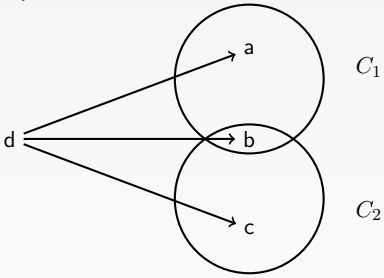
$$
\begin{array}{lcl}
\neg(C_1 \sqcap C_2) & \equiv & (\neg C_1) \sqcup (\neg C_2) \\
\neg(C_1 \sqcup C_2) & \equiv & (\neg C_1) \sqcap (\neg C_2) \\
\neg(\neg C) & \equiv & C \\
\neg(\forall R.C) & \equiv & (\exists R.(\neg C)) \\
\neg(\exists R.C) & \equiv & (\forall R.(\neg C)) \\
\neg \bot & \equiv & \top \\
\neg \top & \equiv & \bot
\end{array}
$$

Beispiel

In $\mathcal{ALC}$ gilt **nicht**: $(\forall R.(C_1 \sqcup C_2)) = (\forall R.C_1) \sqcup (\forall R.C_2)$

Beweis: Gegenbeispiel: Sei I die Interpretation mit

$$
\begin{array}{lcl}
\Delta & = & \{a, b, c, d\} \\
I(C_1) & = & \{a, b\} \\
I(C_2) & = & \{b, c\} \\
I(R) & = & \{(d, b), (d, a), (d, c)\}
\end{array}
$$



$$
\begin{array}{l}
I(\forall R.(C_1 \sqcup C_2)) \\
= \{x \in \{a, b, c, d\} \mid \forall y.(x, y) \in \{(d, b), (d, a), (d, c)\} \Rightarrow y \in \{a, b, c\}\} \\
= \{a, b, c, d\}
\end{array}
$$

$$
\begin{array}{l}
I((\forall R.C_1) \sqcup (\forall R.C_2)) \\
= \{x \in \{a, b, c, d\} \mid \forall y.(x, y) \in \{(d, b), (d, a), (d, c)\} \Rightarrow y \in \{a, b\}\} \\
\quad \cup \{x \in \{a, b, c, d\} \mid \forall y.(x, y) \in \{(d, b), (d, a), (d, c)\} \Rightarrow y \in \{b, c\}\} \\
= \{a, b, c\}
\end{array}
$$

Konsistenz und Subsumtion in $\mathcal{ALC}$

Lemma

Der Subsumtionstest in $\mathcal{ALC}$ lässt sich als Konsistenztest formulieren und umgekehrt: $C_1 \sqsubseteq C_2$ gilt gdw. $(C_1 \sqcap \neg C_2) \equiv \bot$

Und statt $C \not\equiv \bot$ kann man $C \not\sqsubseteq \bot$ testen.
Aber Komplexität ist dann co- bzgl Subsumtion.
In diesem Fall auch PSPACE-vollständig.

Subsumtionsalgorithmus für $\mathcal{ALC}$

Tableauverfahren, das prüft: Ist C konsistent?

Idee:

- Versuche I zu konstruieren mit $I(C) \neq \emptyset$
- Mögliche Ausgänge: Modell exisitiert $\rightarrow$ C konsistent
- Es gibt kein Modell: C inkonsistent

Neue Datenstruktur: **Constraint-Systeme**, um Anforderungen an I zu beschreiben

Subsumtionsalgorithmus für $\mathcal{ALC}$ (2)

Definition

Ein **Constraint** ist eine Folge von Ausdrücken der Form:

$$x : X \qquad xRy \qquad X \sqsubseteq C \qquad X \sqsubseteq Y \sqcup Z \qquad X(\forall R)Y \qquad X(\exists R)Y$$

wobei x, y, z Elemente (von Δ), X, Y, Z Konzeptnamen und C (auch komplexe) Konzepte sind.

Bedeutung (informell):

- $x : X$ entspricht der Bedingung $x \in I(X)$,
- xRy entspricht der Bedingung $(x, y) \in I(R)$,
- $X \sqsubseteq C$ entspricht der Bedingung $I(X) \subseteq I(C)$,
- $X \sqsubseteq Y \sqcup Z$ entspricht der Bedingung $I(X) \subseteq I(Y) \cup I(Z)$,
- $X(\forall R)Y$ entspricht der Bedingung $\forall a \in I(X) : \forall b : (a, b) \in I(R) : b \in I(Y)$
- $X(\exists R)Y$ entspricht der der Bedingung $\forall a \in I(X) : \exists b.(a, b) \in I(R) : b \in I(Y)$

Subsumtionsalgorithmus für $\mathcal{ALC}$ (3)

Subsumtionsalgorithmus: Für C Konsistent?

1. Beginne mit $x : X, X \sqsubseteq C$
2. Entfalte das Constraintsystem
3. Baue Tableau auf mit allen Vervollständigungen
4. Prüfe alle Pfade

Entfaltungsregeln

$$
\begin{array}{lcl}
X \sqsubseteq (\forall R.C) & \rightarrow & X(\forall R)Y, Y \sqsubseteq C, \text{ wobei } Y \text{ ein neuer Name} \\
X \sqsubseteq (\exists R.C) & \rightarrow & X(\exists R)Y, Y \sqsubseteq C, \text{ wobei } Y \text{ ein neuer Name} \\
X \sqsubseteq C \sqcap D & \rightarrow & X \sqsubseteq C, X \sqsubseteq D \\
X \sqsubseteq C \sqcup D & \rightarrow & X \sqsubseteq Y \sqcup Z, Y \sqsubseteq C, Z \sqsubseteq D, \\
 & & \text{wenn } C \text{ oder } D \text{ kein atomares Konzept ist,} \\
 & & \text{und } Y, Z \text{ sind neue Namen sind.} \\
X \sqsubseteq \top & \rightarrow & \text{nichts zu tun}
\end{array}
$$

Ausgabe: Constraint-System, das nur Konzeptnamen und negierte Konzeptnamen für C in $X \sqsubseteq C$ enthält.

Vervollständigung (Tableauregeln)

1. Wenn $x : X$ und $X(\exists R)Y$ da sind, aber keine Variable y mit xRy und $y : Y$, dann füge eine neue Variable y mit den Constraints xRy und $y : Y$ ein.
2. Wenn $x : X, X(\forall R)Y, xRy$ da ist, dann füge $y : Y$ hinzu.
3. Wenn $x : X, X \sqsubseteq Y \sqcup Z$ da sind, aber weder $x : Y$ noch $x : Z$, dann füge $x : Y$ oder $x : Z$ hinzu.

Letzte Regel: Verzweigung im Tableau

Blätter: Vervollständigte Systeme

Prüfen der Pfade

Prüfe ob, Blätter widersprüchliche Constraintsysteme sind:

Definition

Ein Constraint-System ist **widersprüchlich**, wenn eine der folgenden Konstellationen vorkommt:

- $x : X, X \sqsubseteq A, x : Y, Y \sqsubseteq \neg A$, oder
- $x : X, X \sqsubseteq \neg\top$, oder
- $x : X, X \sqsubseteq \bot$

Algorithmus zur Konsistenzprüfung

- Starte mit dem Constraint $x : X, X \sqsubseteq C$
- Falte $x : X, X \sqsubseteq C$ auf
- Baue Tableau auf, indem alle Möglichkeiten zur Vervollständigung betrachtet werden
- Sind alle Blätter widersprüchliche Constraintsysteme, gebe inkonsistent aus
- Gibt es ein nicht-widersprüchliches Blatt, gebe konsistent aus In diesem Fall kann man aus dem Constraintsystem das Modell konstruieren

Beispiel

Seien **Mann** und **hatKind** atomare Konzepte. Wir prüfen Konsistenz von:

$$\textbf{Mann} \sqcap \exists \textbf{hatKind}.(\textbf{Mann} \sqcup \neg \textbf{Mann})$$

Erster Schritt: Starte mit

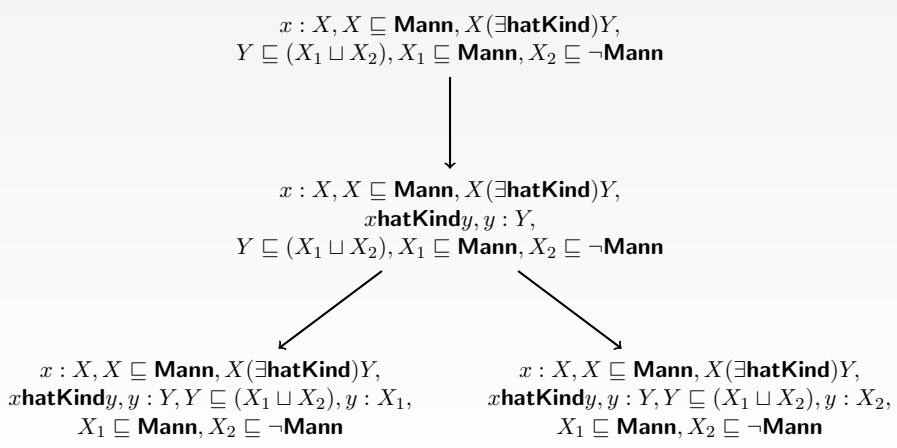
$$x : X, X \sqsubseteq \textbf{Mann} \sqcap \exists \textbf{hatKind}.(\textbf{Mann} \sqcup \neg \textbf{Mann}).$$

Zweiter Schritt: Entfalten:

$$\begin{aligned}
& x : X, X \sqsubseteq \textbf{Mann} \sqcap \exists \textbf{hatKind}.(\textbf{Mann} \sqcup \neg \textbf{Mann}) \\
\rightarrow\ & x : X, X \sqsubseteq \textbf{Mann}, X \sqsubseteq \exists \textbf{hatKind}.(\textbf{Mann} \sqcup \neg \textbf{Mann}) \\
\rightarrow\ & x : X, X \sqsubseteq \textbf{Mann}, X(\exists \textbf{hatKind})Y, Y \sqsubseteq (\textbf{Mann} \sqcup \neg \textbf{Mann}) \\
\rightarrow\ & x : X, X \sqsubseteq \textbf{Mann}, X(\exists \textbf{hatKind})Y, Y \sqsubseteq (X_1 \sqcup X_2), \\
& X_1 \sqsubseteq \textbf{Mann}, X_2 \sqsubseteq \neg \textbf{Mann}
\end{aligned}$$

Beispiel (2)

Vervollständigung ergibt das Tableau:



Beide Blätter: nicht widersprüchlich.

Beispiel (3)

Modelle: an den Blättern:

- Für

  $$x : X, X \sqsubseteq \textbf{Mann}, X(\exists \textbf{hatKind})Y, x\textbf{hatKind}y, y : Y, Y \sqsubseteq (X_1 \sqcup X_2),$$
  $$y : X_1, X_1 \sqsubseteq \textbf{Mann}, X_2 \sqsubseteq \neg\textbf{Mann}$$

  kann man ablesen
  $\Delta = \{x, y\}$, $I(X) = \{x\}$, $I(Y) = \{y\}$, $I(X_1) = \{x, y\}$, $I(\textbf{hatKind}) = \{(x, y)\}$
  und $I(\textbf{Mann}) = \{x, y\}$, $I(X_2) = \emptyset$.
- Für

  $$x : X, X \sqsubseteq \textbf{Mann}, X(\exists \textbf{hatKind})Y, x\textbf{hatKind}y, y : Y, Y \sqsubseteq (X_1 \sqcup X_2),$$
  $$y : X_2, X_1 \sqsubseteq \textbf{Mann}, X_2 \sqsubseteq \neg\textbf{Mann}$$

  kann man ablesen
  $\Delta = \{x, y\}$, $I(X) = \{x\}$, $I(Y) = \{y\}$, $I(X_2) = \{y\}$, $I(\textbf{hatKind}) = \{(x, y)\}$,
  $I(\textbf{Mann}) = \{x\}$, $I(X_1) = \{x\}$.

Eigenschaften

Der Algorithmus terminiert und ist korrekt und vollständig, daher gilt:

Theorem

Subsumtion und Konsistenz in $\mathcal{ALC}$ sind entscheidbar. Der Algorithmus kann in polynomiellem Platz durchgeführt werden.

Aber es gilt:

Theorem

Konsistenz in $\mathcal{ALC}$ ist PSPACE-hart. D.h Konsistenz in $\mathcal{ALC}$ ist PSPACE-vollständig.

Beweis der Härte: Reduktion von QBFs in $\mathcal{ALC}$.

Erweiterungen für $\mathcal{ALCN}$

$\mathcal{ALCN}$: $\mathcal{ALC}$ + number restrictions $(\geq\ n\ R.C)$ und $(\leq\ n\ R.C)$.
Neue Constraints:
$X(\leq\ n\ R)Y$ und $X(\geq\ n\ R)Y$.

Auffalten

$$X \sqsubseteq (\geq\ n\ R.C) \rightarrow X(\geq\ n\ R)Y, Y \sqsubseteq C$$
$$X \sqsubseteq (\leq\ n\ R.C) \rightarrow X(\leq\ n\ R)Y, Y \sqsubseteq C$$

Vervollständigung:

Bei $x : X, X(\leq\ n\ R)Y$ muss man die Variablen y zählen und hinzufügen, evtl. gleichsetzen.
Kodierung der Zahlen n!?

Es gilt: In $\mathcal{ALCN}$ bleibt das Problem $PSPACE$-vollständig

Konsistenztest von A-Boxen

Klar: Füge Constraint entsprechend der A-Box hinzu.

Es gilt:

Theorem

Konsistenz von $\mathcal{ALCN}$-A-Boxen ist entscheidbar und PSPACE-complete.

Komplexität der Subsumtions-Inferenzen

- $\mathcal{FL}^-$ ($\sqcap, \forall$ und $(\exists R)$) hat einen polynomiellen Subsumtionstest
- $\mathcal{FL}$ ($\sqcap, \forall R.C$, $(\exists R.C)$) hat ein PSPACE-vollständiges Subsumtionproblem, Konsistenz ist trivial.
- Erlaubt man nur $\sqcap, \sqcup, \neg$, dann ist der Subsumtionstest co-NP-vollständig, da dies gerade das Komplement von SAT ist.
- In $\mathcal{ALC}$ ($\sqcap, \neg, \sqcup, \forall, (\exists R.C)$) sind Konsistenztest und Subsumtionstest PSPACE-complete.
- $\mathcal{ALC}$ + Pfadgleichungen für bel. Rollen (z.B. (**hatKind**; **studiertFach**) = **studiertFach**): Subsumtion ist unentscheidbar.

Erweiterungen: Rollenterme

$R \sqcap S$	Schnitt von Rollen	$I(R \sqcap S) = I(R) \cap I(S)$
$R \sqcup S$	Vereinigung von Rollen	$I(R \sqcup S) = I(R) \cup I(S)$
$\neg R$	Komplement einer Rolle	$I(\neg R) = \Delta \times \Delta \setminus I(R)$
(R^{-1})	Rolleninversion	$I(R^{-1}) = \{(b,a) \mid (a,b) \in I(R)\}$
$(R \circ S)$	Rollenkomposition	$I(R \circ S) = \{(a,c) \mid \exists b.(a,b) \in I(R), (b,c) \in I(S)\}$
(R^+)	transitiver Abschluss	$I(R^+) =$ trans. Abschluss von $I(R)$.

Einige Eigenschaften:

- $\mathcal{ALCN}$ mit Rollenschnitt hat ein PSPACE-vollständiges Subsumtionsproblem, wenn man Zahlen in Strichcode schreibt.
- $\mathcal{ALC}_{trans}$: $\mathcal{ALC}$ + transitive Rollen hat entscheidbares Subsumtionsproblem. Beziehungen zu propositional dynamic logic

Übersetzung $\mathcal{ALC}$ in Prädikatenlogik

Atomare Konzepte A werden in einstellige Prädikate P_A übersetzt.
Rollen R werden zweistellige Prädikate P_R übersetzt.
$[\![\cdot]\!]_\alpha$: Übersetzung von $\mathcal{ALC}$-Formeln in Prädikatenlogische Formel mit freier Variable α

$$\begin{aligned}
[\![A]\!]_x &= P_A(x) \\
[\![C \sqcap D]\!]_x &= [\![C]\!]_x \wedge [\![D]\!]_x \\
[\![C \sqcup D]\!]_x &= [\![C]\!]_x \vee [\![D]\!]_x \\
[\![\exists R.C]\!]_x &= \exists y.P_R(x,y) \wedge [\![C]\!]_y \\
[\![\forall R.C]\!]_x &= \forall y.P_R(x,y) \implies [\![C]\!]_y
\end{aligned}$$

Übersetzung der Subsumtion:

$$[\![C \sqsubseteq D]\!] = \forall x.([\![C]\!]_x \implies [\![D]\!]_x)$$

Satz

$C \sqsubseteq D$ genau dann, wenn $[\![C \sqsubseteq D]\!]$ eine Tautologie ist.

OWL

- **W**eb **O**ntology **L**anguage = OWL (nicht WOL, in Analogie zu „owl“ (Eule))
- Vom W3C standardisierte formale Beschreibungssprache zur Erstellung von Ontologien
- Bedeutsame Beziehung zum Semantischen Web
- Baut syntaktisch auf RDF (Resource Description Framework) auf
- inbesondere: XML-Notation, zur maschinellen Verarbeitung

Varianten von OWL

1. OWL Lite
2. OWL DL
3. OWL Full

- „OWL Lite $\subset$ OWL DL $\subset$ OWL Full“
- OWL Lite und OWL DL entsprechen Beschreibungslogiken
- OWL Full nicht mehr!

Standards unterscheiden jetzt nur noch zwei Varianten.

Beschreibungslogiken dazu

- OWL Lite entspricht der Beschreibungslogik $\mathcal{SHIN}(\mathbf{D})$
- OWL DL entspricht der Beschreibungslogik $\mathcal{SHOIN}(\mathbf{D})$

wobei

- $\mathcal{S}$ steht für $\mathcal{ALC}$ + transitive Rollen
  Namensgebung: Bezug zur Modallogik S_4.
- $\mathcal{H}$ steht für Rollen**h**ierarchien, $R \sqsubseteq S$
- $\mathcal{I}$ steht für **i**nverse Rollen, R^-
- $\mathcal{N}$: number restrictions $(\leq\ n\ R)$, $(\geq\ n\ R)$
- $\mathcal{O}$ steht für “n**o**minal”: $\{o\}$ (mit o Individuenname)
- $\mathbf{D}$ bedeutet konkrete Datentypen
  (Variante sog. *concrete domains*)).
  In OWL dürfen die XML Schema Datentypen (Integer, Strings, Float, . . .) verwendet werden.

Bezeichnungen

- OWL Lite oder DL-Ontolgie: T-Box + Rollenhierachie
- Statt Konzept sagt man in OWL `Class`
- Statt Rolle sagt man in OWL `Property`

OWL Konstruktoren

Konstruktor	Syntax in DL
`owl:Thing`	$\top$
`owl:Nothing`	$\bot$
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$
`complementOf`	$\neg C$
`oneOf`	$\{a_1, \ldots, a_m\}$
`allValuesFrom`	$\forall R.C$
`someValuesFrom`	$\exists R.C$
`hasValue`	$\exists R.\{a\}$
`minCardinality`	$\geq nR$
`maxCardinality`	$\leq nR$
`inverseOf`	R^-

RDF/XML-Darstellung

Mensch $\sqcap$ **Weiblich** in XML-Notation

```
<owl:Class>
 <owl:intersectionOf rdf:parseType="Collection">
 <owl:Class rdf:about="#Mensch"/>
 <owl:Class rdf:about="#Weiblich"/>
 </owl:intersectionOf>
</owl:Class>
```

$(\leq\ 2$ **hatKind** $\top)$ in XML-Notation:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasChild"/>
  <owl:minCardinality rdf:datatype="&xsd;NonNegativeInteger">
    2
  </owl:minCardinality>
</owl:Restriction>
```


OWL-Syntax für Axiome:

Konstruktor	Syntax in DL
`subClassOf`	$C_1 \sqsubseteq C_2$
`equivalentClass`	$C_1 \equiv C_2$
`subPropertyOf`	$R_1 \sqsubseteq R_2$
`equivalentProperty`	$R_1 \equiv R_2$
`disjointWith`	$C_1 \sqcap C_2 \equiv \bot$ bzw. $C_1 \sqcap \neg C_2$
`sameAs`	$\{a_1\} \equiv \{a_2\}$
`differentFrom`	$\{a_1\} \equiv \neg\{a_2\}$
`TransitiveProperty`	definiert e. transitive Rolle
`FunctionalProperty`	definiert e. funktionale Rolle
`InverseFunctionalProperty`	definiert e. inverse funktionale Rolle
`SymmetricProperty`	definiert e. symmetrische Rolle

Reasoner für OWL-Ontologie (Auswahl)

- Racer (http://www.racer-systems.com/),
- FaCT++ (http://owl.man.ac.uk/factplusplus/)
- Pellet (http://pellet.owldl.com/).

Komplexität

OWL-Variante	passende DL	Subsumption- und Konsistenztest
OWL Lite	$\mathcal{SHIN}(\mathbf{D})$	EXPTIME-vollständig
OWL DL	$\mathcal{SHOIN}(\mathbf{D})$	NEXPTIME-vollständig

Nochmal: Anwendung der DL

Ausdrucksschwache Beschreibungslogik $\mathcal{EL}$:

$$C ::= \top \mid A \mid C \sqcap D \mid \exists R.C$$

Fragestellung:

- Überprüfung der Konsistenz einer Menge von Konzepten
- Überprüfung der Erweiterbarkeit einer Menge von Konzepten.

Die Subsumption in $\mathcal{EL}$ ist polynomiell und damit anwendungsgeeignet.

Passt zu:
SNOMED CT (ein Akronym für Systematized Nomenclature of Medicine – Clinical Terms)