

Logikbasierte Systeme der Wissensverarbeitung

Konzeptbeschreibungssprachen

Prof. Dr. M. Schmidt-Schauß

SoSe 2025

Stand der Folien: 22. Juli 2025

www.uni-frankfurt.de

Wissensverarbeitungssysteme

Wir betrachten die Grundlagen von

Wissensverarbeitungssystemen

(Passt zu: DB und DBMS:
Datenbanken und Datenbankmanagementsysteme)

Kernkomponenten:

- Wissensbasis 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 (DL))

- „Vorgänger“: Semantische Netze und Frames
- Vorteil von DL: Eindeutige Syntax und Semantik
- Viele moderne Konzept- und Repräsentations-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)

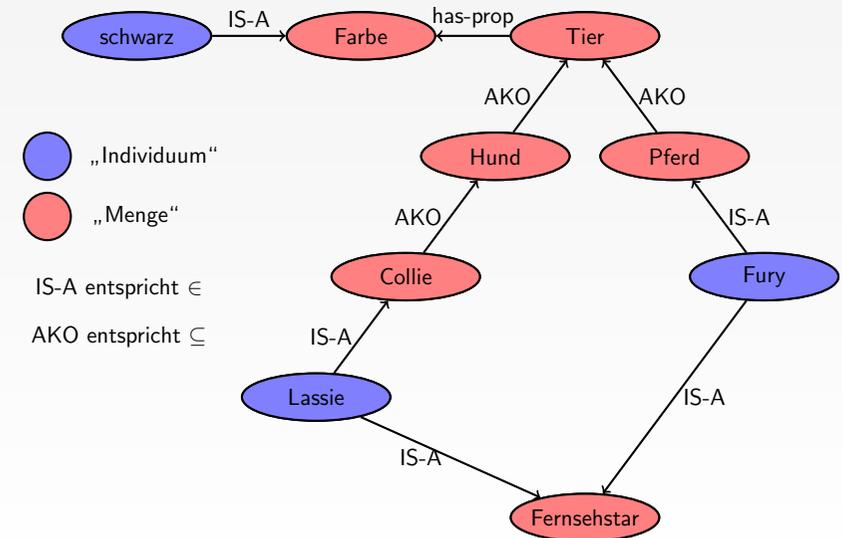
Semantisches Netz: Gerichteter Graph mit

- **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
- Unterscheidung, welche Konzepte durch Teilgraphen definiert werden, und welche gegeben sind.

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:** z.B. Suche nach Verbindungswegen im Netz.

Probleme der semantischen Netze

- Die **Semantik** war **ungenau** definiert und erlaubte Variationen.
- Jedes Programm arbeitete auf (s)einer eigenen **anderen** semantischen Basis.
- Z.B.: Was bedeuten jeweils zirkuläre Links?
- und: Definition neuer Konzepte: wie genau?
- und: Was sind Individuen und was sind Mengen?
- 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 erlaubten 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

Detaillierter und ausführlicher in Vorlesung und Skript

Attributive Konzeptbeschreibungssprachen (2)

Literatur:

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

**The Description Logic Handbook:
Theory, Implementation and Applications**

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

und neuere Veröffentlichungen dazu.

Wissensverarbeitungssystem

Komponenten eines Wissensverarbeitungssystems basierend auf DL:

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

Analog zu Datenbanken:

- T-Box entspricht Datenbankschema
- A-Box entspricht 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 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:

Bei manchen Fragestellungen:

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

Daher: Kompromisse sind notwendig zwischen Ausdruckskraft und effizienter Berechnung

Typische Anfragen / Fragestellungen

Bzgl. der T-Box (Terminologie)

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

Bzgl. der A-Box (Annahmen, Assertions)

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

Vorgehen im Folgenden

- Zuerst Konzentration auf die Syntax und Semantik der DL;
- T-Box und A-Box: lassen wir erstmal weg
- Dh. zuerst betrachten wir: Konstruktion von komplexen Beschreibungen

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

Die Basissprache \mathcal{AL}

- \mathcal{AL} = Attributive Language
- \mathcal{ALC} AL mit Komplementen (Gert Smolka, MSS 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} (eine Basissprache)

Komplexe Konzeptbeschreibungen werden durch **Konzept-Terme** gebildet:

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

Beachte: in \mathcal{AL} ist $\exists R.C$ nicht erlaubt, nur $\exists R.\top$
 $\neg A$ ist erlaubt, aber nicht $\neg C$!

Erweiterungen von \mathcal{AL} später.

Berechnungen zu Konzepten

Man will fragen und berechnen können:

- ob zwei verschiedene Konzepte inhaltlich immer gleich sind;
- ob für zwei definierte Konzepte C, D , immer $C \subseteq D$ gilt.
- Gilt für zwei Konzeptterme C, D (inhaltlich) immer $C = D$?
-?
- Hoffnung, bzw. Designziel der Sprachen: Gute Komplexität der zugehörigen Algorithmen ?

Das ist u.a. der Grund für manche (scheinbar unmotivierten) Einschränkungen.

Anwendungs-Szenario (A-Box)

- Es gibt atomare Konzepte: **Mann, Frau, Haus, ...**
- Es gibt Relationen: zB **wohntIn**.
- konkrete Objekte (**Charles, Franz, Maria, Haus267, ...**)
- konkrete Angaben: **wohntIn(Charles, Haus267)**
- **Benachbart(Haus1, Haus267)**

Aktionen und Aufgaben:

- Komplexere Konzepte definieren, wenn die interessant sind.
- Fragen an die konkreten Daten (d.h. die A-Box) z.B. Welche Personen wohnen benachbart?
- \mathcal{AL} : welche komplexen Konzepte sind definierbar
- Untersuchungen zu \mathcal{AL} beinhalten, wie die abstrakte Ebene strukturiert ist, welche Beziehungen es schon in der abstrakten Sprache gibt, und zu welchen Fragestellungen bzgl. der T-Box (und A-Box) es Algorithmen gibt?

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. \perp

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. $I(R) \subseteq \Delta \times \Delta$.

Definition von I auf komplexen Konzeptbeschreibungen:

$$I(C_1 \sqcap C_2) = I(C_1) \cap I(C_2)$$

$$I(\perp) = \emptyset$$

$$I(\top) = \Delta$$

$$I(\neg A) = \Delta \setminus I(A)$$

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

$$I(\forall R.C) = \{x \in \Delta \mid \forall y.(x, y) \in I(R) \Rightarrow y \in I(C)\}$$

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 :

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

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

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

Beispiele: Interpretationen (2)

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

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

Beispiel: Äquivalenz

Die Konzepte

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

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

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

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

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** $f : \Delta \rightarrow \Delta$ 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 nicht vorkommt)

Erweiterungen (2)

Mögliche weitere Konstrukte für komplexere Konzeptbeschreibungen:

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

R, R_i : Rolle oder atomare Funktion

Symbole

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

AND	entspricht	\sqcap
OR	entspricht	\sqcup
NOT	entspricht	\neg
SOME	entspricht	\exists
ALL	entspricht	\forall
;	entspricht	\circ (als Komposition)
...		

Beispiele (1)

Atomar: **Mensch**, **Frau**, **Mann**, **hatKind** und **studiertFach**
Man hätte gerne als Axiome, bzw. zusätzliche Gleichung.

$$\mathbf{Frau} \sqcap \mathbf{Mann} = \perp$$

$$\mathbf{Mensch} \equiv \mathbf{Frau} \sqcup \mathbf{Mann}$$

Die Axiome sollten / können bei allgemeinen Aussagen und Inferenzen berücksichtigt werden

Beispiele (2)

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

Genauer:

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

Weitere Konzepte:

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

$$\mathbf{Studentin} := \mathbf{Frau} \sqcap (\exists \mathbf{studiertFach.T})$$

Mögliche Inferenz: **Mutter** \subseteq **Frau**.

Beispiele (3)

Beispiel mit Anzahlbeschränkungen:

$$\mathbf{BigamistIn} := (\geq 2 \mathbf{verheiratetMit}) \sqcap (\leq 2 \mathbf{verheiratetMit}) \sqcap (\forall \mathbf{verheiratetMit.Mensch})$$

Beachte: Ist **verheiratetMit** eine (atomare) Funktion
(d.h. es ist keine Relation; nur eine Person ist möglich),

dann liefert die Semantik

$$I(\mathbf{Bigamist}) = \emptyset \text{ für jede Interpretation } I.$$

Beispiele (4)

Beispiele mit Pfadgleichungen:

Rolle **isstGerne**, dann beschreibt:

$$\mathbf{Mensch} \sqcap (\mathbf{verheiratetMit} \circ \mathbf{isstGerne} = \mathbf{isstGerne})$$

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

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

$$\mathbf{Mann} \sqcap (\mathbf{hatNachnamen} = \mathbf{hatMutter} \circ \mathbf{hatNachnamen})$$

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 $I(R) : \Delta \rightarrow \Delta$
(einstellige partielle Funktion)

Da Funktionen auch Relationen sind: Semantik wie Relationen.

Semantik der erweiterten Sprachen (2)

Erweiterung von I auf die Anzahlbeschränkungen:

$$I(C_1 \sqcup C_2) = I(C_1) \cup I(C_2)$$

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

$$I(\leq n R) = \{x \in \Delta \mid |\{y \in \Delta \mid (x, y) \in I(R)\}| \leq n\}$$

$$I(\geq n R) = \{x \in \Delta \mid |\{y \in \Delta \mid (x, y) \in I(R)\}| \geq n\}$$

$$I(\leq n R.C) = \{x \in \Delta \mid |\{y \in \Delta \mid (x, y) \in I(R) \wedge y \in I(C)\}| \leq n\}$$

$$I(\geq n R.C) = \{x \in \Delta \mid |\{y \in \Delta \mid (x, y) \in I(R) \wedge y \in I(C)\}| \geq n\}$$

Semantik der erweiterten Sprachen (3)

Erweiterung von I (Fortsetzung):

$$I(R_1 \circ R_2 \circ \dots \circ R_n = R'_1 \circ R'_2 \circ \dots \circ R'_m)$$

$$= \left\{ x \in \Delta \mid \begin{array}{l} \{y \in \Delta \mid (x, y) \in (I(R_1) \circ \dots \circ I(R_n))\} \\ = \{y \in \Delta \mid (x, y) \in (I(R'_1) \circ \dots \circ I(R'_m))\} \end{array} \right\}$$

wobei \circ die Komposition von Relationen ist

$$(D.h. I(R_1 \circ R_2) = \{(x, z) \mid (x, y) \in I(R_1) \wedge (y, z) \in I(R_2)\})$$

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

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

Beispiel

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

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 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. $\mathcal{AL}\mathcal{E}\mathcal{N}$: $\mathcal{AL} + \exists R.C +$ number restrictions.

Wesentliche **Einschränkung**: keine Rollenoperationen.

(Erweiterungen kommen noch)

$$\mathcal{AL}\mathcal{U}\mathcal{E} = \mathcal{AL}\mathcal{C}$$

Gleichheit $\mathcal{AL}\mathcal{U}\mathcal{E} = \mathcal{AL}\mathcal{C}$, andere Richtung:

Z.B. Stelle Komplemente in $\mathcal{AL}\mathcal{U}\mathcal{E}$ 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

- Einige Sprachen der \mathcal{AL} -Familie sind semantisch gleich.
- Z.B. $\mathcal{AL}\mathcal{U}\mathcal{E} = \mathcal{AL}\mathcal{C}$.
Eine Richtung ist einfach, denn es gilt:

$$C \sqcup D \equiv \neg(\neg C \sqcap \neg D) \quad \text{und} \\ \exists R.C \equiv \neg(\forall R.\neg C).$$

Rechnung für die letzte Gleichung:

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

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

Insgesamt erhält man 8 (echt) verschiedene Sprachen (von 16):

$$\mathcal{AL}, \mathcal{AL}\mathcal{C}, \mathcal{AL}\mathcal{C}\mathcal{N}, \mathcal{AL}\mathcal{N}, \mathcal{AL}\mathcal{U}, \mathcal{AL}\mathcal{E}, \mathcal{AL}\mathcal{U}\mathcal{N}, \mathcal{AL}\mathcal{E}\mathcal{N}$$

Name	Konstrukte	implizite Konstrukte
\mathcal{AL}	$\sqcap, \sqcup, \top, \neg A, \forall R.C, \exists R.\top$	
$\mathcal{AL}\mathcal{C}$	$\mathcal{AL}, \neg C$	$\sqcup, \exists R.C$
$\mathcal{AL}\mathcal{C}\mathcal{N}$	$\mathcal{AL}, \neg C, (\leq n R)$	$\sqcup, \exists R.C, (\geq n R)$
$\mathcal{AL}\mathcal{N}$	$\mathcal{AL}, (\leq n R), (\geq n R)$	
$\mathcal{AL}\mathcal{U}$	\mathcal{AL}, \sqcup	
$\mathcal{AL}\mathcal{E}$	$\mathcal{AL}, \exists R.C$	
$\mathcal{AL}\mathcal{U}\mathcal{N}$	$\mathcal{AL}, \sqcup, (\leq n R), (\geq n R)$	
$\mathcal{AL}\mathcal{E}\mathcal{N}$	$\mathcal{AL}, \exists R.C, (\leq n R), (\geq n R)$	

$\mathcal{AL}\mathcal{C}$ und $\mathcal{AL}\mathcal{C}\mathcal{N}$ sind ausdrucksstark:

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:

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

Schnitt von Rollen: \mathcal{ALCR}

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

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

Namensgebung dann:

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

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

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

Erweiterungen (3)

Konstrukte, um weitere **komplexe Rollen** zu konstruieren:

(R^{-1})	inverse Rolle
R^c	Komplement der Rolle
$R_1 \cup R_2$ und $R_1 \cap R_2$:	Vereinigung und Schnitt
$(R_1 \circ R_2)$	Komposition
(R^+)	transitiver Abschluss
(RESTRICT $R C$)	Rolleneinschränkung

Axiome für Rollen

Reflexiv(R)	Transitiv(R)
Symmetrisch(R)	Antisymmetrisch(R)
funktionale Rolle (R)	...

\mathcal{ALC} -Sprachen haben keine Rollenoperationen.

Subsumtionsalgorithmen zu den entsprechenden Erweiterung können höhere Komplexität haben. Es gibt dazu Untersuchungen und detaillierte Algorithmen und Komplexitäts-Ergebnisse (später..)

Erweiterungen zu Rollen

Beispiele:

älter-als	transitive Rolle
Hat-Farbe	funktionale Rolle
Hat-Adresse	funktionale Rolle
ist-direkt-verbunden-mit	symmetrische Rolle
ist-indirekt-verbunden-mit	symmetrische, transitive Rolle

Beziehungen zwischen Konzepten

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 man verwenden:

(d.h. wenn die entsprechenden Konstrukte erlaubt sind)

- C ist inkonsistent, gdw. $C \equiv \perp$.
- $C \equiv D$ gdw. $C \sqsubseteq D$ und $D \sqsubseteq C$.
- C ist disjunkt zu D gdw. $C \sqcap D$ inkonsistent.
- C inkonsistent, gdw. C wird von \perp subsumiert ($C \sqsubseteq \perp$)
- 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 können Subsumtion, Disjunktheitstest und Konsistenztest (durch leichte Umformulierung der Frage) mit dem gleichen Algorithmus getestet werden, 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 Beschreibungslogiken (Auswahl)

Ontologien der Life-Sciences (Medizin, Biologie),

- SNOMED CT (ein Akronym für Systematized Nomenclature of Medicine – Clinical Terms) ist eine medizinische Nomenklatur, die ca. 350.000 Konzepte umfasst
- The National Cancer Institute Thesaurus, ca. 45.000 Konzepte
- GO: gene ontology: ca. 20.000 Konzepte
- „GALEN was concerned with the computerisation of clinical terminologies.“

aktuelleres (Zahlen, Namen, Verwendung usw..) siehe Internet

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 Subsumtion in \mathcal{EL} ist polynomiell und damit anwendungsgeeignet.

Beispiel (das so nicht gewollt war, aber stimmte):

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

Snomed CT: Erlaubte Operatoren

(aus Wikipedia:)

- Top, bottom
- Primitive roles and concepts with asserted parent(s) for each
- Concept definition and conjunction but NOT disjunction or negation
- Role hierarchy but not role composition
- Domain and range constraints
- Existential but not universal restriction
- A restricted form of role inclusion axiom
($xRy \wedge ySz \implies xRz$) ((**Vermutlich S: Untermenge**))
- General Concept Inclusion Axioms ($A \sqsubseteq B$).



Terminologien

Terminologie

= Vereinbarung (Definition) von Namen für Konzepte und: Axiome bzw. Beschränkungen.

- Menge von terminologischen Axiomen der Formen
 $C \sqsubseteq D$ oder $C \equiv D$ wobei C, D Konzepte (bzw. Konzeptnamen) sind.
- Wenn es Rollenterme gibt, dann auch Axiome der Form
 $R \sqsubseteq S$ oder $R \equiv S$ wobei R, S 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 Definitionen der Form $A = C$ oder $R = S$.

T-Box

Eine Menge von Definitionen von Konzepten und Rollen, wobei jeder Name höchstens einmal definiert wird (d.h. einmal links vorkommt) nennt man **T-Box**

- Eine T-Box ist **zyklisch** (bzw. **rekursiv**), wenn in der Definition mindestens eines Namen dieser Name selbst referenziert wird. (evtl. implizit)
- Anderenfalls ist die T-Box **azyklisch** (**nicht rekursiv**).

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 das i.a. nicht!

Beispiel für eine T-Box

Frau	≡	Person \sqcap Weiblich
Mann	≡	Person \sqcap \neg Frau
Mutter	≡	Frau \sqcap \exists hatKind.Person
Vater	≡	Mann \sqcap \exists hatKind.Person
Eltern	≡	Vater \sqcup Mutter
Grossmutter	≡	Mutter \sqcap \exists hatKind.Eltern
MutterMitVielenKindern	≡	Mutter \sqcap (≥ 3 hatKind)
MutterMitTochter	≡	Mutter \sqcap (\forall hatKind.Frau)
Ehefrau	≡	Frau \sqcap (\exists hatEhemann.Mann)

Diese T-Box ist azyklisch

Nicht definierte Namen und **definierte Namen**

Beispiel: Interpretation

Basisinterpretation I_0 legt nur fest

$\Delta, I_0(\mathbf{Person}), I_0(\mathbf{Weiblich}), I_0(\mathbf{hatKind}), I_0(\mathbf{hatEhemann})$

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

$I(\mathbf{Person})$	$:= I_0(\mathbf{Person})$
$I(\mathbf{Weiblich})$	$:= I_0(\mathbf{Weiblich})$
$I(\mathbf{hatKind})$	$:= I_0(\mathbf{hatKind})$
$I(\mathbf{hatEhemann})$	$:= I_0(\mathbf{hatEhemann})$
$I(\mathbf{Frau})$	$:= I_0(\mathbf{Person}) \cap I_0(\mathbf{Weiblich})$
$I(\mathbf{Mann})$	$:= I_0(\mathbf{Person}) \cap (\Delta \setminus (I_0(\mathbf{Person}) \cap I_0(\mathbf{Weiblich})))$

usw.

Beispiele für zyklische T-Boxen

zyklische T-Boxen überspringen: zu Inklusionen Nr. 75
Menschen sind alle Tiere, deren Eltern nur Menschen sind:

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

Gibt es ein Modell?

- Sei $\Delta, I_0(\text{Tier}), I_0(\text{HatEltern})$ festgelegt
- Idee von vorher:
 $I(\text{Mensch}') := I_0(\text{Tier}) \cap \{x \mid \forall y.(x, y) \in I_0(\text{HatEltern}) \implies y \in I(\text{Mensch}')\}$
- Das ist eine **zyklische Definition!** (evtl. rekursiv interpretierbar), unendlich
- In der Bibel wurde das Problem vermieden durch:
Adam = erster Mensch, Eva = erste Frau.
- Man benötigt eine Interpretation (als Menge)
 $I(\text{Mensch}') \subseteq \Delta$, 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“.

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

Datenstruktur: Binärer Baum:

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

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.

Kleinsten Fixpunkt:

Folge von Interpretationen $I_i, i = 0, 1, 2, \dots$, 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 für 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 $I_i, i = 0, 1, 2, \dots$, 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: $A \equiv \neg A$.

Kleinsten 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 $\Delta = I_1(A) \not\subseteq I_2(A) = \emptyset$.

D.h. es gibt keinen kleinsten Fixpunkt

Größter Fixpunkt:

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

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

D.h. es gibt keinen größten Fixpunkt

Beispiele (2)

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

I_B sei:

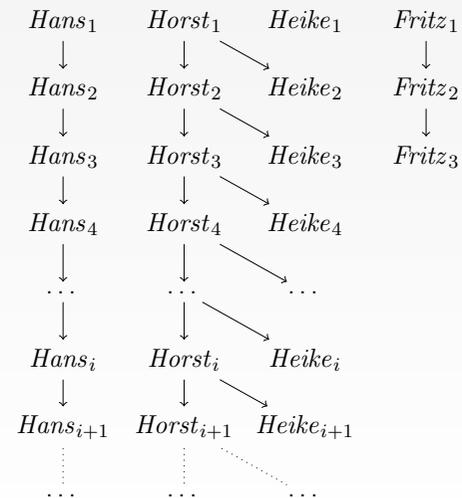
$$\begin{aligned} \Delta &= \{Hans_i \mid i \in \mathbb{N}\} \\ &\cup \{Horst_i \mid i \in \mathbb{N}\} \\ &\cup \{Heike_i \mid i \in \mathbb{N}\} \\ &\cup \{Fritz_i \mid i \in \{1, 2, 3\}\} \end{aligned}$$

$$\begin{aligned} I_B(\mathbf{Mann}) &= \{Hans_i \mid i \in \mathbb{N}\} \\ &\cup \{Horst_i \mid i \in \mathbb{N}\} \\ &\cup \{Fritz_i \mid i \in \{1, 2, 3\}\} \end{aligned}$$

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

Beispiele (3)

Als Graph (gerichtete Kante = „hatKind“)



Beispiele (4)

Kleinsten Fixpunkt:

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

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

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

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

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

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

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

Beispiele (5)

Größten Fixpunkt:

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

$$I_1(\mathbf{MnurS}) = I_B(\mathbf{Mann}) \cap \{x \in \Delta \mid \forall y.(x, y) \in I_B(\text{hatKind}) \Rightarrow y \in I_0(\mathbf{MnurS})\} \\ = I_B(\mathbf{Mann}) \cap (\Delta) \\ = I_B(\mathbf{Mann})$$

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

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

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

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

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, dann kann man sowohl einen kleinsten als auch einen größten Fixpunkt der Interpretationen als Erweiterung einer Basisinterpretation definieren.

Begründung: Monotonie, da „Doppelnegation“

Beispiel

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



Beispiel (2)

T-Box dazu:

$$\text{Huhn} \sqsubseteq (\exists \text{kommtAus.Ei})$$

$$\text{Ei} \sqsubseteq (\exists \text{gelegtVon.Huhn})$$

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

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

Fazit:

- In allen **endlichen** Modellen mit $I(\text{Huhn}) \neq \emptyset$ 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.
- Oder: man erlaubt die Spezifikation von weiteren Eigenschaften von Relationen:
z.B. nicht-zyklisch

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

$$\text{Mann} \sqsubseteq \text{Person}$$

wird zu

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

A-Boxen

- A-Box = Assertions
- Konkrete Annahmen über
 - Existenz von Individuen
 - zur Enthaltensein in Konzepten
 - Relationen zwischen Individuen
- T-Box \approx Datenbankschema
- A-Box \approx 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 Individuename.
- $R(a, b)$ 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.

$(\exists \text{hatKind.Person})(\text{Michael})$



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

- T-Box: Menge von Namen und terminologischen Axiomen; und
- A-Box: Menge von Annahmen über Existenz und Eigenschaft von Objekten



Beispiel

T-Box:

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

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?“
 D.h.: 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

$A \equiv \{a_1, \dots, a_n\}$ wobei a_i Individuennamen sind

Semantik:

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

Beispiel:

Grundfarben $\equiv \{\text{rot, blau, gelb}\}$

Open-World- vs. Closed-World-Semantik

Datenbanken: **Closed-World-Semantik**

- Fakten die in der DB stehen sind wahr.
- Andere sind 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, (Unter Beachtung der Open-World-Assumption)
d.h. Schlussfolgern ist konservativ.



Beispiel

Datenbank / A-Box enthält nur:

hatKind(Maria, Peter)

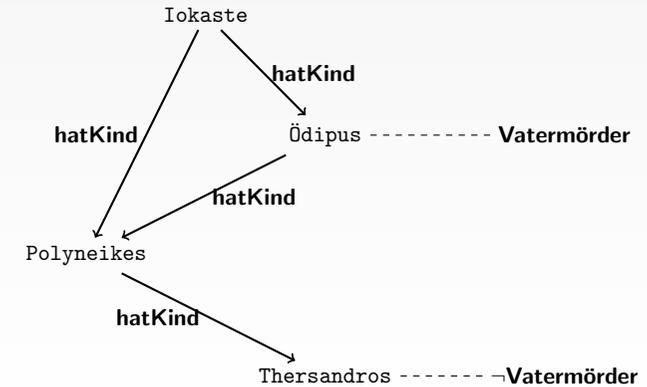
- Closed-World-Semantik: Es gilt: Peter hat keine Geschwister
- Open-World-Semantik: Unbekannt, ob Peter Geschwister hat

Durch (≤ 1 **hatKind**)(Maria) kann sichergestellt werden, dass Peter keine Geschwister hat

Beispiel

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

hatKind(Iokaste, Ödipus) **hatKind**(Iokaste, Polyneikes)
hatKind(Ödipus, Polyneikes) **hatKind**(Polyneikes, Thersandros)
Vatermörder(Ödipus) \neg **Vatermörder**(Thersandros)



Beispiel (2)

hatKind(Iokaste, Ödipus) **hatKind**(Iokaste, Polyneikes)
hatKind(Ödipus, Polyneikes) **hatKind**(Polyneikes, Thersandros)
Vatermörder(Ödipus) \neg **Vatermörder**(Thersandros)

Kann man aus dieser A-Box etwas schließen?

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

- Iokaste hat zwei Kinder, Ödipus und Polyneikes.
- Ödipus erfüllt **Vatermörder**,
- kein Wissen über **Vatermörder**(Polyneikes).
- Trotzdem kann man schließen (auch bzgl. 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 Algorithmen für den Subsumtionstest, d.h. ob gilt:

$C \sqsubseteq D$

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:

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



Struktureller Subsumtionstest

Allgemeines Vorgehen bei strukturellem Subsumtionstest
(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.

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 gibt sogar eine Interpretation, so dass für alle C : $I(C) = \Delta$.

Also:

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

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

Normalform in \mathcal{FL}_0

Normalform in \mathcal{FL}_0 :

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

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

Normalform-Algorithmus für \mathcal{FL}_0

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

Begründung:

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

Vergleich der Normalformen in \mathcal{FL}_0

Sei

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

und

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

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 :

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

Normalformberechnung für C_2 :

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

Subsumtionstest für \mathcal{FL}^-

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

Anpassung des Subsumtionsalgorithmus für \mathcal{FL}^-

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

$$\begin{aligned} &A_1 \sqcap \dots \sqcap A_m \\ &\sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n \\ &\sqcap \exists R'_1.T \sqcap \dots \sqcap \exists R'_k.T \end{aligned}$$

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

Auch der \mathcal{FL}^- -Subsumtions-Algorithmus ist korrekt und vollständig.

Der Zeitaufwand des Algorithmus ist $O(n * \log(n))$.

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: $\forall R_1, \dots$ kommt nur in $NF(C_1)$ vor. Daraus folgt sofort: $C_2 \not\sqsubseteq C_1$.
- Für $C_1 \sqsubseteq C_2$: Für $\forall R_2, \dots$ 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 $\forall R_1, \dots$ rekursiver Vergleich

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

- Daher gilt: $C_1 \sqsubseteq C_2$.

Struktureller Subsumtionstest für weitere DL-Sprachen

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

Subsumtions-Algorithmus für \mathcal{AL}

\mathcal{AL} : Neue Konzepte: \top und $\exists R.C$.

Insgesamt: $A, \top, \perp, \neg A, C \sqcap D, \forall R.C, \exists R.C$

- NF-Berechnung:

- $\top \sqcap C \rightarrow C$
- $\forall R.C \rightarrow \top$
- $\neg \top \rightarrow \perp$
- $\neg \perp \rightarrow \top$

- Vergleich: Beachte $C \sqsubseteq \top$ für alle C

Konflikte bei Anzahlbeschränkungen

Hat man Anzahlbeschränkungen, muss man mehr Konflikte beachten.

Z.B.

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

Weitere Sprachen mit strukturellem Subsumtionsalgorithmus

$\mathcal{AL}\mathcal{E}$ \mathcal{AL} erweitert um $\exists R.C$.

$\mathcal{AL}\mathcal{U}$ \mathcal{AL} erweitert um \sqcup .

$\mathcal{AL}\mathcal{N}$ \mathcal{AL} erweitert um Anzahlbeschränkungen ($\leq n R$).

- $\mathcal{AL}\mathcal{N}$ hat einen polynomiellen und strukturellen Subsumtionsalgorithmus.
- \mathcal{FL} hat **keinen** strukturellen Subsumtionstest: das Subsumtionsproblem ist PSPACE-vollständig. (kommt noch genauer)

Grenzen des strukturellen Subsumtionsalgorithmus

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

Z.B. gibt es für $\mathcal{AL}\mathcal{C}$ **keinen** strukturellen Subsumtionsalgorithmus!

Subsumtion und Äquivalenzen in \mathcal{ALC}

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

In \mathcal{ALC} gilt:

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

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. $(C_1 \sqcap \neg C_2) \equiv \perp$

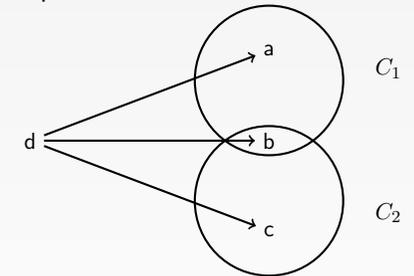
Und statt $C \not\equiv \perp$ kann man $C \not\sqsubseteq \perp$ testen.
Aber Komplexität ist dann co- bzgl Subsumtion.
In diesem Fall auch PSPACE-vollständig.

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

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



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

$$\begin{aligned} I((\forall R.C_1) \sqcup (\forall R.C_2)) &= \{x \in \{a, b, c, d\} \mid \forall y.(x, y) \in \{(d, a), (d, b), (d, c)\} \Rightarrow y \in \{a, b\}\} \\ &\quad \cup \{x \in \{a, b, c, d\} \mid \forall y.(x, y) \in \{(d, a), (d, b), (d, c)\} \Rightarrow y \in \{b, c\}\} \\ &= \{a, b, c\} \quad \text{Denn } a, b, c \text{ haben keine Relation} \end{aligned}$$

Subsumtionsalgorithmus für \mathcal{ALC}

Fallunterscheidungsbaum- (bzw. Tableau-) Verfahren, das prüft:
Ist $C := (C_1 \sqcap \neg C_2)$ konsistent?

Idee:

- Versuche I zu konstruieren mit $I(C) \neq \emptyset$
- Fallunterscheidungen / Untersuchung aller Fälle ist nötig.
- Mögliche Ausgänge: Modell existiert $\rightarrow C$ konsistent
- Es gibt kein Modell: C inkonsistent, d.h. $C_1 \not\sqsubseteq C_2$.

Neue Datenstruktur: **Constraint-Systeme**, bzw. partielle Interpretationen; um Anforderungen an I zu beschreiben

Subsumtionsalgorithmus für \mathcal{ALC} (2)

Definition

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

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

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 $\forall a \in I(X) : \exists b. (a, b) \in I(R) : b \in I(Y)$

Entfaltungsregeln

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

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

Subsumtionsalgorithmus für \mathcal{ALC} (3)

(Subsumtions-) Algorithmus für die Frage: ist C konsistent?

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

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
- $x : X, X \sqsubseteq \perp$

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:

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

Erster Schritt: Starte mit

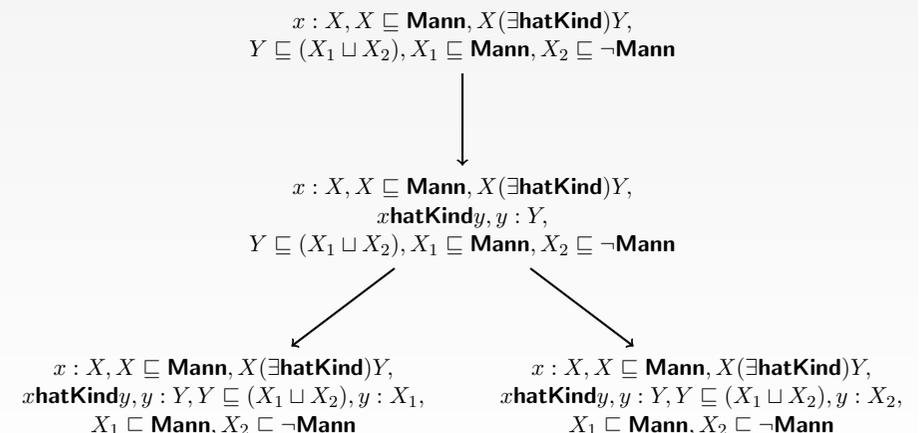
$$x : X, X \sqsubseteq \text{Mann} \sqcap \exists \text{hatKind}.(\text{Mann} \sqcup \neg \text{Mann}).$$

Zweiter Schritt: Entfalten:

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

Beispiel (2)

Vervollständigung ergibt das Tableau:



Beide Blätter: nicht widersprüchlich.

Beispiel (3)

Modelle: an den Blättern:

- Für

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

kann man ablesen

$$\Delta = \{x, y\}, I(X) = \{x\}, I(Y) = \{y\}, I(X_1) = \{x, y\}, I(\mathbf{hatKind}) = \{(x, y)\} \\ \text{und } I(\mathbf{Mann}) = \{x, y\}, I(X_2) = \emptyset.$$

- Für

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

kann man ablesen

$$\Delta = \{x, y\}, I(X) = \{x\}, I(Y) = \{y\}, I(X_2) = \{y\}, I(\mathbf{hatKind}) = \{(x, y)\}, \\ I(\mathbf{Mann}) = \{x\}, I(X_1) = \{x\}.$$

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:

$$X(\leq n R)Y \text{ und } X(\geq n R)Y.$$

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}^- ($\sqcap, \forall R.C$ und $(\exists R.T)$) hat einen **polynomiellen** Subsumtionstest
- Polynomiell:
 - $\mathcal{AL}\mathcal{E}$ \mathcal{AL} erweitert um $\exists R.C$.
 - $\mathcal{AL}\mathcal{U}$ \mathcal{AL} erweitert um \sqcup .
 - $\mathcal{AL}\mathcal{N}$ \mathcal{AL} erweitert um Anzahlbeschränkungen ($\leq n R$).
- \mathcal{FL} ($\sqcap, \forall R.C, (\exists R.C)$) hat ein **PSPACE-vollständiges** Subsumtionsproblem, 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	$I(R^+) = \text{trans. Abschluss von } I(R)$.

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 PDL: propositional dynamic logic.)



Beispiel zu Erweiterungen und Rollenterme

Beispiele

Inseln die durch Brücken verbunden sind, oder mit Fährverbindungen; und Häuser auf den Inseln.

Konzepte: Insel, Haus

Definierte Konzepte: z.B.

- Insel mit Haus: $(\geq 1 \text{ hat-Haus}) \sqcap \text{Insel}$
- Insel mit Brücken zu mehr als drei Inseln: $(\geq 3 \text{ hat-Bruecke-zu.Insel}) \sqcap \text{Insel}$.
- Insel mit Fährverbindung. $(\geq 1 \text{ hat-Faehrverbindung-zu}) \sqcap \text{Insel}$

Beispiel zu Erweiterungen und Rollenterme



Mit Rollentermen:

- verbunden durch Brücken: $(\text{hat-Bruecke-zu} \sqcup \text{hat-Bruecke-zu}^{-1})^+$
- Insel mit Brücken zu mehr als drei Inseln: $(\geq 3 \text{ hat-Bruecke-zu.Insel}) \sqcap \text{Insel}$.
- Insel mit Fährverbindung. $(\geq 1 \text{ hat-Faehrverbindung}) \sqcap \text{Insel}$
- Insel-ohne-Brücke: $(= 0 (\text{hat-Bruecke-zu} \sqcup \text{hat-Bruecke-zu}^{-1})) \sqcap \text{Insel}$

Man kann auch Rollen als symmetrisch, transitiv, reflexiv, usw. definieren.

Beispiel zu Erweiterungen und Rollenterme

- Zwei Inseln sind mit verschiedenen Brücken verbunden?
 - Konzept **Brücke** fehlt.
- Insel ist mit Festland verbunden.
 - Konzept/Insel? **Festland** nicht definiert.
- Insel hat Fährhäfen.
 - nicht modelliert.

Diskussion:

- Wenn die Relation transitiv ist, kann man damit den Weg nicht mehr modellieren
- Wenn die Relation symmetrisch ist, dann kann man die Richtung des Weges nicht modellieren.
- Zeit-Dauern, Kapazität muss man getrennt modellieren.



Übersetzung \mathcal{ALC} in Prädikatenlogik

Atomare Konzepte A werden in einstellige Prädikate P_A übersetzt.
Rollen R werden zweistellige Prädikate P_R übersetzt.

$\llbracket \cdot \rrbracket_\alpha$: Übersetzung von \mathcal{ALC} -Formeln in Prädikatenlogische Formel mit freier Variable α

$$\begin{aligned} \llbracket A \rrbracket_x &= P_A(x) \\ \llbracket C \sqcap D \rrbracket_x &= \llbracket C \rrbracket_x \wedge \llbracket D \rrbracket_x \\ \llbracket C \sqcup D \rrbracket_x &= \llbracket C \rrbracket_x \vee \llbracket D \rrbracket_x \\ \llbracket \exists R.C \rrbracket_x &= \exists y. P_R(x, y) \wedge \llbracket C \rrbracket_y \\ \llbracket \forall R.C \rrbracket_x &= \forall y. P_R(x, y) \implies \llbracket C \rrbracket_y \end{aligned}$$

Übersetzung der Subsumtion:

$$\llbracket C \sqsubseteq D \rrbracket = \forall x. (\llbracket C \rrbracket_x \implies \llbracket D \rrbracket_x)$$

Satz

$C \sqsubseteq D$ genau dann, wenn $\llbracket C \sqsubseteq D \rrbracket$ eine Tautologie ist.

Verbindung zur Modallogik



Description Logic Varianten:

entsprechen teilweise Varianten von
(aussagenlogischen) Modallogiken.

- Objekte \leftrightarrow Welt
- Konzept \leftrightarrow modallogische Aussage.
Konzept nichtleer \iff zug. modallogische Aussage ist wahr.
- $\forall R.C \leftrightarrow$ modallogischer Operator \square mit Rolle R als Erreichbarkeits-Relation
- **Subsumtionstest**: entspricht $F_1 \models F_2$ in Modallogik

Zum Beispiel korrespondiert \mathcal{ALC} mit der Modallogik K .

Tautologie-Erkennung in K ist PSPACE-complete.

OWL



- **Web Ontology Language** = 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
- insbesondere: 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 $SHIN(\mathbf{D})$
- OWL DL entspricht der Beschreibungslogik $SHOIN(\mathbf{D})$

wobei

- S steht für ALC + **transitive Rollen**
Namensgebung: Bezug zur Modallogik S_4 .
- \mathcal{H} steht für Rollen**hierarchien**, $R \sqsubseteq S$
- \mathcal{I} steht für **inverse Rollen**, R^-
- \mathcal{N} : number restrictions ($\leq n R$), ($\geq n R$)
- \mathcal{O} steht für “**nominal**”: $\{o\}$ (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-Ontologie: T-Box + Rollenhierarchie
- 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	\perp
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$
complementOf	$\neg C$
oneOf	$\{a_1, \dots, a_m\}$
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>
```

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

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



Reasoner für OWL-Ontologie

siehe Internet und Literatur

Keine Angaben hier

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	$C_1 \sqcap C_2 \equiv \perp$ 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



Komplexität

OWL-Variante	passende DL	Subsumtion- und Konsistenztest
OWL Lite	$SHIN(\mathbf{D})$	EXPTIME-vollständig
OWL DL	$SHOIN(\mathbf{D})$	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 Subsumtion in \mathcal{EL} ist in polynomieller Zeit testbar, und damit (eher) anwendungsgesegnet.

Passt zu:

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

Aktive Nutzung und Forschung zu Grundlagen und Service