

Beschreibungslogik

Kapitel 2: Grundlagen

Sommersemester 2017

Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

<http://tinyurl.com/ss17-bl>

Vorlesungsübersicht

Kapitel 1: Einleitung

Kapitel 2: Grundlagen

Kapitel 3: Ausdrucksstärke und Modellkonstruktionen

Kapitel 4: Tableau-Algorithmen

Kapitel 5: Komplexität

Kapitel 6: Effiziente Beschreibungslogiken

Kapitel 7: ABoxen und Anfragebeantwortung

Kapitel 2: Grundlagen

- 1 Die Beschreibungslogik ALC
- 2 TBoxen
- 3 Schlussfolgerungsprobleme
- 4 Erweiterungen von ALC

Kapitel 2: Grundlagen

- 1 Die Beschreibungslogik ALC
- 2 TBoxen
- 3 Schlussfolgerungsprobleme
- 4 Erweiterungen von ALC

Es gibt viele verschiedene Beschreibungslogiken:

- viele mögliche Kompromisse bzgl. Ausdrucksstärke vs. Komplexität des Schlussfolgerns
- verschiedene Anwendungen haben unterschiedliche Anforderungen

Hier zunächst *ALC*:

- die einfachste BL mit allen Booleschen Konstruktoren
- eingeführt 1991 von Manfred Schmidt-Schauß & Gert Smolka
- steht für *Attributive (Concept) Language with Complement*

Wir fixieren von nun an

- eine abzählbar unendliche Menge von **Konzeptnamen**
Diese bezeichnen Klassen / unäre Prädikate
z. B. Person, Kurs, Universität, Tafel, Student, etc.
- eine abzählbar unendliche Menge von **Rollennamen**
Diese bezeichnen Relationen / binäre Prädikate
z. B. hört, lehrt, istTeilVon, etc.

Wir nehmen die beiden Mengen als disjunkt an und unterscheiden Konzept- und Rollennamen über Groß- / Kleinschreibung.

Konzepte dienen der *Beschreibung* einer Klasse von Objekten, sind zusammengesetzt aus

- Konzeptnamen
- Rollennamen
- Konzeptkonstruktoren
- (manchmal auch Rollenkonstruktoren)

Es gibt viele verschiedene Konstruktoren.

Verschiedene Konstruktormengen ergeben verschiedene Beschreibungslogiken.

ALC: Syntax

Definition 2.1 (ALC-Konzepte)

Die Menge der **ALC-Konzepte** ist induktiv definiert:

- Jeder Konzeptname ist **ALC-Konzept**.
- Wenn C, D **ALC-Konzepte**, so auch
 - $\neg C$ (Negation)
 - $C \sqcap D$ (Konjunktion)
 - $C \sqcup D$ (Disjunktion)
- Wenn C **ALC-Konzept** und r Rollenname, so sind
 - $\exists r.C$ (Existenzrestriktion)
 - $\forall r.C$ (Werterestriktion)

ALC-Konzepte.

ALC: Syntax

Verwendete Symbole:

- A, B für Konzeptnamen
- C, D für zusammengesetzte Konzepte
- r, s für Rollennamen

Zwei nützliche Abkürzungen: wir schreiben

- \top für $A \sqcup \neg A$
- \perp für $A \sqcap \neg A$

wobei A ein beliebiger Konzeptname ist.

ALC: Syntax

Präzedenzregel:

\neg, \exists, \forall binden stärker als \sqcap und \sqcup

Also zum Beispiel:

$\forall r.(\exists r.A \sqcap B)$ steht für $\forall r.((\exists r.A) \sqcap B)$
und nicht für $\forall r.(\exists r.(A \sqcap B))$

Keine Präzedenz zwischen \sqcap und $\sqcup \rightsquigarrow$ Klammern verwenden!

ALC: Semantik

Definition 2.2 (ALC-Semantik)

Eine *Interpretation* \mathcal{I} ist Paar $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ mit

- $\Delta^{\mathcal{I}}$ nicht-leere Menge (*Domäne*)
- $\cdot^{\mathcal{I}}$ *Interpretationsfunktion* bildet ab:
 - jeden Konzeptnamen A auf Menge $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
 - jeden Rollennamen r auf Relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ T 2.2

Abbildung $\cdot^{\mathcal{I}}$ wird induktiv auf zusammengesetzte Konzepte erweitert:

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

T 2.2 Forts.

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{es gibt } e \in \Delta^{\mathcal{I}} \text{ mit } (d, e) \in r^{\mathcal{I}} \text{ und } e \in C^{\mathcal{I}}\}$$

$$(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{für alle } e \in \Delta^{\mathcal{I}}, (d, e) \in r^{\mathcal{I}} \text{ impliziert } e \in C^{\mathcal{I}}\}$$

ALC: Semantik

Verwendete Symbole:

- \mathcal{I}, \mathcal{J} für Interpretationen
- d, e für Elemente der Domäne

Für Interpretationen verwenden wir übliche Terminologie für Graphen:

- e ist **r -Nachfolger** von d (in \mathcal{I}) wenn $(d, e) \in r^{\mathcal{I}}$.
- e ist **r -Vorgänger** von d (in \mathcal{I}) wenn $(e, d) \in r^{\mathcal{I}}$.
- Wenn r unwichtig, sprechen wir nur von **Nachfolgern/Vorgängern**.
- \mathcal{I} ist **endlich** gdw. $\Delta^{\mathcal{I}}$ endlich ist.

Extension/Modell

Wir nennen

- $C^{\mathcal{I}}$ die **Extension** des Konzeptes C ;
- jedes $d \in C^{\mathcal{I}}$ eine **Instanz** des Konzeptes C ;
- $r^{\mathcal{I}}$ die **Extension** der Rolle r .

Beachte:

- $\top^{\mathcal{I}}$ ist für jede Interpretation \mathcal{I} identisch mit $\Delta^{\mathcal{I}}$
 \top repräsentiert also immer die Menge **aller** Elemente, z. B.:

$\text{Mensch} \sqcap \exists \text{hatKind}.\top$

Menschen, die ein nicht näher spezifiziertes Kind haben

- $\perp^{\mathcal{I}}$ ist für jede Interpretation \mathcal{I} leer.

Intuitiv repräsentiert \perp , dass etwas unmöglich ist, z. B.:

$\text{Mensch} \sqcap \forall \text{hatKind}.\perp$

Menschen, die keine Kinder haben

T 2.3

Erfüllbarkeit, Subsumtion, Äquivalenz

Folgende Begriffe aus der Aussagenlogik sind auch für ALC relevant:

Definition 2.3 (erfüllbar, subsumiert, äquivalent)

Seien C und D ALC -Konzepte. Dann

- ist C **erfüllbar**, wenn es Interpretation \mathcal{I} gibt mit $C^{\mathcal{I}} \neq \emptyset$;
wir nennen \mathcal{I} dann ein *Modell* von C ;
- wird C **von D subsumiert**, wenn $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in allen Interpretationen \mathcal{I} (Notation $C \sqsubseteq D$);
- sind C und D **äquivalent**, wenn $C^{\mathcal{I}} = D^{\mathcal{I}}$ in allen Interpretationen \mathcal{I} (Notation $C \equiv D$).

T 2.4

In Aussagenlogik sagt man „ D folgt aus C “ statt „ D subsumiert C “.

Erfüllbarkeit, Subsumtion, Äquivalenz

Die üblichen aussagenlogischen Äquivalenzen gelten auch in *ALC*,
z. B.:

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D$$

de Morgansches Gesetz

$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D$$

de Morgansches Gesetz

$$C \sqcap D \equiv D \sqcap C$$

Kommutativität von Konjunktion

$$C \sqcup D \equiv D \sqcup C$$

und Disjunktion

$$C \sqcap (D \sqcap E) \equiv (C \sqcap D) \sqcap E$$

Assoziativität von Konjunktion

$$C \sqcup (D \sqcup E) \equiv (C \sqcup D) \sqcup E$$

und Disjunktion

$$C \sqcap \top \equiv C$$

neutrales Element Konjunktion

$$C \sqcup \perp \equiv C$$

neutrales Element Disjunktion

Kapitel 2: Grundlagen

- 1 Die Beschreibungslogik *ALC*
- 2 TBoxen**
- 3 Schlussfolgerungsprobleme
- 4 Erweiterungen von *ALC*

TBoxen

Zur Erinnerung:

TBoxen (terminologische Boxen)

- definieren Konzepte
- setzen diese zueinander in Beziehung

Konzeptdefinition z. B.

$\text{Student} \equiv \text{Mensch} \sqcap \exists \text{hört.Vorlesung}$

Allgemeines Hintergrundwissen/Constraint z. B.

$\text{Student} \sqcap \text{Vorlesungssaal} \sqsubseteq \perp$

TBoxen — Syntax und Semantik

Definition 2.4 (TBox, Syntax)

Konzeptinklusion ist Ausdruck $C \sqsubseteq D$, mit C, D Konzepten.

TBox ist endliche Menge von Konzeptinklusionen.

Wir verwenden $C \equiv D$ als Abkürzung für $C \sqsubseteq D, D \sqsubseteq C$.

Konzeptinklusion $C \sqsubseteq D$ wird gelesen als „ C impliziert D “. **T 2.5**

Definition 2.5 (TBox, Semantik)

Interpretation \mathcal{I}

- **erfüllt** Konzeptinklusion $C \sqsubseteq D$ gdw. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- ist **Modell** von TBox \mathcal{T} gdw. \mathcal{I} alle Konzeptinklusionen in \mathcal{T} erfüllt.

T 2.5 Forts.

TBoxen — Semantik

Beachte: in einer konkreten Interpretation \mathcal{I} werden

- **Konzepte** als **Mengen von Elementen** interpretiert;
- **TBoxen** entweder erfüllt oder nicht erfüllt;
ihnen wird also ein **Wahrheitswert** zugewiesen.

Intuitiv entspricht jede Interpretation einer **möglichen Welt**.

Manche dieser Welten möchten wir **ausschließen**, weil wir sie aufgrund unseres Wissens nicht für möglich halten.

Genau dazu dienen TBoxen:

jede Konzeptinklusion **„eliminiert“ unerwünschte Interpretationen**.

TBoxen — Modellierung

In der Praxis bestehen TBoxen zu einem **großen Teil** aus

- **Konzeptinklusionen** $A \sqsubseteq C$, wobei A ein Konzeptname ist

Intuitiv ist C **notwendige Bedingung** dafür, eine Instanz von A zu sein.

z. B. in SNOMED:

$$\text{Perikardium} \sqsubseteq \text{Gewebe} \sqcap \exists \text{teilVon.Herz}$$

- **Konzeptdefinitionen** $A \equiv C$, wobei A ein Konzeptname ist

Intuitiv ist C **notwendige und hinreichende Bedingung** dafür, eine Instanz von A zu sein

z. B. in SNOMED:

$$\text{Perikarditis} \equiv \text{Entzündung} \sqcap \exists \text{ort.Perikardium}$$

Hinreichende Bedingungen sind für viele Konzepte **schwer** zu finden.

TBoxen — Modellierung

Modellierungsmuster, die **nicht** in dieses Schema passen z. B.:

- **Disjunktheitsconstraints** $C \sqcap D \sqsubseteq \perp$

Kein Objekt kann sowohl zu Konzept C als auch zu Konzept D gehören.

z. B. in SNOMED:

Befund \sqcap Körperstruktur $\sqsubseteq \perp$

- **Komplexe Zusammenhänge** zwischen mehreren Konzepten

z. B.:

Professor $\sqcap \exists \text{hat. Lehrdeputat} \sqsubseteq \exists \text{hält. Vorlesung}$

Erfüllbarkeit, Subsumtion, Äquivalenz

Wir erweitern unsere grundlegenden Begriffe auf TBoxen:

Definition 2.6 (erfüllbar, subsumiert, äquivalent bezügl. einer TBox)

Seien C, D ALC-Konzepte und \mathcal{T} TBox. Dann

- ist C **erfüllbar bzgl. \mathcal{T}** gdw. \mathcal{T} Modell \mathcal{I} hat mit $C^{\mathcal{I}} \neq \emptyset$;
- wird C **von D subsumiert bzgl. \mathcal{T}** gdw. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in allen Modellen \mathcal{I} von \mathcal{T} (Notation $\mathcal{T} \models C \sqsubseteq D$);
- sind C und D **äquivalent bzgl. \mathcal{T}** gdw. $C^{\mathcal{I}} = D^{\mathcal{I}}$ in allen Modellen \mathcal{I} von \mathcal{T} (Notation $\mathcal{T} \models C \equiv D$).

T 2.6

Kleine Übung für Zwischendurch

Aufgabe

① $\mathcal{T} = \{A \sqsubseteq \exists r.B \sqcap \exists s.B, \exists r.B \sqsubseteq \forall r.A\}$

Ist das Konzept A erfüllbar bzgl. der TBox \mathcal{T} ?

② $\mathcal{T} = \{\exists r.T \sqsubseteq A_1, \forall r.B \sqsubseteq A_2\}$

Gilt $\mathcal{T} \models T \sqsubseteq A_1 \sqcup A_2$?

Hinweise:

- Erfüllbarkeit zeigen: Modell angeben
- Unerfüllbarkeit zeigen: semantisch argumentieren
- Subsumtion zeigen: semantisch argumentieren
- Nicht-Subsumtion zeigen: Gegenmodell angeben

Monotonie

Das Erweitern einer TBox um zusätzliche Konzeptinklusionen wirkt sich wie folgt auf Erfüllbarkeit und Subsumption aus:

Lemma 2.7

Seien \mathcal{T}_1 und \mathcal{T}_2 TBoxen mit $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$. Dann gilt:

- 1 Wenn C erfüllbar bzgl. \mathcal{T}_2 , dann ist C erfüllbar bzgl. \mathcal{T}_1 .
- 2 Wenn $\mathcal{T}_1 \models C \sqsubseteq D$, dann $\mathcal{T}_2 \models C \sqsubseteq D$.

Die umgekehrten Aussagen sind im Allgemeinen nicht richtig. **T 2.7**

Eine Logik, die diese Eigenschaften erfüllt, nennt man **monoton**.

(Das Hinzunehmen von Formeln kann nur zu *zusätzlichen* Konsequenzen führen, aber nicht dazu, dass Konsequenzen ungültig werden.)

Kapitel 2: Grundlagen

- 1 Die Beschreibungslogik *ALC*
- 2 TBoxen
- 3 Schlussfolgerungsprobleme**
- 4 Erweiterungen von *ALC*

Schlussfolgerungsprobleme

Zur Erinnerung: Schlussfolgern ...

- ist der wichtigste Bestandteil eines intelligenten Systems
- dient dazu, aus explizit gegebenem Wissen neues Wissen abzuleiten, das vorher nur implizit vorhanden war

Wichtigstes Designkriterium für Beschreibungslogiken:

So viel Ausdruckstärke wie nötig, aber nicht mehr, um möglichst effizientes Schlussfolgern zu erlauben.

Schlussfolgerungsprobleme

Erfüllbarkeitsproblem:

Gegeben C und \mathcal{T} , entscheide ob C erfüllbar bzgl. \mathcal{T} .

Subsumtionsproblem:

Gegeben C, D und \mathcal{T} , entscheide ob $\mathcal{T} \models C \sqsubseteq D$.

Äquivalenzproblem:

Gegeben C, D und \mathcal{T} , entscheide ob $\mathcal{T} \models C \equiv D$.

Diese Entscheidungsprobleme kann man auch mit leerer TBox $\mathcal{T} = \emptyset$ betrachten.

Motivation

Anwendung der Schlussfolgerungsprobleme:

- Modellierungsfehler finden:
Unerfüllbare Konzepte sind im Allgemeinen **unerwünscht**.
- die Struktur der TBox explizit machen, z. B. für Browsing:
Subsumtion (= Is-A-Relation) ist Haupt-**Strukturierungsmittel** für TBoxen.
 $C \sqsubseteq D$ kann gelesen werden als „ D ist genereller als C “.
- Redundanzen finden:
Zwei **äquivalente Konzepte** sind **unter Umständen unerwünscht**.

Subsumtion als Ordnungsrelation

Lemma 2.8

Für jede TBox \mathcal{T} ist die Relation „ \sqsubseteq bzgl. \mathcal{T} “

- reflexiv ($\mathcal{T} \models C \sqsubseteq C$) und
- transitiv ($\mathcal{T} \models C \sqsubseteq D$ und $\mathcal{T} \models D \sqsubseteq E$ impliziert $\mathcal{T} \models C \sqsubseteq E$).

Bis auf fehlende Antisymmetrie ist \sqsubseteq also partielle Ordnung.

Man kann \sqsubseteq als **Hasse-Diagramm** darstellen, dessen Knoten mit Mengen von Konzepten beschriftet sind.

Normalerweise Einschränkung auf die in \mathcal{T} verwendeten Konzeptnamen

T 2.8

Klassifikation

Ein weiteres Schlussfolgerungsproblem:

Klassifikation:

Gegeben \mathcal{T} , berechne das Hasse-Diagramm für \sqsubseteq bzgl. \mathcal{T} , eingeschränkt auf Konzeptnamen in \mathcal{T} .

Ist ein **Berechnungsproblem**, kein Entscheidungsproblem.

In der Praxis:

- berechenbar durch n^2 Subsumtionsberechnungen ($n = \text{Anzahl der Konzeptnamen in } \mathcal{T}$)
- zahlreiche Optimierungen verfügbar

Ontologie-Editor Protégé

koala (http://protege.stanford.edu/plugins/owl/owl-library/koala.owl) : [/Users/schneider/Ontologies/koala/koala.owl]

< > koala (http://protege.stanford.edu/plugins/owl/owl-library/koala.owl) Search...

Data Properties x Annotation Properties x Individuals by class x OWLViz x DL Query x OntoGraf x SPARQL Query x
 Active Ontology x Entities x Classes x Object Properties x

Class hierarchy: Koala **Class hierarchy (inferred)** Class Annotations Class Usage

Annotations: Koala

Annotations +

Description: Koala

Equivalent To +

SubClass Of +

- hasHabitat **some** DryEucalyptForest
- isHardWorking **value** false
- Marsupials

General class axioms +

- owl:Thing
 - Animal
 - Marsupials
 - Koala**
 - KoalaWithPhD
 - Quokka
 - TasmanianDevil
 - Parent
 - Person
 - Degree
 - Female
 - Gender
 - Habitat
 - Male

Asserted

owl:Thing

Annotations: Koala

Annotations +

Description: Koala

Equivalent To +

SubClass Of +

- hasHabitat **some** DryEucalyptForest
- isHardWorking **value** false
- Marsupials

General class axioms +

Klassifikation in Protégé

koala (http://protege.stanford.edu/plugins/owl/owl-library/koala.owl) : [/Users/schneider/Ontologies/koala/koala.owl]

< > koala (http://protege.stanford.edu/plugins/owl/owl-library/koala.owl) Search...

Data Properties x Annotation Properties x Individuals by class x OWLViz x DL Query x OntoGraf x SPARQL Query x

Active Ontology x Entities x Classes x Object Properties x

Class hierarchy: Female Class hierarchy (inferred): F

Class Annotations Class Usage

Annotations: Female

Annotations +

Description: Female

Equivalent To +

● hasGender value female

SubClass Of +

● Animal

General class axioms +

SubClass Of (Anonymous Ancestor)

Annotation property hierarchy Datatypes

Reduktionen

Die 3 Schlussfolgerungsprob. sind wechselseitig polynomiell reduzierbar:

Lemma 2.9

① *Subsumtion ist polynomiell reduzierbar auf (Un)erfüllbarkeit:*

$$\mathcal{T} \models C \sqsubseteq D \quad \text{gdw.} \quad C \sqcap \neg D \text{ unerfüllbar bzgl. } \mathcal{T}$$

② *Erfüllbarkeit polynomiell reduzierbar auf (Nicht-)Äquivalenz:*

$$C \text{ erfüllbar bzgl. } \mathcal{T} \quad \text{gdw.} \quad \mathcal{T} \not\models C \equiv \perp$$

③ *Äquivalenz ist polynomiell reduzierbar auf Subsumtion:*

$$\mathcal{T} \models C \equiv D \quad \text{gdw.} \quad \mathcal{T} \models \top \sqsubseteq (C \sqcap D) \sqcup (\neg C \sqcap \neg D)$$

T 2.9

Algorithmus für eines der Probleme kann also auch für die beiden anderen verwendet werden;

alle drei Probleme haben dieselbe Komplexität (in *ALC*)

Im Folgenden konzentrieren wir uns meist auf Erfüllbarkeit.

Kapitel 2: Grundlagen

- 1 Die Beschreibungslogik ALC
- 2 TBoxen
- 3 Schlussfolgerungsprobleme
- 4 Erweiterungen von ALC**

Erweiterungen von ALC

Wir betrachten exemplarisch **zwei Erweiterungen von ALC** :

- $ALCI$: ALC mit inversen Rollen (Rollenkonstruktor)
- $ALCQ$: ALC mit Zahlenrestriktionen (Konzeptkonstruktor)

Beide Erweiterungen sind in OWL realisiert.

Namenschema:

- ein Buchstabe pro Erweiterung; üblicherweise steht
 - \mathcal{I} für **i**nverse Rollen
 - \mathcal{Q} für **q**uantifizierte Zahlenrestriktionen
 - \mathcal{N} für unquantifizierte Zahlenrestriktionen (**n**umber restrictions)
 \rightsquigarrow Spezialfall von \mathcal{Q}
 - \mathcal{F} für **F**unktionalität \rightsquigarrow Spezialfall von \mathcal{N}
 - \vdots
- kann kombiniert werden, z. B. $ALCQI$

\mathcal{I} : Inverse Rollen

Häufig möchte man über Rollen „in beiden Richtungen“ reden:

- SNOMED: hatTeil / istTeilVon (z. B. in der Anatomie)
- Universitätsbeispiel: hört / wirdGehörtVon ,
gibt / wirdGegebenVon

Verwendet man diese Rollen einfach als Namen in \mathcal{ALC} ,
gibt es unintuitive Konsequenzen.

T 2.10

\mathcal{I} : Inverse Rollen

Definition 2.10 (Inverse Rollen)

Für jeden Rollennamen r ist r^- die **inverse Rolle** zu r .

Wir definieren:

$$(r^-)^{\mathcal{I}} = \{(e, d) \mid (d, e) \in r^{\mathcal{I}}\}$$

ALCI: *ALC* erweitert um die Möglichkeit, inverse Rollen in Existenz- und Wertrestriktionen zu benutzen.

T 2.10 Forts.

Q: Zahlenrestriktionen

Häufig möchte man Rollennachfolger „zählen“ können:

- SNOMED: Eine Hand ist ein Organ mit **genau 5** Teilen, die ein Finger sind.
- Universitätsbeispiel: in jedem Semester werden **mindestens 2** Wahlpflichtmodule angeboten

Wir werden sehen:

In *ALC* ist Zählen nicht ohne Weiteres möglich!

Q: Zahlenrestriktionen

Definition 2.11 (Zahlenrestriktion)

Für jede natürliche Zahl n , jeden Rollennamen r , jedes Konzept C :

- $\leq n r.C$ (Höchstens-Restriktion)
- $\geq n r.C$ (Mindestens-Restriktion)

Die Semantik ist:

$$\leq n r.C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \leq n\}$$

$$\geq n r.C^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \#\{e \mid (d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\} \geq n\}$$

Beachte:

T 2.11

- $\exists r.C$ ist äquivalent zu $\geq 1 r.C$
- $\forall r.C$ ist äquivalent zu $\leq 0 r.\neg C$

ALCQ: ALC erweitert um Zahlenrestriktionen

Weitere Erweiterungen

Es gibt noch viel mehr Erweiterungen:

- spezielle Rolleninterpretationen (transitiv, symm., reflexiv etc.)
- Konzepte, die nur eine einzige Instanz haben können \mathcal{O}
(Nominale)
- Inklusionen zwischen Rollen oder Rollenketten \mathcal{H}, \mathcal{R}
- Numerische Werte (\mathcal{D})
- Fixpunktoperatoren
- temporale Operatoren
- \vdots

Viele davon sind in OWL realisiert.

Literatur für dieses Kapitel (Basis)



F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi und P. F. Patel-Schneider (Hrsg.).

The Description Logic Handbook.

2. Auflage, Cambridge University Press, 2007.

Kapitel 2: Basic Description Logics.

In SUUB zur Ausleihe und elektronisch verfügbar.



C. Lutz.

Theoretische Informatik 1 + 2.

Vorlesungsskript, Uni Bremen, SoSe 2016.

Kapitel 15 (v. a. Def. 15.9), 18, 19 (nur Def. 19.1), 20.

<http://tinyurl.com/ss16-theoinf>



U. Schöning.

Theoretische Informatik – kurzgefasst.

Spektrum Akademischer Verlag, 2008.

In SUUB mehrfach vorhanden.

Literatur für dieses Kapitel (weiterführend)



M. Schmidt-Schauß und G. Smolka.

Attributive concept descriptions with complements.

Artificial Intelligence, 48(1):1–26, 1991.

Links für dieses Kapitel



Stanford Center for Biomedical Informatics Research.
Protégé Ontology Editor.

<http://protege.stanford.edu/>



M. Horridge et al.
Protégé-OWL Tutorial.

<http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/>



E. Zolin.
Description Logic Complexity Navigator.

<http://www.cs.man.ac.uk/~ezolin/dl/>