SoSe 2016

Beschreibungslogik

Mitschriften / Zusammenfassung

Dozent: Prof. Dr. Thomas Schneider
Autoren: Sascha Jongebloed, Robin Nolte

5. Februar 2017

Inhaltsverzeichnis

1 Einleitung

1.1 Wissensrepräsentation

1.1.1 Definition

Entwicklung von Formalismen, mittels derer Wissen über die Welt in abstrakter Weise beschrieben werden kann und die effektiv verwendet werden können, um intelligente Anwendungen zu realisieren.

1.1.2 Wohldefinierte Syntax und Semantik

- Syntax: die Sprache, in der Wissen repräsentiert wird und hier stets symbolisch und logikbasiert
- Semantik: fixiert die Bedeutung des repräsentierten Wissens in exakter, eindeutiger Weise
  - Deklarative Semantik ist unabhängig von verarbeitender Software

1.1.3 In Beschreibungslogik

- Beschränkung auf konzeptuelles Wissen -> Abstraktion
- Schlussfolgern (explizit nach implizit) ist Mehrwert gegenüber Datenbanken
  - Entscheidbarkeit und geringe Komplexität erwünscht
- Beschreibungslogiken sind Logikfamilie

2 Grundlagen

2.1 Attributive Language with Complement (ALC)

2.1.1 Konzept- und Rollennamen

Konzept- und Rollennamen sind abzählbar unendliche und disjunkte (durch Groß- und Kleinschreibung) Mengen.

2.1.2 $\mathcal{ALC}$: Syntax

Definition 2.1. $\mathcal{ALC}$-Konzepte
Die Menge der $\mathcal{ALC}$-Konzepte is induktiv definiert:

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

  $\mathcal{ALC}$-Konzepte

T2.1 Beispiel

Hier einige Beispiel für diese Syntax:

$$Student \sqcap \exists studiert.Naturwissenschaft$$
$$Professor \sqcap Emeritus \sqcap \forall haelt.\neg PlichtVL$$
$$VL \sqcap \neg PlichtVL \sqcap \forall hatUebungsaufgabe(Einfach \sqcup Interessant)$$
$$A \sqcap \exists r.(\neg B \sqcup \forall r.A)$$

Weiteres zur Syntax

Dabei verwenden wir folgende Symbole:

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

Zudem benutzen zudem folgende Abkürzungen: wir schreiben

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

Präzedenzregel

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

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

Desweiteren is keine Präzedenz zwischen $\sqcap$ und $\sqcup$ definiert worden: Daher müssen Klammern verwendet werden!

2.1.3 $\mathcal{ALC}$: Semantik

Definition 2.2. $\mathcal{ALC}$ 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 $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

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

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} |$ es gibt $e \in \Delta^{\mathcal{I}}$ mit $(d,e) \in r^{\mathcal{I}}$ und $e \in C^{\mathcal{I}}\}$
- $(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} |$ für alle $e \in \Delta^{\mathcal{I}}$, $(d,e) \in r^{\mathcal{I}}$ impliziert $e \in C^{\mathcal{I}}\}$

Verwendete Simbole:

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

Für Interpretationen verwenden wir übliche Terminologie für Graphen:

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

2.1.4 Extension/Modell

Wir nennen

- C^I ist *Extension* des Konzeptes oder der Rolle C
- jedes $d \in C^I$ ist eine Instanz des Konzeptes C
- r-Nachfolger, r-Vorgänger

Beachte, das $\top^{\mathcal{I}}$ für jede Interpretation $\mathcal{I}$ identisch mit $\Delta^{\mathcal{I}}$ ist. Intuitivt entspricht $\top$ die Menge alle Elemente.

$\bot^{\mathcal{I}}$ ist für jede Interpretation $\mathcal{I}$ leer, repräsentiert also intuitiv, dass etwas unmöglich ist. Z.B.:

Mensch $\sqcap \forall$hatKind.$\bot$ Menschen, die keine Kinder haben

2.1.5 Erfüllbarkeit,Subsumtion,Äquivalenz

Definition 2.3. Erfüllbar, susumiert, äquivalent
Seien C und D ALC-Konzepte. Dann

- ist C *erfüllbar*, wenn es eine Interpretation I gibt mit $C^{\mathcal{I}} \neq \varnothing$. $\mathcal{I}$ ist 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 $C \equiv D$)

Es gelten die üblichen aussagenlogischen Äquivalenten, wie z.B. de Morgan.

2.2 TBoxen

TBoxen definieren Konzepte und setzten diese zueinander in Beziehung.

2.2.1 TBox-Syntax

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 $C \sqsubseteq D$, $D \sqsubseteq C$.

Wir lesen $C \equiv D$ als "C impliziert D".

2.2.2 TBox-Sematik

Definition 2.5. TBox-Semantik
Eine Interpretation I

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

Intuitiv entsprechen Interpretation mögliche Welten, TBoxen hingegen schließen Welten aus, die wir für nicht möglich halten.

2.2.3 Modellierung

In der Praxis bestehen TBoxen zu einem großen Teil aus:

- Konzeptinklusion $A \sqsubseteq C$, mit A ein Konzeptname: C ist notwendige Bedingung dafür, eine Instanz von A zu sein
- Konzeptdefinition $A \equiv C$, mit A ein Konzeptname: C ist notwendige und hinreichende Bedingung dafür, eine Instanz von A zu sein.
- Disjunktheitsconstraints $C \sqcap D \sqsubseteq \bot$: Kein Objekt kann gleichzeitig zu C und D gehören
- Komplexe Zusammenhänge zwischen mehreren Konzepten. Zum Beispiel:

$$Professor \sqcap \exists hat.Lehrdeputat \sqsubseteq \exists haelt.Vorlesung$$

2.2.4 Erfüllbarkeit, Subsumtion, Äquivalenz

Definition 2.6. Erfüllbar, subsumiert, äquivalent bezüglich einer TBOX
Seine C, D, $\mathcal{ALC}$-Konzepte und $\mathcal{T}$ TBox. Dann

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

Intuitiv gesprochen ist diese Definition wie in 2.1.5, nur ist I jeweils Modell von einer TBox T.

- Erfüllbarkeit zeigen: Modell angeben
- Unerfüllbarkeit / Subsumption zeigen: semantisch Argumentieren
- Nicht-Subsumtion zeigen: Gegenmodell angeben

2.2.5 Monotonie

Lemma 2.7. Seien $\mathcal{T}_1$ und $\mathcal{T}_2$ TBoxen mit $\mathcal{T}_1 \subseteq \mathcal{T}_2$. Dann gilt:

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

T2.8. Beweisskizze.

1. *Beweis.* Sei C erfüllbar bzgl. $\mathcal{T}_2$. Dann gibt es Modell $\mathcal{I}$ von $\mathcal{T}_2$ mit $C^{\mathcal{I}} \neq \varnothing$. Also erfüllt $\mathcal{I}$ alle Konzeptinklusionen in $\mathcal{T}_2$ und wegen $\mathcal{T}_1 \subseteq \mathcal{T}_2$ auch alle Konzeptinklusionen in $\mathcal{T}_1$. Also ist $\mathcal{I}$ Modell von $\mathcal{T}_1$ mit $C^{\mathcal{I}} \neq \varnothing$. Also ist C erfüllbar bzgl. $\mathcal{T}_1$. □

2. *Beweis.* Kontraposition: Wenn $\mathcal{T}_2 \not\models C \sqsubseteq D$ dann $\mathcal{T}_1 \not\models C \sqsubseteq D$.
   Es gelte $\mathcal{T}_2 \not\models C \sqsubseteq D$. D.h. es gibt Modell $\mathcal{I}$ von $\mathcal{T}_2$ mit $C^{\mathcal{I}} \not\subseteq D^{\mathcal{I}}$. Wie im 1. Falls ist $\mathcal{I}$ auch Modell von $\mathcal{T}_1 \subseteq \mathcal{T}_2$. Also $\mathcal{T}_1 \not\models C \sqsubseteq D$ □

Die umgekehrten Aussagen sind im Allgemeinen nicht richtig. Beispiel:

$\mathcal{T}_1 = \varnothing$, $\mathcal{T}_2 = \{A \sqsubseteq B\}$

$\mathcal{T}_1 \subseteq \mathcal{T}_2$
$\mathcal{T}_2 \models A \sqsubseteq B$
$\mathcal{T}_1 \models A \sqsubseteq B$

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)

2.3 Schlussfolgerungsprobleme

Schlussfolgern dient dazu aus explizit gegebenes Wissen neues Wissen abzuleiten, dass vorher nur implizit vorhanden war. Beschreibungslogiken sind so designt, dass sie so viel Ausdruckstärke wie nötig, aber nicht mehr, besitzen, um möglichst effizientes Schlussfolgern zu Erlauben.

Die von uns betrachteten Schlussfolgerungsprobleme sind:

- **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 können auch mit leerer TBox $\mathcal{T} = \varnothing$ betrachtet werden.

Die Schlussfolgerungsprobleme werden dazu genutzt Modellierungsfehler in Ontologien zu finden, die Struktur der TBox explizit zu machen und um Redundanzen zu finden.

2.3.1 Subsumtion als Ordnungsrelation

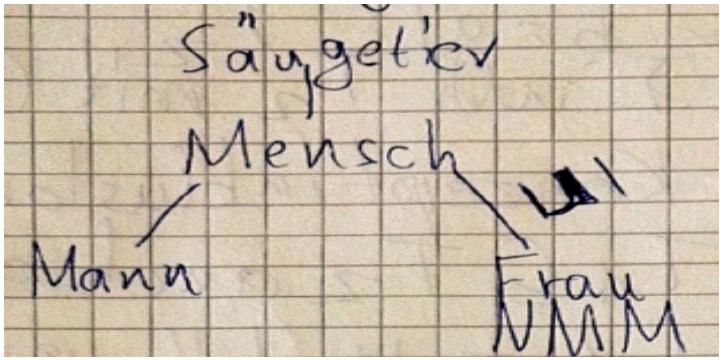
Lemma 2.8. Für jede TBox $\mathcal{T}$ ist die Relation "$\sqsubseteq$ bzgl. $\mathcal{T}$"

- 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 die fehlende Antisymmetrie () ist $\sqsubseteq$ also partielle Ordnung.

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

2.9



2.3.2 Klassifikation

Ein weiteres Schlussforlgerungsproblem:

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

Dies ist ein Berechnungsproblem (kein Entscheidungsproblem), das in der Praxis durch n^2 Subsumtionsberechnungen berechenbar ist und für das zahlreiche Optimierungen verfügbar sind.

2.3.3 Reduktion

Die Entscheidungsprobleme sind wechselseitig polynomiell aufeinander reduzierbar:

1. *Erfüllbarkeit* auf *Nicht-Äquivalenz*
   C erfüllbar bzgl. T gdw. $T \not\models C \equiv \bot$
2. *Subsumtion* auf *Unerfüllbarkeit*
   $T \models C \sqsubseteq D$ gdw. $C \sqcap \neg D$ unerfüllbar bzgl. T
3. *Äquivalenz* auf *Subsumtion*
   $T \models C \equiv D$ *gdw.* $T \models \top \sqsubseteq (C \sqcap D) \sqcup (\neg C \sqcap \neg D)$

Dies heißt für uns, dass ein Algorithmus für eines der Probleme auch für die anderen beiden verwendet werden können. Alle drei Probleme haben dieselbe Komplexität (in $\mathcal{ALC}$). Daher werden wir uns im Folgenden hauptsächlich auch Erfüllbarkeit konzentrieren.

2.4 Erweiterungen von ALC

2.4.1 Inverse Rollen (ALCI)

Definition: Für jeden Rollennamen r ist r^- die *inverse Rolle* zu r. Wir definieren $(r^-)^I = \{(e,d) \mid (d,e) \in r^I\}$

2.4.2 Zahlenrestriktion (ALCQ)

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

- Höchstens-Restriktion: $(\leq n\ r\ C)^I = \left\{d \in \Delta^I \mid \#\left\{e \mid\ (d,e) \in r^I \wedge e \in C^I\right\} \leq n\right\}$
- Mindestens-Restriktion $(\leq n\ r\ C) = \left\{d \in \Delta^I \mid \#\left\{e \mid\ (d,e) \in r^I \wedge e \in C^I\right\} \geq n\right\}$

3 Ausdrucksstärke und Modellkonstruktionen

Die wichtigsten Eigenschaften einer Beschreibungslogik sind *Ausdrucksstärke* und *Komplexität*. Ausdruckstärke kann man nicht linear quantifizieren, sondern nur beschreiben und charakterisieren.

3.1 Bisimulation

3.1.1 Definition

Seien I_1 und I_2 Interpretationen. Relation $\rho \subseteq \Delta^{I_1} \times \Delta^{I_2}$ ist Bisimulation zwischen I_1 und I_2, wenn gilt:

1. Wenn $d_1\ \rho\ \mathrm{d}_2$, dann gilt für alle Konzeptnamen A: $d_1 \in A^{I_1}$ gdw. $d_2 \in A^{I_2}$.
2. Wenn $d_1\ \rho\ \mathrm{d}_2$ und $\left(d_1, d_1^{'}\right) \in r^{I_1}$ für beliebigen Rollennamen r, dann gibt es ein $d_2^{'} \in \Delta^{I_2}$ mit $d'_1\ \rho\ d'_2$ und $\left(d_2, d_2^{'}\right) \in r^{I_2}$.
3. Wenn $d_1\ \rho\ \mathrm{d}_2$ und $\left(d_2, d_2^{'}\right) \in r^{I_2}$ für beliebigen Rollennamen r, dann gibt es ein $d_1^{'} \in \Delta^{I_1}$ mit $d'_1\ \rho\ d'_2$ und $\left(d_1, d_1^{'}\right) \in r^{I_1}$.

Seien I_1 und I_2 Interpretationen, $d_1 \in \Delta^{I_1}$, $d_2 \in \Delta^{I_2}$:

$(I_1, d_1) \sim (I_2, d_2)$: Es gibt Bisimulation ρ zwischen I_1 und I_2 mit $d_1\ \rho\ \mathrm{d}_2$. Die leere Relation ist immer Bisimulation.

3.1.2 Theorem 3.2

Seien I_1, I_2 Interpretationen, $d_1 \in \Delta^{I_1}$ und $d_2 \in \Delta^{I_2}$. Wenn $(I_1, d_1) \sim (I_2, d_2)$, dann gilt für alle ALC-Konzepte C:

$d_1 \in C^{I_1}$ gdw. $d_2 \in C^{I_2}$

Beweisskizze per Induktion über die Struktur von C. Sei ρ eine Bisimulation zwischen I_1 und I_2 mit $d_1\ \rho\ \mathrm{d}_2$.

I.A. $C = A$ ist Konzeptname. Nach Bedingung 1. der Bisimulation gilt $d_1 \in A^{I_1}$ gdw. $d_2 \in A^{I_2}$.

I.S. Unterscheide Fälle gemäß des äußersten Konstruktes von C. Es genügen ¬,⊓, ∃r.C:

1. $C = \neg D$

   $d_1 \in C^{I_1}$ gdw. $d_1 \notin D^{I_1}$ (Semantik) gdw. $d_2 \notin D^{I_2}$ (I.V.) gdw. $d_2 \in C^{I_2}$ (Semantik)

1. $C = D_1 \sqcap D_2$

   $d_1 \in C^{I_1}$ gdw. $d_1 \in D_1^{I_1}$und $d_1 \in D_2^{I_1}$ (Semantik) gdw. $d_2 \in D_1^{I_2}$ und $d_2 \in D_2^{I_2}$ (I.V.) gdw. $d_2 \in C^{I_2}$ (Semantik)

1. $C = \exists r.D$

   Hinrichtung und Rückrichtung analog über Semantik, 2. Bedingung der Bisimulation, I.V., Semantik.

3.2 Ausdrucksstärke

3.2.1 Definition

Eine *Eigenschaft* E ist eine Menge von Paaren (I, d), wobei I eine Interpretation und $d \in \Delta^I$ ein Element in I ist. E ist *ausdrückbar in ALC*, wenn es ein ALC-Konzept C gibt, so dass für alle I und $d \in \Delta^I$ gilt: $(I,\ d) \in E$ gdw. $d \in C^I$.

3.2.2 Theorem 3.4

Die zusätzlichen Eigenschaften von ALCI und ALCQ sind in ALC nicht ausdrückbar.

Beweisskizze. Finde Bisimulation für die dies nicht gilt.

3.2.3 Theorem 3.5

Sei E eine Eigenschaft. Wenn es Interpretation I_1, I_2 und Elemente $d_1 \in \Delta^{I_1}$ und $d_2 \in \Delta^{I_2}$ gibt, so dass

- $(I_1, d_1) \in E$ und $(I_2, d_2) \in E$ sowie
- $(I_1, d_1) \sim (I_2, d_2)$

dann ist E nicht in ALC ausdrückbar.

3.2.4 Theorem 3.6

Wenn ein ALC-Konzept C bzgl. einer ALC-TBox T erfüllbar ist, dann haben C und T ein gemeinsames Baummodell I. I Baum, Wurzel in C^I.

3.2.5 Unravelling

Sei I eine Interpretation und $d \in \Delta^I$. d-Pfad in I: Sequenz $d_0 d_1 \dots d_{n-1}$, $n > 0$ mit

- $d_0 = d$
- für alle $i < n$: es gibt Rollenname r mit $(d_i, d_{i+1}) \in r^I$.

Wir setzen $\text{end}\,(d_0 \dots d_{n-1}) = d_{n-1}$.

Definition: Unravelling von I an Stelle d ist folgende Interpretation J:

- Δ^J = Menge aller d-Pfade in I
- $A^J = \left\{ p \in \Delta^J \;\middle|\; \text{end}\,(p) \in A^I \right\}$
- $r^J = \left\{ (p, p') \in \Delta^J \times \Delta^J \mid \exists e : p' = p \cdot e \text{ und } (\text{end}\,(p), e) \in r^I \right\}$

Erklärung: Erzeuge Knoten, die den Folgen entsprechen, füge sie den Konzepten hinzu, die als letztes Element in der Folge vorkommen und erzeuge Kanten die den weiterführenden Rollen entsprechen.

3.2.6 Lemma 3.8

Sei J Unravelling von I an Stelle d. Für alle ALC-Konzepte C und alle $p \in \Delta^J$ gilt: $\text{end}(p) \in C^I$ gdw. $p \in C^J$.

Beweisskizze. Zeige Bisimulation $(I, end(p)) \sim (J, p)$, z.B. $\rho = \{(\text{end}(p), p) \in \Delta^I \times \Delta^J \mid \text{p ist d}-\text{Pfad}\}$ alle Knoten in J auf ihr Ende ab).

1. gilt per Definition von J.
2. Angenommen e,ρ, p und $(e, e') \in r^I$. Dann $e = end(p)$ per Konstruktion von J. Wegen $(e, e') \in r^I$ ist pe' Pfad. Nach Konstruktion von J gilt $(p, pe') \in r^J$.

3.2.7 Theorem 3.6

Wenn ein ALC-Konzept C bzgl. einer TBox T erfüllbar ist, dann haben C und T ein gemeinsames Baummodell I.

Beweisskizze. Wurzel von Unravelling J von I an der Stelle $d \in C^I$ nach Lemma 3.8 in C^J. Zu zeigen: J ist Modell von T. Sei $D \sqsubseteq E$ in T und $p \in D^J$. Nach Lemma 3.8 ist dann $\text{end}(p) \in D^I$ und weil I Modell von T folgt $\text{end}(p) \in E^I$. Mit Lemma 3.8 folgt $p \in E^J \Rightarrow J \vDash D \sqsubseteq E$.

3.2.8 Bisimulation versus Ausdrucksstärke

Bisimulation entspricht nicht der Ausdrucksstärke von ALC.

3.2.9 Bisimulation in ALCI

Füge 2 Regeln hinzu, sodass Vorgänger auch simuliert sein müssen.

3.3 Ausdrucksstärke und Modellkonstruktion

3.3.1 Größe von Konzepten und TBoxen

Größe $|C|$ eines ALC-Konzeptes C ist induktiv definiert:

- $|A| = 1$
- $|\neg C| = |C| + 1$
- $|C \sqcap D| = |C \sqcup D| = |C| + |D| + 1$
- $|\exists r.C| = |\forall r.C| = |C| + 3$

Größe $|C|$ einer TBox T ist

- $\sum_{C \sqsubseteq D \in T} |C| + |D| + 1$

3.3.2 Teilkonzepte

- $\text{sub}(C)$ ist Menge der Teilkonzepte von C (einschließlich C)
- $\text{sub}(T) \bigcup_{C \sqsubseteq D \in T} \text{sub}(C) \cup sub(D)$
- $\text{sub}(C,T)\, sub(C) \cup sub(D)$

3.3.3 Lemma 3.13

$|\text{sub}(C,T)| \leq |C| + |T|$

3.3.4 Typ

Sei I eine Interpretation, $d \in \Delta^I$. Der *Typ* $t_I(d)$ *von* d *in* I ist $t_I(d) = \{D \in sub(C,T) \mid d \in D^I\}$.

Erklärung: Alle Konzepte in T und C, die ein Objekt d erfüllt.

3.3.5 Lemma 3.15

Für jede Interpretation I gilt: $\#\left\{t_I(d) \mid d \in \Delta^I\right\} \leq 2^{|C|+|T|}$

3.3.6 Filtration

Sei I Interpretation. Definiere Äquivalenzrelation $\sim$ auf Δ^I: $d \sim e$ gdw. $t_I(d) = t_I(e)$. Wir bezeichnen diese Äquivalenzklasse von $d \in \Delta^I$ bzgl. $\sim$ mit $[d]$. Die Filtration von I bzgl. C und T ist folgende Interpretation J:

- $\Delta^J = \left\{[d] \mid d \in \Delta^I\right\}$
- $A^J = \left\{[d] \mid d \in A^I\right\}$ für alle $A \in sub(C,T)$
- $r^J = \left\{([d],[e]) \mid \exists d' \in [d],\ e' \in [e] : \left(d', e'\right) \in r^I\right\}$ für alle Rollennamen r

Theorem 3.17 Wenn I Modell von C und T, so auch J, bzw. für alle $d \in \Delta^I$ und $D \in sub(C,T)$ gilt: $d \in D^I$ gdw. $[d] \in D^J$

Beweisskizze per Induktion über die Struktur von D.

I.A. $C = A$ folgt aus Definition A^J.

I.S.

1. $\neg$, $\sqcap$ einfach mittels Semantik und I.A.
2. $D = \exists r.E$

    a. Hinrichtung

    $d \in (\exists r.E)^I \Leftrightarrow$ (Semantik) es gibt $e \in \Delta^I$ mit $(d,e) \in r^I$ und $e \in E^I \Rightarrow$ (Definition r^J und I.V.) es gibt $e \in \Delta^I$ mit $([d],[e]) \in r^J$ und $[e] \in E^J \Leftrightarrow$ (Semantik $\exists$) $[d] \in (\exists r.E)^J$

a. Rückrichtung

$[d] \in (\exists r.E)^J \Leftrightarrow$ (Semantik $\exists$) es gibt $e \in \Delta^J$ mit $([d], [e]) \in r^J$ und $[e] \in E^J \Leftrightarrow$ (Definition r^J und I.V.) es gibt $e \in \Delta^J$, es gibt $d' \in [d]$, $e' \in [e]$, $(d', e') \in r^I$ und $e' \in E^I \Rightarrow$ (Semantik $\exists$) $d' \in (\exists r.E)^I \Rightarrow (d \sim d')\ d \in (\exists r.E)^I$

3.3.7 Theorem 3.11 – Beschränkte Modelleigenschaft

Wenn ein Konzept C bzgl. einer TBox T erfüllbar ist, dann haben C und T ein gemeinsames Modell der *Kardinalität* $\leq 2^{|C|+|T|}$.

Beweis. Folgt aus Theorem 3.17 und Lemma 3.15.

3.3.8 Theorem 3.18

ALCQI hat nicht die endliche Modelleigenschaft.

Beweis: A hat nur unendliche Modelle bzgl. folgender TBox:

- $\top \sqsubseteq \exists r.\neg A$
- $T \sqsubseteq (\leq 1\ r^-\ \top)$

3.3.9 Theorem 3.11 – Erfüllbarkeit

Wenn C erfüllbar bzgl. T, dann haben C und T Modell der Größe $\leq 2^{|C|+|T|}$. Erfüllbarkeit ist also entscheidbar (Erzeuge alle Interpretationen mit $|\Delta|^I \leq 2^n$ und prüfe, ob ein Modell darunter ist).

3.3.10 Lemma 3.19

Gegeben sei ein Konzept C und endliche Interpretation I. Man kann in polynomieller Zeit – genauer in Zeit $O\left(|C| \cdot \left|\Delta^I\right|\right)$ – die Extension C^I berechnen.

Beweis: Rekursiver Algorithmus über die Definition der Konzeptsemantik.

3.3.11 Korollar 3.20

Gegeben seien C, T und endliche Interpretation I. Man kann in polynomieller Zeit – genauer: in Zeit $O((|T| + |C|) \cdot \left|\Delta^I\right|)$ – entscheiden, ob I ein Modell von C und T ist.

3.3.12 Theorem 3.21

In ALC ist Erfüllbarkeit bzgl. TBoxen entscheidbar.

4 Tableau-Algorithmen

4.1 ALC ohne TBoxen

4.1.1 Negationsnormalform

Konzept ist in *Negationsnormalform* (NNF) gdw. Negation nur auf Konzeptnamen angewendet wird.

4.1.2 Lemma 4.2

Jedes Konzept kann in Linearzeit in ein äquivalentes Konzept in NNF umgewandelt werden.

Beweisskizze. Wende Gesetzte der doppelten Negation, de Morgan und Dualität von $\exists$, $\forall$ an.

4.1.3 I-Baum

I-Baum für C_0 (in NNF) ist knoten- und kantenbeschrifteter Baum (V, E, L) mit

- V Knotenmenge
- E ist Menge beschrifteter Kanten $\left(v, r, v^{'}\right)$ mit v, $v^{'} \in V$, r Rollenname
- L: $V \rightarrow 2^{sub(C_0)}$

4.1.4 Tableau-Algorithmus

Berechnet Folge $M_0, M_1, \ldots$ von Mengen von I-Bäumen:

$M_0 = \{B_{\text{ini}}\}$ mit B_{ini} *initialer I-Baum* für C_0:

- $V\,\{v_{\text{ini}}\}$
- $E\varnothing$
- $L\,(v_{\text{ini}})\,\{C_0\}$

M_{i+1} entsteht aus M_i durch Anwendung der Tableau-Regeln auf irgendeinen I-Baum in M_i und anschließendes Austauschen des verwendeten Baumes durch den neu erzeugten (Sei (V, E, L) I-Baum):

- $\sqcap$-Regel
  - Wähle $v \in V$ und $C \sqcap D \in L\,(v)$ so dass *nicht* $\{C, D\} \subseteq L\,(v)$
  - erweitere $L(v)$ um C und D
- $\sqcup$-Regel
  - Wähle $v \in V$ und $C \sqcup D \in L\,(v)$ so dass $\{C, D\} \cap L\,(v) = \varnothing$
  - erweitere $L(v)$ um C oder D (ergibt zwei I-Bäume)

- $\exists$-Regel
  - Wähle $v \in V$ und $\exists r.C \in L(v)$ so dass es kein $v' \in V$ gibt mit $(v, r, v') \in E$ und $C \in L(v')$
  - erweitere V um neuen Konten v' und E um (v, r, v'); setze $L(v') = \{C\}$
- $\forall$-Regel
  - Wähle $v, v' \in V$ und $\forall r.C \in L(v)$ so dass $(v, r, v') \in E$ und $C \notin L(v)$
  - erweitere $L(v')$ um C

Stoppe, wenn alle Regeln erschöpfend angewandt wurden. Gib „erfüllbar“ zurück, falls es einen I-Baum ohne offensichtlichen Widerspruch ($\{A, \neg A\} \subseteq L(v)$) gibt; „unerfüllbar“ sonst.

4.1.5 Definition Rollentiefe

Rollentiefe $\mathrm{rd}(C)$ von Konzepten $C \in sub(C_0)$ ist induktiv definiert:

- $\mathrm{rd}(A) = rd(\neg A) = 0$
- $\mathrm{rd}(C \sqcap D) = rd(C \sqcup D) = \max(\mathrm{rd}(C), rd(D))$
- $\mathrm{rd}(\exists r.C) = rd(\forall r.C) = 1 + rd(C)$

Lemma 4.4 Für alle $C \in sub(C_0)$ gilt $\mathrm{rd}(C) \leq |C|$.

4.1.6 Multimengen

Multimengen sind Mengen, in denen Elemente mehrfach vorkommen dürfen. Formal: Abbildung $M : S \to \mathbb{N}$, welche jedes Element auf die Anzahl seines Vorkommens abbildet.

$MM(S)$ ist die Menge aller Multimengen über der Menge S.

Gegeben strikte partielle Ordnung $(S, <)$, ist die Multimengenerweiterung $(\mathrm{MM}(S), <_{\mathrm{mul}})$ definiert als $M_2 <_{\mathrm{mul}} M_1$ gdw. $\exists X, Y \in MM(S)$, so dass

- $\varnothing \neq X \subseteq M_1$
- $M_2 = (M_1 \setminus X) \cup Y$
- $\forall y \in Y \exists x \in X : x > y$

Beispiel: $\{3,1\} >_{\mathrm{mul}} \{2,2,2\} >_{\mathrm{mul}} \{2,2\} >_{\mathrm{mul}} \{2,1,1,1\}$

Theorem 4.6 Wenn $(S, <)$ wohlfundiert (hat keine unendlichen absteigenden Ketten) ist, dann ist auch $(\mathrm{MM}(S), <_{\mathrm{mul}})$ wohlfundiert.

4.1.7 Proposition 4.5 (Terminierung)

Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis in 4 Schritten:

1. Es werden nur I-Bäume mit einem Verzweigungsgrad $\leq |C_0|$ generiert.

Beweisskizze. Nur die $\exists$-Regel generiert Nachfolger, aber höchstens einen für jedes Konzept $\exists r.C \in sub(C_0)$. Nach Lemma 3.13 enthält sub (C_o) höchstens $|C_0|$ viele Konzepte.

1. Es werden nur I-Bäume mit einer Tiefe $\leq |C_0|$ generiert.

Beweisskizze über die Anzahl der Regelanwendungen. Zu zeigen: Wenn v Knoten mit Tiefe i ist, dann gilt rd $(C) \leq rd\,(C_0) - i$ für alle $C \in L\,(v)$.

I.A. Es gibt nur Knoten v_{ini} mit $L\,(v_{\text{ini}}) = \{C_0\}$. I.V. gilt, da $i = 0$.

I.S. Fallunterscheidung nach angewandter Regel (exemplarisch $\sqcap$, $\exists$):

a. $\sqcap$-Regel

   $C \sqcap D \in L(v)$ und $L(v)$ wird durch C, D erweitert. Nach I.V. gilt: rd $(C \sqcap D) \leq rd\,(C_0) - i$, also auch rd $(C) \leq rd\,(C_0) - i$, weil rd $(C) \leq rd\,(C \sqcap D)$. Analog für D.

a. $\exists$-Regel

   Dann $\exists r.C \in L\,(v)$ und es wird neues v' auf Tiefe $i+1$ generiert mit $L\left(v'\right) = \{C\}$. Es gilt rd $(C) = rd\,(\exists r.C) - 1 \leq rd\,(C_0) - i - 1 = rd\,(C_0) - (i+1)$

1. Sei $M_0, M_1, \ldots$ die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal $|C_0|^{|C_0|} \cdot |C_0| \leq 2^{2|C_0|^2} = n$ Regeln entstanden (Knoten im Baum mal Größe Knotenbeschriftung).

4.1.8 Proposition 4.7 (Korrektheit)

Wenn der Tableau-Algorithmus „erfüllbar“ zurückgibt, so ist C_0 erfüllbar.

Beweisskizze per Induktion über die Struktur von C. „erfüllbar“-Ausgabe bedeutet widerspruchsfreier, vollständiger I-Baum $B = (V, E, L)$ gefunden. Konstruiere Interpretation I:

- $\Delta^I = V$
- $r^I = \left\{\left(v, v'\right) \middle| \left(v, r, v'\right) \in E\right\}$ für alle Rollennamen r
- $A^I = \{v | A \in L\,(v)\}$ für alle Konzeptnamen A

Behauptung: Für alle Konzepte C und $v \in V$ gilt $C \in L\,(v)$ impliziert $v \in C^I$. Da $C_0 \in L\,(v_{\text{ini}})$ in B_{ini} gilt auch $C_0 \in L\,(v_{\text{ini}})$ in B. Also $v_{\text{ini}} \in C_0^I$ nach Behauptung, weswegen dann C_0 erfüllbar.

I.A. $C = A$ (Konzeptname) I.B. gilt nach Definition von I.

I.S. An den Beispielen $\neg$, $\sqcap$

- $C = \neg A$

A Konzeptname. Da B keinen offensichtlichen Widerspruch hat, folgt das $A \notin L(v)$. Nach Definition von I gilt $v \notin A^I$. Also $v \in (\neg A)^I$.

- $C = D \sqcap E$

$C \in L(v) \Rightarrow$ ($\sqcap$-Regel nicht anwendbar) $D \in L(v)$, $E \in L(v) \Rightarrow$ (I.V.) $v \in D^I$, $v \in E^I \Rightarrow$ (Semantik) $v \in (D \sqcap E)^I$

4.1.9 Realisierbarkeit

Sei $B = (V, E, L)$ ein I-Baum. Interpretation I *realisiert* B gdw. es gibt eine Funktion $\pi : V \to \Delta^I$ so dass

- $\left(v, r, v'\right) \in E$ impliziert $\left(\pi(v), \pi\left(v'\right)\right) \in r^I$
- $C \in L(v)$ impliziert $\pi(v) \in C^I$

B ist realisierbar, wenn es Interpretation I gibt, die B realisiert. Menge M von I-Bäumen ist realisierbar gdw. ein $B \in M$ realisierbar.

4.1.10 Proposition 4.9 (Vollständigkeit)

Wenn C_0 erfüllbar, so gibt der Tableau-Algorithmus „erfüllbar“ zurück.

Beweisskizze per Induktion über i. Sei C_0 erfüllbar. Nach Proposition 4.5 berechnet der Algorithmus endlich Folge $M_0, \ldots, M_n$. Wir zeigen: M_i ist realisierbar für alle $0 \leq i \leq n$. Daraus folgt: Es gibt realisierbaren Baum $B \in M_n$ und damit enthält B keinen offensichtlichen Widerspruch. Also gibt der Algorithmus „erfüllbar“ zurück.

I.A. $i = 0$. $M_0 = \{B_{\text{ini}}\}$. B_{ini} ist realisierbar, weil C_0 erfüllbar.

I.S. Fallunterscheidung gemäß der Regel, mit der M_{i+1} aus M_i erzeugt wurde. Sei B realisierbarer Baum aus M_i, auf welchen Regel angewandt wird. Beispielhaft $\sqcup$-Regel:

1. $\sqcup$-Regel

Dann wird $B = (V, E, L)$ ersetzt durch $B' = \left(V, E, L'\right) \in M_{i+1}$ und $B'' = \left(V, E, L''\right) \in M_{i+1}$ und es gibt $v \in V$ mit

- $(C \sqcup D) \in L(v)$
- $L'(v) = L(v) \cup \{C\}$, $L''(v) = L(v) \cup \{D\}$
- $L'(u) = L''(u) = L(u)$ für alle $u \neq v$

Es genügt zu zeigen, dass wenn B realisiertbar, dann B' oder B'' realisierbar. Sei I Interpretation, die B realisiert und $\pi : V \to \Delta^I$ Abbildung wie in Definition 4.8. Dann gilt $\pi(v) \in (C \sqcup D)^I$. Nach Semantik: $\pi(v) \in C^I$ oder $\pi(v) \in D^I$. Also realisiert I den Baum B' oder B''.

4.1.11 Praktikabilität

I-Bäume können höchstens exponentiell groß werden.

4.2 ALC mit generellen TBoxen

Jede TBox T ist äquivalent zu einer TBox der Form $\{\top \sqcap C_T\}$. Setzte $C_T \prod_{C \sqsubseteq D \in T} \neg C \sqcup D$.

4.2.1 TBox-Regel

Wähle $v \in V$ so dass $C_T \notin L(V)$ und erweitere $L(v)$ um C_T.

Problem: Terminiert nicht.

4.2.2 Blockieren

Sei (V, E, L) ein I-Baum und $u, v \in V$. Dann ist v direkt blockiert durch u, wenn

1. u Vorgänger von v in B ist und
2. $L(v) \subseteq L(u)$

v ist blockiert, wenn v direkt blockiert ist oder einen direkt blockierten Vorgänger hat.

4.2.3 Neue $\exists$-Regel ($\exists'$-Regel)

- Wähle $v \in V$ und $\exists r.C \in L(v)$ so dass v *nicht blockiert ist und* es kein $v' \in V$ gibt mit $(v, r, v') \in E$ und $C \in L(v')$
- erweitere V um neuen Konten v' und E um (v, r, v'); setze $L(v') = \{C\}$

4.2.4 Vollständigkeit

Beweis wie ohne TBoxen: Alle $M_0, \ldots, M_n$ sind realisierbar bzgl. I (Induktion), also enthält M_n einen Baum ohne offensichtlichen Widerspruch (Nur neue Fallunterscheidung für TBox-Regel und Realisierbarkeitsbegriff auf TBoxen erweitert).

4.2.5 Korrektheit

Beweisskizze per Induktion über die Struktur von C. Definiere Interpretation I:

- $\Delta^I = \{v \in V | \text{v nicht blockiert}\}$
- $r^I = \{(v, v') \mid (v, r, v') \in E\} \cup \{(v, u) \mid \exists (v, r, v') \in E \text{ und } v' \text{ direkt blockiert durch } u\}$
- $A^I = \{v | A \in L(v)\}$

Behauptung: Für alle ALC-Konzepte C und $v \in \Delta^I$ gilt: $C \in L(v) \Rightarrow v \in C^I$. Die Behauptung impliziert wie gewünscht, dass

- I Modell von T ist.

Da die TBox-Regel nicht anwendbar ist, gilt $C_T \in L(v)$ für alle $v \in V$. Also $v \in C_T^I$ für alle $v \in \Delta^I$.

- I Modell von C_0 ist.

Da $C_0 \in L(v_{\text{ini}})$ gilt nach Behauptung $v_{\text{ini}} \in C_0^I$.

I.A. Siehe Beweis zu Proposition 4.7.

I.S. Schritte wie in Beweis zu Proposition 4.7, außer:

- $C = \exists r.D$

Sei $\exists r.D \in L(v)$. Da die $\exists'$-Regel nicht anwendbar ist, gibt es $v' \in V$ mit $\left(v, r, v'\right) \in E$ und $D \in L(v)$. Fallunterscheidung:

1. v' unblockiert. Dann $\left(v, v'\right) \in r^I$ (Definition I), $v' \in D^I$ (**I.V.**) $\Rightarrow v \in (\exists r.D)^I$
2. v' blockiert. Da der direkte Vorgänger v von v' unblockiert ist, ist v' direkt blockiert von unblockiertem Vorgänger u. Es gilt:

- $(v, u) \in r^I$ nach Definition r^I
- $D \in L(v) \subseteq L(u)$ (Blockierungsbedingung)
- $\Rightarrow u \in D^I$ (**I.V.**)

  Also $v \in (\exists r.D)^I$.

- $C = \forall r.D$

Ähnlich zu oberem Fall.

4.2.6 Terminierung

Beweis analog, aber mit Einbezug der TBox.

5 Komplexität

5.1 Komplexität mit TBoxen, obere Schranke

5.1.1 Theorem 5.1

In ALC ist die Erfüllbarkeit von Konzepten bzgl. TBoxen ExpTime-Vollständig.

5.1.2 Definition 5.2 (Typ)

Ein Typ für C_0 und T ist Teilmenge $t \subseteq sub\,(C_0, T)$, so dass

1. $A \in t$ gdw. $\neg A \notin t$ für alle $\neg A \notin sub\,(C_0, T)$
2. $C \sqcap D \in t$ gdw. $C \in t$ und $D \in t$ für alle $C \sqcap D \in sub\,(C_0, T)$
3. $C \sqcup D \in t$ gdw. $C \in t$ oder $D \in t$ für alle $C \sqcup D \in sub\,(C_0, T)$
4. $C_T \in t$

5.1.3 Typelimination

1. Generiere alle Typen für C_0 und T (exponentiell viele)
2. Eliminiere wiederholt Typen, die in keinem Modell von T vorkommen können
3. Überprüfe, ob ein Typ überlebt hat, der C_0 enthält
4. Wenn ja, antworte „erfüllbar“, sonst „unerfüllbar“

5.1.4 Schlechter Typ

Sei Γ Typenmenge und $t \in \Gamma$. Dann ist t *schlecht in* Γ, wenn für ein $\exists r.C \in t$ gilt: Es gibt kein $t' \in \Gamma$ mit $\{C\} \cup \{\mathrm{D} \mid \forall r.D \in t\} \subseteq t'$.

Erklärung: t braucht einen Nachfolger, es gibt aber keinen geeigneten.

Proposition 5.4 (Terminierung) Beweisskizze. Sei $n = |C_0| + |T|$. Proposition folgt aus:

1. Es gibt nur 2^n Typen mit $n\,|C_0| + T$ (Lemma 3.15)
2. In jedem Schritt, der repeat-Schleife wird mindestens ein Typ eliminiert; die Schleife terminiert spätestens nach 2^n Durchläufen.
3. Die restlichen Operationen (prüfen, ob ein Typ schlecht ist usw.) können leicht in Zeit $2^{O(n)}$ implementiert werden.

Proposition 5.5 Korrektheit. Beweisskizze per Induktion über Struktur von C.

Antworte ALC$-Elim(C_0, T)$ „erfüllbar“ und sei Γ_i die resultierende Typmenge. Dann gibt es $t_0 \in \Gamma_i$ mit $C_0 \in t_0$. Definiere Interpretation I:

- $\Delta^I = \Gamma_i$
- $A^I = \{t \in \Gamma_i | A \in t\}$
- $r^i = \left\{ \left(t, t^{'}\right) \mid \forall r.C \in t \text{ und } C \in t^{'} \right\}$

Zu zeigen: I ist Modell von C_0 und T.

Behauptung: Für alle $C \in sub\left(C_0, T\right)$ und alle $t \in \Gamma_i$ gilt $C \in t \Rightarrow t \in C^I$. Daraus folgt:

1. Wegen $C_0 \in t_0$ ist $t_0 \in C_0^I$
2. Wegen $C_T \in t$ für alle $t \in \Gamma_i$ folgt $t \in C_T^I$.

Also ist I Modell von T.

I.A. $C = A$. Folgt direkt aus Definition I.

$C = \neg A$. Nach Definition „Typ“ gilt $A \notin t$. Nach Definition I ist $t \notin A^I$.

I.S. Fallunterscheidung:

- $C = D \sqcap D^{'}$

Nach Definition „Typ“ ist $D \in t$ und $D^{'} \in t$. Nach **I.V.**: $t \in D^I$ und $t \in \left(D^{'}\right)^I$. Nach Semantik: $t \in \left(D \sqcap D^{'}\right)^I$.

- $C = D \sqcup D^{'}$

Analog.

- $C = \forall r.D$

Sei $\forall r.D \in t$ und $\left(t, t^{'}\right) \in r^I$. Nach Definition r^I muss $D \in t^{'}$ gelten. Nach **I.V.**: $t^{'} \in D^I$, also $t \in \left(\forall r.D\right)^I$

- $C = \exists r.D$

Sei $\exists r.D \in t$. Da $t \in \Gamma_i$ (also nicht schlecht), gibt es $t^{'} \in \Gamma_i$ mit $D \in t^{'}$ und $E \in t^{'}$ für alle $\forall r.E \in t$. Nach **I.V.** gilt $t^{'} \in D^I$ und nach Definition von r^I gilt $\left(t, t^{'}\right) \in r^I$. Also $t \in \left(\exists r.D\right)^I$

Vollständigkeit. Beweisskizze per Induktion über i.

Sei C_0 erfüllbar bzgl. T und sei I Modell von T mit $d_0 \in C_0^I$. Sei $\Gamma = \left\{ t_I\left(d\right) \mid d \in \Delta^I \right\}$. Sei $\Gamma_0, \ldots, \Gamma_k$ die von ALC$-$Elim $\left(C_0, T\right)$ erzeugte Sequenz. Behauptung: $\Gamma \subseteq \Gamma_i$ für alle $i \geq 0$. Daraus folgt wegen $d_0 \in C_0^I$, dass $C_0 \in t_i\left(d_0\right) \in \Gamma \subseteq \Gamma_k$. Also gibt ALC$-Elim(C_0, T)$ „erfüllbar“ zurück.

I.A. $i = 0$. Einfach jedes Element von Γ (semantisch Typ) ist auch ein syntaktischer Schritt. **???**

I.S. Gelte $\Gamma \subseteq \Gamma_i$ (I.V.). Zu zeigen: $\Gamma \subseteq \Gamma_{i+1}$. Sei $t \in \Gamma$. Es genügt zu zeigen: t ist nicht schlecht in Γ_i. Sei $\exists r.C \in t$ und $S = \{C\} \cup \{D | \forall r.D \in T\}$. Da $t \in \Gamma$, gibt es $d \in \Delta^I$ mit $t_i\left(d\right) = t_0$. Also gibt es nach Semantik $e \in \Delta^I$, $(d, e) \in r^I$, $e \in D^I$ für alle $D \in S$. $t's$ gilt also $S \subseteq t_I\left(e\right)$ und $t_I\left(e\right) \in \Gamma \subseteq \Gamma_i$. Es folgt: t nicht schlecht

5.1.5 Theorem 5.6

In ALC ist die Erfüllbarkeit von Konzepten bzgl. TBoxen entscheidbar in ExpTime.

5.2 Komplexität mit TBoxen, untere Schranke

5.2.1 ExpTime-Spiele

- Zwei Spieler spielen auf gegebener aussagenlogischer Formel φ
- Jede Variable in φ gehört entweder Spieler 1 oder Spieler 2
- Das Spiel beginnt auf einer gegebenen Anfangsbelegung π_0 der Variablen
- Spieler 1 beginnt, die Spieler wechseln sich ab
- In jedem Zug ändert Spieler Wahrheitswert einer seiner Variablen; es ist erlaubt, zu passen
- Spieler 1 gewinnt, wenn φ jemals wahr wird (egal, welcher Spieler gezogen hat)
- Spieler 2 gewinnt, wenn das Spiel unendlich weitergeht ohne dass φ wahr wird

5.2.2 Definition 5.7

- *Spiel*: Tupel $(\varphi,\ \Gamma_1, \Gamma_2, \pi_0)$ mit Γ_1, Γ_2 Partitionierung der Variablen in φ_1 und π_0 Anfangsbelegung
- *Konfiguration*: Paar (i, π) mit $i \in \{1, 2\}$ aktiver Spieler und π Belegung
- π ist *j-Variation* von π' ($j \in \{1, 2\}$) wenn $\pi = \pi'$ oder π und $\pi^{'}$ unterscheiden sich nur in einer Variablen $p \in \Gamma_j$

„π ist j-Variation von π'“ bedeutet: Spieler j kann π in π' transformieren (oder umgekehrt).

5.2.3 Definition 5.8 (Gewinnstrategie)

Gewinnstrategie für Spieler 2 in Spiel $(\varphi, \Gamma_1, \Gamma_2, \pi_0)$ ist unendlicher knotenbeschrifteter Baum (V, E, l), wobei l jedem Knoten $v \in V$ Konfiguration $l(v)$ zuweist so, dass

a) Wurzel beschriftet mit $(1, \pi_0)$

b) wenn $l(v) = (2, \pi)$, dann hat v Nachfolger $v^{'}$ mit $l(v') = \left(1, \pi^{'}\right)$, wobei $\pi^{'}$ 2-Variation von π

c) wenn $l(v) = (1, \pi)$, dann hat v Nachfolger $v_0, \ldots, v_{|\Gamma_1|}$ mit $l(v_1) = (2, \pi_i)$ wobei $\pi_0, \ldots, \pi_{|\Gamma_1|}$ alle existierenden 1-Variationen von π

d) wenn $l(v) = (i, \pi)$, dann nicht $\pi \vDash \varphi$

5.2.4 Definition 5.9

Spiel$_1$ ist das folgende Problem: Gegeben Spiel $(\varphi, \Gamma_1, \Gamma_2, \pi_1)$, entscheide ob Spieler 2 eine Gewinnstrategie hat.

5.2.5 Theorem 5.10

Spiel_1 ist ExpTime-Vollständig

5.2.6 Reduktion

Reduziere Spiel_1: Gegeben Spiel $(\varphi, \Gamma_1, \Gamma_2, \pi_0)$, konstruiere in Polynomialzeit Konzept C_S und TBox T_S so, dass: Spieler 2 hat Gewinnstrategie in S gdw. C_S erfüllbar bzgl. T_S.

Beweisskizze.

- Hinrichtung: Erzeuge aus der Gewinnstrategie Interpretation und zeige, dass diese Modell ist. **Hinrichtung Sascha**
- Rückrichtung: Nimm an es gibt Baummodell und erzeuge daraus Gewinnstrategie.

5.2.7 Theorem 5.12

In ALC ist die Erfüllbarkeit von TBoxen ExpTime-hart.

Daraus ergibt sich zusammen mit Theorem 5.6 ExpTime-Vollständigkeit.

5.3 Komplexität ohne TBoxen obere Schranke

5.3.1 Theorem 5.13

In ALC ist die Erfüllbarkeit von Konzepten (ohne TBoxen) PSpace-Vollständig.

5.3.2 ALC-Worlds

Wenn C erfüllbar, dann hat C ein Baummodell. Ohne TBox ist dessen Tiefe mit $|C|$ beschränkt. In PSpace:

- Ein linear Tiefer Baum ist exponentiell groß
- Gesamtes Modell im Speicher: nicht PSpace
- Stattdessen Prüfe Existenz des Baumes mittels Tiefensuche; halte zu jeder Zeit nur einen Pfad des Baumes im Speicher

Theorem 5.14 PSpace = NPSpace

Definition 5.15 (i-Konzepte) Für $i \geq 0$ ist die Menge der i-Konzepte definiert als: $\text{sub}_i(C_0)\,\{C \in sub(C_0) \mid \text{rd}(C) \leq i\}$.

Definition 6.16 (i-Typ) Sei $i \geq 0$. i-Typ für C_0 ist Teilmenge $t \subseteq sub_i(C_0)$ so, dass

1. $A \in t$ gdw. $\neg A \notin t$ für alle $\neg A \in sub_i(C_0)$

2. $C \sqcap D \in t$ gdw. $C \in t$ und $D \in t$ für alle $C \sqcap D \in sub_i(C_0)$
3. $C \sqcup D \in t$ gdw. $C \in t$ oder $D \in t$ für alle $C \sqcup D \in sub_i(C_0)$

ALC-Worlds Rekursion über i-Typen.

Proposition 5.17 ALC-Worlds(C_0) terminiert und benötigt polynomiellen Platz.

Beweisskizze. Stelle als Rekursionsbaum dar. Verzweigungsgrad beschränkt durch Anzahl der $\exists$. Tiefe Beschränkt durch $rd(C_0)$. Der Rekursionsstack hat höchstens Tiefe $|C_0|$ und der Platzbedarf pro Aufruf ist polynomiell.

5.3.3 Proposition 5.18

Korrektheitsbeweis per Induktion über C_0:

I.A. $C = A$. Folgt direkt aus Definition I.

$C = \neg A$. Da der Lauf erfolgreich ist, ist $p_1(v)$ Typ für C_0. Nach Definition „Typ“ gilt $A \notin p_1(v)$. Nach Definition I ist $v \notin A^I$.

I.S. Fallunterscheidung:

- $C = D \sqcap D'$

Nach Definition „Typ“ ist $D \in p_1(v)$ und $D' \in p_1(v)$. Nach **I.V.**: $v \in D^I$ und $v \in \left(D'\right)^I$. Nach Semantik: $v \in \left(D \sqcap D'\right)^I$.

- $C = D \sqcup D'$

Analog.

- $C = \forall r.D$

Sei $\left(v, v'\right) \in r^I$. Dann ist $\left(v, v'\right) \in E$ und $\sigma\left(v'\right) = r$. Wegen $\forall r.D \in p_1(v)$ ist auch $D \in p_1\left(v'\right)$. Nach I.V. gilt $v' \in D$. Also $v \in (\forall r.D)^I$

- $C = \exists r.D$

Ähnlich

Vollständigkeitsbeweis Sascha

5.3.4 Theorem 5.19

In ALC ist die Erfüllbarkeit von Konzepten in PSpace.

5.4 Komplexität ohne TBoxen untere Schranke

5.4.1 Definition 5.20

- *Spiel*: Aussagenlogische Formel φ mit Variablen $p_1, \ldots, p_n$, n gradzahlig
- *Konfiguration*: Wort $\pi \in \{0,1\}*$

5.4.2 Definition 5.21

Gewinnstrategie für Spieler 1 in Spiel φ ist endlicher knotenbeschrifteter Baum (V, E, L), wobei l jedem Koten $v \in V$ Konfiguration $l(v)$ zuweist, sodass

a) Wurzel beschriftet mit ε (leere Konfiguration)

b) wenn $l(v) = w$ mit $|w|$ gerade und $|w| < n$ (Also Spieler 1 am Zug), dann hat v Nachfolger v' mit $l(v') \in \{w0, w1\}$

c) wenn $l(v) = w$ mit |w| ungerade (also Spieler 2 am Zug), dann hat v Nachfolger v' und v''mit $l(v') = w0$ und $l(v'') = w1$

d) wenn $l(v) = w$ mit $|w| = n$, dann $w \vDash \varphi$

5.4.3 Theorem 5.25

In ALC ist die Erfüllbarkeit von Konzepten bzgl. leerer TBoxen PSpace-Hart.

Beweisskizze. Konstruiere Konzept C_φ, so dass Spieler 1 hat Gewinnstrategie in φ gdw. C_φ erfüllbar.

5.5 Unentscheidbare Erweiterungen

5.6 Konkrete Bereiche

Ein *Konkreter Bereich* ist ein Paar $B = (\Delta^B, \Phi^B)$ wobei

- Δ^B eine Menge von *Werten* ist und
- Φ^B eine Menge von *Prädikaten*

sodass jedes $P \in \Phi^B$ mit einer Stelligkeit $n \geq 0$ ausgestattet ist und mit einer Extension $P^B \subseteq (\Delta^B)^n$.

5.6.1 Definition 5.27 (ALCB Syntax)

Sei B ein konkreter Bereich. Mit ALC(B) bezeichnen wir die Erweiterung von ALC um B, d.h. um

- *Featurenamen* (eine zusätzliche Art von Rolle) und
- die Konstruktoren $\exists R_1, \ldots, \ R_n.\mathrm{P}$ und $\forall R_1, \ldots, R_n.\mathrm{P}$

wobei $P \in \Phi^B$ n-Stellig ist und die R_i *Rollenkomposition* der Form $r_1; \ldots; r_k; f$ sind mit r_j Rollenname und f Featurename.

5.6.2 Definition 5.28 (ALCB Semantik)

Eine Interpretation I ordnet nun zusätzlich zu jedem Featurenamen f eine Funktion $f^I : \Delta^I \to \Delta^B$ zu. Für jede Rollenkomposition $r = r_1; \ldots r_k; f$ bezeichnet R^I die Komposition der Interpretationen: $R^I = r_1^I \circ \ldots \circ r_k^I \circ f$

Die Semantik der zusätzlichen Konstruktoren ist nun:

$$(\exists R_1, \ldots,\ R_k.\mathrm{P})^I = \left\{d \in \Delta^I \mid \exists d_1, \ldots, d_k : (d, d_i) \in R_i^I \text{ fur } 1 \leq i \leq k \text{ und } (d_1, \ldots,\ d_k) \in P^B\right\}$$

$$(\forall R_1, \ldots,\ R_k.\mathrm{P})^I = \left\{d \in \Delta^I \mid \forall d_1, \ldots, d_k : (d, d_i) \in R_i^I \text{ fur } 1 \leq i \leq k \text{ impliziert } (d_1, \ldots,\ d_k) \in P^B\right\}$$

5.6.3 2-Registermaschinen

- Endlich viele Zustände
- Zwei Register mit Werten aus $\mathbb{N}$
- Instruktionen um
  - Register zu inkrementieren
  - Register auf null zu testen und bei Wert $\neq 0$ zu dekrementieren. Der Folgezustand hängt davon ab, ob das Register 0 war.

5.6.4 Definition 5.30

(Deterministische) *2-Registermaschine* (2RM) ist Paar $M = (Q, P)$ mit $Q = \{q_0, \ldots,\ q_l\}$ Menge von *Zuständen* und $P = I_0, \ldots, I_{l-1}$ *Instruktionsfolge*. Per Definition ist q_0 Startzustand, q_l Stoppzustand. Jede Instruktion ist I_i hat eine der folgenden Formen:

- $I_i = +(p, q_1)$ mit $p \in \{1, 2\}$ *Register* und q_j Folgezustand: Inkrementierunsanweisung
- $I_i = -(p, q_j, q_k)$ mit $p \in \{1, 2\}$ Register und q_j, q_k Folgezustände: Dekrementierungsanweisung mit Folgezustand q_j, wenn Register p den Wert 0 enthält und q_k sonst.

5.6.5 Definition 5.31

Konfiguration und Konfigurationsübergänge $(q, m, n) \vdash_M (q^{'}, m', n^{'})$. Berechnung als eindeutige längste Konfigurationsfolge.

5.6.6 Theorem 5.29

Das Erfüllbarkeitsproblem in $ALC(B_1)$ ist unentscheidbar.

- $\Delta^{B_1} = \mathbb{N}$
- $\Phi^{B_1} = \{=_0, =, +_1\}$, wobei $=_0$ einstellig, die anderen Zweistellig.

Beweisskizze. Gegeben 2RM M, konstuiereALC (B_1)-TBox T_M und wähle einen Konzeptnamen J sodass: M hält auf $(0, 0)$ gdw. J unerfüllbar bzgl. T_M. Zeige dies jeweils für Hinrichtung und Rückrichtung per Kontraposition. **Wo kommen i und j her?**

6 Effiziente Beschreibungslogiken

6.1 EL

Ein EL-Konzept ist ein ALC-Konzept, in dem nur die Konstruktoren $\top$, $\sqcap$ und $\exists r.C$ verwendet werden.

6.2 Simulation

Simulation ist gerichtete Bisimulation. $(I_1, d_1) \precsim (I_2, d_2)$: es gibt Simulation p von I_1 nach I_2 mit d_1 p d_2.

6.2.1 Lemma 6.3

Seien I_1, I_2 Interpretationen, $d_1 \in \Delta^{I_1}$ und $d_2 \in \Delta^{I_2}$. Wenn $(I_1, d_1) \precsim (I_2, d_2)$, dann gilt für alle EL-Konzepte C: $d_1 \in C^{I_1}$ impliziert $d_2 \in C^{I_2}$.

Beweisskizze per Induktion über die Struktur von C. Sei ρ eine Simulation zwischen I_1 und I_2 mit d_1 ρ d_2.

I.A. $C = A$ ist Konzeptname. Nach Bedingung 1. der Bisimulation gilt $d_1 \in A^{I_1}$ impliziert $d_2 \in A^{I_2}$.

I.S. Unterscheide Fälle gemäß dem äußersten Konstrukt von C.

1. $C = D_1 \sqcap D_2$

   $d_1 \in C^{I_1}$ gdw. $d_1 \in D_1^{I_1}$und $d_1 \in D_2^{I_1}$ (Semantik) impliziert. $d_2 \in D_1^{I_2}$ und $d_2 \in D_2^{I_2}$ (I.V.) gdw. $d_2 \in C^{I_2}$ (Semantik)

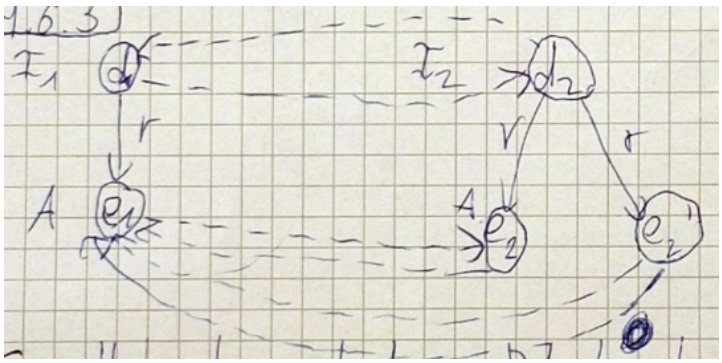
1. $C = \exists r.D$

   Hinrichtung und Rückrichtung analog über Semantik, 2. Bedingung der Simulation, I.V., Semantik.

6.2.2 Lemma 6.4

Bisimulation und wechselseitige Simulation sind nicht dasselbe.

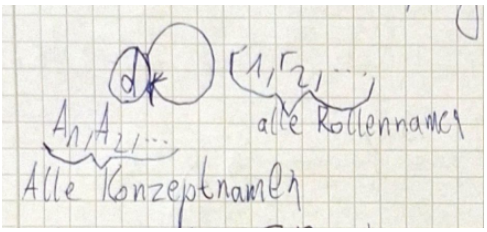
Beweisskizze: Zeige Wechselseitige Simulation, die keine Biosimulation ist:



6.2.3 Lemma 6.6

Jedes EL-Konzept ist erfüllbar bzgl. jeder TBox.

Beweisskizze per Induktion über die Struktur von C:



6.3 Subsumtion ohne TBox

Eine Subsumption $C \sqsubseteq D$ gilt in EL im Prinzip gdw mann D syntaktisch in C „wiederfindet“. Werden Konzepte als Baummodell dargestellt entspricht „Wiederfinden“ Simulation von D-Baum in C-Baum (also Teilgraphenproblem).

6.3.1 Definition kanonisches Modell

Baue aus dem gegebenen Konzept C intuitiv Baummodell.

6.3.2 Lemma 6.8

Für alle EL-Konzepte C gilt: Die Interpretation I_C ist Modell von C mit $d_w \in C^{I_C}$.

Beweisskizze per Induktion über die Struktur von C.

6.3.3 Lemma 6.9

Für alle EL-Konzepte C, Interpretation I und $e \in \Delta^I$ gilt: $e \in C^I$ gdw. $(I_C, d_w) \precsim (I, e)$.

Beweisskizze.

- Hinrichtung per Induktion über C. Schau jeweils Simulation nach I.V. an und ergänze diese.
- Rückrichtung. Angenommen $(I_C, d_w) \precsim (I, e)$. Lemma 6.8 liefert $d_w \in C^{I_C}$. Nach Theorem 6.3 ist $e \in C^I$.

6.3.4 Lemma 6.10

Für alle EL-Konzepte C, D gilt: $C \sqsubseteq D$ gdw. $(I_D, d_w) \precsim (I_C, d_w)$

Beweisskizze.

- Hinrichtung: Betrachte kanonisches Modell I_C von C. Wegen Lemma 6.8 gibt es $d_w \in C^{I_C}$. Mit $C \sqsubseteq D$ folgt $d_W \in D^{I_C}$. Mit Lemma 6.9 folgt für D, I_C, d dass $d_w \in D^{I_C}$ gdw. $(I_D, d_w) \precsim (I_C, d_w)$.

- Rückrichtung. Angenommen $(I_D, d_w) \precsim (I_C, d_w)$. Betrachte beliebige Interpretation I und $d \in C^I$. Zu zeigen: $d \in D^I$. Wegen $d \in C^I$ und Lemma 6.9 gilt $(I_C, d_w) \precsim (I, d)$. Verkette die Simulationen so dass $(I_D, d_w) \precsim (I, d)$. Mit Lemma 6.9 folgt $d_w \in D^{I_C}$.

6.3.5 Theorem 6.11

Subsumtion in EL kann in polynomieller Zeit entschieden werden:

- Konstruiere I_C und I_D in polynomieller Zeit.
- Überprüfe in polynomieller Zeit, ob $(I_D, d_w) \precsim (I_C, d_w)$
  - Berechne maximale Simulation ς
  - Teste ob $(d_w, d_W) \in \varsigma$
  - **Sascha**

6.4 Subsumption mit TBox

6.4.1 Lemma 6.12

Seien C, D zwei beliebige EL-Konzepte und T eine EL-TBox. Sei weiterhin $T' = T \cup \{A_C \sqsubseteq C, D \sqsubseteq A_D\}$. mit Konzeptnamen A_C, A_D, die nicht in C, D, T vorkommen. Dann gilt; $T \vDash C \sqsubseteq D$ gdw $T' \vDash A_C \sqsubseteq A_D$.

6.4.2 Normalform

Eine TBox ist in *Normalform*, wenn sie nur Inklusionen folgender Form enthält:

$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq A$ $A \sqsubseteq \exists r.A_1$ $\exists r.A \sqsubseteq A_1$

6.4.3 Lemma 6.14

Jede EL-TBox T kann in polynomieller Zeit in eine TBox T' in Normalform gewandelt werden, so dass für alle Konzeptnamen A, B in T gilt: $T \vDash A \sqsubseteq B$ gdw $T \vDash A \sqsubseteq B$. Dann ist T' *konservative Erweiterung* von T.

6.4.4 Lemma 6.14

Jede EL-TBox T kann durch linear viele Regelanwendungen in TBox in Normalform transformiert werden, die konservative Erweiterung von T ist.

Die Regeln fügen jeweils Zwischenkonzepte ein.

Beweisskizze: Grad der Abnormalität definieren und zeigen, dass

1. Der Grad ist beschränkt durch $|T|$
2. Jede Regelanwendung verringert den Grad

3. TBoxen vom Grad 0 sind in Normalform

6.4.5 Algorithmus

Wende Regeln erschöpfend an um alle Subsumptionen zu berechnen:

$\frac{}{A \sqsubseteq A}$ (Wenn A in T vorkommt) $\frac{}{A \sqsubseteq \top}$ (Wenn A in T vorkommt)

$\frac{A \sqsubseteq A_1, \ldots, A \sqsubseteq A_n, A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B}{A \sqsubseteq B}$ $\frac{A \sqsubseteq \exists r.A_1, \; A_1 \sqsubseteq B_1, \exists r.B_1 \sqsubseteq B}{A \sqsubseteq B}$

Wo gehört das x hin Vorlesung 17?

Für eine EL-TBox T sei $T*$ das Ergebnis erschöpfender Regelanwendungen, die *Saturierung.*

6.4.6 Theorem 6.16

Für alle Konzeptnamen A, B in T gilt: $T \vDash A \sqsubseteq B$ gdw $A \sqsubseteq B \in T*$

Terminierung Beweisskizze. Jede Regelanwendung erzeugt eine neue Konzeptinklusion $A \sqsubseteq B$, wobei A, B Konzeptnamen aus T. Es gibt nur endlich viele solcher Inklusionen.

Korrektheit Beweisskizze. Sei $T = T_0, \; \ldots, (T_n = T*)$ die durch Regelanwendungen erzeugte Folge von TBoxen. Es genügt zu zeigen: $T_i \vDash T_{i+1}$. Zeige dies durch Vorbedingung der Regelanwendung und der Semantik.

Vollständigkeit

Kanonische Interpretation Die Kanonische Interpretation I ist:

- $\Delta^I = \{d_A \mid A \text{ Konzeptname in T}*\} \cup \{d_\top\}$
- $A^I = \{d_\text{B} \mid B \sqsubseteq A \in T*\}$
- $r^I = \left\{(d_A, d_B) \;\middle|\; A \sqsubseteq A' \in T* \text{ und } A' \sqsubseteq \exists r.B \in T*, \; A' \text{ Konzeptname}\right\}$

Erklärung: Konstruiere Intuitiv Modell aus allen Konzeptnamen.

Lemma 6.18 Die kanonische Interpretation ist ein Model von $T*$.

Beweis: Zeige, dass alle Inklusionen in $T*$ von I erfüllt werden. Verwende die Inklusionen in Normalform. Überlege dazu, was aus den Inklusionen mithilfe der Definition gefolgert werden kann.

Vollständigkeit Angenommen $A \sqsubseteq B \notin T*$. Betrachte Element d_A der kanonischen Interpretation I. Wegen R1: $A \sqsubseteq A \in T*$, also nach Def. I: $d_A \in A^I$. Def. von I und $A \sqsubseteq B \notin T*$ liefern $d_A \notin B$. Da I Modell von $T*$ (Lemma 6.18), und damit von T, folgt nicht $T \vDash A \sqsubseteq B$.

6.5 Erweiterungen von EL

6.5.1 EL mit Disjunktion und Bottom

Erfüllbarkeit in $\mathrm{ELU}_\bot$ (mit Disjunktion) ist ExpTime-Vollständig.

Beweisskizze per Reduktion von Erfüllbarkeit von Konzeptname A bzgl. ALC-TBox T:

1. Ersetze Werterestriktion in T durch Existenzrestriktion.
2. Bringe T in Negationsnormalform.
3. Ersetze $\neg X$ durch X mit $\top \sqsubseteq X \sqcup X$ und $X \sqcap X \sqsubseteq \bot$

6.5.2 ELU (mit Disjunktion)

Erfüllbarkeit in ELU ist ExpTime-Vollständig.

Beweisskizze per Reduktion von 6.5.1. Ersetzte $\bot$ durch L mit $\exists r.L \sqsubseteq L$ für alle Rollennamen r in T.

7 ABoxen und Anfragebeantwortung

8 Übersichten

Literatur

Schneider, T. (2016). Vorlesung Beschreibungslogik. http://http://www.informatik.uni-bremen.de/tdki/lehre/ss16/bl/. Last accessed on 2016-12-04.