

SoSe 2016

# **Beschreibungslogik**

Mitschriften / "Lecture Notes"

Dozent: Prof. Dr. Thomas Schneider  
Mitschrift/Kommentare von: Sascha Jongebloed, Robin Nolte

8. Februar 2017

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
1.1 Wissensrepräsentation . . . . .	3
1.1.1 Definition . . . . .	3
1.1.2 Wohldefinierte Syntax und Semantik . . . . .	3
1.1.3 In Beschreibungslogik . . . . .	3
<b>2 Grundlagen</b>	<b>4</b>
2.1 Attributive Language with Complement ( $\mathcal{ALC}$ ) . . . . .	4
2.1.1 Konzept- und Rollennamen . . . . .	4
2.1.2 $\mathcal{ALC}$ : Syntax . . . . .	4
2.1.3 $\mathcal{ALC}$ : Semantik . . . . .	5
2.1.4 Extension/Modell . . . . .	5
2.1.5 Erfüllbarkeit, Subsumtion, Äquivalenz . . . . .	6
2.2 TBoxen . . . . .	6
2.2.1 TBox-Syntax . . . . .	6
2.2.2 TBox-Semantik . . . . .	6
2.2.3 Modellierung . . . . .	7
2.2.4 Erfüllbarkeit, Subsumtion, Äquivalenz . . . . .	7
2.2.5 Monotonie . . . . .	7
2.3 Schlussfolgerungsprobleme . . . . .	8
2.3.1 Subsumtion als Ordnungsrelation . . . . .	8
2.3.2 Klassifikation . . . . .	9
2.3.3 Reduktion . . . . .	9
2.4 Erweiterungen von $\mathcal{ALC}$ . . . . .	9
2.4.1 Inverse Rollen ( $\mathcal{ALCI}$ ) . . . . .	9
2.4.2 Zahlenrestriktion ( $\mathcal{ALCQ}$ ) . . . . .	10
<b>3 Ausdrucksstärke und Modellkonstruktionen</b>	<b>11</b>
3.1 Bisimulation . . . . .	11
3.2 Ausdrucksstärke . . . . .	12
3.2.1 Anwendungen von Bisimulation I . . . . .	12
3.2.2 Anwendungen von Bisimulation II . . . . .	13
3.2.3 Unravelling . . . . .	14
3.2.4 Bisimulation versus Ausdrucksstärke . . . . .	15
3.2.5 Bisimulation für Erweiterungen von $\mathcal{ALCI}$ . . . . .	15
3.3 Ausdrucksstärke und Modellkonstruktion . . . . .	16
3.3.1 Größe von Konzepten und TBoxen . . . . .	16
3.3.2 Endliche/beschränkte Modelleigenschaft . . . . .	16
3.3.3 Typ . . . . .	16
3.3.4 Typ . . . . .	17
3.3.5 Filtration . . . . .	18
3.3.6 Endliche/Beschränkte Modelleigenschaft . . . . .	20
3.3.7 Entscheidbarkeit . . . . .	20
<b>4 Tableau-Algorithmen</b>	<b>22</b>
4.1 Ziel . . . . .	22

4.2	ALC ohne TBoxen . . . . .	22
4.2.1	Negationsnormalform . . . . .	22
4.2.2	I-Baum . . . . .	22
4.2.3	Tableau-Algorithmus . . . . .	23
4.2.4	Definition Rollentiefe . . . . .	24
4.2.5	Multimengen . . . . .	24
4.2.6	Terminierung . . . . .	25
4.2.7	Korrektheit und Vollständigkeit . . . . .	26
4.2.8	Komplexitätsanalyse . . . . .	29
4.2.9	Praktikabilität . . . . .	29
4.3	ALC mit generellen TBoxen . . . . .	29
4.3.1	TBox-Regel . . . . .	30
4.3.2	Blockieren . . . . .	30
4.3.3	Neue $\exists$ -Regel ( $\exists'$ -Regel) . . . . .	30
4.3.4	Vollständigkeit . . . . .	31
4.3.5	Korrektheit . . . . .	31
4.3.6	Terminierung . . . . .	32
4.3.7	Komplexität . . . . .	33
4.3.8	Bemerkung zur TBox-Regel . . . . .	33
4.3.9	Erweiterungen . . . . .	33
<b>5</b>	<b>Komplexität</b>	<b>34</b>
5.1	Komplexität mit TBoxen, obere Schranke . . . . .	34
5.1.1	Theorem 5.1 . . . . .	34
5.1.2	Definition 5.2 (Typ) . . . . .	34
5.1.3	Typelimination . . . . .	34
5.1.4	Schlechter Typ . . . . .	34
	Proposition 5.4 (Terminierung) . . . . .	34
	Proposition 5.5 . . . . .	34
5.1.5	Theorem 5.6 . . . . .	35
5.2	Komplexität mit TBoxen, untere Schranke . . . . .	36
5.2.1	ExpTime-Spiele . . . . .	36
5.2.2	Definition 5.7 . . . . .	36
5.2.3	Definition 5.8 (Gewinnstrategie) . . . . .	36
5.2.4	Definition 5.9 . . . . .	36
5.2.5	Theorem 5.10 . . . . .	37
5.2.6	Reduktion . . . . .	37
5.2.7	Theorem 5.12 . . . . .	37
5.3	Komplexität ohne TBoxen obere Schranke . . . . .	37
5.3.1	Theorem 5.13 . . . . .	37
5.3.2	ALC-Worlds . . . . .	37
	Theorem 5.14 . . . . .	37
	Definition 5.15 (i-Konzepte) . . . . .	37
	Definition 6.16 (i-Typ) . . . . .	37
	ALC-Worlds . . . . .	38
	Proposition 5.17 . . . . .	38
5.3.3	Proposition 5.18 . . . . .	38
5.3.4	Theorem 5.19 . . . . .	38

5.4	Komplexität ohne TBoxen untere Schranke . . . . .	38
5.4.1	Definition 5.20 . . . . .	38
5.4.2	Definition 5.21 . . . . .	39
5.4.3	Theorem 5.25 . . . . .	39
5.5	Unentscheidbare Erweiterungen . . . . .	39
5.6	Konkrete Bereiche . . . . .	39
5.6.1	Definition 5.27 (ALCB Syntax) . . . . .	39
5.6.2	Definition 5.28 (ALCB Semantik) . . . . .	39
5.6.3	2-Registermaschinen . . . . .	40
5.6.4	Definition 5.30 . . . . .	40
5.6.5	Definition 5.31 . . . . .	40
5.6.6	Theorem 5.29 . . . . .	40
<b>6</b>	<b>Effiziente Beschreibungslogiken</b>	<b>41</b>
6.1	EL . . . . .	41
6.2	Simulation . . . . .	41
6.2.1	Lemma 6.3 . . . . .	41
6.2.2	Lemma 6.4 . . . . .	41
6.2.3	Lemma 6.6 . . . . .	42
6.3	Subsumtion ohne TBox . . . . .	42
6.3.1	Definition kanonisches Modell . . . . .	42
6.3.2	Lemma 6.8 . . . . .	42
6.3.3	Lemma 6.9 . . . . .	42
6.3.4	Lemma 6.10 . . . . .	42
6.3.5	Theorem 6.11 . . . . .	43
6.4	Subsumption mit TBox . . . . .	43
6.4.1	Lemma 6.12 . . . . .	43
6.4.2	Normalform . . . . .	43
6.4.3	Lemma 6.14 . . . . .	43
6.4.4	Lemma 6.14 . . . . .	43
6.4.5	Algorithmus . . . . .	44
6.4.6	Theorem 6.16 . . . . .	44
Terminierung . . . . .	44	
Korrektheit . . . . .	44	
Vollständigkeit . . . . .	44	
6.5	Erweiterungen von EL . . . . .	45
6.5.1	EL mit Disjunktion und Bottom . . . . .	45
6.5.2	ELU (mit Disjunktion) . . . . .	45
<b>7</b>	<b>ABoxen und Anfragebeantwortung</b>	<b>46</b>
<b>8</b>	<b>Übersichten</b>	<b>47</b>

# 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 ( $\mathcal{ALC}$ )

#### 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 ist 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  $r$  Rollename, so sind
  - $\exists r.C$  (Existenzrestriktion)
  - $\forall r.C$  (Werterestriktion)

$\mathcal{ALC}$ -Konzepte

#### T2.1 Beispiel

Hier einige Beispiel für diese Syntax:

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

#### 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  $A \sqcup \neg A$
- $\perp$  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  $\forall r.(\exists r.(A \sqcap B))$ .

Des Weiteren ist keine Präzedenz zwischen  $\sqcap$  und  $\sqcup$  definiert worden: Daher müssen Klammern verwendet werden!

### 2.1.3 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  $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

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

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

Verwendete Symbole:

- $\mathcal{I}, \mathcal{J}$  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. Intuitiv entspricht  $\top$  die Menge alle Elemente.

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

Mensch  $\sqcap \forall \text{hatKind}.\perp$  Menschen, die keine Kinder haben

### 2.1.5 Erfüllbarkeit, Subsumtion, Äquivalenz

**Definition 2.3.** Erfüllbar, susumiert, äquivalent

Seien  $C$  und  $D$   $\mathcal{ALC}$ -Konzepte. Dann

- ist  $C$  *erfüllbar*, wenn es eine Interpretation  $I$  gibt mit  $C^{\mathcal{I}} \neq \emptyset$ .  $\mathcal{I}$  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-Semantik

**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.
- Disjunktionsconstraints  $C \sqcap D \sqsubseteq \perp$ : Kein Objekt kann gleichzeitig zu  $C$  und  $D$  gehören
- Komplexe Zusammenhänge zwischen mehreren Konzepten. Zum Beispiel:

$$\textit{Professor} \sqcap \exists \textit{hat.Lehrdeputat} \sqsubseteq \exists \textit{haelt.Vorlesung}$$

### 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  $C^{\mathcal{I}} \neq \emptyset$
- 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  $\mathcal{T} \models C \sqsubseteq D$ )
- sind  $C$  und  $D$  äquivalent bzgl.  $\mathcal{T}$ , gdw.  $C^{\mathcal{I}} = D^{\mathcal{I}}$  in allen Modellen  $\mathcal{I}$  von  $\mathcal{T}$  (Notation  $\mathcal{T} \models C \equiv D$ )

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-Subsumption 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  $\mathcal{T}_1 \models C \sqsubseteq D$ , dann  $\mathcal{T}_2 \models C \sqsubseteq D$ .

**T2.8.** Beweisskizze.

1. *Beweis.* Sei  $C$  erfüllbar bzgl.  $\mathcal{T}_2$ . Dann gibt es Modell  $\mathcal{I}$  von  $\mathcal{T}_2$  mit  $C^{\mathcal{I}} \neq \emptyset$ . 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  $C^{\mathcal{I}} \neq \emptyset$ . Also ist  $C$  erfüllbar bzgl.  $\mathcal{T}_1$ .  $\square$

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  $\mathcal{T}_1 \not\models C \sqsubseteq D$   $\square$

Die umgekehrten Aussagen sind im Allgemeinen nicht richtig. Beispiel:

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

$$\mathcal{T}_1 \subseteq \mathcal{T}_2$$

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

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

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  $\mathcal{T} = \emptyset$  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

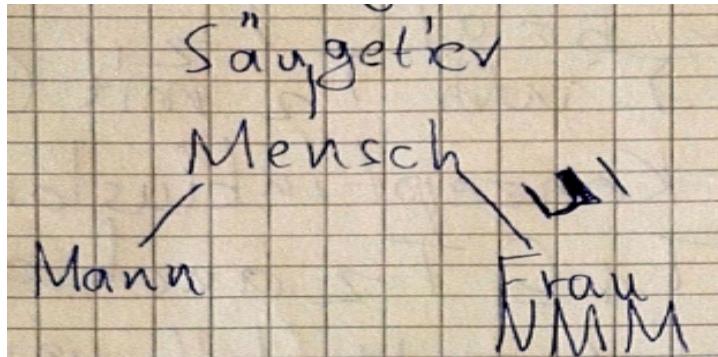
**Lemma 2.8.** Für jede TBox  $\mathcal{T}$  ist die Relation “ $\sqsubseteq$  bzgl.  $\mathcal{T}$ ”

- 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-Diagramm darstellen, dessen Knoten mit Mengen von Konzepten beschriftet sind.

## 2.9



### 2.3.2 Klassifikation

Ein weiteres Schlussfolgerungsproblem:

- **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:

**Lemma 2.9.** 1. Erfüllbarkeit auf Nicht-Äquivalenz

$C$  erfüllbar bzgl.  $T$  gdw.  $T \not\models C \equiv \perp$

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 $\mathcal{ALC}$

Wir betrachten exemplarisch die beiden Erweiterungen  $\mathcal{ALCI}$  und  $\mathcal{ALCQ}$  von  $\mathcal{ALC}$ . Es existieren viel mehr Erweiterungen (z.B.: um spezielle Rolleninterpretationen, temporale Operatoren etc.). Diese können auch kombiniert werden, z.B.:  $\mathcal{ALCQI}$ .

### 2.4.1 Inverse Rollen ( $\mathcal{ALCI}$ )

Zunächst betrachten wir  $\mathcal{ALCI}$ , das  $\mathcal{ALC}$  um die Möglichkeit inverse Rollen in Existenz- und Werterestriktionen zu benutzen erweitert.

**Definition 2.10.** Inverse Rollen 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 ( $\mathcal{ALCQ}$ )

**Definition 2.11.** Zahlenrestriktion Für jede natürliche Zahl  $n$ , jeden Rollennamen  $r$  und jedes Konzept  $C$ :

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

Die Semantik ist

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

Beachte:

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

### 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

Bisimulation ist ein graphentheoretische Begriff, die “Ähnlichkeit” von Graphen beschreibt und eng mit der Ausdruckstärke von  $\mathcal{ALC}$  zusammenhängt.

**Definition 3.1.** Bisimulation

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

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  $\rho$  zwischen  $I_1$  und  $I_2$  mit  $d_1 \rho d_2$ . Die leere Relation ist immer Bisimulation.

**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  $C$ :

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

#### T3.2.

*Beweis.* Beweisskizze per Induktion über die Struktur von  $C$ . Sei  $\rho$  eine Bisimulation zwischen  $I_1$  und  $I_2$  mit  $d_1 \rho d_2$ .

**I.A.**  $C = A$  ist Konzeptname. Nach Bedingung 1. der Bisimulation gilt

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

**I.S.** Unterscheide Fälle gemäß des äußersten Konstruktes von  $C$ . Es genügen  $\neg, \sqcap, \exists r.C$ :

1.  $C = \neg D$

$$\begin{aligned} d_1 \in C^{I_1} &\stackrel{\text{Sem.}}{\text{gdw.}} d_1 \notin D^{I_1} \\ &\stackrel{\text{I.V.}}{\text{gdw.}} d_2 \notin D^{I_2} \\ &\stackrel{\text{Sem.}}{\text{gdw.}} d_2 \in C^{I_2} \end{aligned} \tag{1}$$

1.  $C = D_1 \sqcap D_2$

$$\begin{aligned}
 d_1 \in C^{\mathcal{I}_1} &\stackrel{\text{Sem.}}{gdw.} d_1 \in D_1^{\mathcal{I}_1} \text{ und } d_1 \in D_2^{\mathcal{I}_1} \\
 &\stackrel{\text{I.V.}}{gdw.} d_2 \in D_1^{\mathcal{I}_2} \text{ und } d_2 \in D_2^{\mathcal{I}_2} \\
 &\stackrel{\text{Sem.}}{gdw.} d_2 \in C^{\mathcal{I}_2}
 \end{aligned} \tag{2}$$

1.  $C = \exists r.D$

$$\begin{aligned}
 d_1 \in C^{\mathcal{I}_1} &\stackrel{\text{Sem.}}{\Rightarrow} \exists e_1 : (d_1, e_1) \in r^{\mathcal{I}_1} \text{ und } e \in D^{\mathcal{I}_1} \\
 &\stackrel{\text{2.Bed.}}{\Rightarrow} \exists e_2 \in \Delta_2^{\mathcal{I}_2} : (d_2, e_e) \in r^{\mathcal{I}_2} \text{ und } (e_1 \rho e_2) \\
 &\stackrel{\text{I.V.}}{\Rightarrow} e_2 \in D^{\mathcal{I}_2} \\
 &\stackrel{\text{Sem.}}{\Rightarrow} d_2 \in (\exists r.D)^{\mathcal{I}_2} \\
 &\stackrel{\text{Sem.}}{gdw.} d_2 \in C^{\mathcal{I}_2}
 \end{aligned} \tag{3}$$

Rückrichtung analog, nur mit der 3. Bedingung.

□

## 3.2 Ausdrucksstärke

**Definition 3.3.** Eigenschaft, Ausdrückbarkeit

Eine *Eigenschaft*  $E$  ist eine Menge von Paaren  $(\mathcal{I}, d)$ , wobei  $\mathcal{I}$  eine Interpretation und  $d \in \Delta^{\mathcal{I}}$  ein Element in  $\mathcal{I}$  ist.

$E$  ist *ausdrückbar in  $\mathcal{ALC}$* , wenn es ein  $\mathcal{ALC}$ -Konzept  $C$  gibt, so dass für alle  $\mathcal{I}$  und  $d \in \Delta^{\mathcal{I}}$  gilt:

$$(\mathcal{I}, d) \in E \text{ gdw. } d \in C^{\mathcal{I}}$$

### 3.2.1 Anwendungen von Bisimulation I

**Theorem 3.4.** In  $\mathcal{ALC}$  ist nicht ausdrückbar:

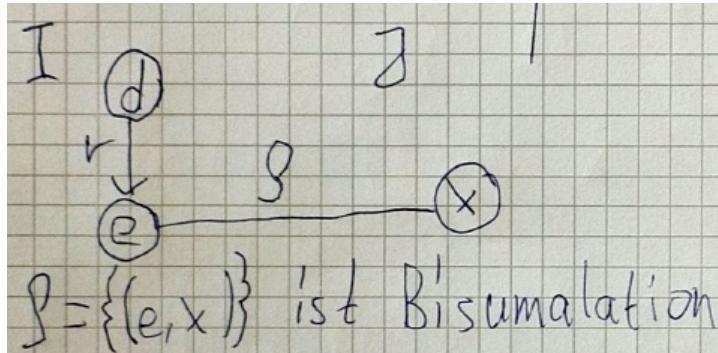
- das  $\mathcal{ALCI}$ -Konzept  $\exists r^-. \top$
- die  $\mathcal{ALCQ}$ -Konzepte
  - $(\leq n r \top)$  für alle  $n > 0$  und
  - $(\geq n r \top)$  für alle  $n > 1$

#### T.3.3.

Beweisskizze. Finde Bisimulation für die dies nicht gilt.

*Beweis.*  $\exists r. \top$

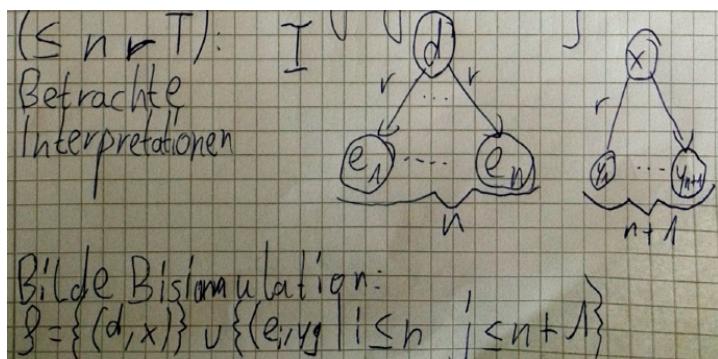
Betrachte 2 Interpretationen  $\mathcal{I}$  und  $\mathcal{J}$



Angenommen, es gäbe ein  $\mathcal{ALC}$ -Konzept  $C$  mit  $C \equiv \exists r^-. \top$ . Dann gilt  $e \in C^\mathcal{I}$ . Mit Theorem 3.2 folgt  $x \in C^\mathcal{J}$ , also hat  $x$  auch einen  $r$ -Vorgänger in  $\mathcal{J}$ .  $\nabla$   $\square$

*Beweis.*  $\leq n r \top$

Betrachte 2 Interpretationen  $\mathcal{I}$  und  $\mathcal{J}$



Angenommen, es gäbe ein  $\mathcal{ALC}$ -Konzept  $C$  mit  $C \equiv (\leq n r \top)$ . Dann gilt  $d \in C^\mathcal{I}$ . Mit Theorem 3.2 folgt  $x \in C^\mathcal{J}$ , also  $x \in (\leq n r \top)$ .  $\nabla$   $\square$

Man sieht, dass die Argumentation zum Beweis der Nicht-Ausdrückbarkeit immer auf dasselbe hinausläuft:

**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  $\mathcal{ALC}$  ausdrückbar.

### 3.2.2 Anwendungen von Bisimulation II

Interpretation ist *Baum* gdw.  $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$  Baum (endl. oder unendl.).  $\mathcal{ALC}$  hat die *Baummodelleigenschaft*

**Theorem 3.6.** Wenn ein ALC-Konzept  $C$  bzgl. einer ALC-TBox  $\mathcal{T}$  erfüllbar ist, dann haben  $C$  und  $\mathcal{T}$  ein gemeinsames Baummodell  $\mathcal{I}$ . (mit  $\mathcal{I}$  Baum, Wurzel in  $C^\mathcal{I}$ .)

### 3.2.3 Unravelling

Sei  $\mathcal{I}$  eine Interpretation und  $d \in \Delta^{\mathcal{I}}$ .

$d$ -Pfad in  $\mathcal{I}$ : Sequenz  $d_0 d_1 \dots d_{n-1}$ ,  $n > 0$  mit

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

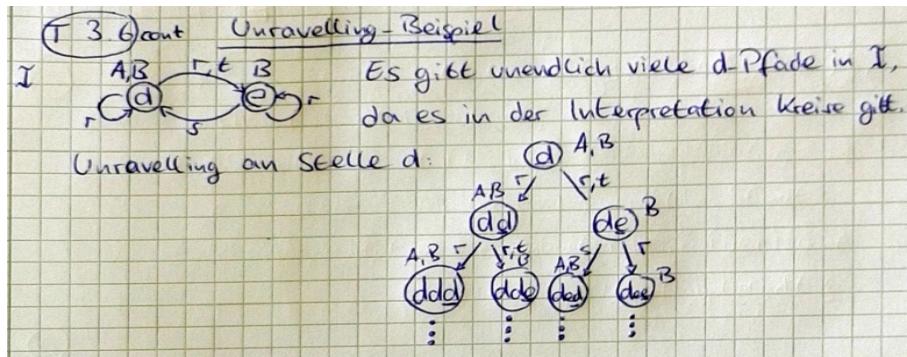
Wir setzen  $\text{end}(d_0 \dots d_{n-1}) = d_{n-1}$ .

**Definition 3.7.** Unravelling

Unravelling von  $\mathcal{I}$  an Stelle  $d$  ist folgende Interpretation  $\mathcal{J}$ :

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

für alle Konzeptnamen  $A$  und Rollennamne  $r$



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.

**Lemma 3.8.** Sei  $\mathcal{J}$  Unravelling von  $\mathcal{I}$  an Stelle  $d$ . Für alle  $\mathcal{ALC}$ -Konzepte  $C$  und alle  $p \in \Delta^{\mathcal{J}}$  gilt:

$$\text{end}(p) \in C^{\mathcal{I}} \text{ gdw. } p \in C^{\mathcal{J}}$$

### T3.7

*Beweis.* Mit Theorem genügt es folgende Bisimulation zu zeigen:

$$(I, \text{end}(p)) \sim (J, p)$$

Definiere Relation:

$$\rho = \{(\text{end}(p), p) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}} \mid p \text{ ist } d\text{-Pfad}\}$$

(Bilde alle Knoten in  $\mathcal{J}$  auf ihr Ende ab).

Zur Bisimulation:

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

□

Nun folgt Theorem 3.6.:

**Theorem 3.6** Wenn ein  $\mathcal{ALC}$ -Konzept  $C$  bzgl. einer TBox  $\mathcal{T}$  erfüllbar ist, dann haben  $C$  und  $\mathcal{T}$  ein gemeinsames Baummodell  $\mathcal{I}$ .

### T3.8

*Beweis.* Sei  $C$  erfüllbar bezüglich  $\mathcal{T}$ . Dann gibt es Modell  $\mathcal{I}$  von  $C$  und  $\mathcal{T}$ . Sei  $\mathcal{J}$  das Unravelling von  $\mathcal{I}$  an der Stelle  $d_0 \in C^{\mathcal{I}}$ .

Mit Lemma 3.8 ist die Wurzel  $d_0$  von  $\mathcal{J}$  in  $C^{\mathcal{J}}$ . Noch zu zeigen:  $\mathcal{J}$  ist Modell von  $\mathcal{T}$ .

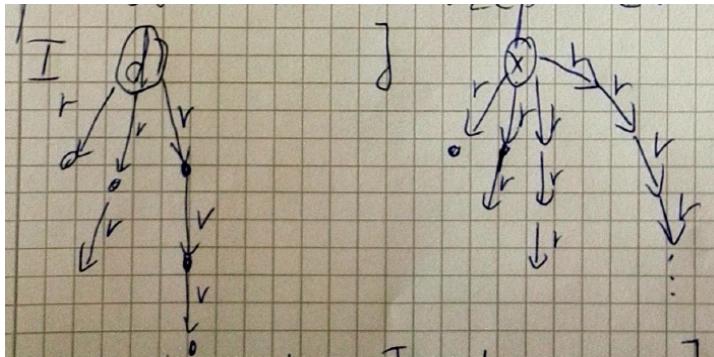
Sei  $D \sqsubseteq E$  in  $\mathcal{T}$  und  $p \in D^{\mathcal{J}}$ . Nach Lemma 3.8 ist dann  $\text{end}(p) \in D^{\mathcal{I}}$  und weil  $\mathcal{I}$  Modell von  $\mathcal{T}$  folgt  $\text{end}(p) \in E^{\mathcal{I}}$ . Mit Lemma 3.8 folgt  $p \in E^{\mathcal{J}} \Rightarrow J \models D \sqsubseteq E$ . □

#### 3.2.4 Bisimulation versus Ausdrucksstärke

Bisimulation entspricht nicht der Ausdrucksstärke von  $\mathcal{ALC}$ , denn die Gegenrichtung von Theorem 3.2 gilt nicht:

Es gibt Interpretationen  $\mathcal{I}$  und  $\mathcal{J}$  und  $d \in \mathcal{I}$ ,  $x \in \mathcal{J}$ , so dass  $d \in C^{\mathcal{I}}$  gdw.  $x \in C^{\mathcal{J}}$  für alle  $\mathcal{ALC}$ -Konzepte  $C$ , aber  $(\mathcal{I}, d) \not\sim (\mathcal{J}, x)$ .

Beispiel:



Es gilt:

- $d \in C^{\mathcal{I}}$  gdw.  $x \in C^{\mathcal{J}}$ , falls ALC-Konzept C
- $(\mathcal{I}, d) \not\sim (\mathcal{J}, x)$ : für  $y$  in  $\mathcal{J}$  gibt es kein adäquates Element  $e \in \Delta^{\mathcal{I}}$  mit  $e \rho y$

Sie gilt allerdings für verschiedene Klassen von Interpretationen, wie die Klasse aller endlichen Interpretationen oder die Klasse aller Interpretationen mit endlicher Verweigungszahl.

#### 3.2.5 Bisimulation für Erweiterungen von $\mathcal{ALCI}$

Für  $\mathcal{ALCI}$ ,  $\mathcal{ALCQ}$  und  $\mathcal{ALQI}$  gibt es ebenfalls Bisimulationsbegriffe.

Fpr  $\mathcal{ALCI}$  füge 2 Regeln hinzu, sodass Vorgänger auch simuliert sein müssen.

So kann man zudem zeigen, dass  $\mathcal{ALCI}$ ,  $\mathcal{ALCQ}$  und  $\mathcal{ALQI}$  die Baummodelleigenschaft haben.

### 3.3 Ausdrucksstärke und Modellkonstruktion

In diesem Kapitel wird die endliche Modelleigenschaft und Filtration eingeführt.

#### 3.3.1 Größe von Konzepten und TBoxen

**Definition 3.9.** 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  $\mathcal{T}$  ist

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

Intuitiv entspricht dies der Anzahl von Symbolen in  $C$  bzw.  $\mathcal{T}$ .

#### 3.3.2 Endliche/beschränkte Modelleigenschaft

$\mathcal{ALC}$  hat *endliche Modelleigenschaft*:

**Theorem 3.10.** Wenn ein  $\mathcal{ALC}$ -Konzept  $C$  bzgl. einer  $\mathcal{ALC}$ -TBox  $\mathcal{T}$  erfüllbar ist, dann haben  $C$  und  $\mathcal{T}$  ein gemeinsames *endliches* Modell.

$\mathcal{ALC}$  hat sogar *beschränkte Modelleigenschaft*:

**Theorem 3.11.** Wenn ein  $\mathcal{ALC}$ -Konzept  $C$  bzgl. einer  $\mathcal{ALC}$ -TBox  $\mathcal{T}$  erfüllbar ist, dann haben  $C$  und  $\mathcal{T}$  ein gemeinsames Modell der Kardinalität  $\leq 2^{|C|+|\mathcal{T}|}$

#### 3.3.3 Typ

Im Folgenden sei  $C$   $\mathcal{ALC}$ -Konzept und  $\mathcal{T}$  TBox, so dass  $C$  erfüllbar bzgl.  $\mathcal{T}$ .

Wir definieren den Begriff eines Typs:

- ist Menge von Konzepten
- beschreibt einen Punkt  $d \in \Delta^{\mathcal{T}}$  in einer Interpretation  $\mathcal{I}$
- Einschränkung auf Teilkonzepte von  $C$  und  $\mathcal{T}$  (um Endlichkeit zu erreichen)

### 3.3.4 Typ

**Definition 3.12.** Teilkonzepte

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

### T3.10

Beispiele:

$$\text{sub}(\forall r. \exists r. (A \sqcap B)) = \{A, B, A \sqcap B, \exists r. (A \sqcap B), \forall r. \exists r. (A \sqcap B)\}$$

$$\text{sub}(\{A \sqsubseteq r.B, \forall r.B \sqsubseteq A\}) = \{A, B, \exists r.B, \forall r.B\}$$

**Lemma 3.13.**  $|\text{sub}(C, T)| \leq |C| + |T|$

**Definition 3.14.** Typ von  $d$

Sei  $\mathcal{I}$  eine Interpretation,  $d \in \Delta^{\mathcal{I}}$ . Der Typ  $t_{\mathcal{I}}(d)$  von  $d$  in  $\mathcal{I}$  ist

$$t_{\mathcal{I}}(d) = \{D \in \text{sub}(C, T) \mid d \in D^{\mathcal{I}}\}$$

Erklärung: Alle Teilkonzepte von  $\mathcal{T}$  und  $C$ , die ein Objekt  $d$  erfüllt.

### T3.11

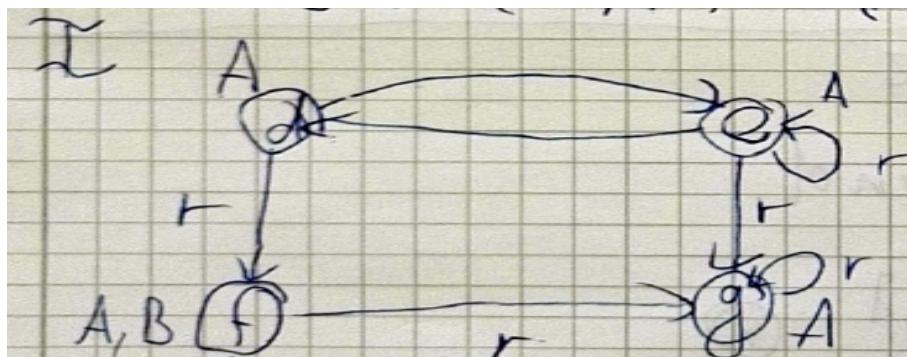
Beispiel:

Sei

$$\mathcal{T} = \{A \subseteq \exists r.A\}$$

$$C = A \sqcap B$$

$$\text{sub}(C, \mathcal{T}) = \{A, \exists r.A, B, A \sqcap B\}$$



$$t_{\mathcal{I}}(d) = \{A, \exists r.A\}$$

$$t_{\mathcal{I}}(e) = \{A, \exists r.A\}$$

$$t_{\mathcal{I}}(f) = \{A, \exists r.A, A \sqcap B, B\}$$

$$t_{\mathcal{I}}(g) = \{A, \exists r.A\}$$

**Lemma 3.15.** Für jede Interpretation  $\mathcal{I}$  gilt:  $\#\{t_{\mathcal{I}}(d) \mid d \in \Delta^{\mathcal{I}}\} \leq 2^{|C|+|\mathcal{T}|}$

### 3.3.5 Filtration

Idee:

- Gegeben Interpretation  $\mathcal{I}$ , identifiziere alle Elemente gleichen Typs
- Danach kommt also jeder Typ nur einmal vor
- Nach Lemma 3.15 gibt es nur  $2^{|C|+|\mathcal{T}|}$  viele Typen
- Wenn  $\mathcal{I}$  Modell von  $C$  und  $\mathcal{T}$ , so auch das Resultat.

**Definition 3.16.** Filtration

Sei  $\mathcal{I}$  Interpretation. Definiere Äquivalenzrelation  $\sim$  auf  $\Delta^{\mathcal{I}}$ :

$$d \sim e \text{ gdw. } t_{\mathcal{I}}(d) = t_{\mathcal{I}}(e)$$

Wir bezeichnen diese Äquivalenzklasse von  $d \in \Delta^{\mathcal{I}}$  bzgl.  $\sim$  mit  $[d]$ .

Die Filtration von  $\mathcal{I}$  bzgl.  $C$  und  $\mathcal{T}$  ist folgende Interpretation  $\mathcal{J}$ :

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

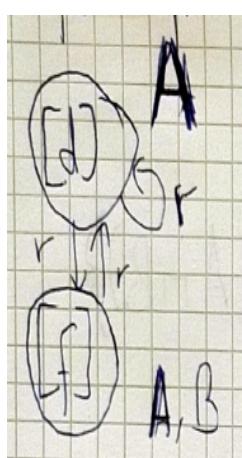
Beachte:  $A^{\mathcal{J}}$  ist wohldefiniert (Repräsentantenunabhängigkeit)

### T3.11cont

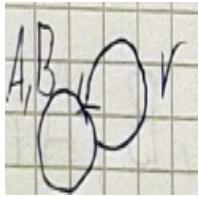
Wenden wir diese Definition auf das Beispiel T3.11 an.

$$\begin{aligned}[d] &= \{d, e, g\} \\ [f] &= \{f\}\end{aligned}$$

Die Interpretation  $\mathcal{J}$  sieht wie folgt aus:



Offensichtlich bringt Filtration aber nicht immer das minimalste Modell, denn folgende Interpretation wäre auch ein Modell:



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

### T3.12

*Beweis.* Beh.: Für alle  $d \in \Delta^{\mathcal{I}}$  und  $D \in sub(C, \mathcal{T})$  gilt:

$$d \in D^{\mathcal{I}} \text{ gdw. } [d] \in D^{\mathcal{J}}$$

Beweis per Induktion über die Struktur von  $D$ .

**I.A.**  $C = A^*$  folgt aus Definition  $A^{\mathcal{J}}$ .

**I.S.**

1.  $\neg, \sqcap$  einfach mittels Semantik und I.A.

2.  $D = \exists r.E$

a. Hinrichtung

$$\begin{aligned} d &\in (\exists r.E)^{\mathcal{I}} \\ \Leftrightarrow &(\text{Semantik}) \text{ es gibt } e \in \Delta^{\mathcal{I}} \text{ mit } (d, e) \in r^{\mathcal{I}} \text{ und } e \in E^{\mathcal{I}} \\ \Rightarrow &(\text{Definition } r^{\mathcal{J}} \text{ und I.V.}) \text{ es gibt } e \in \Delta^{\mathcal{I}} \text{ mit } ([d], [e]) \in r^{\mathcal{J}} \text{ und} \\ &[e] \in E^{\mathcal{J}} \\ \Leftrightarrow &(\text{Semantik } \exists) [d] \in (\exists r.E)^{\mathcal{J}} \end{aligned}$$

b. Rückrichtung

$$\begin{aligned} [d] &\in (\exists r.E)^{\mathcal{J}} \\ \Leftrightarrow &(\text{Semantik } \exists) \text{ es gibt } [e] \in \Delta^{\mathcal{J}} \text{ mit } ([d], [e]) \in r^{\mathcal{J}} \text{ und } [e] \in E^{\mathcal{J}} \\ \Leftrightarrow &(\text{Definition } r^{\mathcal{J}} \text{ und I.V.}) \text{ es gibt } [e] \in \Delta^{\mathcal{J}}, \text{ es gibt } d' \in [d], e' \in [e], \\ &(d', e') \in r^{\mathcal{J}} \text{ und } e' \in E^{\mathcal{J}} \\ \Rightarrow &(\text{Semantik } \exists) d' \in (\exists r.E)^{\mathcal{I}} \\ \Rightarrow &(d \sim d') d \in (\exists r.E)^{\mathcal{I}} \end{aligned}$$

□

Sei  $d \in C^{\mathcal{I}}$ . Nach Behauptung gilt  $[d] \in C^{\mathcal{J}}$ , also ist  $\mathcal{J}$  Modell von  $C$ .

$\mathcal{J}$  ist ebenfalls Modell von  $\mathcal{T}$ :

Sei  $C \sqsubseteq D$  in  $\mathcal{T}$ ,  $[d] \in C^{\mathcal{J}}$

Nach Beh. gilt  $d \in C^{\mathcal{I}}$

Weil  $\mathcal{I}$  Modell von  $C \sqsubseteq D$ , gilt  $d \in D^{\mathcal{I}}$

Nach Beh. gilt  $[d] \in D^{\mathcal{J}}$

### 3.3.6 Endliche/Beschränkte Modelleigenschaft

#### Theorem 3.11

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

Beweis. Folgt aus Theorem 3.17 und Lemma 3.15.

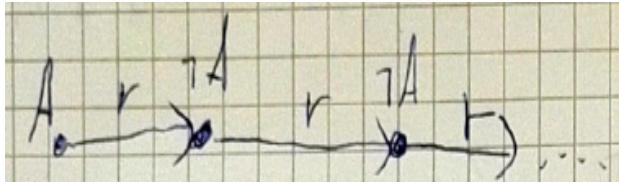
Ähnliches (mit der selben Schranke) lässt sich für  $\mathcal{ALCI}$  und  $\mathcal{ALCQ}$  beweisen. (mit derselben Schranke)

**Theorem 3.18.**  $\mathcal{ALQT}$  hat nicht die endliche Modelleigenschaft.

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

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

#### T3.13



Erklärung: Diese Interpretation müsste unendlich erweitert werden, damit es die TBox erfüllt.

### 3.3.7 Entscheidbarkeit

#### Theorem 3.11:

Wenn  $C$  erfüllbar bzgl.  $\mathcal{T}$ , dann haben  $C$  und  $\mathcal{T}$  Modell der Größe  $\leq 2^{|C|+|\mathcal{T}|}$ .

Erfüllbarkeit ist also entscheidbar:

Gegeben  $C$  und  $\mathcal{T}$ , so dass  $|C| + |\mathcal{T}| = n$ ,

- erzeuge alle Interpretationen  $\mathcal{I}$  mit  $|\Delta|^{\mathcal{I}} \leq 2^n$  (es gibt höchstens  $2^{2^{5n}}$  viele davon)
- und überprüfe, ob  $\mathcal{I}$  Modell von  $C$  und  $\mathcal{T}$  ist. (in Zeit polynomiell in  $\mathcal{I}$ ,  $C$ , und  $\mathcal{T}$ )

**Lemma 3.19.** Gegeben sei ein  $\mathcal{ALC}$ -Konzept  $C$  und endliche Interpretation  $\mathcal{I}$ . Man kann in polynomieller Zeit – genauer in Zeit  $O(|C| \cdot |\Delta^{\mathcal{I}}|)$  – die Extension  $C^{\mathcal{I}}$  berechnen.

Beweisskizze zu Lemma 3.19: Rekursiver Algorithmus über die Definition der Konzeptsemantik. Dessen Zeitaufwand ist  $O(|C| \cdot |\Delta^{\mathcal{I}}|)$ :

- Anzahl der (rekursiven Aufruf =  $|sub(C)| \leq |C|$ )
- pro Aufruf Zeitaufwand  $O(|\Delta^{\mathcal{I}}|)$ :  
simple Operationen auf  $\leq 2$  Teilmengen von  $\Delta^{\mathcal{I}}$

**Korollar 3.20.** Gegeben seien  $C$ ,  $\mathcal{T}$  in  $\mathcal{ALC}$  und endliche Interpretation  $\mathcal{I}$ . Man kann in polynomieller Zeit – genauer: in Zeit  $O((|\mathcal{T}| + |C|) \cdot |\Delta^{\mathcal{I}}|)$  – entscheiden, ob  $\mathcal{I}$  ein Modell von  $C$  und  $\mathcal{T}$  ist.

**Theorem 3.21.** In  $\mathcal{ALC}$  ist Erfüllbarkeit bzgl. TBoxen entscheidbar.

Die Komplexität liegt aber bei 2-ExpTime: 2-exponentiell viele Interpretationen müssen geprüft werden, jede Prüfung braucht polynomielle Zeit.

Dieser Ansatz ist kaum tauglich für die Praxis.

## 4 Tableau-Algorithmen

### 4.1 Ziel

Automatisches Schlussfolgern spielt eine zentrale Rolle für BLen. Insbesondere ist die Ausdruckstärke von BLen stark darauf zugeschnitten.

Dabei ist aber wichtig, dass die relevanten Schlussfolgerungsprobleme entscheidbar sind, sie eine möglichst geringe Komplexität haben und/oder Algorithmen existieren, die sich in der Praxis performant verhalten.

Von uns wird hauptsächlich das Problem der Erfüllbarkeit betrachtet.

In der Praxis haben sich hauptsächlich Tableau-Algorithmen und Resolutionsverfahren als effizient herausgestellt.

### 4.2 ALC ohne TBoxen

#### 4.2.1 Negationsnormalform

**Definition 4.1.** Negationsnormalform

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

**Lemma 4.2.** Jedes Konzept kann in Linearzeit in ein äquivalentes Konzept in NNF umgewandelt werden.

#### T4.1

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

#### 4.2.2 I-Baum

**Definition 4.3.** I-Baum

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

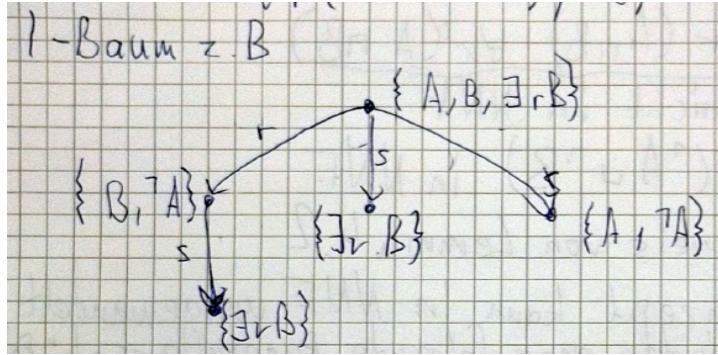
- $V$  Knotenmenge
- $E$  ist Menge beschrifteter Kanten  $(v, r, v')$  mit  $v, v' \in V$ ,  $r$  Rollenname
- $L: V \rightarrow 2^{sub(C_0)}$

#### T4.2

Bsp.

$$C_0 = A \sqcap \forall r. (\neg A \sqcap \exists r. B)$$

$$sub(C_0) = \{A, \neg A, B, \exists r. B, \neg A \sqcap \exists r. B, \forall r. (\neg A \sqcap \exists r. B)\}$$



#### 4.2.3 Tableau-Algorithmus

Berechnet Folge

$$M_0, M_1, \dots$$

von Mengen von I-Bäumen:

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

- $V := \{v_{\text{ini}}\}$
- $E := \emptyset$
- $L := (v_{\text{ini}}) \{C_0\}$

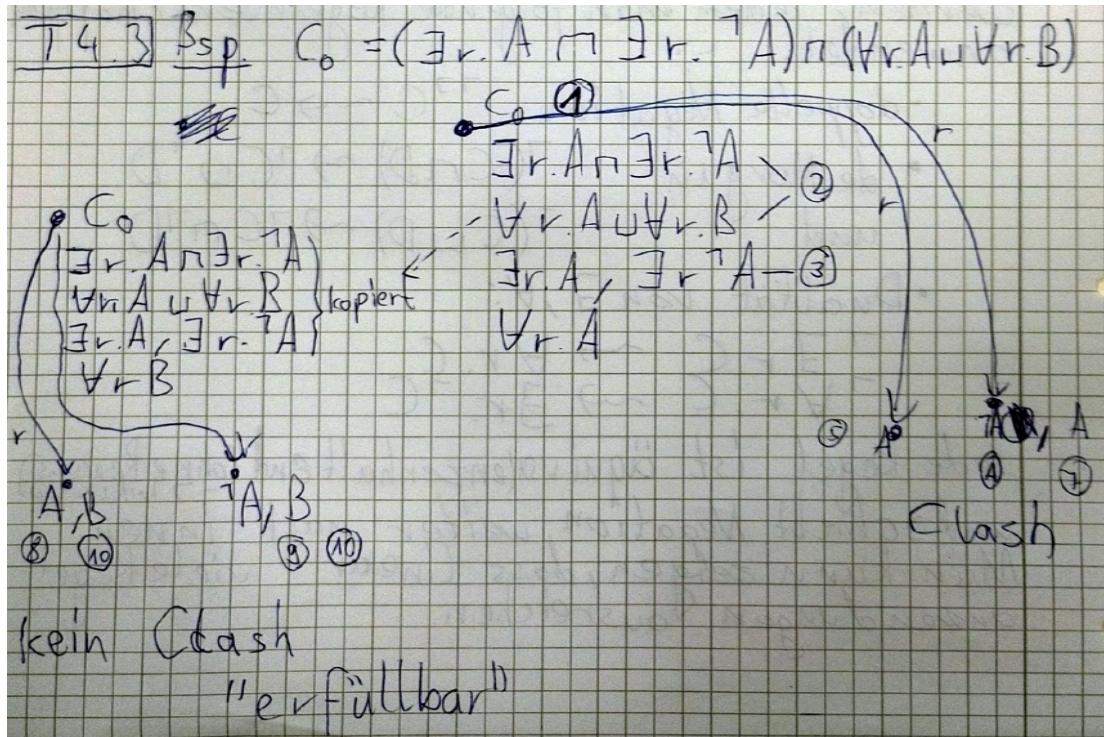
$M_{i+1}$  entsteht aus  $M_i$  durch Anwendung der Tableau-Regeln auf irgendeinen I-Baum in  $M_i$  und anschließendes Austauschen des verwendeten Baumes durch den neu erzeugten (Sei  $(V, E, L)$  I-Baum):

- $\Box$ -Regel
  - Wähle  $v \in V$  und  $C \Box D \in L(v)$  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  $\{C, D\} \cap L(v) = \emptyset$
  - 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.

### T4.3

Beispiel:



#### 4.2.4 Definition Rollentiefe

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

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

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

#### 4.2.5 Multimengen

Multimengen sind Mengen, in denen Elemente mehrfach vorkommen dürfen, z.B.:

$$\{1, 1, 2, 3, 4, 4, 5, 6, 6, 6\}$$

Formal: Multimengen über die Menge  $S$  ist Abbildung

$$M : S \rightarrow \mathbb{N}$$

, welche jedes Element auf die Anzahl seines Vorkommens abbildet.

Die meisten Begriffe übertragen sich von Mengen auf Multimengen:

- Leere Menge  $\emptyset : s \mapsto 0$  für alle  $s \in S$
- Vereinigung  $(M_1 \cup M_2)(s) := M_1(s) + M_2(s)$

- Element:  $s \in M$  gdw.  $M(s) > 0$
- Differenz:  $(M_1 \setminus M_2)(s) = X(m, n) = \begin{cases} M_1(s) - M_2(s) & , \text{ wenn } M_1(s) \geq M_2(s) \\ 0 & \text{sonst} \end{cases}$

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

Gegeben strikte partielle Ordnung  $(S, <)$ , ist die *Multimengenerweiterung*  $(MM(S), <_{\text{mul}})$  definiert als:

$M_2 <_{\text{mul}} M_1$  gdw.  $\exists X, Y \in MM(S)$ , so dass

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

Also erhält man  $M_2$  aus  $M_1$ , indem man einige Elemente entfernt und durch endlich viele *kleinere* ersetzt.

Beispiel:

$$\{3, 1\} >_{\text{mul}} \{2, 2, 2\} >_{\text{mul}} \{2, 2\} >_{\text{mul}} \{2, 1, 1, 1\}$$

Es ist leicht zu zeigen das diese Ordnung eine strikte partielle Ordnung ist. Zudem ist sie wohlfundiert, wenn  $(S, <)$  wohldefiniert ist: Es gibt keine unendlich  $<$  absteigenden Ketten.

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

#### 4.2.6 Terminierung

**Proposition 4.5.** Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis in 4 Schritten:

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

##### T4.5a

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

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

##### T4.5b

*Beweis.* Induktion über die Anzahl der Regelanwendungen. Zu zeigen: Wenn  $v$  Knoten mit Tiefe  $\mathcal{I}$  ist, dann gilt  $\text{rd}(C) \leq \text{rd}(C_0) - i$  für alle  $C \in L(v)$ .

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

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

a)  $\Box$ -Regel

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

b)  $\exists$ -Regel

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

□

3. Sei  $M_0, M_1, \dots$  die erzeugte Folge und  $B \in M_{\mathcal{I}}$  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).

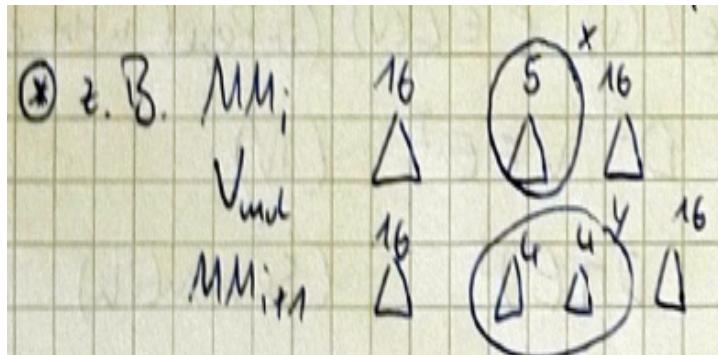
Nun kann die Terminierung mittel Behauptung 3 beweisen:

#### T4.5d

*Beweis.* Wir ordnen jedem  $M_i$  eine Multimenge  $MM_i$  zu. Für jedes  $B \in M_i$  enthält  $MM_i$  die Zahl der "n-Anzahl der Regelanwendungen, mittels derer  $B$  generiert wurde."

Weil  $<$  auf  $\mathcal{N}$  wohldefiniert ist, ist  $<_{mul}$  auf  $MM(\{0, \dots, n\})$  wohldefiniert.

Offenbar gilt  $MM_{i+1} <_{mul} MM_i$  für alle  $i \geq 0$  \*. Also werden nur endlich oft Regeln angewendet. □



#### 4.2.7 Korrektheit und Vollständigkeit

**Proposition 4.7.** Wenn der Tableau-Algorithmus „erfüllbar“ zurückgibt, so ist  $C_0$  erfüllbar.

#### T4.6

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

- $\Delta^{\mathcal{I}} = V$
- $r^{\mathcal{I}} = \{(v, v') \mid (v, r, v') \in E\}$  für alle Rollennamen  $r$

- $A^{\mathcal{I}} = \{v \mid A \in L(v)\}$  für alle Konzeptnamen  $A$

Behauptung: Für alle Konzepte  $C$  und  $v \in V$  gilt

$$C \in L(v) \text{ impliziert } v \in C^{\mathcal{I}}$$

Da  $C_0 \in L(v_{\text{ini}})$  in  $B_{\text{ini}}$  gilt auch  $C_0 \in L(v_{\text{ini}})$  in  $B$ . Also  $v_{\text{ini}} \in C_0^{\mathcal{I}}$  nach Behauptung, weswegen dann  $C_0$  erfüllbar.

**I.A.**  $C = A$  (Konzeptname) Gilt nach Definition von  $\mathcal{I}$ .

**I.S.**

- $C = \neg A$

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

- $C = D \sqcap E$

$$\begin{aligned} & C \in L(v) \\ \Rightarrow & (\sqcap\text{-Regel nicht anwendbar}) D \in L(v), E \in L(v) \\ \Rightarrow & (\text{I.V.}) v \in D^{\mathcal{I}}, v \in E^{\mathcal{I}} \\ \Rightarrow & (\text{Semantik}) v \in (D \sqcap E)^{\mathcal{I}} \end{aligned}$$

- $C = D \sqcup E$

analog.

- $C = \exists r.D$

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

Nach I.V.:  $v' \in D^{\mathcal{I}}$ ; nach Konstruktion  $(v, v') \in r^{\mathcal{I}}$ . Nach Semantik gilt dann:  $v \in (\exists r.D^{\mathcal{I}})$

- $C = \forall r.D$

ähnlich.

□

**Definition 4.8.** Realisierbarkeit

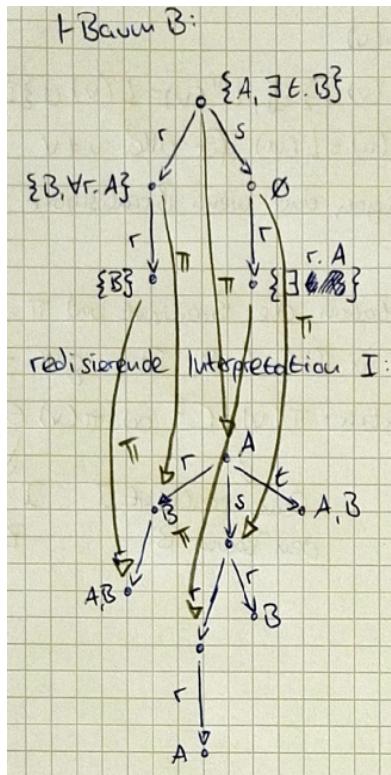
Sei  $B = (V, E, L)$  ein I-Baum. Interpretation  $\mathcal{I}$  realisiert  $B$  gdw. es gibt eine Funktion  $\pi : V \rightarrow \Delta^{\mathcal{I}}$  so dass

- $(v, r, v') \in E$  impliziert  $(\pi(v), \pi(v')) \in r^{\mathcal{I}}$
- $C \in L(v)$  impliziert  $\pi(v) \in C^{\mathcal{I}}$

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

Beachte: realisierbarer I-Baum enthält keinen offensichtlichen Widerspruch!

#### T4.7



#### Proposition 4.9. Vollständigkeit

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

#### T4.8

*Beweis.* Per Induktion über  $\mathcal{I}$ .

Sei  $C_0$  erfüllbar. Nach Proposition 4.5 berechnet der Algorithmus endlich Folge  $M_0, \dots, M_n$ . 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.**  $i = 0$ .  $M_0 = \{B_{\text{ini}}\}$ .  $B_{\text{ini}}$  ist realisierbar, weil  $C_0$  erfüllbar.

**I.S.** Fallunterscheidung gemäß der Regel, mit der  $M_{i+1}$  aus  $M_{\mathcal{I}}$  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  $B' = (V, E, L') \in M_{i+1}$  und  $B'' = (V, E, L'') \in M_{i+1}$  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  $B$  realisierbar, dann  $B'$  oder  $B''$  realisierbar.

Sei  $\mathcal{I}$  Interpretation, die  $B$  realisiert und  $\pi : V \rightarrow \Delta^{\mathcal{I}}$  Abbildung wie in Definition 4.8. Dann gilt  $\pi(v) \in (C \sqcup D)^{\mathcal{I}}$ . Nach Semantik:  $\pi(v) \in C^{\mathcal{I}}$  oder  $\pi(v) \in D^{\mathcal{I}}$ . Also realisiert  $\mathcal{I}$  den Baum  $B'$  oder  $B''$ .

□

#### 4.2.8 Komplexitätsanalyse

Wir beobachten:

I-Bäume können höchstens exponentiell groß werden.

Dieser Fall kann tatsächlich eintreffen. Beispielhaft, der Erfüllbarkeitstest von:

$$\prod_{i < n} \forall r^i. (\exists r. B \sqcap \exists r. \neg B)$$

generiert Baum der Größe  $2^n$ .

Also: exponentieller Zeit- und Platzverbrauch (sogar 2-exponentiell)

#### 4.2.9 Praktikabilität

Offenbar wäre eine naive Implementierung nicht effizient. Dabei kann man aber einige Hinweise/Optimierungen bei der Implementierung beachten:

- Es wird nur ein Baum zur Zeit generiert, keine Menge
- bei der  $\sqcup$ -Regel muss man sich also entscheiden (Heuristik); ggf. Entscheidung revidieren (Backtracking).
- Es wird nur ein Teil des Baumes (Pfad) im Speicher gehalten.
- Backjumping: Führe Buch über die “Herkunft” von Knotenbeschriftungen und Kanten mittels Dependenzmengen. Wenn Backtracking nötig, springe direkt zu einer der Ursachen des Widerspruches zurück.

### 4.3 ALC mit generellen TBoxen

Nun wollen wir einen Tableau-Algorithmus für die Erfüllbarkeit in  $\mathcal{ALC}$  bzgl. TBoxen.

Jede TBox  $\mathcal{T}$  ist äquivalent zu einer TBox der Form  $\{\top \sqsubseteq C_{\mathcal{T}}\}$ :

$$\text{setze } C_{\mathcal{T}} := \prod_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D.$$

**T4.11** Beispiel

$$\mathcal{T} = \{A \sqsubseteq \exists r. B, A \sqcup B \sqsubseteq \forall r. B\}$$

Daraus wird

$$\{\top \sqsubseteq (\neg A \sqcup \exists r.B) \sqcap (\neg(A \sqcup B) \sqcup \forall r.B)\}$$

in NNF:

$$\mathcal{T}' = \{\top \sqsubseteq (\neg A \sqcup \exists r.B) \sqcap ((\neg A \sqcap \neg B) \sqcup \forall r.B)\}$$

Desweiteren nehmen wir an, dass:

- Eingabe  $C_0$  in NNF;
- Eingabe  $\mathcal{T}$  hat Form  $\{\top \sqsubseteq C_{\mathcal{T}}\}$  mit  $C_{\mathcal{T}}$  in NNF

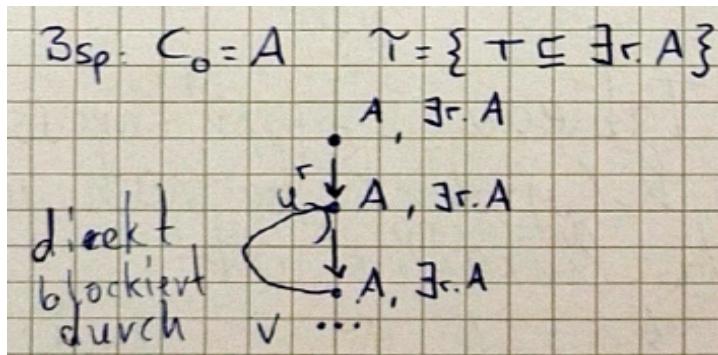
Nun modifiziere den vorigen Algorithmus durch Hinzufügen folgender Regel:

### 4.3.1 TBox-Regel

Wähle  $v \in V$  so dass  $C_{\mathcal{T}} \notin L(v)$  und erweitere  $L(v)$  um  $C_{\mathcal{T}}$ .

Problem: Terminiert nicht!

#### T4.12



### 4.3.2 Blockieren

Dies lösen wir, indem wir nur ein endliches Anfangsstück eines Baummodells anhand dessen sich die Existenz eines vollständigen Modells entscheiden lässt konstruieren. Dazu müssen wir die Anwendung der  $\exists$ -Regel einschränken.

**Definition 4.10.** Blockiert

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

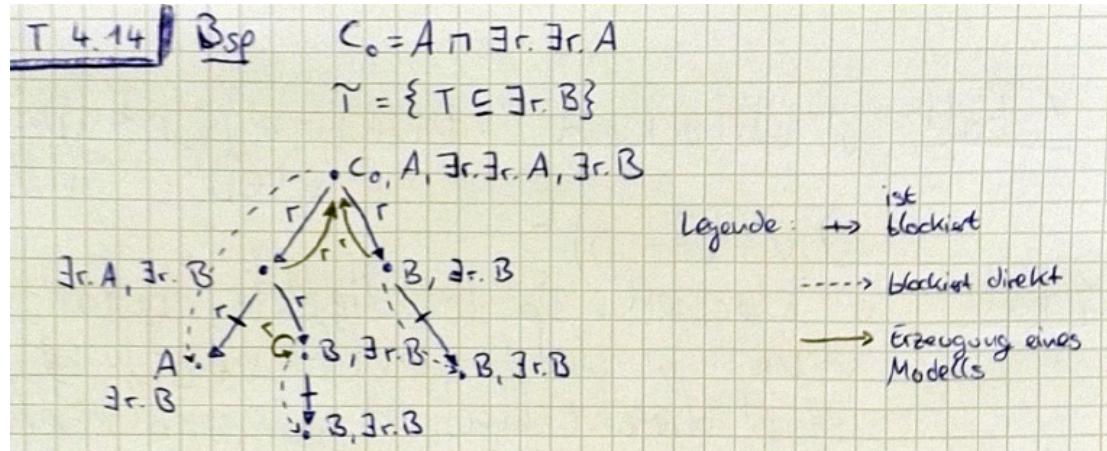
1.  $u$  Vorgänger von  $v$  in  $B$  ist und
2.  $L(v) \subseteq L(u)$

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

### 4.3.3 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  $L(v') = \{C\}$



#### 4.3.4 Vollständigkeit

**Proposition 4.11.** Wenn  $C_0$  erfüllbar bzgl.  $\mathcal{T}$ , so gibt der Algorithmus erfüllbar zurück.

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

#### 4.3.5 Korrektheit

**Proposition 4.12.** Wenn der Algorithmus “erfüllbar” zurückgibt, so ist  $C_0$  erfüllbar bzgl.  $\mathcal{T}$

#### T4.15

Beweisskizze per Induktion über die Struktur von  $C$ . Definiere Interpretation  $\mathcal{I}$ :

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

Behauptung: Für alle ALC-Konzepte  $C$  und  $v \in \Delta^{\mathcal{I}}$  gilt:

$$C \in L(v) \Rightarrow v \in C^{\mathcal{I}}$$

Die Behauptung impliziert wie gewünscht, dass

- $\mathcal{I}$  Modell von  $\mathcal{T}$  ist.

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

- $\mathcal{I}$  Modell von  $C_0$  ist.

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

**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  $(v, r, v') \in E$  und  $D \in L(v)$ . Fallunterscheidung:

1.  $v'$  unblockiert. Dann  $(v, v') \in r^{\mathcal{I}}$  (Definition  $\mathcal{I}$ ),  $v' \in D^{\mathcal{I}}$  (**I.V.**)  $\Rightarrow v \in (\exists r.D)^{\mathcal{I}}$
  2.  $v'$  blockiert. Da der direkte Vorgänger  $v$  von  $v'$  unblockiert ist, ist  $v'$  direkt blockiert von unblockiertem Vorgänger  $u$ . Es gilt:
    - $(v, u) \in r^{\mathcal{I}}$  nach Definition  $r^{\mathcal{I}}$
    - $D \in L(v) \subseteq L(u)$  (Blockierungsbedingung)
    - $\Rightarrow u \in D^{\mathcal{I}}$  (**I.V.**)

Also  $v \in (\exists r.D)^{\mathcal{I}}$ .
- $C = \forall r.D$

Ähnlich zu oberem Fall.

#### 4.3.6 Terminierung

**Proposition 4.13.** Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis analog zu den ohne TBoxen (Prop. 4.5), aber mit Einbezug der TBox. Wir zeigen es also in den selben Schritten:

Beweis in 4 Schritten:

1. Es werden nur I-Bäume mit einem Verzweigungsgrad  $\leq |C_0| + |\mathcal{T}|$  generiert.
2. Es werden nur I-Bäume mit einer Tiefe  $2^k$  generiert.

#### T4.16

Angenommen, es wird ein I-Baum der Tiefe  $> 2^k$  erzeugt.

Dann wird irgendwann die  $\exists'$ -Regel auf einen Knoten  $v$  der Tiefe  $2^k$  angewendet.

Betrachte Pfad  $v_0, \dots, v_{2^k}$  von der Wurzel bis  $v$ . Dieser Pfad hat  $2^k + 1$  Knoten.

Weil es nur  $2^k$  möglich Knotenbeschriftungen gibt, muss es auf dem Pfad zwei Knoten  $v_i$  und  $v_j$  geben, mit  $0 \leq i < j \leq 2^k$ , welche dieselben Knotenbeschriftungen haben, also  $L(v_i) = L(v_j)$ . Also ist  $v_j$  durch  $v_i$  blockiert, weswegen auch  $v$  blockiert ist.  $\sharp$

Widerspruch zur Anwendung der  $\exists'$ -Regel auf  $v_j$ , also ist die Annahme falsch.

3. Sei  $M_0, M_1, \dots$  die erzeugte Folge und  $B \in M_{\mathcal{I}}$  für ein  $i \geq 0$ . Dann ist  $B$  durch die Anwendung von maximal  $k^{2^k} \cdot k \leq 2^{2^{3k}} = n$  Regeln entstanden (Knoten im Baum mal Größe Knotenbeschriftung).

Danach kann Terminierung wie gehabt mittels Behauptung 3 bewiesen werden.

#### 4.3.7 Komplexität

Im Beweis zum 3. Schritt der Terminierung haben wir gesehen, dass die I-Bäume höchstens doppelt exponentiell groß werden.

Dieser Worst-Case kann eintreten!

**Lemma 4.14.** Es gibt Eingabe  $C_0, \mathcal{T}$  für die der Tableau-Algorithmus einen Baum von exponentieller Tiefe generiert.

Also: 2-exponentieller Zeit- und Platzaufwand (sogar 3-exponentiell!).

#### 4.3.8 Bemerkung zur TBox-Regel

#### 4.3.9 Erweiterungen

## 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  $t \subseteq \text{sub}(C_0, T)$ , so dass

1.  $A \in t$  gdw.  $\neg A \notin t$  für alle  $\neg A \notin \text{sub}(C_0, T)$
2.  $C \sqcap D \in t$  gdw.  $C \in t$  und  $D \in t$  für alle  $C \sqcap D \in \text{sub}(C_0, T)$
3.  $C \sqcup D \in t$  gdw.  $C \in t$  oder  $D \in t$  für alle  $C \sqcup D \in \text{sub}(C_0, T)$
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  $\Gamma$  Typenmenge und  $t \in \Gamma$ . Dann ist  $t$  schlecht in  $\Gamma$ , wenn für ein  $\exists r.C \in t$  gilt: Es gibt kein  $t' \in \Gamma$  mit  $\{C\} \cup \{D \mid \forall r.D \in t\} \subseteq t'$ .

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

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

Zu zeigen:  $I$  ist Modell von  $C_0$  und  $T$ .

Behauptung: Für alle  $C \in \text{sub}(C_0, T)$  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  $t \in (D')^I$ . Nach Semantik:  $t \in (D \sqcap D')^I$ .

- $C = D \sqcup D'$

Analog.

- $C = \forall r.D$

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

- $C = \exists r.D$

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  $(t, t') \in r^I$ . Also  $t \in (\exists r.D)^I$

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

I.A.  $i = 0$ . Einfach jedes Element von  $\Gamma$  (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  $\Gamma_i$ . Sei  $\exists r.C \in t$  und  $S = \{C\} \cup \{D \mid \forall r.D \in T\}$ . Da  $t \in \Gamma$ , gibt es  $d \in \Delta^I$  mit  $t_i(d) = t_0$ . 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  $S \subseteq t_I(e)$  und  $t_I(e) \in \Gamma \subseteq \Gamma_i$ . 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  $\varphi$
- Jede Variable in  $\varphi$  gehört entweder Spieler 1 oder Spieler 2
- Das Spiel beginnt auf einer gegebenen Anfangsbelegung  $\pi_0$  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  $\varphi$  jemals wahr wird (egal, welcher Spieler gezogen hat)
- Spieler 2 gewinnt, wenn das Spiel unendlich weitergeht ohne dass  $\varphi$  wahr wird

### 5.2.2 Definition 5.7

- *Spiel*: Tupel  $(\varphi, \Gamma_1, \Gamma_2, \pi_0)$  mit  $\Gamma_1, \Gamma_2$  Partitionierung der Variablen in  $\varphi_1$  und  $\varphi_2$  Anfangsbelegung
- *Konfiguration*: Paar  $(i, \pi)$  mit  $i \in \{1, 2\}$  aktiver Spieler und  $\pi$  Belegung
- $\pi$  ist  $j$ -Variation von  $\pi'$  ( $j \in \{1, 2\}$ ) wenn  $\pi = \pi'$  oder  $\pi$  und  $\pi'$  unterscheiden sich nur in einer Variablen  $p \in \Gamma_j$

„ $\pi$  ist  $j$ -Variation von  $\pi'$ “ bedeutet: Spieler  $j$  kann  $\pi$  in  $\pi'$  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  $l(v') = (1, \pi')$ , wobei  $\pi'$  2-Variation von  $\pi$
- c) wenn  $l(v) = (1, \pi)$ , dann hat  $v$  Nachfolger  $v_0, \dots, v_{|\Gamma_1|}$  mit  $l(v_i) = (2, \pi_i)$  wobei  $\pi_0, \dots, \pi_{|\Gamma_1|}$  alle existierenden 1-Variationen von  $\pi$
- d) wenn  $l(v) = (i, \pi)$ , dann nicht  $\pi \models \varphi$

### 5.2.4 Definition 5.9

*Spiel<sub>1</sub>* 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

$\text{Spiel}_1$  ist ExpTime-Vollständig

### 5.2.6 Reduktion

Reduziere  $\text{Spiel}_1$ : 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:  $\text{sub}_i(C_0) \{C \in \text{sub}(C_0) \mid \text{rd}(C) \leq i\}$ .

**Definition 6.16 (i-Typ)** Sei  $i \geq 0$ .  $i$ -Typ für  $C_0$  ist Teilmenge  $t \subseteq \text{sub}_i(C_0)$  so, dass

1.  $A \in t$  gdw.  $\neg A \notin t$  für alle  $\neg A \in \text{sub}_i(C_0)$

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

Korrekteitsbeweis 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  $v \in (D')^I$ . Nach Semantik:  $v \in (D \sqcap D')^I$ .

- $C = D \sqcup D'$

Analog.

- $C = \forall r.D$

Sei  $(v, v') \in r^I$ . Dann ist  $(v, v') \in E$  und  $\sigma(v') = r$ . Wegen  $\forall r.D \in p_1(v)$  ist auch  $D \in p_1(v')$ . 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  $\varphi$  mit Variablen  $p_1, \dots, p_n$ ,  $n$  gradzahlig
- *Konfiguration*: Wort  $\pi \in \{0, 1\}^*$

### 5.4.2 Definition 5.21

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

- a) Wurzel beschriftet mit  $\varepsilon$  (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  $w \models \varphi$

### 5.4.3 Theorem 5.25

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

Beweisskizze. Konstruiere Konzept  $C_\varphi$ , so dass Spieler 1 hat Gewinnstrategie in  $\varphi$  gdw.  $C_\varphi$  erfüllbar.

## 5.5 Unentscheidbare Erweiterungen

## 5.6 Konkrete Bereiche

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

- $\Delta^B$  eine Menge von *Werten* ist und
- $\Phi^B$  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  $\text{ALC}(B)$  bezeichnen wir die Erweiterung von ALC um  $B$ , d.h. um

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

wobei  $P \in \Phi^B$   $n$ -Stellig ist und die  $R_i$  Rollenkomposition der Form  $r_1; \dots; r_k; f$  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  $f^I : \Delta^I \rightarrow \Delta^B$  zu. Für jede Rollenkomposition  $r = r_1; \dots; r_k; f$  bezeichnet  $R^I$  die Komposition der Interpretationen:  $R^I = r_1^I \circ \dots \circ r_k^I \circ f$

Die Semantik der zusätzlichen Konstruktoren ist nun:

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

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

### 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  $Q = \{q_0, \dots, q_l\}$  Menge von Zuständen und  $P = I_0, \dots, I_{l-1}$  Instruktionsfolge. Per Definition ist  $q_0$  Startzustand,  $q_l$  Stopzustand. Jede Instruktion ist  $I_i$  hat eine der folgenden Formen:

- $I_i = +(p, q_j)$  mit  $p \in \{1, 2\}$  Register und  $q_j$  Folgezustand: Inkrementierungsanweisung
- $I_i = -(p, q_j, q_k)$  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  $M$ , konstuiere  $ALC(B_1)$ -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  $\rho$  eine Simulation zwischen  $I_1$  und  $I_2$  mit  $d_1 \rho d_2$ .

**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)

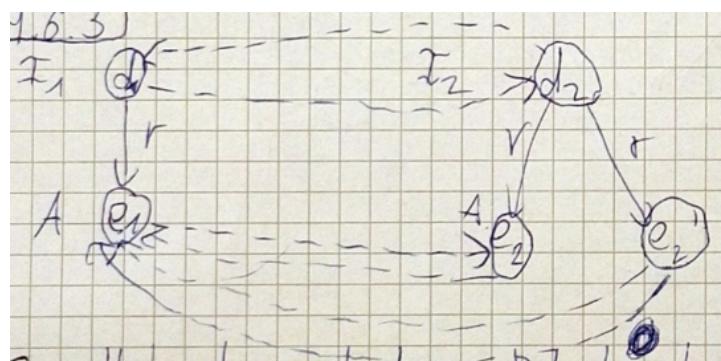
1.  $C = \exists r.D$

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.

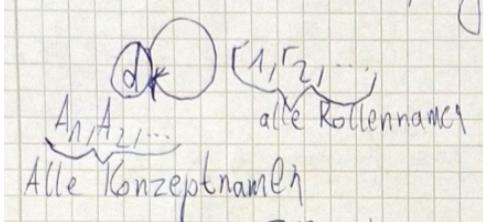
Beweisskizze: Zeige Wechselseitige Simulation, die keine Biosimulation ist:



### 6.2.3 Lemma 6.6

Jedes EL-Konzept ist erfüllbar bzgl. jeder TBox.

Beweisskizze per Induktion über die Struktur von  $C$ :



## 6.3 Subsumption ohne TBox

Eine Subsumption  $C \sqsubseteq D$  gilt in EL im Prinzip gdw man  $D$  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  $\varsigma$
  - 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;  $T \models C \sqsubseteq D$  gdw  $T' \models A_C \sqsubseteq A_D$ .

### 6.4.2 Normalform

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

$$A_1 \sqcap \dots \sqcap A_n \sqsubseteq A \quad A \sqsubseteq \exists r.A_1 \quad \exists r.A \sqsubseteq A_1$$

### 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:  $T \models A \sqsubseteq B$  gdw  $T' \models A \sqsubseteq B$ . 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:

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

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

**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:  $T \models A \sqsubseteq B$  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  $T = T_0, \dots, (T_n = T^*)$  die durch Regelanwendungen erzeugte Folge von TBoxen. Es genügt zu zeigen:  $T_i \models T_{i+1}$ . Zeige dies durch Vorbedingung der Regelanwendung und der Semantik.

#### Vollständigkeit

**Kanonische Interpretation** Die Kanonische Interpretation  $I$  ist:

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

Erklärung: Konstruiere Intuitiv Modell aus allen Konzeptnamen.

**Lemma 6.18** Die kanonische Interpretation ist ein Modell 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  $T \models A \sqsubseteq B$ .

## 6.5 Erweiterungen von EL

### 6.5.1 EL mit Disjunktion und Bottom

Erfüllbarkeit in  $\text{ELU}_\perp$  (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  $X \sqcap X \sqsubseteq \perp$

### 6.5.2 ELU (mit Disjunktion)

Erfüllbarkeit in  $\text{ELU}$  ist ExpTime-Vollständig.

Beweisskizze per Reduktion von 6.5.1. Ersetzte  $\perp$  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://www.informatik.uni-bremen.de/tdki/lehre/ss16/b1/>. Last accessed on 2016-12-04.