

Struktur Vorlesung

- Kapitel 1: Einleitung
- Kapitel 2: Grundlagen
- Kapitel 3: Ausdrucksstärke und Modellkonstruktionen
- Kapitel 4: Tableau Algorithmen
- Kapitel 5: Komplexität
- Kapitel 6: ABoxen und Anfragebeantwortung
- Kapitel 7: Effiziente Beschreibungslogiken

Grundlagen

Die Beschreibungslogik *ALC*

ALC

Es gibt viele verschiedene Beschreibungslogiken:

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

Hier zunächst *ALC*:

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

ALC

Von nun an sei

- \mathbf{N}_C eine unendliche Menge von Konzeptnamen
Diese bezeichnen Klassen / unäre Prädikate
z.B. Person, Kurs, Universität, Tafel, Student, etc.
- \mathbf{N}_R eine unendliche Menge von Rollennamen
Diese bezeichnen Relationen / binäre Prädikate
z.B. hört, lehrt, istTeilVon, etc.

Wir nehmen \mathbf{N}_C und \mathbf{N}_R als disjunkt und abzählbar unendlich an

Beachte Gross- / Kleinschreibung!

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

- Konzeptnamen
- Rollennamen
- Konzeptkonstruktoren (manchmal auch Rollenkonstruktoren)

Es gibt viele verschiedene Konstruktoren

Versch. Konstruktormengen ergeben versch. Beschreibungslogiken

ALC Syntax

Definition 2.1 (*ALC*-Konzepte).

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

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

T2.1

ALC Syntax

Verwendete Symbole:

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

Zwei nützliche Abkürzungen: wir schreiben

- \top für $A \sqcup \neg A$ (Top Konzept)
- \perp für $A \sqcap \neg A$ (Bottom Konzept)

wobei A ein beliebiger Konzeptname ist.

ALC Syntax

Präzedenzregel:

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

Also zum Beispiel:

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

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

Keine Präzedenz zwischen \sqcap und \sqcup : Klammern verwenden!

ALC Semantik

Definition 2.2 (**ALC** Semantik).

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

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

T2.2

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

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

T2.2 cont.

ALC Semantik

Verwendete Symbole:

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

Für Interpretationen verwenden wir übliche Terminologie für Graphen:

- e ist r -Nachfolger von d (in \mathcal{I}) wenn $(d, e) \in r^{\mathcal{I}}$;
- e ist r -Vorgänger von d (in \mathcal{I}) wenn $(e, d) \in r^{\mathcal{I}}$;
- wenn r unwichtig, sprechen wir nur von *Nachfolgern* / *Vorgängern*;
- e ist *erreichbar* von d wenn es d_0, \dots, d_n gibt ($n \geq 0$) mit $d_0 = d$, $d_n = e$ und d_{i+1} ist Nachfolger von d_i für alle $i < n$;
- \mathcal{I} is *endlich* gdw. $\Delta^{\mathcal{I}}$ endlich ist;

T2.2 cont.

Extension/ Modell

Wir nennen

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

Beachte:

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

Mensch \sqcap \exists hatKind. \top

Menschen, die ein nicht näher
spezifiziertes Kind haben

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

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

Mensch \sqcap \forall hatKind. \perp

Menschen, die keine Kinder haben

Erfüllbarkeit, Subsumption, Äquivalenz

Folgende Begriffe aus der Aussagenlogik spielen auch für \mathcal{ALC} eine Rolle:

Definition 2.3 (Erfüllbar, subsumiert, äquivalent)

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

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

T2.3

In der Aussagenlogik sagt man “ D folgt aus C ” statt “ D subsumiert C ”

Erfüllbarkeit, Subsumption, Äquivalenz

Die üblichen aussagenlogischen Äquivalenzen gelten auch in \mathcal{ALC} , z.B.:

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D \quad \text{de Morgansches Gesetz}$$

$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D \quad \text{de Morgansches Gesetz}$$

$$C \sqcap D \equiv D \sqcap C \quad \text{Kommutativität von Konjunktion}$$

$$C \sqcup D \equiv D \sqcup C \quad \text{und Disjunktion}$$

$$C \sqcap (D \sqcap E) \equiv (C \sqcap D) \sqcap E \quad \text{Assoziativität von Konjunktion}$$

$$C \sqcup (D \sqcup E) \equiv (C \sqcup D) \sqcup E \quad \text{und Disjunktion}$$

$$C \sqcap \top \equiv C \quad \text{Neutrales Element Konjunktion}$$

$$C \sqcup \perp \equiv C \quad \text{Neutrales Element Disjunktion}$$

Grundlagen

TBoxen

TBoxen

Zur Erinnerung:

TBoxen (terminologische Boxen)

- definieren Konzepte
- setzen diese zueinander in Beziehung

Konzeptdefinition z.B.

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

Allgemeines Hintergrundwissen / Constraint z.B.

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

TBoxen

Wir führen zwei Arten von TBoxen ein:

- Generelle TBoxen
Ausdrucksstark, aber recht hohe Komplexität beim Schlussfolgern
- Azyklische TBoxen.
Erlauben nur die Definition von Konzepten, keine allgemeinen Constraints
für viele Anwendungen ausreichend, z.B. SNOMED

Folgende Definitionen können für beliebige Beschreibungslogiken verwendet werden, nicht nur für *ALC*.

Generelle TBox - Syntax und Semantik

Definition 2.4 (Generelle TBox—Syntax)

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

Generelle TBox ist endliche Menge von Konzeptinklusionen.

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

T2.4

Definition 2.5 (Generelle TBox—Semantik)

Interpretation \mathcal{I}

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

Beachte: \mathcal{I} erfüllt $C \equiv D$ gdw. $C^{\mathcal{I}} = D^{\mathcal{I}}$

T2.4 cont

TBoxen - Semantik

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

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

Von generellen zu azyklischen TBoxen

Oft nur Konzeptnamen auf der linken Seite:

Perikardium \sqsubseteq Gewebe \sqcap \exists teilVon.Herz

Perikarditis \equiv Entzündung \sqcap \exists ort.Perikardium

Entzündung \sqsubseteq Krankheit \sqcap \exists wirktAuf.Gewebe

Intuition:

- Definition $A \equiv C$ gibt notwendige und hinreichende Bedingungen dafür, eine Instanz des Konzeptes A zu sein.
- Inklusion $A \sqsubseteq C$ gibt nur notwendige Bedingungen.

Azyklische TBoxen schränken sich auf diesen Fall ein.

Azyklische TBox - Syntax

Definition 2.6 (Azyklische TBox)

- *Konzeptdefinition* ist Ausdruck $A \equiv C$
- *Primitive Konzeptinklusion* ist Ausdruck $A \sqsubseteq C$.

mit A Konzeptname und C Konzept.

Azyklische TBox ist endliche Menge von Konzeptdefinitionen und primitiven Konzeptinklusionen so dass

- linke Seiten eindeutig
- keine Zyklen vorkommen, also keine Ausdrücke

$$A_0 \sim_0 C_0 \quad , \quad \dots \quad , \quad A_{n-1} \sim_{n-1} C_{n-1}, \quad \text{mit } \sim_i \subseteq \{\sqsubseteq, \equiv\}$$

so dass A_i in C_{i+1} vorkommt, für alle $i < n$ (modulo n)

T2.5

Azyklische TBox - Semantik

Da azyklische TBoxen Spezialfall von generellen TBoxen, ist Semantik bereits definiert.

Für azyklische TBox \mathcal{T} :

- Konzeptname A is *definiert* in \mathcal{T} gdw. \mathcal{T} enthält $A \equiv C$
- sonst ist A *primitiv* in \mathcal{T} .

T2.6

Im folgenden verwenden wir meistens generelle TBoxen,
verwenden “TBox” synonym zu “generelle TBox”

Grundlagen

Schlussfolgerungsprobleme

Schlussfolgerungsprobleme

Zur Erinnerung: Schlussfolgern...

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

Wichtigstes Designkriterium für Beschreibungslogiken:

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

Erfüllbarkeit, Subsumtion

Definition 2.7 (Erfüllbar, subsumiert, äquivalent bezüglich einer TBox)

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

T2.7

Schlussfolgerungsprobleme

Erfüllbarkeitsproblem:

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

Subsumtionsproblem:

gegeben C, D und \mathcal{T} , entscheide ob $\mathcal{T} \models C \sqsubseteq D$;

Äquivalenzproblem:

gegeben C, D und \mathcal{T} , entscheide ob $\mathcal{T} \models C \equiv D$;

Untersch. TBox-Formalisen geben untersch. Entscheidungsprobleme.

(Extremfall: gar keine TBox erlaubt = TBox muss leer sein)

Motivation

Anwendung:

- Modellierungsfehler finden:
unerfüllbare Konzepte im allgemeinen unerwünscht;
- die Struktur der TBox explizit machen, z.B. für Browsing:
Subsumption (= Is-A Relation) haupt-Strukturierungsmittel für TBoxen
 $C \sqsubseteq D$ kann gelesen werden als " D ist genereller als C "
- Redundanzen finden:
zwei äquivalente Konzepte sind u.U. unerwünscht.

Subsumtion als Ordnungsrelation

Lemma 2.8

Seit \mathcal{T} TBox mit mind. einem Modell. Die Relation " \sqsubseteq bzgl. \mathcal{T} " auf der Menge aller \mathcal{ALC} -Konzepte ist

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

Bis auf fehlende Antisymmetrie ist \sqsubseteq also partielle Ordnung.

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

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

T2.8

Klassifikation

Ein weiteres Schlussfolgerungsproblem:

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

Berechnungsproblem, kein Entscheidungsproblem.

In der Praxis:

- berechenbar durch n^2 Subsumtionsberechnungen;
- zahlreiche Optimierungen verfügbar.

Doors Protégé 3.2 beta (file:/home/ast/turhan/GonMoRe/Ontologies/Final-Tests/Door-Tests/Doors.pprj, OWL / RDF Files)

File Edit Project OWL Code Tools Window Help

protégé

OWLClasses Properties Forms Individuals Metadata OWL-DL Individuals OWLViz

SUBCLASS EXPLORER

For Project: Doors

Asserted Hierarchy

- owl:Thing
 - Activity
 - AuthorizedPerson
 - Context
 - DeliveryGoods
 - Device
 - AudioEnabledDevice
 - AudioEnabledConnectedDevice
 - InstantMessageEnabledDevice
 - InstantMessageEnabledConnectedDevice
 - MobileDevice
 - Handy
 - Laptop
 - PDA
 - SmartPhone
 - SMSEnabledDevice
 - StationaryDevice
 - TextEnabledConnectedDevice
 - TextEnabledDevice
 - VideoEnabledConnectedDevice
 - VideoEnabledDevice
 - Location
 - Network
 - Person
 - Presence
 - ValuePartition

SUBCLASS EXPLORER

For Project: Doors

Inferred Hierarchy

- owl:Thing
 - Activity
 - Context
 - DeliveryGoods
 - Device
 - AudioEnabledDevice
 - AudioEnabledConnectedDevice
 - DoorBell
 - TextEnabledDevice
 - MobileDevice
 - Handy
 - InstantMessageEnabledDevice
 - InstantMessageEnabledConnectedDevice
 - Laptop
 - PDA
 - SmartPhone
 - StationaryDevice
 - VideoEnabledDevice
 - Location
 - Network
 - Person
 - Presence
 - ValuePartition

CLASS EDITOR

For Class: InstantMessageEnabledConnectedDevice

Name: InstantMessageEnabledConnectedDevice

Annotations: rdfs:comment

Asserted Conditions:

- InstantMessageEnabledDevice (NECESSARY & SUFFICIENT)
- isConnectedVia some Network (NECESSARY)
- Laptop or PDA [from InstantMessageEnabledDevice] (INHERITED)

Properties:

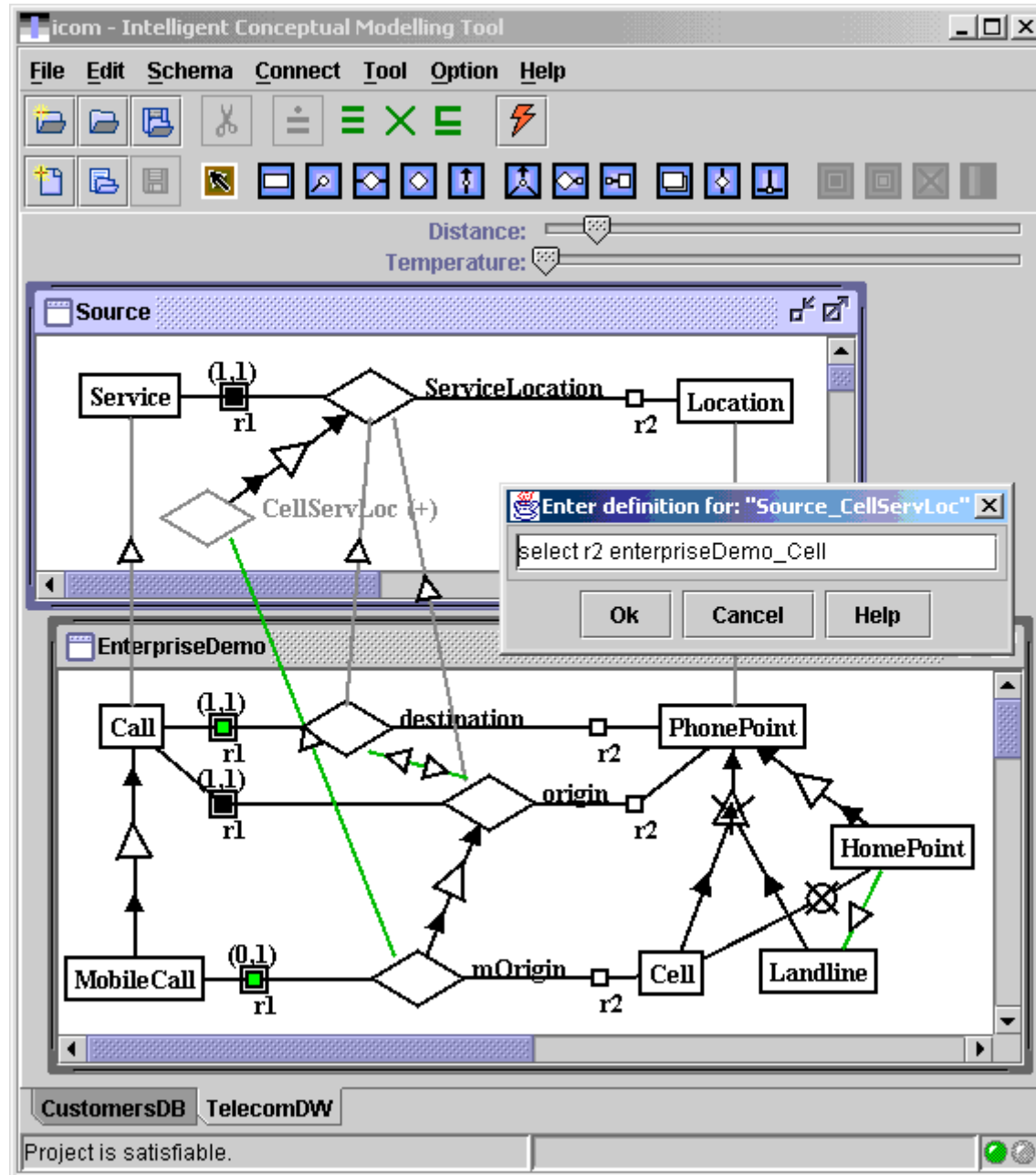
- isConnectedVia (multiple Network)
- Network
- isOwnedBy (multiple Person)
- isUsedBy (single Person)
- supportsLink (multiple Network)

Dis joints

Logic View Properties View

| Class | Changed direct superclasses |
|--------------------------------------|--|
| AudioReachableInUrgencyContext | Removed PresenceContext |
| AudioReachableResidentContext | Removed ReachableContext |
| AuthorisedPersonRingingContext | Added PresenceContext |
| AuthorizedNeighbourRingingContext | Moved from DoorContext to AuthorisedPersonRingingContext |
| AuthorizedPerson | Moved from owl:Thing to Person |
| Building | Removed Location |
| ConnectedResident | Added ConnectedPerson |
| DoorBell | Added AudioEnabledDevice |
| DoorbellReachableInUrgencyContext | Removed PresenceContext |
| DoorbellReachableResidentContext | Removed ReachableContext |
| DoorContext | Moved from Context to ConnectedContext, PresenceContext |
| Handy | Added SMSEnabledDevice |
| InstantMessageEnabledConnectedDevice | Added TextEnabledConnectedDevice |
| InstantMessageEnabledDevice | Moved from Device to TextEnabledDevice, VideoEnabledDevice, MobileDevice |

Classification Results



Reduktionen

Erfüllbarkeit, Subsumtion, Äquivalenz wechselseitig polynomiell reduzierbar:

Lemma 2.9.

1. Subsumtion polynomiell reduzierbar auf (Un)erfüllbarkeit:

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

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

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

3. Äquivalenz polynomiell reduzierbar auf Subsumtion:

$$\mathcal{T} \models C \equiv D \text{ gdw. } \mathcal{T} \models \top \sqsubseteq (C \sqcap D) \sqcup (\neg C \sqcap \neg D) \quad \mathbf{T2.9}$$

Algorithmus für eines der Probleme kann also auch für die beiden anderen verwendet werden; alle drei Probleme haben dieselbe Komplexität (in \mathcal{ALC})

Im folgenden konzentrieren wir uns meist auf Erfüllbarkeit

Reduktionen

Wir möchten zeigen:

Theorem 2.10

Erfüllbarkeit bzgl. azyklischer TBoxen kann reduziert werden auf Erfüllbarkeit ohne TBoxen.

Also kann Algorithmus, der Konzepterfüllbarkeit ohne TBoxen entscheidet auch in Gegenwart azyklischer TBoxen verwendet werden

Mit Lemma 2.9 folgt dasselbe für Subsumtion und Äquivalenz.

Reduktionen

Zu zeigen also:

Gegeben ein Konzept C und eine azyklische TBox \mathcal{T} kann man effektiv ein Konzept D konstruieren, so dass gilt:

C ist erfüllbar bzgl. \mathcal{T} gdw. D ist erfüllbar

Wir gehen in drei Schritten vor:

1. Ersetzen von primitiven Konzeptinklusionen durch Konzeptdefinitionen
2. Eliminieren von definierten Konzeptnamen auf den rechten Seiten von Konzeptdefinitionen
3. Expandieren der definierten Konzeptnamen im Eingabekonzept.

Beweis von Theorem 2.10

Schritt 1:

Ersetzen von primitiven Konzeptinklusionen durch Konzeptdefinitionen

Elimination von Inklusionen

Sei \mathcal{T} azyklische TBox.

Ersetze jede Inklusion $A \sqsubseteq C \in \mathcal{T}$ durch

$$A \equiv X_A \sqcap C, \text{ mit } X_A \text{ neuer Konzeptname}$$

Lemma 2.11

Für alle Konzepte C , die die neuen Konzeptnamen der Form X_A nicht verwenden, gilt:

$$C \text{ erfüllbar bzgl. } \mathcal{T} \text{ gdw. } C \text{ erfüllbar bzgl. } \mathcal{T}' \quad \text{T2.10}$$

Beweis von Theorem 2.10

Schritt 2:

Eliminieren von definierten Konzeptnamen auf den rechten Seiten von
Konzeptdefinitionen
= “Expandieren” der TBox

Azyklische TBox - Expandieren

Sei \mathcal{T} azyklische TBox ohne Konzeptinklusionen.

Definiere Folge von TBoxen $\mathcal{T}_0, \mathcal{T}_1, \dots$ wie folgt:

- $\mathcal{T}_0 = \mathcal{T}$;
- \mathcal{T}_{i+1} entsteht aus \mathcal{T}_i wie folgt:
wähle $A \equiv C \in \mathcal{T}_i$, so dass C definierten Konzeptnamen B enthält
ersetze alle Vorkommen von B in C durch D wenn $B \equiv D \in \mathcal{T}_i$

Da \mathcal{T} azyklisch, gibt es schliesslich kein solches $A \equiv C \in \mathcal{T}_i$ mehr.

Das letzte \mathcal{T}_i ist die *Expansion* von \mathcal{T} .

T2.11

Azyklische TBox - Expandieren

Zwei TBoxen sind *äquivalent* gdw. sie dieselben Modelle haben.

Lemma 2.12

Jede azyklische TBox ohne Konzeptinklusionen ist äquivalent zu ihrer Expansion.

T2.12

Bemerkung:

Intuitiv zeigt Lemma 2.12, dass primitive Konzeptdefinitionen nicht mehr sind als Makros

Azyklische TBox - Expandieren

Expansion kann zu einem exponentiellen Aufblasen der TBox führen:

$$\begin{array}{l} \text{Expandieren von} \\ A_0 \equiv \exists r. A_1 \sqcap \exists s. A_1 \\ A_1 \equiv \exists r. A_2 \sqcap \exists s. A_2 \\ \dots \\ A_{n-1} \equiv \exists r. A_n \sqcap \exists s. A_n \end{array}$$

gibt TBox der Grösse $> 2^n$.

T2.13

Beweis von Theorem 2.10

Schritt 3:

Expandieren der definierten Konzeptnamen im Eingabekonzept.

Beweis von Theorem 2.10

Sei C ein Konzept und \mathcal{T} eine azyklische TBox, die

1. keine Konzeptinklusionen und
2. keine definierten Konzeptnamen auf den rechten Seiten von Konzeptdefinitionen

enthält.

D ergibt sich aus C durch das Ersetzen aller definierten Konzeptnamen A durch E , wenn $A \equiv E \in \mathcal{T}$.

Lemma 2.13

C ist erfüllbar bzgl. \mathcal{T} gdw. D erfüllbar ist.

T2.14

Grundlagen

Erweiterungen von *ACC*

Erweiterungen von *ALC*

Wir betrachten exemplarisch zwei Erweiterungen von *ALC*:

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

Beide Erweiterungen sind in OWL realisiert.

Namenschema:

- ein Buchstabe pro Erweiterung
- kann kombiniert werden, z.B. *ALCQI*

Inverse Rollen

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

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

Verwendet man diese Rollen einfach als Namen in *ACC*,
gibt es unintuitive Konsequenzen.

T2.15

Inverse Rollen

Definition 2.14. (Inverse Rollen)

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

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

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

T2.15 cont.

Zahlenrestriktionen

Häufig möchte man Rollennachfolger “zählen” können:

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

Wir werden sehen:

In *ACC* ist zählen nicht ohne weiteres möglich

Zahlenrestriktionen

Definition 2.15. (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

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

Beachte:

T2.16

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

ALCQ: *ALC* erweitert mit Zahlenrestriktionen.

Weitere Erweiterungen

Es gibt noch viel mehr Erweiterungen:

- Spezielle Rolleninterpretationen (transitiv, symmetrisch, reflexiv, etc)
- Konzepte, die nur eine einzige Instanz haben koennen (Nominale)
- Inklusionen zwischen Rollen oder Rollenketten
- Numerische Werte
- Fixpunktoperatoren
- temporale Operatoren
- ...

Viele davon sind in OWL realisiert.