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 $\mathcal{ALC}$

ALC

Es gibt viele verschiedene Beschreibungslogiken:

- viele mögliche Kompromisse bzgl. Ausdrucksstärke vs. Komplexität des Schlussfolgerns

- verschiedene Anwendungen haben unterschiedliche Anforderungen

Hier zunächst $\mathcal{ALC}$:

- die einfachste BL mit allen Boolschen 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!

ALC

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 ($\mathcal{ALC}$-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 Rollenname, so sind
  - $\exists r.C$ (Existenzrestriktion)
  - $\forall r.C$ (Werterestriktion)

  $\mathcal{ALC}$-Konzepte.

T2.1

ALC Syntax

Verwendete Symbole:

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

Abkürzungen:

- $A \rightarrow B$ für $\neg A \sqcup B$ (Implikation)
- $A \leftrightarrow B$ für $(A \rightarrow B) \sqcap (B \rightarrow A)$ (Biimplikation)
- $\top$ für $A \sqcup \neg A$ (Top Konzept)
- $\bot$ für $\neg \top$ (Bottom Konzept)

ALC Syntax

Präzedenzregel:

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

Also zum Beispiel:

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

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

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

ALC Semantik

Definition 2.2 ($\mathcal{ALC}$ Semantik).
Eine *Interpretation* $\mathcal{I}$ ist Paar $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ mit

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

T2.2

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

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
- $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ es gibt $e \in \Delta^{\mathcal{I}}$ mit $(d, e) \in r^{\mathcal{I}}$ und $e \in C^{\mathcal{I}}\}$,
- $(\forall r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid$ für alle $e \in \Delta^{\mathcal{I}}$, $(d, e) \in r^{\mathcal{I}}$ 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 $d_0, \ldots, d_n$ 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
- $\bot^{\mathcal{I}}$ ist für jede Interpretation $\mathcal{I}$ leer
  Intuitiv repräsentiert $\bot$, dass etwas unmöglich ist, z.B.:

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

Grundlagen

TBoxen

TBoxen

Zur Erinnerung:

TBoxen (terminologische Boxen)

- definieren Konzepte
- setzen diese zueinander in Beziehung

Konzeptdefinition z.B.

$$\mathsf{Student} \equiv \mathsf{Mensch} \sqcap \exists \mathsf{h\ddot{o}rt.Vorlesung}$$

Allgemeines Hintergrundwissen / Constraint z.B.

$$\mathsf{Student} \sqcap \mathsf{Vorlesungssaal} \sqsubseteq \bot$$

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 $\mathcal{ALC}$.

Generelle TBox - Syntax und Semantik

Definition 2.3 (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 $C \sqsubseteq D$, $D \sqsubseteq C$.

T2.3

Definition 2.4 (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. $C^\mathcal{I} = D^\mathcal{I}$

T2.3 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 Konzept*namen* auf der linken Seite:

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

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

$$\text{Entzündung} \sqsubseteq \text{Krankheit} \sqcap \exists \text{wirktAuf}.\text{Gewebe}$$

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 2.3 (Azyklische TBox)

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

mit $\boldsymbol{A}$ Konzeptname und $\boldsymbol{C}$ Konzept.

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

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

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

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

T2.4

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 $\mathcal{T}$.

T2.5

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 möglich, aber wenig genug, um effizientes/entscheidbares Schlussfolgern zu erlauben.

Erfüllbarkeit, Subsumtion

Definition 2.6 (Erfüllbar, subsumiert, äquivalent)
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.* $\mathcal{T}$ ($\mathcal{T} \models C \sqsubseteq D$) gdw $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in allen Modellen $\mathcal{I}$ von $\mathcal{T}$.
- sind C und D *äquivalent bzgl.* $\mathcal{T}$ ($\mathcal{T} \models C \equiv D$) gdw $C^{\mathcal{I}} = D^{\mathcal{I}}$ in allen Modellen $\mathcal{I}$ von $\mathcal{T}$.

T2.6

Wir verwenden diese Begriffe manchmal auch *ohne TBox*, z.B.:

Konzept C ist *erfüllbar* wenn es Interpretation $\mathcal{I}$ gibt mit $C^{\mathcal{I}} \neq \emptyset$

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-Formalismen 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-Stukturierungsmittel 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 2.7
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-Diagram darstellen, dessen Knoten mit Mengen von Konzepten beschriftet sind.

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

T2.7

Klassifikation

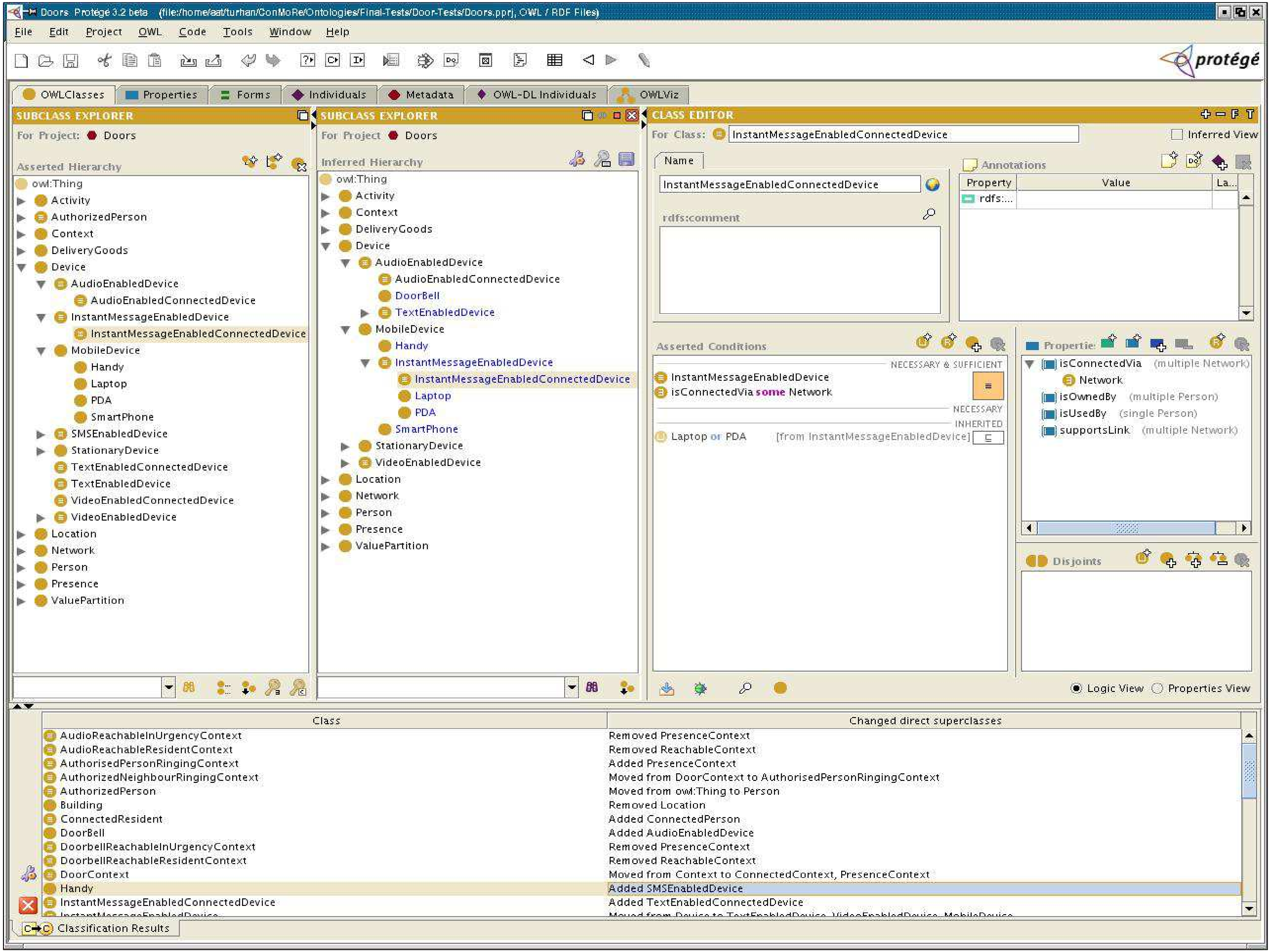
Ein weiteres Schlussfolgerungsproblem:

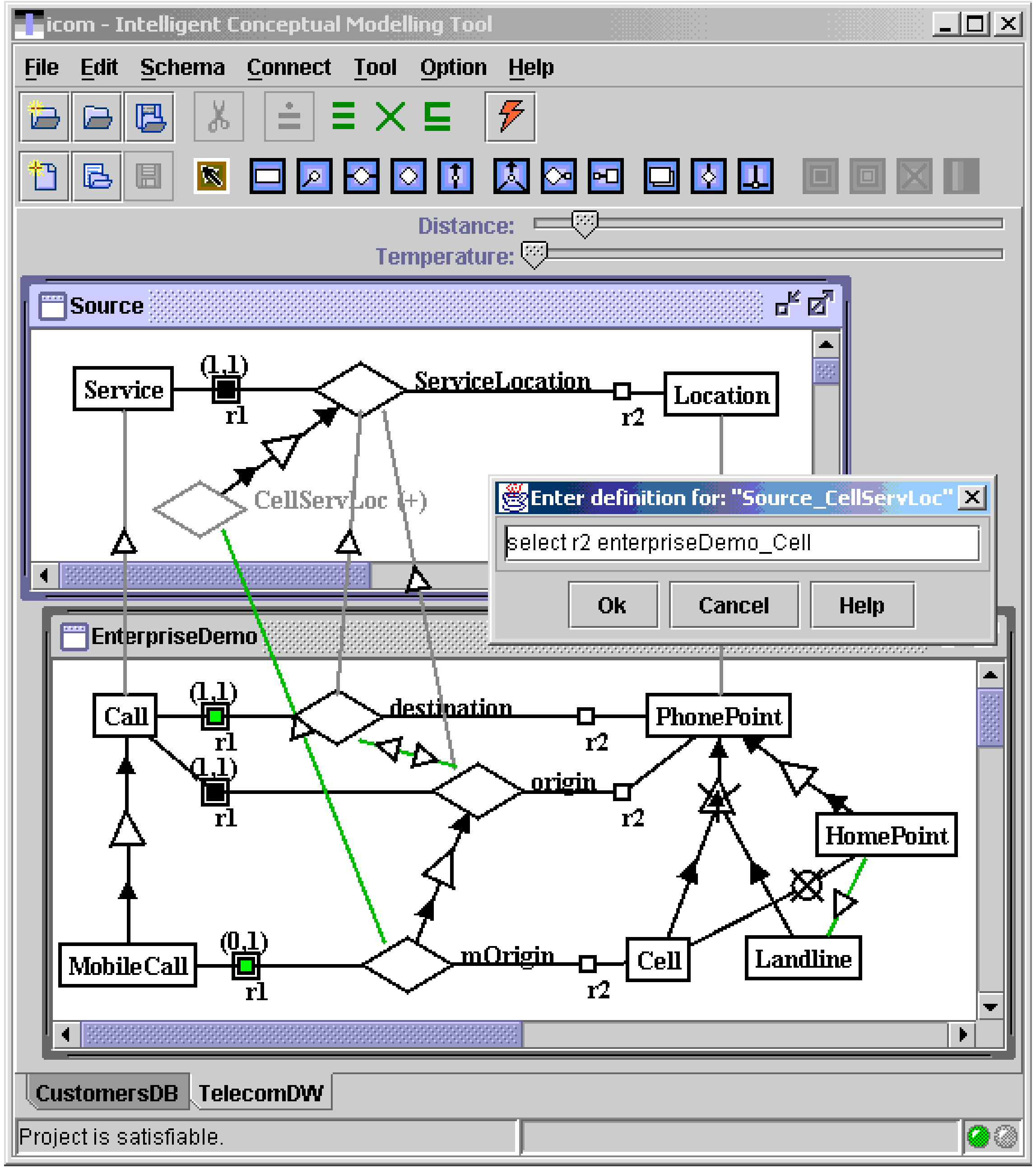
- *Klassifikation*: gegeben $\mathcal{T}$, berechne das Hasse-Diagramm für $\sqsubseteq$ bzgl. $\mathcal{T}$, eingeschränkt auf Konzeptnamen in $\mathcal{T}$.

Berechnungsproblem, kein Entscheidungsproblem.

In der Praxis:

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


Doors Protégé 3.2 beta (file:/home/aat/turhan/ConMoRe/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
Inferred View
Name
InstantMessageEnabledConnectedDevice
rdfs:comment
Annotations
Property Value La...
rdfs:...
Asserted Conditions
NECESSARY & SUFFICIENT
InstantMessageEnabledDevice
isConnectedVia some Network
NECESSARY
INHERITED
Laptop or PDA [from InstantMessageEnabledDevice]
Propertie
isConnectedVia (multiple Network)
Network
isOwnedBy (multiple Person)
isUsedBy (single Person)
supportsLink (multiple Network)
Disjoints
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
Classification Results


icom - Intelligent Conceptual Modelling Tool
File
Edit
Schema
Connect
Tool
Option
Help
Distance:
Temperature:
Source
Service
(1,1)
r1
ServiceLocation
r2
Location
CellServLoc (+)
Enter definition for: "Source_CellServLoc"
select r2 enterpriseDemo_Cell
Ok
Cancel
Help
EnterpriseDemo
Call
(1,1)
r1
destination
r2
PhonePoint
(1,1)
r1
origin
r2
HomePoint
MobileCall
(0,1)
r1
mOrigin
r2
Cell
Landline
CustomersDB
TelecomDW
Project is satisfiable.

Reduktionen

Erfüllbarkeit, Subsumtion, Äquivalenz wechselseitig polynomiall reduzierbar:

Lemma 2.8.

- $\mathcal{T} \models C \sqsubseteq D$ gdw. $C \sqcap \neg D$ unerfüllbar bzgl. $\mathcal{T}$
- C erfüllbar bzgl. $\mathcal{T}$ gdw. $\mathcal{T} \not\models C \equiv \bot$
- $\mathcal{T} \models C \equiv D$ gdw. $\mathcal{T} \models \top \sqsubseteq C \leftrightarrow D$ T2.8

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 2.9
Erfüllbarkeit bzgl. azyklischer TBoxen kann reduziert werden auf Erfüllbarkeit ohne TBoxen.

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

Mit Lemma 2.8 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:

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

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 2.9

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:

$$C \text{ erfüllbar bzgl. } \mathcal{T} \text{ gdw. } C \text{ erfüllbar bzgl. } \mathcal{T}'$$ T2.9

Beweis von Theorem 2.9

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 $\mathcal{T}_0, \mathcal{T}_1, \ldots$ 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 $\mathcal{T}$.

T2.10

Azyklische TBox - Expandieren

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

Lemma 2.13
Jede azyklische TBox ohne Konzeptinklusionen ist äquivalent zu ihrer Expansion. T2.11

Bemerkungen:

Intuitiv zeigt Lemma 2.13, dass primitive Konzeptdefinitionen nicht mehr sind als Makros

Azyklische TBox - Expandieren

Expansion kann zu einem exponentiellen Aufblasen der TBox führen:

Expandieren von

$$\begin{aligned} A_0 &\equiv \exists r.A_1 \sqcap \exists s.A_1 \\ A_1 &\equiv \exists r.A_2 \sqcap \exists s.A_2 \\ &\cdots \\ A_{n-1} &\equiv \exists r.A_n \sqcap \exists s.A_n \end{aligned}$$

gibt TBox der Grösse $> 2^n$.

T2.12

Beweis von Theorem 2.9

Schritt 3:

Expandieren der definierten Konzeptnamen im Eingabekonzept.

Beweis von Theorem 2.9

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

1. keine Konzeptinklusionen und
2. keine definierten Konzeptnamen auf den rechten Seiten von Konzept-definitionen

enthält.

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

Lemma 2.14
C ist erfüllbar bzgl. $\mathcal{T}$ gdw. D erfüllbar ist.

T2.14

Grundlagen

Erweiterungen von $\mathcal{ALC}$

Erweiterungen von ALC

Wir betrachten exemplarisch zwei Erweiterungen von $\mathcal{ALC}$:

- $\mathcal{ALCI}$: $\mathcal{ALC}$ mit inversen Rollen (Rollenkonstruktor)
- $\mathcal{ALCQ}$: $\mathcal{ALC}$ mit Zahlenrestriktionen (Konzeptkonstruktor)

Beide Erweiterungen sind in OWL realisiert.

Namenschema:

- ein Buchstabe pro Erweiterung
- kann kombiniert werden, z.B. $\mathcal{ALCQI}$

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 $\mathcal{ALC}$,
gibt es unintuitive Konsequenzen.

T2.15

Inverse Rollen

Definition 2.16. (Inverse Rollen)
Für jeden Rollennamen r ist r^- die *inverse Rolle* zu r. Wir definieren

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

$\mathcal{ALCI}$: $\mathcal{ALC}$ erweitert um die Möglichkeit, inverse Rollen in Existenz- und Werterestriktionen 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 $\mathcal{ALC}$ ist zählen nicht ohne weiteres möglich

Zahlenrestriktionen

Definition 2.17. (Zahlenrestriktion)
Für jede natürliche Zahl n, jeden Rollennamen r und jedes Konzept C:

- $(\leqslant n\ r\ C)$ (Höchstens-Restriktion);
- $(\geqslant n\ r\ C)$ (Mindestens-Restriktion);

Die Semantik ist

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

Beachte:

T2.16

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

$\mathcal{ALCQ}$: $\mathcal{ALC}$ 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.