

ABoxen und Anfragebeantwortung

Ziel des Kapitels

TBoxen repräsentieren allgemeines, begriffliches Wissen

Um *konkrete Situationen* zu repräsentieren, braucht man *Instanzen*.

Für medizinische Ontologien wie SNOMED z.B. Patientendaten:

Patient(p_1)	Patient(p_2)
Medikament(m)	Krankheit(k)
erhält(p_1, m)	erhält(p_2, m)
heilt(m, k)	hat(p_1, k)

Patient, Medikament, etc können in TBox definiert sein.

Ziel des Kapitels

Eine relationale Datenbank ist auch nichts weiter als Menge solcher Daten, allerdings sind die Konzepte/Relationen dort nicht formal beschrieben

Ziel des Kapitels:

- Bestimmen eines Formalismus für Instanzdaten (ABox)
- Auswahl Schlußfolgerungsprobleme, um mit Instanzdaten zu arbeiten (insb. Anfragebeantwortung)
- Algorithmen entwickeln / Komplexität studieren.

ABoxen und Anfragebeantwortung

Grundlagen

ABox - Syntax

Von nun an sei

- \mathbf{N}_I eine unendliche Menge von *Individuennamen*
Diese entsprechen Konstanten im FO-Sinne

Wir verwendet a, b, \dots für Individuennamen.

Definition 6.1 (ABox Syntax)

Eine

- *Konzeptassertion* hat die Form $C(a)$
- *Rollenassertion* hat die Form $r(a, b)$

Eine *ABox* ist eine endliche Menge von (Konzept- und Rollen-)assertionen.

T6.1

ABox – Semantik

Definition 6.1 (ABox Semantik)

Interpretation \mathcal{I}

- bildet jedes $a \in \mathbf{N}_I$ auf Element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ ab;
- erfüllt $C(a)$ gdw. $a^{\mathcal{I}} \in C^{\mathcal{I}}$ ($\mathcal{I} \models C(a)$);
- erfüllt $r(a, b)$ gdw. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ ($\mathcal{I} \models r(a, b)$).

\mathcal{I} ist *Modell* von \mathcal{A} ($\mathcal{I} \models \mathcal{A}$) gdw. \mathcal{I} alle Assertionen in \mathcal{A} erfüllt.

T6.1 cont

ABox – Unvollständigkeit

ABoxen repräsentieren *unvollständiges Wissen*:

- Eine ABox hat viele Modelle!
- Es gibt ABoxen \mathcal{A} und Assertionen $C(a)$ so dass

weder $\mathcal{A} \models C(a)$ noch $\mathcal{A} \models \neg C(a)$

T6.2

\Rightarrow *open world assumption (OWA)*, im Ggs. zur *closed world assumption (CWA)* bei Datenbanken und der logischen Programmierung.

CWA: was nicht explizit als wahr angegeben ist, ist falsch.

OWA: es sind nur solche Dinge falsch, die explizit als falsch angegeben sind

T6.2 cont

Wissensbasis

Definition 6.2 (Wissensbasis)

Wissensbasis (WB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ besteht aus TBox \mathcal{T} und ABox \mathcal{A} .

Interpretation \mathcal{I} ist *Modell* von \mathcal{K} ($\mathcal{I} \models \mathcal{K}$) gdw. $\mathcal{I} \models \mathcal{T}$ und $\mathcal{I} \models \mathcal{A}$.

In Anwendungen haben \mathcal{T} und \mathcal{A} verschiedenen Status:

- \mathcal{T} wird einmal erstellt, ändert sich danach üblicherweise nicht mehr
- \mathcal{A} ändert sich häufig (wie Datenbank)

Man kann \mathcal{T} als Ontologie verstehen oder \mathcal{K} .

Sicht von OWL: Ontologie enthält auch Instanzdaten, entspricht WB

ABoxen und Anfragebeantwortung

Schlussfolgerungsprobleme

Grundlegende Schlussfolgerungsprobleme

Definition 6.3 (Konsistenz, Instanz)

Sei \mathcal{K} Wissensbasis, C Konzept, a Individuename. Dann

- ist \mathcal{K} *konsistent* gdw. \mathcal{K} Modell hat;
- ist a eine *Instanz* von C bzgl. \mathcal{K} gdw. $(\mathcal{K} \models C(a))$
 $\mathcal{I} \models C(a)$ für jedes Modell \mathcal{I} von \mathcal{K} .

T6.3

Grundlegende Schlussfolgerungsprobleme

Konsistenzproblem:

gegeben \mathcal{K} , entscheide ob \mathcal{K} konsistent;

Instanzproblem:

gegeben \mathcal{K} , C und a , entscheide ob $\mathcal{K} \models C(a)$;

Beide Probleme auch ohne (mit leerer) TBox.

Untersch. TBox-Formalisten geben untersch. Entscheidungsprobleme.

Reduktionen

Konsistenz- und (Nicht-)Instanzproblem wechselseitig polynomiell reduzierbar.

Lemma 6.4.

- \mathcal{K} konsistent gdw. $\mathcal{K} \not\models \perp(a)$
- $(\mathcal{T}, \mathcal{A}) \models C(a)$ gdw. $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$ inkonsistent.

T6.4

Erfüllbarkeit (von Konzepten) polynomiell reduzierbar auf Konsistenz.

Lemma 6.5.

C erfüllbar bzgl. \mathcal{T} gdw. $(\mathcal{T}, \{C(a)\})$ konsistent.

Natürlich auch korrekt wenn $\mathcal{T} = \emptyset$, erhält Azyklizität von \mathcal{T} .

Anfragebeantwortung

Viele Anwendungen verwenden ABoxen wie (semantische) Datenbanken

Verschiedene Anfragesprachen möglich:

- *Instanzanfrage*: gegeben \mathcal{K} und C , ermittle alle $a \in \mathbf{N}_C$ mit $\mathcal{K} \models C(a)$;
- *Konjunktive Anfragen*: generalisieren Instanzanfragen, Definition später.

Berechnungsproblem, kein Entscheidungsproblem.

ABoxen vs. Datenbanken

Unterschiedliche Semantik: CWA vs. OWA

Anfragebeantwortung

- in relationalen Datenbanken: Model checking (Datenbank=Modell)
- in Beschreibungslogik: logisches Schlussfolgern (KB=logische Theorie)

Enger Zshg. zum Gebiet der unvollständigen Datenbanken:

- dort macht man ebenfalls die OWA,
- interessiert sich für *certain answers*: Antworten, bezüglich derer alle möglichen Modelle übereinstimmen.

OWA Anfragebeantwortung ist i.d.R. komplexer!

ABoxen vs. Datenbanken

TBox vs. konzeptuelles DB-Schema:

- konzeptuelles Schema formuliert als ER Diagram, UML Diagram, etc.
- konzeptuelles Schema dient dem DB-Design, wird bei Anfragebeantwortung nicht verwendet.

TBox vs. Constraints:

- Constraints: Datenbankschlüssel, funktionale Abhängigkeiten, etc.
- DB System stellt sicher, dass die Datenbank keine Constraints verletzt
ABoxen können der TBox widersprechen
- Anfragebeantwortung: Constraints nicht zum Schlussfolgern verwendet
sondern zur Optimierung

ABoxen und Anfragebeantwortung

Instanzanfragen

Übersicht

Instanzanfragen:

- berechenbar durch n -fache Berechnung des Instanzproblems;
- Instanzproblem kann auf Konsistenzproblem reduziert werden

Wir konzentrieren uns also auf das Entscheiden von Konsistenz

(in der Praxis ist das allerdings nicht sehr effizient)

Mögliche Algorithmen für Konsistenz:

- Erweiterung von *ALC*-Elim und *ALC*-Worlds
- Erweiterung von Tableau Algorithmen
- Reduktion auf Erfüllbarkeit

Vervollständigung

Wir betrachten den Fall von generellen TBoxen

Vervollständigung (Precompletion): Turing-Reduktion auf Erfüllbarkeit, also:
es wird auf nicht nur einen Erfüllbarkeitstest reduziert, sondern auf mehrere

Grundideen:

- Zerlege ABox in Teile: eine pro Individuenname $a \in \text{Ind}(\mathcal{A})$.
- Für jedes $a \in \text{Ind}(\mathcal{A})$, konstruiere Konzept C_a und teste ob C_a erfüllbar bzgl. \mathcal{T}
- Aus den gesammelten Modellen kann man Modell von \mathcal{K} konstruieren.
- Grundidee: $C_a = \bigsqcap_{D(a) \in \mathcal{A}} D$ (funktioniert so aber noch nicht)

T6.5

Vervollständigung

Bzgl. der Eingabe $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ nehmen wir an:

- Alle Konzepte in \mathcal{A} sind in NNF
- \mathcal{T} hat die Form $\{\top \sqsubseteq C_{\mathcal{T}}\}$ mit $C_{\mathcal{T}}$ in NNF

Notation: $\sim C$ bezeichnet NNF von $\neg C$.

Definition 6.6. (Teilkonzepte, Closure)

Für alle ABoxen \mathcal{A} und Wissensbasen $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

- $\text{sub}(\mathcal{A}) = \bigcup_{C(a) \in \mathcal{A}} \text{sub}(C)$
- $\text{sub}(\mathcal{K}) = \text{sub}(\mathcal{T}) \cup \text{sub}(\mathcal{A})$
- $\text{cl}(\mathcal{K}) = \{C, \sim C \mid C \in \text{sub}(\mathcal{K})\}$.

Vervollständigung

Notation: $\text{Ind}(\mathcal{A})$ ist Menge aller Individuennamen in \mathcal{A}

Definition 6.7. (Vervollständigung)

Vervollständigung von \mathcal{ALC} -Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ ist WB $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ so dass:

1. $\mathcal{A} \subseteq \mathcal{A}'$;
2. für alle $a \in \text{Ind}(\mathcal{A})$ und $C \in \text{cl}(\mathcal{K})$: $C(a) \in \mathcal{A}'$ oder $\sim C(a) \in \mathcal{A}'$
3. $\{r(a, b), \forall r.C(a)\} \in \mathcal{A}'$ impliziert $C(b) \in \mathcal{A}'$
4. es gibt kein $\mathcal{A}'' \subsetneq \mathcal{A}'$, das ebenfalls Punkte 1-3 erfüllt. T6.6

Vervollständigung

Lemma 6.8.

\mathcal{K} konsistent gdw. es gibt Vervollständigung $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ von \mathcal{K} so dass

$C_a := \bigcap_{C(a) \in \mathcal{A}'} C$ erfüllbar bzgl. \mathcal{T} für alle $a \in \text{Ind}(\mathcal{A}')$. T6.6 cont, T6.7

Es gibt offensichtlich nur endlich viele Vervollständigungen

Theorem 6.9.

ABox Konsistenz in \mathcal{ALC} ist entscheidbar.

T6.8

Wir haben damit auch einen Algorithmus für Anfragebeantwortung.

Komplexität

Mit wenigen Ausnahmen:

Konsistenzproblem hat selbe Komplexität wie Erfüllbarkeit

Für *ACC* also zu erwarten:

- EXPTIME-vollständig mit generellen TBoxen;
- PSPACE-vollständig ohne TBoxen.

Untere Schranken:

Kapitel 5 + Lemma 6.5

Obere Schranken: im folgenden aus

Kapitel 5 + Lemma 6.8

Resultat

Definition 6.10.

Größe $|\mathcal{A}|$ einer ABox \mathcal{A} ist

$$\sum_{C(a) \in \mathcal{A}} |C| + 3 \quad + \quad 6 \cdot |\{r(a, b) \in \mathcal{A}\}|$$

Größe $|\mathcal{K}|$ einer WB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ ist $|\mathcal{T}| + |\mathcal{A}|$.

Lemma 6.11.

Für jede WB \mathcal{K} gibt es höchstens $2^{\mathcal{O}(n^2)}$ Vervollständigungen, $n = |\mathcal{K}|$.

T6.9

Theorem 6.12.

Das Instanzproblem in \mathcal{ALC} ist

1. EXPTIME-vollständig mit generellen TBoxen;
2. PSPACE-vollständig ohne TBoxen.

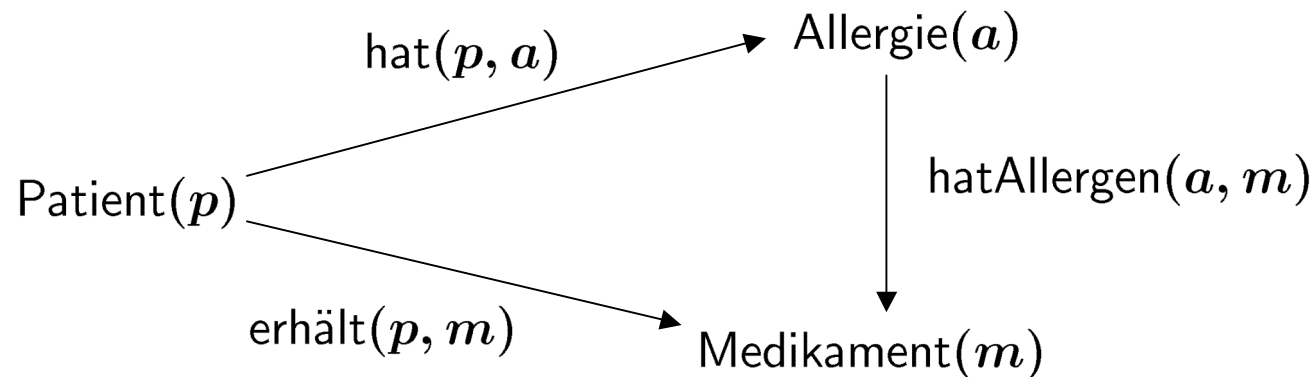
T6.10

ABoxen und Anfragebeantwortung

Konjunktive Anfragen

Instanzanfragen und ABoxen

ABoxen können komplexe relationale Strukturen erzwingen:



Konzept kann solche Struktur (mit p als Wurzel) nicht “abfragen”

(denn die Struktur ist nicht baumförmig, siehe Kapitel 3 + Unravelling)

Unbefriedigend!

Konjunktive Anfragen

Konjunktive Anfragen

- generalisieren Instanzanfragen in natürlicher Weise;
- erlauben Anfragen bzgl. der *relationalen Struktur* von ABoxen;
- entsprechen einem Fragment von SQL;
- sind in der Datenbanktheorie sehr weit verbreitet.

Konjunktive Anfrage - Syntax

Von nun an sei

- \mathbf{N}_V eine Menge von *Variablen*.
- ein *Term* entweder eine Variable oder ein Individuenname.

Definition 6.13. (Konjunktive Anfrage)

Ein

- *Konzeptatom* hat die Form $C(t)$, mit t Term;
- *Rollenatom* hat die Form $r(t, t')$, mit t, t' Term;

Ein *Atom* ist entweder ein Konzeptatom oder ein Rollenatom

Eine *Konjunktive Anfrage* hat die Form

$$q = \exists y_0, \dots, y_{m-1} . \alpha_0 \wedge \dots \wedge \alpha_{k-1}$$

wobei $y_0, \dots, y_{m-1} \in \mathbf{N}_V$ und $\alpha_0, \dots, \alpha_{k-1}$ Atome.

Konjunktive Anfrage - Syntax

Sei

$$q = \exists y_0, \dots, y_{m-1} \cdot \alpha_0 \wedge \dots \wedge \alpha_{k-1}$$

konjunktive Anfrage.

Die Variablen y_0, \dots, y_{m-1} heissen *quantifizierte Variablen*, alle anderen Variablen in $\alpha_0, \dots, \alpha_{k-1}$ *Antwortvariablen*.

T6.11

Konjunktive Anfrage - Notation

Wir schreiben

- t, t' , etc für Terme;
- x_0, x_1, \dots für Antwortvariablen
- y_0, y_1, \dots für quantifizierte Variablen, schreiben Sequenz y_0, \dots, y_{m-1} als \bar{y}
- φ für Konjunktionen von Atomen
- q, q' , etc für konjunktive Anfragen

For konjunktive Anfrage q ist $\text{Term}(q)$ die Menge der Terme in q .

(Antwortvariablen + quantifizierte Variablen
+ Individuennamen)

Konjunktive Anfrage - Semantik

Definition 6.14. (Treffer, Antwort bzgl. Interpretation)

Sei \mathcal{I} Interpretation, $q = \exists \bar{y}. \varphi$ konjunktive Anfrage und

$\tau : \text{Term}(q) \rightarrow \Delta^{\mathcal{I}}$ Abbildung mit

$$\tau(a) = a^{\mathcal{I}} \text{ für alle } a \in \text{Term}(q) \cap \mathbf{N}_I.$$

τ erfüllt

- Konzeptatom $C(t)$ wenn $\tau(t) \in C^{\mathcal{I}}$;
- Rollenatom $r(t, t')$ wenn $(\tau(t), \tau(t')) \in r^{\mathcal{I}}$.

τ is *Treffer* für q und \mathcal{I} wenn τ alle Atome in φ erfüllt.

Antwort auf q bzgl. \mathcal{I} : Tupel

$$(a_1, \dots, a_n) \in \mathbf{N}_I^n$$

so dass es Treffer τ für q und \mathcal{I} gibt mit $\tau(x_i) = a_i$ für $1 \leq i \leq n$. **T6.12**

Konjunktive Anfrage - Semantik

Definition 6.15. (Antwort bzgl. Wissensbasis)

Sei $q = \exists \bar{y}. \varphi$ konjunktive Anfrage mit Antwortvariablen x_1, \dots, x_n ,
 \mathcal{K} Wissensbasis

Antwort auf q bzgl. \mathcal{K} : Tupel

$$(a_1, \dots, a_n) \in \mathbf{N}_I^n$$

das Antwort auf q ist bzgl. aller Modelle \mathcal{I} von \mathcal{K} .

Beantwortung konjunktiver Anfragen: gegeben q und \mathcal{K} , berechne
alle Antworten auf q bzgl. \mathcal{K}

T6.12 cont

Konjunktive Anfragen vs. Instanzanfragen

Konjunktive Anfrage q kann als knoten- und kantenbeschrifteter Graph (V, E, \mathcal{L}) gesehen werden:

- $V = \text{Term}(q)$;
- $(t, r, t') \in E$ wenn $r(t, t')$ Atom in q ist;
- $\mathcal{L}(t) = \{C \mid C(t) \text{ ist Atom in } q\}$.

Intuitiv:

Instanzanfragen = baumförmige konjunktive Anfragen.

mit Wurzel einziger Antwortvariable

T6.13

Konjunktive Anfragen und SQL

Datenbanktheorie:

- SQL entspricht FO (wechselseitig polynomiell übersetzbar);
- KAen entsprechen i.d.S. select-project-join queries

In Datenbanksystemen

- sind $>90\%$ aller gestellten Anfragen select-project-join;
- sind die query engines speziell auf solche Anfragen hin optimiert.

Theorem 6.16.

Die Beantwortung von FO-Anfragen über *ACC*-Wissensbasen ist unentscheidbar.

T6.14

Boolsche Anfragen

Für Komplexitätsresultate brauchen wir Entscheidungsproblem

Definition 6.17. (Boolsche Anfrage)

Eine konjunktive Anfrage heisst *Boolsch* wenn sie keine Antwortvariablen hat.

T6.15

Boolsche Anfrage liefert keine Individuennamen als Antworten:

ist für jeder Interpretation entweder wahr (= es gibt Treffer)

oder falsch (= es gibt keinen Treffer)

Notation: wir schreiben

- $\mathcal{I} \models q$ wenn q in \mathcal{I} wahr ist;
- $\mathcal{K} \models q$ wenn für jedes Modell \mathcal{I} von \mathcal{K} gilt $\mathcal{I} \models q$

Wir sagen: q folgt aus \mathcal{K}

Anfrageimplikation

Anfrageimplikationsproblem:

Gegeben Boolesche KA q und WB \mathcal{K} , entscheide ob $\mathcal{K} \models q$.

Anfrageimplikation vs. Anfragebeantwortung:

- Implikation ist Spezialfall von Beantwortung
- Beantwortung n -ärer KA kann durch $|\text{Ind}(\mathcal{A})|^n$ Implikationsanfragen realisiert werden

Wir betrachten von nun an Anfrageimplikation statt -beantwortung.

Komplexität und Entscheidbarkeit

Das folgende Resultat ist deutlich schwieriger zu zeigen als die entsprechenden Resultate für das Instanzproblem.

Theorem 6.18.

Anfrageimplikation ist

- EXPTIME -vollständig in \mathcal{ALL} und \mathcal{ALLQ} ;
- 2EXPTIME -vollständig in \mathcal{ALLI} und \mathcal{ALLQI} .

(in beiden Fällen mit generellen TBoxen)

Die unteren Schranken gelten bereits, wenn ABox die Form $\{A(a)\}$ hat!

Konjunktive Anfragen mit Ungleichheit

Schon leichte Erweiterungen machen Anfrageimplikation unentscheidbar.

Erweiterte konjunktive Anfrage:

Konjunktive Anfrage mit zusätzlichem Typ von Atom $t \neq t'$.

Erweiterte Anfrageimplikation:

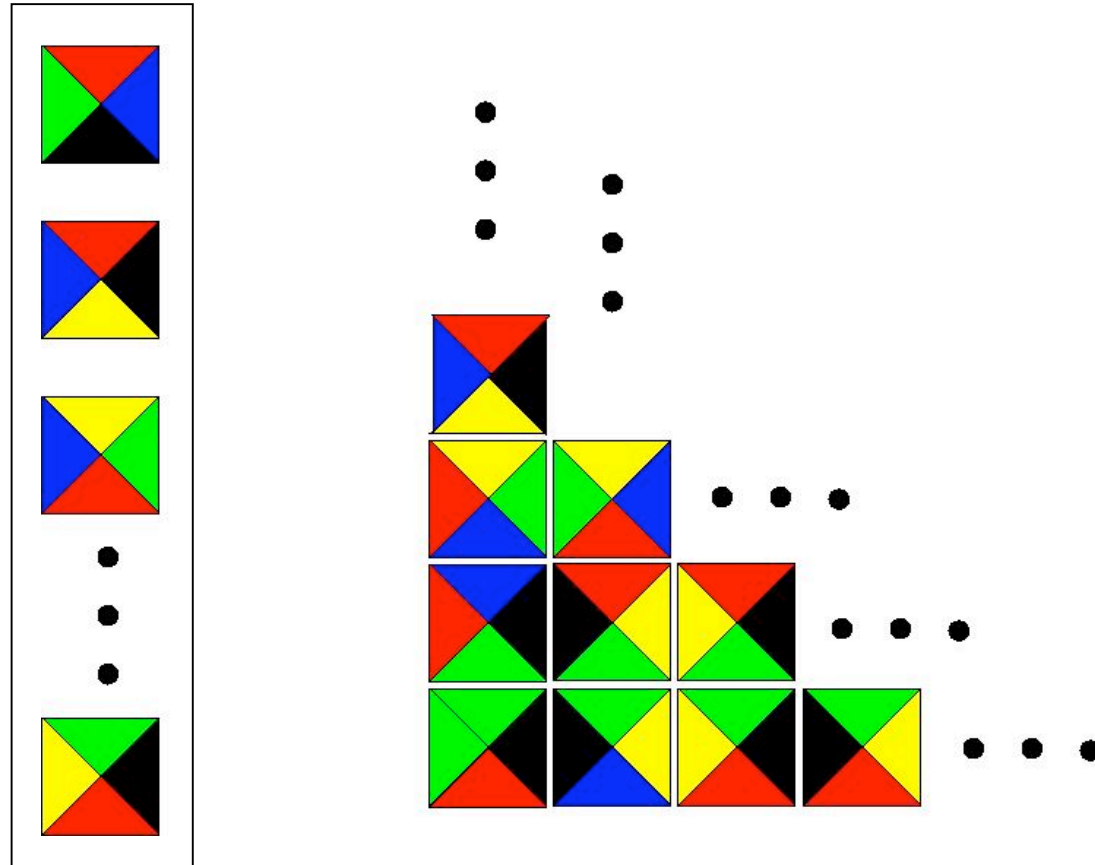
Wie Anfrageimplikation, aber mit erweiterten Anfragen. **T6.16**

Theorem 6.19

Das erweiterte Anfrageimplikationsproblem in *ALC* mit generellen TBoxen ist unentscheidbar.

Beweis: Reduktion des unentscheidbaren Dominoproblems.

Domino Problem



Gegeben: endliche Menge von Domino-*Typen*.

Frage: kann man damit den ersten Quadranten der Ebene parkettieren so dass alle benachbarten Kanten dieselbe Farbe haben?

Domino Problem

Definition 6.20 [Domino System]

Domino System ist Paar $\mathcal{D} = (F, T)$ mit

- F endliche Menge von *Farben*;
- T endliche Menge von 4-Tupeln $t = (t_o, t_u, t_\ell, t_r) \in F^4$,
den *Domino Typen*

Abbildung $\pi : \mathbb{N} \times \mathbb{N} \rightarrow T$ ist *Lösung* für \mathcal{D} wenn

- $\pi(i, j)_r = \pi(i + 1, j)_\ell$ für alle $i, j \geq 0$;
- $\pi(i, j)_o = \pi(i, j + 1)_u$ für alle $i, j \geq 0$.

Theorem 6.21

Es ist unentscheidbar ob ein gegebenes Dominosystem eine Lösung hat.

Die Reduktion

Ziel:

Gegeben Domino System \mathcal{D} , konstruiere \mathcal{ALC} -Wissensbases $\mathcal{K}_{\mathcal{D}}$ und erweiterte konjunktive Anfrage q , so dass \mathcal{D} Lösung hat gdw. $\mathcal{K}_{\mathcal{D}} \models q$.

Intuition:

Modelle \mathcal{I} von $\mathcal{K}_{\mathcal{D}}$ mit $\mathcal{I} \models q$ repräsentieren Lösungen für \mathcal{D} .

Signatur von $\mathcal{K}_{\mathcal{D}}$ und q :

- Rollennamen v, h für vertikale/horizontale Nachbarschaft
- Konzeptname A_t für jedes $t \in T$.

T6.17

Die Reduktion

Jeder Punkt hat horizontalen und vertikalen Nachfolger:

$$\top \sqsubseteq \exists h. \top \sqcap \exists v. \top$$

Jeder Punkt hat genau einen Typ:

$$\top \sqsubseteq \bigsqcup_{t \in T} (A_t \sqcap \bigsqcap_{t' \in T, t \neq t'} \neg A_{t'})$$

Benachbarte Kanten haben dieselbe Farbe:

$$\top \sqsubseteq \bigsqcap_{t \in T} (A_t \rightarrow \forall h. \bigsqcup_{t' \in T \text{ und } t_r = t'_\ell} A_{t'})$$

$$\top \sqsubseteq \bigsqcap_{t \in T} (A_t \rightarrow \forall v. \bigsqcup_{t' \in T \text{ und } t_o = t'_u} A_{t'})$$

Die ABox ist leer!

Die Reduktion

Was noch fehlt:

- die Rollen h und v sind *konfluent*, also:
jeder hv -Nachfolger ist auch vh -Nachfolger.

Wir erreichen das mittels q :

$$\exists v_0, v_1, v_2, v', v''. h(v_0, v_1), v(v_1, v_2), v(v_0, v'), h(v', v''), v_2 \neq v''$$

T6.18

Beachte: für Korrektheit müssen h und v nicht unbedingt Funktionen sein

Lemma 6.22

\mathcal{D} hat Lösung gdw. $\mathcal{K}_{\mathcal{D}} \not\models q$.

T6.19

Theorem 6.19 folgt unmittelbar.

Zusammenfassung

- In vielen Anwendungen reichen Erfüllbarkeit und Subsumption nicht aus
- ABoxen machen aus DLs “semantische Datenbanken”
- Instanzanfragen sind genauso schwierig wie Erfüllbarkeit
- Konjunktive Anfragen sind ausdrucksstärker, aber auch schwerer zu beantworten
- Noch mehr Ausdrucksstärke führt meist zu Unentscheidbarkeit