

# 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

# Effiziente Beschreibungslogiken

## Ziel des Kapitels

Für manche Anwendungen ist  $\mathcal{ALC}$  zu komplex:

- Auch hoch-optimierte Reasoner können große Ontologien wie SNOMED CT nicht verarbeiten (oder nur nach intensivem Tuning)
- In der Anfragebeantwortung muss man oft mit sehr grossen Datenmengen umgehen und braucht schnelle Antworten (scalability)

Wir betrachten die Beschreibungslogik  $\mathcal{EL}$ :

- viel weniger ausdrucksstark als  $\mathcal{ALC}$ , Basisoperatoren  $\sqcap$  und  $\exists r.C$
- Erfüllbarkeit und Subsumption in Polyzeit entscheidbar
- Skalierbare Anfragebeantwortung mit Standard-SQL-Datenbanken möglich

# EL

## Definition 7.1

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

Beliebt für biomedizinische Ontologien (groß, hoher Abstraktionsgrad):

Perikardium  $\sqsubseteq$  Gewebe  $\sqcap$   $\exists$ teilVon.Herz

Perikarditis  $\equiv$  Entzündung  $\sqcap$   $\exists$ ort.Perikardium

Entzündung  $\sqsubseteq$  Krankheit  $\sqcap$   $\exists$ wirktAuf.Gewebe

SNOMED CT ist in unwesentlicher Erweiterung von EL formuliert

$\mathcal{EL}$  ist Grundlage des OWL EL Profiles von OWL2

# Simulation

Intuitiv:  $\mathcal{EL}$  ist die “Hälfte von  $\mathcal{ALLC}$ ”, entspricht der “Hälfte von Bisimulation”

## Definition 7.2 (Simulation)

Seien  $\mathcal{I}_1$  und  $\mathcal{I}_2$  Interpretationen

Relation  $\rho \subseteq \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$  ist *Simulation* von  $\mathcal{I}_1$  nach  $\mathcal{I}_2$  wenn

1.  $d_1 \rho d_2$  und  $d_1 \in A^{\mathcal{I}_1}$  impliziert  $d_2 \in A^{\mathcal{I}_2}$ , für alle  $A \in \mathbf{N}_C$
2.  $d_1 \rho d_2$  und  $(d_1, d'_1) \in r^{\mathcal{I}_1}$  impliziert die Existenz eines  $d'_2 \in \Delta^{\mathcal{I}_2}$  mit  
 $d'_1 \rho d'_2$  und  $(d_2, d'_2) \in r^{\mathcal{I}_1}$ , für alle  $r \in \mathbf{N}_R$  T7.1

Beachte: im Ggs. zu Bisimulationen sind Simulationen gerichtet!

# Simulation

Seien  $\mathcal{I}_1$  und  $\mathcal{I}_2$  Interpretationen,  $d_1 \in \Delta^{\mathcal{I}_1}$ ,  $d_2 \in \Delta^{\mathcal{I}_2}$ .

$(\mathcal{I}_1, d_1) \lesssim (\mathcal{I}_2, d_2)$ : es gibt Simulation  $\rho$  von  $\mathcal{I}_1$  nach  $\mathcal{I}_2$  mit  
 $d_1 \rho d_2$  (wir sagen:  $d_2$  *simuliert*  $d_1$ ).

## Theorem 7.3.

Seien  $\mathcal{I}_1, \mathcal{I}_2$  Interpretationen,  $d_1 \in \Delta^{\mathcal{I}_1}$  und  $d_2 \in \Delta^{\mathcal{I}_2}$ .

Wenn  $(\mathcal{I}_1, d_1) \lesssim (\mathcal{I}_2, d_2)$ , dann gilt für alle  $\mathcal{EL}$ -Konzepte  $C$ :  
 $d_1 \in C^{\mathcal{I}_1}$  impliziert  $d_2 \in C^{\mathcal{I}_2}$ .

# Simulation

Intuitiv: Wenn  $(\mathcal{I}_1, d_1) \preceq (\mathcal{I}_2, d_2)$  und  $(\mathcal{I}_2, d_2) \preceq (\mathcal{I}_1, d_1)$ , dann kann  $\mathcal{EL}$  nicht zwischen  $d_1$  und  $d_2$  “unterscheiden”.

Natürlich sind Bisimulation und wechselseitige Simulation nicht dasselbe:

**Lemma 7.4.**

Es gibt  $(\mathcal{I}_1, d_1)$  und  $(\mathcal{I}, d_2)$  so dass

- $(\mathcal{I}_1, d_1) \preceq (\mathcal{I}_2, d_2)$  und  $(\mathcal{I}_2, d_2) \preceq (\mathcal{I}_1, d_1)$
- $(\mathcal{I}_1, d_1) \not\sim (\mathcal{I}_2, d_2)$

T7.2

Man kann nun wieder Nicht-Ausdrückbarkeitsresultate zeigen:

**Lemma 7.5.**

Das  $\mathcal{ALC}$ -Konzept  $\forall r.A$  ist nicht in  $\mathcal{EL}$  ausdrückbar.

T7.2 cont

# EL

In  $\mathcal{EL}$  ist Erfüllbarkeit kein interessantes Schlussfolgerungsproblem:

## Lemma 7.6

Jedes  $\mathcal{EL}$ -Konzept ist erfüllbar bzgl. jeder TBox.

T7.3

Darum konzentrieren wir uns auf Subsumtion

Wir gehen in 2 Schritten vor:

- ohne TBox: kanonische Modelle für Konzepte
- mit TBox: kanonische Modelle für TBoxen



# Subsumtion ohne TBox

# Subsumtion ohne TBox

Subsumtion  $C \sqsubseteq D$  zwischen  $\mathcal{EL}$ -Konzepten  $C, D$  gilt im Prinzip genau dann, wenn man  $D$  syntaktisch "in  $C$  wiederfindet"

Z.B.:  $C = A \sqcap B$

$$\sqcap \exists r. (\exists s. A \sqcap \exists s. B)$$

$$\sqcap \exists r. (A \sqcap \exists r. B)$$

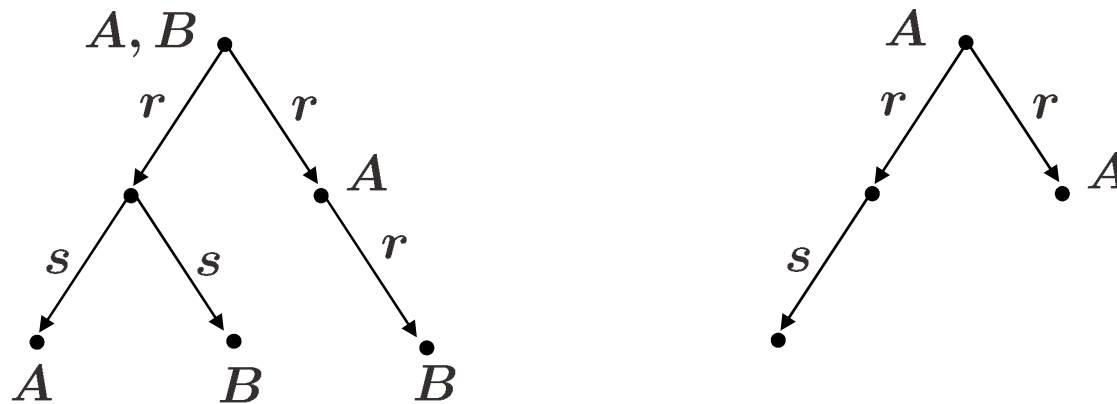
$$D = A$$

$$\sqcap \exists r. \exists s. \top$$

$$\sqcap \exists r. A$$

Konzepte dargestellt als Bäume:

"Wiederfinden" entspricht Simulation von  $D$ -Baum in  $C$ -Baum (Richtung!)



## Subsumtion ohne TBox

Ohne formale Definition:

Wir fassen diese Bäume als Interpretation auf; Baum für Konzept  $C$  nennen wir das *kanonische Modell*  $\mathcal{I}_C$  für das Konzept  $C$ .

Damit ist auch klar, was eine Simulation zwischen solchen Bäumen ist.

Die Wurzel von  $\mathcal{I}_C$  bezeichnen wir mit  $d_W$ .

Die zentrale Eigenschaft kanonischer Modelle:

**Lemma 7.7.**

Für alle  $\mathcal{EL}$ -Konzepte  $C$ , Interpretation  $\mathcal{I}$  und  $e \in \Delta^{\mathcal{I}}$  gilt:

$e \in C^{\mathcal{I}}$  gdw.  $(\mathcal{I}_C, d_W) \lesssim (\mathcal{I}, e)$ .

T7.4

## Subsumtion ohne TBox

Da trivialerweise  $(\mathcal{I}_C, d_W) \simeq (\mathcal{I}_C, d_W)$ , folgt daraus:

**Lemma 7.8.**

$d_W \in C^{\mathcal{I}_C}$ , also ist  $\mathcal{I}_C$  tatsächlich Modell von  $C$ .

Zentrale Beobachtung für unseren Subsumptions-Algorithmus:

**Lemma 7.9.**

Für alle  $\mathcal{EL}$ -Konzepte  $C, D$  gilt:  $C \sqsubseteq D$  gdw.  $(\mathcal{I}_D, d_W) \preceq (\mathcal{I}_C, d_W)$ .

T7.5

**Theorem 7.10.**

Subsumtion in  $\mathcal{EL}$  kann in polynomieller Zeit entschieden werden:

- konstruiere  $\mathcal{I}_C$  und  $\mathcal{I}_D$  in polynomieller Zeit;
- überprüfe in polynomieller Zeit, ob  $(\mathcal{I}_D, d_W) \preceq (\mathcal{I}_C, d_W)$

T7.6

# Subsumtion mit TBox

# Subsumtion mit TBox

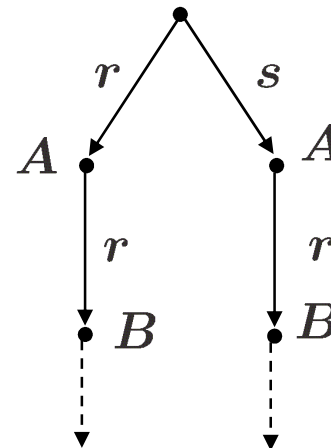
Der beschriebene Algorithmus kann im Prinzip auf TBoxen erweitert werden

Problem: kanonische Modelle werden unendlich groß

Z.B.:  $C = \exists r.A \sqcap \exists s.A$

TBox

$\{A \sqsubseteq \exists r.A\}$



Man verwendet daher eine kompakte (polynomiell große), aber nicht mehr baumförmige Version des unendlichen kanonischen Modells

# Subsumtion mit TBox

Zwei vereinfachende Annahmen:

- Algorithmus entscheidet Subsumtion zwischen Konzeptnamen aus  $\mathcal{T}$ :

O.B.d.A. denn  $\mathcal{T} \models C \sqsubseteq D$  gdw.  $\mathcal{T}' \models A_C \sqsubseteq A_D$

mit  $\mathcal{T}' = \mathcal{T} \cup \{A_C \equiv C, D \equiv A_D\}$

- $\mathcal{T}$  enthält nur Inklusionen der Form

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

wobei  $A, A_1, \dots, A_n$  Konzeptnamen oder  $\top$

T7.7

Dann:

Kanonisches Modell  $\mathcal{I}_{\mathcal{T}}$  für die TBox  $\mathcal{T}$ , verwendet für *alle* Subsumtionen

$\mathcal{T} \models A \sqsubseteq B$ , mit  $A$  und  $B$  Konzeptnamen

## Subsumtion mit TBox

Sei  $\mathcal{T}$  eine  $\mathcal{EL}$ -TBox in der beschriebenen Normalform

Folgender Algorithmus konstruiert Folge  $\mathcal{I}_0, \mathcal{I}_1, \dots, \mathcal{I}_k$  von Interpretationen,  $\mathcal{I}_k$  ist dann das gesuchte kanonische Modell  $\mathcal{I}_{\mathcal{T}}$ .

$\mathcal{I}_0$  ist definiert wie folgt:

- $\Delta^{\mathcal{I}_0} = \{d_A \mid A \text{ Konzeptname in } \mathcal{T}\} \cup \{d_{\top}\}$ ;
- $A^{\mathcal{I}_0} = \{d_A\}$  für alle Konzeptnamen  $A$ ;
- $r^{\mathcal{I}_0} = \emptyset$  für alle Rollennamen  $r$ .



## Subsumtion mit TBox

$\mathcal{I}_{i+1}$  erhält man aus  $\mathcal{I}_i$  durch (einmaliges) Anwenden einer der folgenden Regeln:

R1 Wenn  $d \in (A_1 \sqcap \dots \sqcap A_n)^{\mathcal{I}_i}$ ,  $A_1 \sqcap \dots \sqcap A_n \sqsubseteq A \in \mathcal{T}$  und  $d \notin A^{\mathcal{I}_i}$   
dann  $\mathcal{I}_{i+1}$  wie  $\mathcal{I}_i$ , aber  $A^{\mathcal{I}_{i+1}} = A^{\mathcal{I}_i} \cup \{d\}$

R2 Wenn  $d \in A^{\mathcal{I}_i}$ ,  $A \sqsubseteq \exists r.B \in \mathcal{T}$  und  $(d, d_B) \notin r^{\mathcal{I}_i}$   
dann  $\mathcal{I}_{i+1}$  wie  $\mathcal{I}_i$ , aber  $r^{\mathcal{I}_{i+1}} = r^{\mathcal{I}_i} \cup \{(d, d_B)\}$

R2 Wenn  $(d, d') \in r^{\mathcal{I}_i}$ ,  $d' \in A^{\mathcal{I}_i}$  und  $\exists r.A \sqsubseteq B \in \mathcal{T}$   
dann  $\mathcal{I}_{i+1}$  wie  $\mathcal{I}_i$ , aber  $B^{\mathcal{I}_{i+1}} = B^{\mathcal{I}_i} \cup \{d\}$

T7.8

# Subsumtion mit TBox

Man sieht leicht:

- Die Vorbedingungen können in Polyzeit geprüft werden
- Es sind max.  $|\Delta^{\mathcal{I}_0}| \cdot |\text{sub}(\mathcal{T})| \leq |\mathcal{T}|^2$  Regelanwendungen möglich.

**Lemma 7.11.**

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

T7.9

Die zentrale Eigenschaft kanonischer Modelle, analog zu Lemma 7.7:

**Lemma 7.12.**

Für alle Modelle  $\mathcal{I}$  von  $\mathcal{T}$ ,  $d \in \Delta^{\mathcal{I}}$  und Konzeptnamen  $A$  gilt:

$d \in A^{\mathcal{I}}$  gdw.  $(\mathcal{I}_{\mathcal{T}}, d_A) \lesssim (\mathcal{I}, d)$ .

T7.10

## Subsumtion mit TBox

Zum Entscheiden von Subsumtion könnte man nun Simulationen zwischen kanonischen Modellen berechnen.

Da wir uns auf Konzeptnamen beschränken, geht es aber auch viel einfacher: Subsumtionen können direkt aus  $\mathcal{I}_{\mathcal{T}}$  "abgelesen werden"

**Lemma 7.13.**

Für alle  $\mathcal{EL}$ -Konzepte  $D$  und Konzeptnamen  $A, B$  in  $\mathcal{T}$  gilt:

$\mathcal{T} \models A \sqsubseteq B$  gdw.  $d_A \in B^{\mathcal{I}_{\mathcal{T}}}$ .

T7.11

Da die Konstruktion von  $\mathcal{I}_{\mathcal{T}}$  nur polynomielle Zeit erfordert:

**Theorem 7.14.**

Subsumtion in  $\mathcal{EL}$  bzgl. TBoxen kann in polynomieller Zeit entschieden werden.

# Vergleich mit Tableau-Algorithmus

Regeln:

- R1 entspricht der Konjunktionsregel, R2 der  $\exists$ -Regel
- R3 hat keine Entsprechung, Tableau-Algorithmus behandelt  $\exists r.A \sqsubseteq B$  über TBox-Normalform als  $\forall r.\neg A \sqcup B$ , also  $\sqcup$ -Regel und  $\forall$ -Regel.

Zentrale Unterschiede:

- Tableau Algorithmus kennt  $\Delta^{\mathcal{I}}$  anfangs nicht, führt Elemente nach Bedarf ein
- Tableau Algorithmus konstruiert Baummodell
- Tableau Algorithmus behandelt Konzeptinklusionen  $C \sqsubseteq D$  nicht als Regeln 'wenn  $C$ , dann  $D$ '

# Erweiterungen von EL

# Erweiterungen von EL

Der Algorithmus kann angepasst werden für  $\mathcal{EL}$  erweitert mit:

- $\perp$
- Range Restrictions  $\top \sqsubseteq \forall r.C$  und Domain Restrictions  $\top \sqsubseteq \forall r^-.C$
- allgemeine Rolleninklusionen  $r_1 \circ \dots \circ r_n \sqsubseteq r$
- ...

Dies (und mehr) ist im OWL EL Profil von OWL2 realisiert.

Viele andere Erweiterungen lassen die Komplexität zurück auf ExpTime springen

Wir betrachten exemplarisch  $\mathcal{ELU}$ , die Erweiterung von  $\mathcal{EL}$  mit  $\sqcup$

$\mathcal{EL}_{\forall}$ , die Erweiterung von  $\mathcal{EL}$  mit  $\forall r.C$

$\mathcal{EL}^{\geq 2}$ , die Erweiterung von  $\mathcal{EL}$  mit  $(\geq 2 r \top)$

# Erweiterungen von EL

Theorem 7.15.

Erfüllbarkeit in  $\mathcal{ELU}_{\perp}$  bzgl. TBoxen ist ExpTime-vollständig.

Beweis: Reduktion von Erfüllbarkeit von Konzeptname  $A$  bzgl.  $\mathcal{ALC}$ -TBox  $\mathcal{T}$

Schritt 1: Ersetze Wertrestriktionen in  $\mathcal{T}$  durch Existenzrestriktionen:

$$\forall r.C \quad \text{wird} \quad \neg \exists r. \neg C$$

Schritt 2: Modifiziere  $\mathcal{T}$  so dass Negation nur vor Konzeptnamen auftritt:

$$A \sqsubseteq \exists s.(B' \sqcup \neg \exists r.B) \quad \text{wird} \quad A \sqsubseteq \exists s.(B' \sqcup \neg X)$$
$$X \equiv \exists r.B$$

( $X$  neuer Konzeptname)

# Erweiterungen von EL

Theorem 7.15.

Erfüllbarkeit in  $\mathcal{ELU}_{\perp}$  bzgl. TBoxen ist ExpTime-vollständig.

Beweis: Reduktion von Erfüllbarkeit von Konzeptname  $A$  bzgl.  $\mathcal{ALC}$ -TBox  $\mathcal{T}$

Schritt 3: Entferne Negation vollständig aus  $\mathcal{T}$ :

- ◆ Ersetze jedes  $\neg X$  durch  $\overline{X}$ ,  $\overline{X}$  neuer Konzeptname
- ◆ Erzwingte korrektes Verhalten von  $\overline{X}$ :

$$\begin{aligned} \top &\sqsubseteq X \sqcup \overline{X} \\ X \sqcap \overline{X} &\sqsubseteq \perp \end{aligned}$$

$\mathcal{T}'$  sei die resultierende  $\mathcal{ELU}_{\perp}$ -TBox.

Lemma 7.16

T7.12

Für alle Konzeptnamen  $A$  gilt:  $A$  erfüllbar bzgl.  $\mathcal{T}$  gdw.  $A$  erfüllbar bzgl.  $\mathcal{T}'$ .



# Erweiterungen von EL

Theorem 7.17.

Subsumtion in  $\mathcal{ELU}$  bzgl. TBoxen ist ExpTime-vollständig.

Beweis: Reduktion von Erfüllbarkeit von Konzeptname  $A$  bzgl.  $\mathcal{ELU}_\perp$ -TBox  $\mathcal{T}$

Konstruiere  $\mathcal{ELU}$ -TBox  $\mathcal{T}'$ :

- nimm o.B.d.A. an, dass  $\perp$  nur in der Form  $C \sqsubseteq \perp$  vorkommt
- ersetze  $\perp$  durch neuen Konzeptnamen  $L$
- füge hinzu:

$$\exists r.L \sqsubseteq L \text{ für alle Rollennamen } r \text{ in } \mathcal{T}$$

Lemma 7.18

$A$  unerfüllbar bzgl.  $\mathcal{T}$  gdw.  $\mathcal{T}' \models A \sqsubseteq L$ .

T7.13

# Erweiterungen von EL

$\mathcal{EL}^\forall$  ist  $\mathcal{EL}$  erweitert um  $\forall r.C$

Theorem 7.19. In  $\mathcal{EL}^\forall$  ist Subsumtion bzgl. TBoxen ExpTime-vollständig.

Beweis: Reduktion von Subsumtion zwischen Konzeptnamen bzgl.  $\mathcal{ELU}$ -TBox  $\mathcal{T}$

Wir können annehmen, dass Disjunktion nur in den folgenden Formen vorkommt:

$$\begin{array}{c}
 A_1 \sqcup A_2 \sqsubseteq A \quad \text{und} \quad A \sqsubseteq B_1 \sqcup B_2 \\
 \underbrace{\hspace{10em}} \\
 = A_1 \sqsubseteq A, A_2 \sqsubseteq A
 \end{array}
 \quad \begin{array}{c}
 \text{ersetze in } \mathcal{T}' \text{ durch} \\
 A \sqcap \exists r.T \sqsubseteq B_1 \\
 A \sqcap \forall r.X \sqsubseteq B_2
 \end{array}
 \quad r, X \text{ neu}$$

Lemma 7.20

$\mathcal{T} \models A \sqsubseteq B$  gdw.  $\mathcal{T}' \models A \sqsubseteq B$ . T7.14

# Erweiterungen von EL

$\mathcal{EL}^{\geq 2}$  ist  $\mathcal{EL}$  erweitert um  $(\geq 2 r \top)$

Theorem 7.21. In  $\mathcal{EL}^{\geq 2}$  ist Subsumtion bzgl. TBoxen ExpTime-vollständig.

Beweis: Reduktion von Subsumtion zwischen Konzeptnamen bzgl.  $\mathcal{ELU}$ -TBox  $\mathcal{T}$

Wir können annehmen, dass Disjunktion nur in den folgenden Formen vorkommt:

$$\begin{array}{ccc}
 \underbrace{A_1 \sqcup A_2 \sqsubseteq A}_{=} & \text{und} & A \sqsubseteq B_1 \sqcup B_2 \\
 = A_1 \sqsubseteq A, A_2 \sqsubseteq A & & \text{ersetze in } \mathcal{T}' \text{ durch} \\
 & & A \sqsubseteq \exists r.X \sqcap \exists r.Y \\
 & & A \sqcap \exists r.(X \sqcap Y) \sqsubseteq B_1 \quad r, X, Y \text{ neu} \\
 & & A \sqcap (\geq 2 r) \sqsubseteq B_2
 \end{array}$$

Lemma 7.22

$\mathcal{T} \models A \sqsubseteq B$  gdw.  $\mathcal{T}' \models A \sqsubseteq B$ .

# Erweiterungen von EL

Erweiterung von  $\mathcal{EL}$  ist *konvex* wenn für alle TBoxen  $\mathcal{T}$  und Konzepte  $C, D_1, D_2$ :

$$\mathcal{T} \models C \sqsubseteq D_1 \sqcup D_2 \quad \text{impliziert} \quad \mathcal{T} \models C \sqsubseteq D_1 \sqcup D_2 \quad \text{für ein } i \in \{1, 2\}$$

$\mathcal{EL} + \forall r.C$  ist nicht konvex:

$$\mathcal{T} \models \exists r.T \sqcup \forall r.X, \text{ aber } \mathcal{T} \not\models \exists r.T \text{ und } \mathcal{T} \not\models \forall r.X$$

Unsere Beweise zeigen: jede nicht-konvexe Erweiterung von  $\mathcal{EL}$  ist ExpTime-hart

Aber auch konvexe Erweiterungen sind leider nicht zwangsläufig in PTIME:

Zum Beispiel ist  $\mathcal{ELI}$  ( $\mathcal{EL}$  erweitert mit  $\exists r^-.C$ ) konvex,  
aber EXPTIME-vollständig.

# Diskussion

Die  $\mathcal{EL}$ -Familie von BLen:

- Erlaubt Schlußfolgern in polynomieller Zeit
- Es gibt verschiedene Reasoner wie CEL, SNOROCKET und CB
- Skaliert auch auf große Terminologien wie SNOMED CT  
( $> 400.000$  Konzepte, wird in  $< 2$  Min. klassifiziert)
- Die Beantwortung konjunktiver Anfragen ist NP-vollständig,...
- ...kann aber skalierbar mit normalen relationalen Datenbanksystemen implementiert werden