

Kapitel 5

Komplexität

Ziel des Kapitels

Automatisches Schlussfolgern spielt zentrale Rolle für BLen:

- ermöglicht die Entwicklung intelligenter Anwendungen
- die Ausdruckstärke von BLen ist stark darauf zugeschnitten

Wichtig für automatisches Schlussfolgern:

1. Entscheidbarkeit der relevanten Schlussfolgerungsprobleme
2. möglichst geringe Komplexität
3. Algorithmen, die sich in der Praxis performant verhalten

Dieses Kapitel: 2

Komplexität

Komplexität mit generellen TBoxen,
obere Schranke

Obere Schranke

Wir wollen zeigen:

Theorem 5.1. In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten bzgl. genereller TBoxen EXPTIME -vollständig.

Mit Lemma 2.8: Subsumtion und Äquivalenz EXPTIME -vollständig.

Wir beginnen mit oberer Schranke (Enthaltensein in EXPTIME):

- wir verwenden ein Verfahren aus der Modallogik: Typelimination [Pratt78]
- basiert auf *syntaktischem* Typ-Begriff

Syntaktische Typen

Wir nehmen an, dass

- das Eingabe-Konzept C_0 in NNF ist
- die Eingabe-TBox die Form $\{\top \sqsubseteq C_{\mathcal{T}}\}$ hat mit $C_{\mathcal{T}}$ in NNF.

Definition 5.2. (Typ)

Ein *Typ* für C_0 und \mathcal{T} ist Teilmenge $t \subseteq \text{sub}(C_0, \mathcal{T})$ so dass

1. $A \in t$ gdw. $\neg A \notin t$ für all $A \in \text{sub}(C_0, \mathcal{T})$;
2. $C \sqcap D \in t$ impliziert $C \in t$ und $D \in t$;
3. $C \sqcup D \in t$ impliziert $C \in t$ oder $D \in t$;
4. $C_{\mathcal{T}} \in t$;

T5.1

Typelimination

Generelle Idee der Typelimination bei Eingabe C_0 , \mathcal{T} :

- generiere alle Typen für C_0 und \mathcal{T} (exponentiell viele)
- eliminiere wiederholt Typen, die in keinem Modell von \mathcal{T} vorkommen können
- überprüfe, ob ein Typ überlebt hat, der C_0 enthält
- antworte "ja", wenn dem so ist, sonst "nein"

Schlechte Typen

Formalisierung von "Typen, die in keinem Modell vorkommen können"

Definition 5.3. (schlecht)

Sei Γ Typenmenge und $t \in \Gamma$.

Dann ist t *schlecht in Γ* wenn es ein $\exists r.C \in t$ gibt so dass

für alle $t' \in \Gamma$: $\{C\} \cup \{D \mid \forall r.D \in t\} \not\subseteq t'$.

Intuitiv: Typ ist schlecht wenn er eine existentielle Restriktion enthält,
für die es keinen "Zeugen" gibt.

T5.1 cont

Typelimination

```
define procedure  $\mathcal{ALC}$ -Elim( $C_0, \mathcal{T}$ )
  berechne  $\Gamma_0$ : Menge aller Typen für  $C_0$  und  $\mathcal{T}$ 
   $i := 0$ 
  repeat
     $i := i + 1$ 
     $\Gamma_i := \{t \in \Gamma_{i-1} \mid t \text{ nicht schlecht in } \Gamma_{i-1}\}$ 
  until  $\Gamma_i = \Gamma_{i-1}$ 
  if es gibt  $t \in \Gamma_i$  mit  $C_0 \in t$  then
    return true
  else return false
```

T5.1 cont

Terminierung und Korrektheit

Proposition 5.4.

\mathcal{ALC} -Elim(C_0, \mathcal{T}) terminiert nach $2^{\mathcal{O}(|C_0|+|\mathcal{T}|)}$ Schritten. T5.2

Proposition 5.5.

\mathcal{ALC} -Elim(C_0, \mathcal{T})=true gdw C_0 erfüllbar bzgl. \mathcal{T} . T5.3

Theorem 5.6.

In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten bzgl. genereller TBoxen in EXPTIME.

Typelimination

Bemerkungen:

- Typelimination ist verwandt mit Filtration, z.B.:
Filtrieren des Modells im Beweis von Prop 5.5 gibt selbes Modell
- Typelimination kann leicht auf *ALCI*, *ALCQ* erweitert werden
- auch in *ALCI*, *ALCQ* ist also Erfüllbarkeit bzgl. genereller TBoxen in EXPTIME
- Typelimination für *ALCQI* auch möglich, aber aufwändiger (fehlende endliche Modelleigenschaft); trotzdem in EXPTIME .

Zusammenhang mit Tableau-Algorithmen

Offensichtliche Entsprechungen:

- \neg -Regel, \sqcup -Regel, TBox-Regel finden sich wieder in der Definition eines Typs;
- \exists -Regel und \forall -Regel finden sich wieder in Def. von “schlecht”
- Freiheit von offensichtlichen Widersprüchen findet sich wieder in der Definition eines Typs;

Unterschiede:

- Tableau-Algorithmus benötigt im Worst Case dreifach exponentielle Laufzeit
- Typelimination benötigt im Best Case exponentielle Laufzeit
- Im Korrektheitsbeweis des Tableau-Algorithmus wird grösseres Modell konstruiert

Komplexität

Komplexität mit generellen TBoxen,
untere Schranke

Untere Schranke

Standard-Ansatz zum Beweis von EXPTIME-Härte:

Reduktion des Wortproblems für polynomiell platzbeschränkte, alternierende Turing Maschinen.

Wir reduzieren stattdessen ein spieltheoretisches Problem

(mit obigem verwandt, aber intuitiver)

ExpTime Spiele

- Zwei Spieler spielen auf gegebener aussagenlogischer Formel φ
- Jede Variable in φ gehört entweder Spieler 1 oder Spieler 2
- Das Spiel beginnt auf einer gegebenen Anfangsbelegung der Variablen
- Spieler 1 beginnt, die Spieler wechseln sich ab
- In jedem Zug ändert Spieler Wahrheitswert einer seiner Variablen;
es ist erlaubt, zu passen und eine Variable mehrfach zu ändern
- Spieler 1 gewinnt wenn φ jemals wahr wird
(egal, welcher Spieler gezogen hat)
- Spieler 2 gewinnt, wenn das Spiel unendlich weitergeht
ohne dass φ wahr wird

T5.4

ExpTime Spiele

Definition 5.7.

- *Spiel*: Tupel $(\varphi, \Gamma_1, \Gamma_2, \pi_0)$
 Γ_1, Γ_2 Partitionierung der Variablen in φ , π_0 Anfangsbelegung
- *Konfiguration*: Paar (i, π) mit $i \in \{1, 2\}$ aktiver Spieler und π Belegung
- π ist *j-Variation* von π' ($j \in \{1, 2\}$) wenn $\pi = \pi'$ oder π und π' unterscheiden sich nur in einer Variablen $p \in \Gamma_j$

π *j*-Variation von π' : Spieler *j* kann π in π' transformieren.

Das hier relevante Entscheidungsproblem bezieht sich auf

Gewinnstrategien für Spieler 2.

Gewinnstrategie

Intuitiv:

- eine Gewinnstrategie sagt Spieler 2 nach jedem möglichen Spielverlauf wie er spielen muss um zu gewinnen.
- wenn Spieler 2 eine Gewinnstrategie hat, so kann er das Spiel gewinnen

Gewinnstrategie

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 $\ell(v)$ zuweist so dass

- (a) Wurzel beschriftet mit $(1, \pi)$;
- (b) wenn $\ell(v) = (2, \pi)$, dann hat v Nachfolger v' mit $\ell(v') = (1, \pi')$, wobei π' 2-Variation von π ;
- (c) wenn $\ell(v) = (1, \pi)$, dann hat v Nachfolger $v_0, \dots, v_{|\Gamma_1|}$ mit $\ell(v_i) = (2, \pi_i)$ wobei $\pi_0, \dots, \pi_{|\Gamma_1|}$ alle existierenden 1-Variationen von π ;
- (d) wenn $\ell(v) = (i, \pi)$, dann $\pi \not\equiv \varphi$

T5.5

Gewinnstrategie

Beachte:

- man kann auch Gewinnstrategien für Spieler 1 definieren
dann ist jedes Spiel *determiniert*
- man kann zeigen, dass die Existenz einer Gewinnstrategie die Existenz einer *gedächtnislosen* Gewinnstrategie impliziert
(Bei der der nächste Zug nicht von der Spielhistorie abhängt)

ExpTime Spiele

Definition 5.9.

$Spiel_1$ ist das folgende Problem: gegeben Spiel $(\varphi, \Gamma_1, \Gamma_2, \pi_0)$, entscheide ob Spieler 2 eine Gewinnstrategie hat.

Theorem 5.10.

$Spiel_1$ ist EXPTIME-vollständig.

EXPTIME-Härte von Erfüllbarkeit in \mathcal{ALC} bzgl. generellen TBoxen:

Beweis per Reduktion von $Spiel_1$

ExpTime-Härte

Gegeben Spiel $S = (\varphi, \Gamma_1, \Gamma_2, \pi_0)$, konstruiere Konzept C_S und TBox \mathcal{T}_S so dass:

Spieler 2 hat Gewinnstrategie in S gdw. C_S erfüllbar bzgl. \mathcal{T}_S .

Idee:

(Baum)-Modelle von C_S und \mathcal{T}_S kodieren Gewinnstrategien

ExpTime-Härte

Sei $\Gamma_1 = \{p_0, \dots, p_{m-1}\}$ und $\Gamma_2 = \{p_m, \dots, p_{n-1}\}$, mit $m \leq n$

Signatur von C_S und \mathcal{T}_S :

- Rollenname r für Kanten im Baum
- Konzeptname W für die Wurzel
- Konzeptnamen P_0, \dots, P_{n-1} für die Variablen
- Konzeptnamen S_1, S_2 für aktiven Spieler
- Konzeptnamen V_0, \dots, V_{n-1} für Variable, deren Wert zum Erreichen der aktuellen Konfiguration geändert wurde.

ExpTime-Härte

1. Anfangskonfiguration ist korrekt:

$$W \sqsubseteq S_1 \sqcap \prod_{i < n, \pi_0(p_i)=0} \neg P_i \sqcap \prod_{i < n, \pi_0(p_i)=1} P_i$$

2. Wenn Spieler 1 am Zug gibt es $m+1$ Nachfolger:

$$S_1 \sqsubseteq \exists r. (\neg V_0 \sqcap \dots \sqcap \neg V_{n-1}) \sqcap \prod_{i < m} \exists r. V_i$$

3. Wenn Spieler 2 am Zug, gibt es einen Nachfolger:

$$S_2 \sqsubseteq \exists r. (\neg V_0 \sqcap \dots \sqcap \neg V_{n-1}) \sqcup \bigsqcup_{n \leq i < m} \exists r. V_i$$

4. Es ändert sich höchstens eine Variable pro Zug:

$$\top \sqsubseteq \prod_{i < j < n} \neg (V_i \sqcap V_j)$$

ExpTime-Härte

5. Ausgewählte Variable ändern ihren Wahrheitswert:

$$\top \sqsubseteq \prod_{i < n} ((P_i \rightarrow \forall r.(V_i \rightarrow \neg P_i)) \sqcap (\neg P_i \rightarrow \forall r.(V_i \rightarrow P_i)))$$

6. Alle anderen Variablen behalten ihren Wert:

$$\top \sqsubseteq \prod_{i < n} ((P_i \rightarrow \forall r.(\neg V_i \rightarrow P_i)) \sqcap (\neg P_i \rightarrow \forall r.(\neg V_i \rightarrow \neg P_i)))$$

7. Die Spieler wechseln sich ab:

$$S_1 \sqsubseteq \forall r.S_2, \quad S_2 \sqsubseteq \forall r.S_1, \quad S_1 \sqsubseteq \neg S_2$$

8. Die Formel φ ist immer falsch:

$$\top \sqsubseteq \neg \varphi$$

Korrektheit

Setze $C_S = W$.

Lemma 5.11.

Spieler 2 hat Gewinnstrategie in S gdw. C_S erfüllbar bzgl. \mathcal{T}_S .

T5.6

Theorem 5.12.

In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten bzgl. genereller TBoxen EXPTIME-hart.

Zusammen mit Theorem 5.6: EXPTIME-Vollständigkeit (Theorem 5.1)

Komplexität

Komplexität ohne TBoxen
obere Schranke

Obere Schranke

Wir wollen zeigen:

Theorem 5.13. In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten (ohne TBoxen) $PSPACE$ -vollständig.

Mit Lemma 2.8: Subsumtion und Äquivalenz $PSPACE$ -vollständig.

Wir beginnen mit oberer Schranke (Enthaltensein in $PSPACE$),
benutzen ein Verfahren aus der Modallogik: K-Worlds

Baummodelle

Mehr zu Baummodellen:

- Wenn C erfüllbar, dann hat C Baummodell (Theorem 3.4)
- mit genereller TBox \mathcal{T} kann es sein, dass alle Baummodelle unendlich sind:

$$A \text{ erfüllbar bzgl. } \mathcal{T} = \{A \sqsubseteq \exists r.A\}$$

- ohne TBox gibt es stets ein Baummodell, dessen Tiefe polynomiell in $|C|$ beschränkt ist

Es genügt, die Existenz solcher Modelle zu überprüfen.

(Dasselbe gilt übrigens für azyklische TBoxen)

ALC-Worlds

In PSpace:

- ein polynomiell tiefer Baum ist exponentiell gross
- gesamtes Modell im Speicher: nicht PSPACE
- stattdessen: prüfe Existenz des Baumes mittels Depth-first Traversal, halte zu jeder Zeit nur einen Pfad des Baumes im Speicher

Wir entwickeln nicht-deterministischen Algorithmus, verwenden

Theorem 5.14. (Savitch)

$\text{PSPACE} = \text{NPSPACE}$.

ALC-Worlds

Wir nehmen an, dass Eingabe C_0 in NNF.

Wir verwenden wieder Typen, definieren diese jedoch differenzierter.

Zur Erinnerung:

Rollentiefe $rd(C)$ von Konzepten $C \in \text{sub}(C_0)$ 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 4.9. Für alle $C \in \text{sub}(C_0)$ gilt $rd(C) \leq |C|$.

ALC-Worlds

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 5.16 (*i*-Typ)

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

1. $A \in t$ impliziert $\neg A \notin t$
2. $C \sqcap D \in t$ impliziert $C \in t$ und $D \in t$
3. $C \sqcup D \in t$ impliziert $C \in t$ oder $D \in t$

T5.7

ALC-Worlds

define procedure \mathcal{ALC} -Worlds(C_0)

$i := \text{rd}(C_0)$

rate $t \subseteq \text{sub}_i(C_0)$ mit $C_0 \in t$

recurse(t, i, C_0)

define procedure recurse(t, i, C_0)

if t kein i -Typ für C_0 then

return false

for all $\exists r.C \in t$ do

$S := \{C\} \cup \{D \mid \forall r.D \in t\}$

rate $t' \subseteq \text{sub}_{i-1}(C_0)$ mit $S \subseteq t'$

if recurse($t', i - 1, C_0$) = false then

return false

return true

T5.7 cont

Terminierung und Korrektheit

Proposition 5.17.

\mathcal{ALC} -Worlds(C_0) terminiert und benötigt polynomiellen Platz (in $|C_0|$).

T5.8

Proposition 5.18.

\mathcal{ALC} -Worlds(C_0)=true gdw C_0 erfüllbar.

T5.9

Theorem 5.19.

In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten in PSPACE.

Zusammenhang mit Tableau-Algorithmen

Offensichtliche Entsprechungen:

- \top -Regel und \perp -Regel finden sich wieder in Definition von i -Typ;
- \exists -Regel und \forall -Regel finden sich wieder in rekursivem Aufruf
- Freiheit von offensichtlichen Widersprüchen findet sich wieder in der Definition eines i -Typs;
- Korrektheitsbeweise sehr ähnlich.

Unterschiede:

- Tableau-Algorithmus ist deterministisch, hat dafür “teure” \perp -Regel;
- Tableau-Algorithmus ist nicht platzoptimiert.

Erweiterungen von ALC

Bemerkungen:

- \mathcal{ALC} -Worlds kann auf azyklische TBoxen erweitert werden (andere Definition von Rollentiefe benötigt)
- Erfüllbarkeit in \mathcal{ALC} bzgl. azyklischer TBoxen also auch in PSPACE
- \mathcal{ALC} -Worlds kann auf \mathcal{ALCI} , \mathcal{ALCQ} , \mathcal{ALCQI} erweitert werden
- auch in diesen Logiken ist Erfüllbarkeit bzgl. azyklischer/leerer TBoxen also in PSPACE

Komplexität

Komplexität ohne TBoxen
untere Schranke

Untere Schranke

Standard-Ansatz zum Beweis von PSPACE-Härte:

Reduktion des Gültigkeitsproblems für QBFs

(quantifizierte Bool'sche Formeln)

Wir reduzieren stattdessen wieder ein spieltheoretisches Problem

(mit obigem verwandt, aber intuitiver)

PSPACE Spiele

- Zwei Spieler spielen auf gegebener aussagenlogischer Formel φ
- Jede Variable in φ gehört entweder Spieler 1 oder Spieler 2
- Jedem Spieler gehören gleich viele Variablen
- Die Variablen der Spieler sind linear geordnet
- Spieler 1 beginnt, die Spieler wechseln sich ab
- In jedem Zug wählt Spieler Wahrheitswert seiner nächsten Variablen
- Spieler 1 gewinnt wenn φ am Ende wahr ist, sonst gewinnt Spieler 2

T5.10

PSPACE Spiele

Unterschiede zu EXPTIME-Spielen:

- Das Spiel endet immer, die Anzahl der Schritte ist vorbestimmt
- Der Spieler hat keine Freiheit in der Wahl seiner Variablen
- Jede Variable bekommt nur einmal einen Wahrheitswert zugewiesen
- Man darf nicht passen
- Es wird keine Anfangsbelegung benötigt.

PSPACE Spiele

Definition 5.20.

- *Spiel*: Propositionale Formel φ mit Variablen p_1, \dots, p_n ,
 n geradzahlig
- *Konfiguration*: Wort $\pi \in \{0, 1\}^*$

Intuition:

- Variablen p_i mit i ungerade gehören Spieler 1, die anderen Spieler 2
- Konfiguration w ist partielle Wertzuweisung:
 i -tes Symbol in w ist Wert von p_i

Das hier relevante Entscheidungsproblem bezieht sich auf

Gewinnstrategien für Spieler 1.

Gewinnstrategie

Definition 5.21. (Gewinnstrategie)

Gewinnstrategie für Spieler 1 in Spiel φ ist endlicher knotenbeschrifteter Baum (V, E, ℓ) , wobei ℓ jedem Knoten $v \in V$ Konfiguration $\ell(v)$ zuweist so dass

- (a) Wurzel beschriftet mit ε (leere Konfiguration);
- (b) wenn $\ell(v) = w$ mit $|w|$ gerade,
dann hat v Nachfolger v' mit $\ell(v') \in \{t_0, t_1\}$;
- (c) wenn $\ell(v) = w$ mit $|w|$ ungerade und $|w| < n$,
dann hat v Nachfolger v' und v'' mit $\ell(v') = t_0$ und $\ell(v'') = t_1$;
- (d) wenn $\ell(v) = w$ mit $|w| = n$, dann $w \models \varphi$

T5.11

PSpace Spiele

Definition 5.22.

*Spiel*₂ ist das folgende Problem: gegeben Spiel φ , entscheide ob Spieler 1 eine Gewinnstrategie hat.

Theorem 5.23.

*Spiel*₂ ist PSPACE-vollständig.

PSPACE-Härte von Erfüllbarkeit in *ALC* bzgl. leeren TBoxen:

Beweis per Reduktion von *Spiel*₂

PSpace-Härte

Gegeben Spiel φ , konstruiere Konzept C_φ so dass

Spieler 1 hat Gewinnstrategie in φ gdw. C_φ erfüllbar.

Idee:

(Baum)-Modelle von C_φ kodieren Gewinnstrategien

PSpace-Härte

Die Variablen in φ seien p_1, \dots, p_n , n geradzahlig

Signatur von C_φ :

- Rollenname r für Kanten im Baum
- Konzeptnamen P_1, \dots, P_n für die Wahrheitswerte der Variablen in partiellen Wertzuweisungen

Wir schreiben $\forall r^i.C$ für $\underbrace{\forall r. \dots \forall r.C}_{i \text{ mal}}$

C_φ ist Konjunktion mit Konjunkten wie folgt.

PSpace-Härte

1. $|w|$ gerade gdw. Spieler 1 am Zug gdw. Knoten auf Tiefe i , i gerade
Dann gibt es einen Nachfolger, der Wert für P_{i+1} auswählt

$$C_1 := \bigwedge_{i \in \{0, 2, \dots, n-2\}} \forall r^i. \exists r. \top$$

Wert von P_{i+1} wird implizit ausgewählt, da P_{i+1} am Nachfolger entweder wahr oder falsch sein muss.

2. $|w|$ ungerade gdw. Spieler 2 am Zug gdw. Knoten auf Tiefe i , i ungerade
Dann gibt zwei Nachfolger für beide Werte von P_{i+1}

$$C_2 := \bigwedge_{i \in \{1, 3, \dots, n-1\}} \forall r^i. (\exists r. \neg P_{i+1} \sqcap \exists r. P_{i+1})$$

PSpace-Härte

3. Einmal gewählte Wahrheitswerte bleiben erhalten:

$$C_3 := \prod_{1 \leq i \leq j < n} \forall r^j. ((P_i \rightarrow \forall r. P_i) \sqcap (\neg P_i \rightarrow \forall r. \neg P_i))$$

4. An den Blättern ist φ wahr:

$$C_4 := \forall r^n. \varphi$$

Korrektheit

Setze $C_\varphi = C_1 \sqcap C_2 \sqcap C_3 \sqcap C_4$.

Lemma 5.24.

Spieler 1 hat Gewinnstrategie in φ gdw. C_φ erfüllbar.

Theorem 5.25.

In \mathcal{ALC} ist die Erfüllbarkeit von Konzepten bzgl. leerer TBoxen PSPACE-hart.

Zusammen mit Theorem 5.19: PSPACE-Vollständigkeit (Theorem 5.13)

Komplexität

Zusammenfassung + Nachbemerkung

Zusammenfassung

- Erfüllbarkeit in *ALL* EXPTIME-vollständig mit generellen TBoxen
- Erfüllbarkeit in *ALL* PSPACE-vollständig ohne TBoxen / mit azyklischen TBoxen
- Dasselbe gilt für *ALLI*, *ALLQ*, *ALLQI*
- Baummodelle spielen in allen Fällen eine wichtige Rolle
- PSPACE vs. EXPTIME: polynomiell tiefe vs. unendliche Bäume
- *Typen* sind wichtiger Begriff zum Entwickeln von Algorithmen