Struktur Vorlesung

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

Kapitel 5

Komplexität

Ziel des Kapitels

Automatisches Schlussfolgern spielt zentrale Rolle für BLen:

- ermöglicht die Entwicklung intelligenter Anwendungen
- die Ausdrucksstärke 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

Zur Erinnerung

2ExpTime

$\subsetneq$

ExpTime

$\subseteq$

PSpace

$\subseteq$

NP

$\subseteq$

P

2ExpTime-vollständige Probleme sind erwiesenermaßen nicht in Expzeit lösbar

ExpTime-vollständige Probleme sind erwiesenermaßen nicht in Polyzeit lösbar

NP-/PSpace-vollständige Probleme sind wahrscheinlich nicht in Polyzeit lösbar

$\approx$ effizient lösbar

Für alle Klassen ist Härte mittels Polyzeitreduktionen definiert.

Siehe auch Skript zur VL „Theoretische Informatik 1/2“, §18–20:
http://tinyurl.com/ss16-theoinf

Komplexität

Komplexität mit TBoxen,
obere Schranke

Obere Schranke

Wir wollen zeigen:

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

Mit Lemma 2.9: 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 $t \subseteq \mathsf{sub}(C_0, \mathcal{T})$, so dass

1. $A \in t$ gdw. $\neg A \notin t$ — für alle $\neg A \in \mathsf{sub}(C_0, \mathcal{T})$
2. $C \sqcap D \in t$ gdw. $C \in t$ und $D \in t$ — für alle $C \sqcap D \in \mathsf{sub}(C_0, \mathcal{T})$
3. $C \sqcup D \in t$ gdw. $C \in t$ oder $D \in t$ — für alle $C \sqcup D \in \mathsf{sub}(C_0, \mathcal{T})$
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.
- Wenn ja, antworte „erfüllbar“, sonst „unerfüllbar“.

Schlechte Typen

Formalisierung von „Typen, die in keinem Modell vorkommen können“

Definition 5.3 (schlechter Typ)
Sei Γ Typenmenge und $t \in \Gamma$.
Dann ist t *schlecht in* Γ, wenn für ein $\exists r.C \in t$ gilt:

es gibt kein $t' \in \Gamma$ mit $\{C\} \cup \{D \mid \forall r.D \in t\} \subseteq t'$.

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

T5.1 cont.

Typelimination

procedure $\mathcal{ALC}$-Elim$(C_0, \mathcal{T})$

berechne Γ_0: Menge aller Typen für C_0 und $\mathcal{T}$

$i := 0$

repeat

  $i := i + 1$

  $\Gamma_i := \{t \in \Gamma_{i-1} \mid t$ 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 „erfüllbar“

else return „unerfüllbar“

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$(C_0, \mathcal{T})$ antwortet „erfüllbar“ gdw. C_0 erfüllbar bzgl. $\mathcal{T}$. T5.3

Theorem 5.6
In $\mathcal{ALC}$ ist die Erfüllbarkeit von Konzepten bzgl. TBoxen entscheidbar in ExpTime.

Zusammenhang mit Tableau-Algorithmen

Offensichtliche Entsprechungen:

- $\sqcap$-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

Komplexität

Komplexität mit TBoxen,
untere Schranke

Untere Schranke

Standard-Ansatz zum Beweis von ExpTime-Härte:

> Reduktion des Wortproblems für polynomiell platzbeschränkte, alternierende Turingmaschinen.

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 π_0 der Variablen
- Spieler 1 beginnt, die Spieler wechseln sich ab
- In jedem Zug ändert Spieler Wahrheitswert einer seiner Variablen;
  es ist erlaubt, zu passen
- 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 ist j-Variation von π'“ bedeutet:
Spieler j kann π in π' transformieren (oder umgekehrt).

Das hier relevante Entscheidungsproblem bezieht sich auf
Gewinn*strategien* 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 $(1, \pi_0)$;

(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 $v_0, \ldots, v_{|\Gamma_1|}$ mit $\ell(v_i) = (2, \pi_i)$ wobei $\pi_0, \ldots, \pi_{|\Gamma_1|}$ alle existierenden 1-Variationen von π;

(d) wenn $\ell(v) = (i, \pi)$, dann $\pi \not\models \varphi$

T5.5

ExpTime-Spiele als Entscheidungsproblem

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. TBoxen:

Beweis per Reduktion von Spiel_1

ExpTime-Härte

Gegeben Spiel $S = (\varphi, \Gamma_1, \Gamma_2, \pi_0)$, konstruiere (in Polynomialzeit) 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

Details der Reduktion

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

Signatur von C_S und $\mathcal{T}_S$:

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

Details der Reduktion

1. Anfangskonfiguration ist korrekt:

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

2. Wenn Spieler 1 am Zug ist, gibt es $\boldsymbol{m+1}$ Nachfolger:

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

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

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

4. Es ändert sich höchstens eine Variable pro Zug:

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

Details der Reduktion

5. Ausgewählte Variable ändern ihren Wahrheitswert:

$$\top \sqsubseteq \bigsqcap_{i<n} \big(\, (\, \boldsymbol{P_i} \rightarrow \forall \boldsymbol{r}.(\boldsymbol{V_i} \rightarrow \neg \boldsymbol{P_i}) \,) \sqcap (\, \neg \boldsymbol{P_i} \rightarrow \forall \boldsymbol{r}.(\boldsymbol{V_i} \rightarrow \boldsymbol{P_i}) \,) \, \big)$$

6. Alle anderen Variablen behalten ihren Wert:

$$\top \sqsubseteq \bigsqcap_{i<n} \big(\, (\, \boldsymbol{P_i} \rightarrow \forall \boldsymbol{r}.(\neg \boldsymbol{V_i} \rightarrow \boldsymbol{P_i}) \,) \sqcap (\, \neg \boldsymbol{P_i} \rightarrow \forall \boldsymbol{r}.(\neg \boldsymbol{V_i} \rightarrow \neg \boldsymbol{P_i}) \,) \, \big)$$

7. Die Spieler wechseln sich ab:

$$\boldsymbol{S_1} \sqsubseteq \forall \boldsymbol{r}.\boldsymbol{S_2}, \quad \boldsymbol{S_2} \sqsubseteq \forall \boldsymbol{r}.\boldsymbol{S_1}, \quad \boldsymbol{S_1} \sqsubseteq \neg \boldsymbol{S_2}$$

8. Die Formel φ ist immer falsch:

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

Korrektheit der Reduktion

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. 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 sind dann auch
Subsumtion und Äquivalenz PSPACE-vollständig.

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

Baummodelle

Zur Erinnerung:

- Wenn C erfüllbar, dann hat C Baummodell (Theorem 3.4)
- mit 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 durch $|C|$ beschränkt ist

Es genügt, die Existenz solcher Modelle zu überprüfen.

ALC-Worlds: Grundidee

In PSpace:

- ein linear tiefer Baum ist exponentiell groß
- gesamtes Modell im Speicher: nicht PSPACE
- stattdessen: prüfe Existenz des Baumes mittels Tiefensuche; halte zu jeder Zeit nur einen Pfad des Baumes im Speicher

Wir entwickeln nichtdeterministischen Algorithmus, verwenden

Theorem 5.14. (Savitch)
PSPACE = NPSPACE.

Vorbereitungen

Wir nehmen an, dass Eingabe C_0 in NNF.

Wir verwenden wieder Typen, definieren diese jedoch differenzierter.

Zur Erinnerung:

Rollentiefe $\mathsf{rd}(C)$ von Konzepten $C \in \mathsf{sub}(C_0)$ induktiv definiert:

- $\mathsf{rd}(A) = \mathsf{rd}(\neg A) = 0$;
- $\mathsf{rd}(C \sqcap D) = \mathsf{rd}(C \sqcup D) = \max(\mathsf{rd}(C), \mathsf{rd}(D))$;
- $\mathsf{rd}(\exists r.C) = \mathsf{rd}(\forall r.C) = 1 + \mathsf{rd}(C)$.

Lemma 4.9. Für alle $\mathcal{ALC}$-Konzepte C gilt $\mathsf{rd}(C) \leq |C|$.

i-Typen

Definition 5.15. (i-Konzepte)
Für $i \geq 0$ ist die Menge der *i-Konzepte* definiert als:

$$\mathsf{sub}_i(C_0) := \{C \in \mathsf{sub}(C_0) \mid \mathsf{rd}(C) \leq i\}.$$

Definition 5.16 (i-Typ)
Sei $i \geq 0$. *i-Typ* für C_0 ist Teilmenge $t \subseteq \mathsf{sub}_i(C_0)$ so dass

1. $A \in t$ gdw. $\neg A \notin t$ — für alle $\neg A \in \mathsf{sub}_i(C_0)$;
2. $C \sqcap D \in t$ gdw. $C \in t$ und $D \in t$ — für alle $C \sqcap D \in \mathsf{sub}_i(C_0)$;
3. $C \sqcup D \in t$ gdw. $C \in t$ oder $D \in t$ — für alle $C \sqcup D \in \mathsf{sub}_i(C_0)$;

T5.7

ALC-Worlds

procedure $\mathcal{ALC}$-Worlds(C_0)
- $i := \mathsf{rd}(C_0)$
- rate $t \subseteq \mathsf{sub}_i(C_0)$ mit $C_0 \in t$
- recurse(t, i, C_0)

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 $t' \subseteq \mathsf{sub}_{i-1}(C_0)$ 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.

Theorem 5.19.
In $\mathcal{ALC}$ ist die Erfüllbarkeit von Konzepten in PSPACE.

Beweis der Korrektheit

$\mathcal{ALC}$-Worlds(C_0) = true $\Rightarrow$ C_0 erfüllbar

Beweis.

Sei $\mathcal{ALC}$-Worlds(C_0) = true und $T = (V, E, \ell)$ der Rekursionsbaum eines erfolgreichen Laufes, mit Wurzel v_0.

Für jeden Knoten $v \in V_0 \setminus \{v_0\}$ sei $\sigma(v)$ der Rollenname r des Konzeptes $\exists r.C$, für das der Aufruf v gemacht wurde.

Definiere Interpretation $\mathcal{I}$:

$$\begin{aligned} \Delta^{\mathcal{I}} &= V \\ r^{\mathcal{I}} &= \{\, (v, v') \in E \mid \sigma(v') = r \,\} \\ A^{\mathcal{I}} &= \{\, v \mid A \in p_1(v) \,\} \qquad (p_1(v) = \text{1. Parameter in } \ell(v)) \end{aligned}$$

Behauptung: Für alle $v \in V$ und $C \in \text{sub}(C_0)$ gilt:

$C \in p_1(v)$ impliziert $v \in C^{\mathcal{I}}$ T5.9a

Da $C_0 \in p_1(v_0)$, ist auch $v_0 \in C_0^{\mathcal{I}}$; also ist $\mathcal{I}$ ein Modell von C_0.

Beweis der Vollständigkeit

C_0 erfüllbar $\Rightarrow$ $\mathcal{ALC}$-Worlds$(C_0) =$ true

Beweis.

Sei (C_0) erfüllbar.

Sei $\mathcal{I}$ ein Modell von C_0 und $d_0 \in C_0^{\mathcal{I}}$.

Idee:
Verwenden $\mathcal{I}$, um die nichtdeterministischen Entscheidungen von $\mathcal{ALC}$-Worlds(C_0) zu einem erfolgreichen Lauf zu „lenken“. T5.9b

Zusammenhang mit Tableau-Algorithmen

Offensichtliche Entsprechungen:

- $\sqcap$-Regel und $\sqcup$-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 recht ähnlich.

Unterschiede:

- Tableau-Algorithmus ist deterministisch, hat dafür „teure“ $\sqcup$-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 ist 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.

Ankündigung

Anhörung im Berufungsverfahren für die Vertretungsprofessur „Theorie der künstlichen Intelligenz“, **Mi. 15.6. 13:00 Uhr, Cartesium, Rotunde**

- 13:00–13:30 Uhr Probelehrveranstaltung:

  „Entscheidbarkeit und unentscheidbare Probleme“

- 13:45–14:15 Uhr Forschungsvortrag:

  „Der Ausdrucksstarken Zähmung: Grundlagen für effizientes Schlussfolgern in komplexen Modal- und Beschreibungslogiken“

Komplexität

Komplexität ohne TBoxen
untere Schranke

Untere Schranke

Standard-Ansatz zum Beweis von PSPACE-Härte:

Reduktion des Gültigkeitsproblems für QBFs
(quantifizierte Boolesche 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*: Aussagenlogische Formel φ mit Variablen $p_1, \ldots, p_n$,
  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 Belegung:
  i-tes Symbol in w ist Wahrheitswert 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 und $|w| < n$ (also Spieler 1 am Zug), dann hat v Nachfolger v' mit $\ell(v') \in \{w0, w1\}$;

(c) wenn $\ell(v) = w$ mit $|w|$ ungerade (also Spieler 2 am Zug), dann hat v Nachfolger v' und v'' mit $\ell(v') = w0$ und $\ell(v'') = w1$;

(d) wenn $\ell(v) = w$ mit $|w| = n$, dann $w \models \varphi$

T5.11

PSpace-Spiele als Entscheidungsproblem

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 $\mathcal{ALC}$ bzgl. leeren TBoxen:

Beweis per Reduktion von Spiel_2

PSpace-Härte

Gegeben Spiel φ, konstruiere (in Polynomialzeit) Konzept C_φ so dass

Spieler 1 hat Gewinnstrategie in φ gdw. C_φ erfüllbar.

Idee:

(Baum)-Modelle von C_φ kodieren Gewinnstrategien

Details der Reduktion

Die Variablen in φ seien $p_1, \ldots, p_n$, n geradzahlig

Signatur von C_φ:

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

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

C_φ ist Konjunktion mit Konjunkten wie folgt.

Details der Reduktion

1. $|w|$ gerade gdw. Spieler 1 am Zug gdw. Knoten auf Tiefe i, $2 \mid i$
   Dann gibt es einen Nachfolger, der Wert für P_{i+1} auswählt.

$$C_1 := \bigsqcap_{i \in \{0,2,\ldots,n-2\}} \forall r^i.\left(\exists r.\neg P_{i+1} \sqcup \exists r.P_{i+1}\right)$$

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

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

Details der Reduktion

3. Einmal gewählte Wahrheitswerte bleiben erhalten:

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

4. An den Blättern ist φ wahr:

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

Korrektheit der Reduktion

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)

Zusammenfassung

- Erfüllbarkeit in $\mathcal{ALC}$ mit TBoxen ist ExpTime-vollständig.
- Erfüllbarkeit in $\mathcal{ALC}$ ohne TBoxen ist PSpace-vollständig.
- Dasselbe gilt für $\mathcal{ALCI}$, $\mathcal{ALCQ}$, $\mathcal{ALCQI}$.
- Baummodelle spielen in allen Fällen eine wichtige Rolle.
- PSpace vs. ExpTime: polynomiell tiefe vs. unendliche Bäume
- *Typen* sind wichtiger Begriff zum Entwickeln von Algorithmen.

Gründe für Entscheidbarkeit?

Es gab eine Zeitlang Diskussionen darüber, was die beste Erklärung für die Entscheidbarkeit von Modal- und Beschreibungslogiken ist.

- Existenz von Baummodellen
- Einbettbarkeit in das *2-Variablen-Fragment* der Prädikatenlogik
- Einbettbarkeit in das *Guarded Fragment* der Prädikatenlogik

Siehe z. B.:

Erich Grädel: *Why are Modal Logics so Robustly Decidable?*
Current Trends in Theoretical Computer Science 2001: 393–408.

Komplexität

Unentscheidbare Erweiterungen

Konkrete Bereiche

Einige Erweiterungen von $\mathcal{ALC}$, die zunächst vielleicht harmlos erscheinen, können zu Unentscheidbarkeit führen.

Wir betrachten hier beispielhaft *konkrete Bereiche*, die es erlauben, Zahlen, Strings und andere Datentypen zu verwenden.

Definition 5.26. (Konkreter Bereich)
Ein *konkreter Bereich* ist ein Paar $\mathcal{B} = (\Delta^{\mathcal{B}}, \Phi^{\mathcal{B}})$ wobei

- $\Delta^{\mathcal{B}}$ eine Menge von *Werten* ist und
- $\Phi^{\mathcal{B}}$ eine Menge von *Prädikaten*

so dass jedes $P \in \Phi^{\mathcal{B}}$ mit einer Stelligkeit $n \geq 0$ ausgestattet ist und mit einer Extension $P^{\mathcal{B}} \subseteq (\Delta^{\mathcal{B}})^n$.

T5.12

ALC(B)

Definition 5.27. ($\mathcal{ALC}(\mathcal{B})$ Syntax)
Sei $\mathcal{B}$ ein konkreter Bereich. Mit $\mathcal{ALC}(\mathcal{B})$ bezeichnen wir die Erweiterung von $\mathcal{ALC}$ um $\mathcal{B}$, d. h. um

- *Featurenamen* (eine zusätzliche Art von Rolle) und
- die Konstruktoren $\exists R_1, \ldots, R_n.P$ und $\forall R_1, \ldots, R_n.P$

wobei $P \in \Phi^{\mathcal{B}}$ n-stellig ist und die R_i *Rollenkompositionen* der Form

$$r_1; r_2; \cdots ; r_k; f$$

sind mit $r_1, \ldots, r_k$ Rollennamen und f Featurename.

T5.13

ALC(B)

Definition 5.28. ($\mathcal{ALC}(\mathcal{B})$ Semantik)
Eine Interpretation $\mathcal{I}$ ordnet nun zusätzlich jedem Featurenamen f eine *Funktion* $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{B}}$ zu. Für jede Rollenkomposition

$$R = r_1; r_2; \cdots ; r_k; f$$

bezeichnet $R^{\mathcal{I}}$ die Komposition der Interpretationen:

$$R^{\mathcal{I}} = r_1^{\mathcal{I}} \circ r_2^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \circ f^{\mathcal{I}}.$$

Die Semantik der zusätzlichen Konstruktoren ist nun:

$$(\exists R_1, \ldots, R_k.P)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d_1, \ldots, d_k : (d, d_i) \in R_i^{\mathcal{I}} \text{ für } 1 \leq i \leq k \text{ und } (d_1, \ldots, d_k) \in P^{\mathcal{B}}\}$$

$$(\forall R_1, \ldots, R_k.P)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \forall d_1, \ldots, d_k : (d, d_i) \in R_i^{\mathcal{I}} \text{ für } 1 \leq i \leq k \text{ impliziert } (d_1, \ldots, d_k) \in P^{\mathcal{B}}\}$$

T5.14

Inkrementierung

Wir zeigen, dass bereits scheinbar einfache konkrete Bereiche zu Unentscheidbarkeit führen können. Betrachten Bereich $\mathcal{B}_1$ mit:

$$\Delta^{\mathcal{B}_1} = \mathbb{N}$$
$$\Phi^{\mathcal{B}_1} = \{=_0, =, +_1\}$$

wobei $=_0$ einstellig ist und $=$, $+_1$ zweistellig,
mit den natürlichen Extensionen.

Theorem 5.29

Das Erfüllbarkeitsproblem in $\mathcal{ALC}(\mathcal{B}_1)$ ist unentscheidbar.

Der Beweis ist per Reduktion des Halteproblems für 2-Registermaschinen.

2-Registermaschinen

2-Registermaschinen (2RMs) sind ähnlich zu Turingmaschinen:

- es gibt endlich viele Zustände
- statt eines Arbeitsbandes gibt es zwei Register mit Werten aus $\mathbb{N}$
- statt einer Übergangsfunktion gibt es Instruktionen

Die Instruktionen erlauben es,

- ein Register zu inkrementieren
- ein Register auf null zu testen und bei Wert $\neq \mathbf{0}$ zu dekrementieren

Bei der zweiten Art Instruktion hängt der Folgezustand davon ab,
ob das Register null war.

2-Registermaschinen

Definition 5.30

(Deterministische) 2-Registermaschine (2RM) ist Paar $M = (Q, P)$ mit $Q = \{q_0, \ldots, q_\ell\}$ Menge von *Zuständen* und $P = I_0, \ldots, I_{\ell-1}$ *Instruktionsfolge*.

Per Definition ist q_0 Startzustand, q_ℓ Stoppzustand.

Jede Instruktion I_i hat eine der folgenden Formen:

- $I_i = +(p, q_j)$ mit $p \in \{1, 2\}$ Register und q_j Folgezustand: Inkrementierungsanweisung
- $I_i = -(p, q_j, q_k)$ mit $p \in \{1, 2\}$ Register und q_j, q_k Folgezustände: Dekrementierungsanweisung mit Folgezustand q_j wenn Register p den Wert 0 enthält und q_k sonst

2-Registermaschinen

Definition 5.30

Konfiguration ist Tripel (q, m, n) mit q aktueller Zustand und $m, n \in \mathbb{N}$ Registerinhalte. *Konfigurationsübergänge* $(q, m, n) \vdash_M (q', m, n')$ sind wie folgt definiert:

- Wenn $I_i = +(1, q_j)$, dann $(q_i, m, n) \vdash_M (q_j, m+1, n)$
- Wenn $I_i = +(2, q_j)$, dann $(q_i, m, n) \vdash_M (q_j, m, n+1)$
- Wenn $I_i = -(1, q_j, q_k)$, dann $(q_i, 0, n) \vdash_M (q_j, 0, n)$
  und $(q_i, m, n) \vdash_M (q_k, m-1, n)$, falls $m > 0$
- Wenn $I_i = -(2, q_j, q_k)$, dann $(q_i, m, 0) \vdash_M (q_j, m, 0)$
  und $(q_i, m, n) \vdash_M (q_k, m, n-1)$, falls $n > 0$

Berechnung von M auf Eingabe $(m, n) \in \mathbb{N}^2$ ist die eindeutige (längste) Konfigurationsfolge

$$(q_0, m, n) = (p_0, m_0, n_0) \vdash_M (p_1, m_1, n_1) \vdash_M \cdots .$$

Die Reduktion

Das *Halteproblem für 2RMs* ist: gegeben 2RM $\boldsymbol{M}$, entscheide ob $\boldsymbol{M}$ gestartet auf Eingabe $(\mathbf{0}, \mathbf{0})$ anhält (also $\boldsymbol{q_\ell}$ erreicht).

Theorem 5.31

Das Halteproblem für 2RMs ist unentscheidbar.

Für Beweis von Theorem 5.29:

Gegeben 2RM $\boldsymbol{M}$, konstruiere $\mathcal{ALC}(\mathcal{B}_1)$-TBox $\boldsymbol{\mathcal{T}_M}$
und wähle einen Konzeptnamen $\boldsymbol{I}$, so dass:

$$\boldsymbol{M} \text{ hält auf } (\mathbf{0}, \mathbf{0}) \quad \text{gdw.} \quad \boldsymbol{I} \text{ unerfüllbar bzgl. } \boldsymbol{\mathcal{T}_M}$$

Die Reduktion

Wir verwenden die folgenden Symbole:

- Konzeptnamen $Q_0, \ldots, Q_\ell$ für die Zustände $q_0, \ldots, q_\ell$
- Featurenamen f_1 und f_2, um die Registerinhalte zu speichern
- Rollennamen r, um Nachfolgekonfigurationen zu verbinden
- Konzeptnamen I, der die initiale Konfiguration anzeigt

Wir nehmen o. B. d. A. an: $q_0 \neq q_\ell$

Die Reduktion

Die TBox $\mathcal{T}_M$ enthält folgende Konzeptinklusionen:

1. Start in Zustand q_0 und mit Registerwerten 0:

$$I \sqsubseteq Q_0 \sqcap \exists f_1.{=}_0 \sqcap \exists f_2.{=}_0$$

2. Inkrementierung. Für alle $I_i = +(p, q_j)$:

$$Q_i \sqsubseteq \exists r.Q_j \sqcap \forall f_p\,, (r; f_p).{+}_1 \sqcap \forall f_{\overline{p}}\,, (r; f_{\overline{p}}).{=}$$

wobei $\overline{1} = 2$ und $\overline{2} = 1$

3. Dekrementierung. Für alle $I_i = -(p, q_j, q_k)$:

$$Q_i \sqcap \exists f_p.{=}_0 \sqsubseteq \exists r.Q_j \sqcap \forall f_p\,, (r; f_p).{=} \sqcap \forall f_{\overline{p}}\,, (r; f_{\overline{p}}).{=}$$
$$Q_i \sqcap \neg\exists f_p.{=}_0 \sqsubseteq \exists r.Q_k \sqcap \forall (r; f_p)\,, f_p.{+}_1 \sqcap \forall f_{\overline{p}}\,, (r; f_{\overline{p}}).{=}$$

4. Haltezustand wird nie erreicht:

$$\top \sqsubseteq \neg Q_\ell$$

Die Reduktion

Lemma 5.32.

M hält auf $(0, 0)$ gdw. I unerfüllbar bzgl. $\mathcal{T}_M$. T5.16

Andere konkrete Bereiche

Man kann zeigen, dass „=“ für die Reduktion nicht gebraucht wird.

Es führen also schon sehr schwache arithmetische konkrete Bereiche zu Unentscheidbarkeit.

Auch mit Wörtern und Konkatenation ist man sehr schnell unentscheidbar.

Entscheidbar bleibt man aber, wenn man sich auf Vergleichsoperatoren konzentriert wie etwa $\mathbb{N}$ mit $=$, $>$, $<$, $\geq$, $\leq$