Logik Teil 4: Prädikatenlogik zweiter Stufe

Übersicht Teil 4

- Kapitel 4.1: Syntax und Semantik
- Kapitel 4.2: Entscheidbarkeit und Komplexität
- Kapitel 4.3: MSO über linearen Strukturen
- Kapitel 4.4: Temporallogik
- Kapitel 4.5: Logik und Komplexitätstheorie

Kapitel 4.1: Syntax und Semantik

Logik zweiter Stufe

Die Logik zweiter Stufe (SO) erweitert die Logik erster Stufe, behebt die meisten Unzulänglichkeiten in der Ausdrucksstärke

Grundidee:

Man kann nicht nur über Elemente von Strukturen quantifizieren, sondern auch über Mengen von Elementen und Relationen

Zum Beispiel definiert folgende SO-Formel die transitive Hülle von E:

$$\varphi(x,y) = \forall X \; \big(\; (X(\underline{x}) \land \forall z \forall z' \, (X(z) \land E(z,z') \to X(z'))) \\ \to X(\underline{y}) \; \big)$$

„Jede Knotenmenge, die x enthält und unter Nachfolgern abgeschlossen ist, enthält auch y."

Logik zweiter Stufe

Wir werden sehen, dass die Logik zweiter Stufe (SO)

- eine sehr befriedigende Ausdrucksstärke hat
- eng mit anderen Gebieten der Informatik zusammenhängt, insbesondere den formalen Sprachen und der Komplexitätstheorie

Diesen Vorteilen steht aber eine (noch) schlechtere Berechnungskomplexität als in FO gegenüber

SO sollte als interessante Logik zur Definition interessanter Eigenschaften betrachtet werden, weniger als Logik zu Deduktion.

Wir fixieren für jedes $n \geq 1$ eine abzählbar unendliche Menge $\mathrm{VAR}^n = \{X_1, X_2, X_3, \dots\}$ von n-*ären Relationsvariablen*.

Logik zweiter Stufe

Definition SO-Formeln

Sei τ eine Signatur. Die Menge SO(τ) der τ-*Formeln* der *Prädikatenlogik zweiter Stufe* ist induktiv wie folgt definiert:

- sind $t_1, t_2 \in T(\tau)$, dann ist $t_1 = t_2$ eine Formel
- sind $t_1, \ldots, t_n \in T(\tau)$ und $P \in R^n(\tau)$, dann ist $P(t_1, \ldots, t_n)$ eine Formel
- sind $t_1, \ldots, t_n \in T(\tau)$ und $X \in \mathrm{VAR}^n(\tau)$, dann ist $X(t_1, \ldots, t_n)$ eine Formel
- wenn φ und ψ Formeln sind, dann auch $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$
- wenn φ eine Formel ist und $x \in \mathrm{VAR}$, dann sind $\exists x\, \varphi$ und $\forall x\, \varphi$ Formeln
- wenn φ eine Formel ist und $X \in \mathrm{VAR}^n$, dann sind $\exists X\, \varphi$ und $\forall X\, \varphi$ Formeln

Ist die konkrete Signatur unwichtig, so schreiben wir auch einfach SO.

Sprechweisen und Konventionen

- *Atome* haben nun drei mögliche Formen:

  $t = t'$, $P(t_1, \ldots, t_n)$, $X(t_1, \ldots, t_n)$

- Die Quantoren $\exists X$ und $\forall X$ binden genau wie $\exists x$ und $\forall x$ (stärker als $\wedge$ und $\vee$, die wiederum stärker als $\rightarrow$, $\leftrightarrow$).

- Relationsvariablen können ebenso wie Objektvariablen *frei* oder *gebunden* vorkommen.

- Ein *Satz* ist eine Formel ohne freie Variablen *beider* Arten.

Semantik

Wir gehen wieder in zwei Schritten vor: zunächst Terme, dann Formeln

Definition SO-Zuweisung, SO-Semantik

Sei $\mathfrak{A}$ eine τ-Struktur. Eine *SO-Zuweisung in* $\mathfrak{A}$ ist eine Abbildung β, die

- jeder Objektvariablen $x \in$ VAR ein Element $\beta(x) \in A$ und
- jeder n-ären Relationsvariablen $X \in \text{VAR}^n$ eine n-äre Relation $\beta(X) \subseteq A^n$

zuweist. Wie in FO erweitert man β induktiv auf τ-Terme.

Erweiterung der Erfülltheitsrelation $\models$ auf SO:

- $\mathfrak{A}, \beta \models X(t_1, \ldots, t_n)$ gdw. $(\beta(t_1), \ldots, \beta(t_n)) \in \beta(X)$
- $\mathfrak{A}, \beta \models \exists X\, \varphi$ mit $X \in \text{VAR}^n$ gdw. ein $R \subseteq A^n$ existiert mit $\mathfrak{A}, \beta[X/R] \models \varphi$
- $\mathfrak{A}, \beta \models \forall X\, \varphi$ mit $X \in \text{VAR}^n$ gdw. für alle $R \subseteq A^n$ gilt, dass $\mathfrak{A}, \beta[X/R] \models \varphi$

Logik zweiter Stufe – Beispiele

Erreichbarkeit:

$$\varphi(x,y) = \forall X \Big(\big(X(\underline{x}) \wedge \forall z \forall z' \, (X(z) \wedge E(z,z') \rightarrow X(z')) \big) \rightarrow X(\underline{y}) \Big)$$

„Jede Knotenmenge, die x enthält und unter Nachfolgern abgeschlossen ist, enthält auch y."

$\mathsf{EVEN} = \{\mathfrak{A} \mid |A| \text{ geradzahlig } \}$

$$\mathsf{Func}(R) = \forall x \forall y \forall y' \, (\, R(x,y) \wedge R(x,y') \rightarrow y = y' \,)$$

$$\mathsf{Func}^{-}(R) = \forall x \forall y \forall y' \, (\, R(y,x) \wedge R(y',x) \rightarrow y = y' \,)$$

$$\begin{aligned}\varphi = \exists R \big(& \forall x \exists y \, R(x,y) \vee R(y,x) \\ & \wedge \forall x \, (\, \exists y \, R(x,y) \rightarrow \neg \exists y \, R(y,x) \,) \\ & \wedge \mathsf{Func}(R) \wedge \mathsf{Func}^{-}(R) \,\big)\end{aligned}$$

„A kann ohne Überlappungen mit $R \in \mathsf{VAR}^2$ überdeckt werden."

Logik zweiter Stufe – Beispiele

Unendliche Strukturen:

$$
\begin{aligned}
\varphi_\infty = \exists R \Big(\; & \exists x \exists y \, R(x,y) \wedge \\
& \forall x \forall y \, \big(\, R(x,y) \rightarrow \exists z \, R(y,z) \, \big) \wedge \\
& \forall x \, \neg R(x,x) \wedge \\
& \forall x \forall y \forall z \, \big(\, (R(x,y) \wedge R(y,z)) \rightarrow R(x,z) \, \big) \Big)
\end{aligned}
$$

„Man kann eine Teilmenge des Universums in einer irreflexiven, transitiven Ordnung ohne größtes Element anordnen.“

Endliche Strukturen: $\neg\varphi_\infty$

Auch Abzählbarkeit und Überabzählbarkeit sind definierbar.

Löwenheim-Skolem gilt also nicht (weder aufwärts noch abwärts).

Logik zweiter Stufe – Beispiele

Betrachte folgende Menge von SO-Sätzen:

$$\Gamma = \{\neg\varphi_\infty\} \cup \{\varphi_n \mid n \geq 1\}$$

wobei $\varphi_n = \exists x_1 \cdots \exists x_n \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j$

„Die Struktur hat Größe $\geq n$."

Offensichtlich:

- Γ ist unerfüllbar
- Jede endliche Teilmenge $\Delta \subseteq \Gamma$ ist erfüllbar
  (in einer Struktur der Größe max$\{n \mid \varphi_n \in \Delta\}$)

SO hat also nicht die Kompaktheits-Eigenschaft

Logik zweiter Stufe – Beispiele

Betrachte Signatur mit Konstantensymbol 0, unärem Funktionssymbol nf

Die Peano-Axiome (in leicht angepasster Form):

- $\forall x \, \mathsf{nf}(x) \neq 0$
- $\forall x \forall y \, (\mathsf{nf}(x) = \mathsf{nf}(y) \rightarrow x = y)$
- $\forall X \, \Big(\big(X(0) \land \forall x \, (X(x) \rightarrow X(\mathsf{nf}(x))) \big) \rightarrow \forall x \, X(x) \Big)$

Lemma

$\mathfrak{A} \models \Gamma$ gdw. $\mathfrak{A}$ isomorph zu $(\mathbb{N}, \mathsf{nf}, 0)$.

Wegen des aufsteigenden Satzes von Löwenheim-Skolem für FO gibt es keine FO-Formel, die das leistet!

In SO lassen sich basierend auf 0 und nf auch $+$ und $\cdot$ definieren; also lässt sich Arithmetik bis auf Isomorphie definieren!

Einfache Resultate

Folgende Resultate überträgt man leicht von FO nach SO:

- Koinzidenzlemma und Isomorphielemma
- Existenz äquivalenter Formeln in Negationsnormalform und Pränex-Normalform (alle herstellbar in Linearzeit)

Die Dualität der Quantoren gilt auch für SO-Quantoren:

$$\neg\exists R\,\neg\varphi \equiv \forall R\,\varphi$$
$$\neg\forall R\,\neg\varphi \equiv \exists R\,\varphi$$

für Relationsvariablen R beliebiger Stelligkeit

MSO

Für viele Zwecke genügen bereits unäre Relationsvariablen.

Das resultierende MSO hat oft bessere Eigenschaften als volles SO.

Definition: Monadische Logik zweiter Stufe (MSO)

Eine SO-Formel φ heißt *monadisch*,
wenn sie keine Relationsvariablen mit Arität > 1 enthält.

MSO(τ) ist die Menge aller monadischen Formeln aus SO(τ).

Beispiel: Zusammenhang von Graphen ist MSO-ausdrückbar

$$
\begin{aligned}
\varphi = \forall X \Big(\Big(& \exists x\, X(x) \wedge \\
& \forall x \forall y \,\big(\, (X(x) \wedge E(x,y)) \to X(y) \,\big) \wedge \\
& \forall x \forall y \,\big(\, (X(x) \wedge E(y,x)) \to X(y) \,\big) \Big) \\
& \to \forall x\, X(x) \Big)
\end{aligned}
$$

„Jede Zusammenhangs-komponente enthält alle Knoten.“

Kapitel 4.2: Entscheidbarkeit und Komplexität

Auswertungsproblem

Der Algorithmus für das Auswerten von FO-Formeln
kann leicht auf SO-Formeln erweitert werden.

Der erweiterte Algorithmus benötigt

- polynomiell viel Platz für MSO
- exponentiell viel Platz für volles SO

Dieser Platzverbrauch ist optimal, denn

- das MSO-Auswertungsproblem ist PSpace-vollständig
- das SO-Auswertungsproblem ist „ungefähr ExpSpace-vollständig“

Eine Analyse der *Zeit*komplexität zeigt aber wichtige Unterschiede zwischen FO und MSO!

Zur Erinnerung: der Algorithmus für FO

ausw$(\mathfrak{A}, \beta, \varphi)$

case

$\varphi = (t = t')$: return 1 if $\beta(t) = \beta(t')$, else return 0

$\varphi = P(t_1, \ldots, t_k)$: return 1 if $(\beta(t_1), \ldots, \beta(t_k)) \in P^{\mathfrak{A}}$, else return 0

$\varphi = \neg\psi$: return $1-$ausw$(\mathfrak{A}, \beta, \psi)$

$\varphi = \psi \wedge \vartheta$: return min{ausw$(\mathfrak{A}, \beta, \psi)$, ausw$(\mathfrak{A}, \beta, \vartheta)$}

$\varphi = \psi \vee \vartheta$: return max{ausw$(\mathfrak{A}, \beta, \psi)$, ausw$(\mathfrak{A}, \beta, \vartheta)$}

$\varphi = \exists x\, \psi$:

rufe ausw$(\mathfrak{A}, \beta[x/a], \psi)$ für alle $a \in A$

return 1 if ein Ruf erfolgreich, else return 0

$\varphi = \forall x\, \psi$:

rufe ausw$(\mathfrak{A}, \beta[x/a], \psi)$ für alle $a \in A$

return 1 if alle Rufe erfolgreich, else return 0

endcase

Erweiterung des Algorithmus auf SO

Fälle für die SO-Teilformeln:

case

$\varphi = X(t_1, \ldots, t_\ell)$: return 1 if $(\beta(t_1), \ldots, \beta(t_\ell)) \in \beta(X)$,

else return 0

$\varphi = \exists X\, \psi$ wobei X eine ℓ-stellige Relationsvariable:

rufe ausw$(\mathfrak{A}, \beta[X/B], \psi)$ für alle $B \subseteq A^\ell$

return 1 if ein Ruf erfolgreich, else return 0

$\varphi = \forall X\, \psi$ wobei X eine ℓ-stellige Relationsvariable:

rufe ausw$(\mathfrak{A}, \beta[X/B], \psi)$ für alle $B \subseteq A^\ell$

return 1 if alle Rufe erfolgreich, else return 0

endcase

Platzkomplexität von Auswertung

Lemma

Der Algorithmus benötigt

1. polynomiell viel Platz, wenn die Eingabe eine MSO-Formel ist
2. exponentiell viel Platz im Allgemeinen

Das Auswertungsproblem ist damit für MSO PSpace-vollständig, genauso wie für FO.

Warum beruht SQL dann auf FO und nicht auf dem deutlich ausdrucksstärkeren MSO?

Zur Antwort betrachten wir die *Zeit*komplexität des Auswertungsproblems

Zeitkomplexität von Auswertung

Lemma

Bei Eingabe einer Struktur $\mathfrak{A}$ der Größe n und einer Formel φ der Größe k benötigt der Algorithmus

1. Zeit $\mathcal{O}(n^k)$, wenn φ eine FO-Formel ist
2. Zeit $2^{\mathcal{O}(nk)}$, wenn φ eine MSO-Formel ist
3. Zeit $2^{\mathcal{O}(n^{2k})}$ im Allgemeinen.

Diese Komplexitäten kann man (wahrscheinlich) nicht wesentlich verbessern

Beachte: in Anwendungen ist ...

- n meist sehr groß (Datenbank, zu verifizierendes System, ...)
- k meist eher klein (Datenbankanfrage, zu verifizierende Eigenschaft, ...)

Diese Beobachtungen führen zur Idee der *Datenkomplexität*

Datenkomplexität von Auswertung

Bei der *Datenkomplexität* betrachtet man

- nur die Struktur $\mathfrak{A}$ als Eingabe (die *Daten*bank bzw. das System)
- die Formel φ als fixiert, also ist ihre Größe k eine Konstante.

Der Algorithmus hat dann folgende Komplexität:

Lemma

Bei Eingabe einer Struktur $\mathfrak{A}$ der Größe n benötigt der Algorithmus

1. Zeit $\mathsf{poly}(n)$ wenn eine FO-Formel ausgewertet wird
2. Zeit $2^{\mathcal{O}(n)}$ wenn eine MSO-Formel ausgewertet wird
3. Zeit $2^{\mathsf{poly}(n)}$ wenn eine SO-Formel ausgewertet wird

Aus dieser Perspektive:
die Zeitkomplexität von (M)SO ist für praktischen Einsatz zu groß!

Komplexität von Gültigkeit

Gültigkeit ist in SO (und MSO) ein noch schwierigeres Problem als in FO

Theorem

Wenn τ mind. ein binäres Relationssymbol enthält,
ist die Menge der Tautologien in SO(τ) nicht rekursiv aufzählbar.

Auch die erfüllbaren Formeln und unerfüllbaren Formeln
sind natürlich nicht rekursiv aufzählbar; das alles gilt bereits in MSO.

Das bedeutet natürlich:
Theorembeweisen im Sinne von FO ist in SO nicht möglich.

Es gibt trotzdem SO-Beweiser (wie Isabelle/HOL),
die aber vom Benutzer „geleitet“ werden müssen.

Kapitel 4.3: MSO über linearen Strukturen

MSO über linearen Strukturen

Eine wichtige Eigenschaft von MSO ist die Entscheidbarkeit von Erfüllbarkeit, Gültigkeit, etc. über eingeschränkten Strukturklassen:

- endliche und unendliche lineare Strukturen
- endliche und unendliche Baumstrukturen

Diese Strukturen liefern auch einen interessanten Zusammenhang zu den formalen Sprachen, denn

endliche lineare Struktur ≈ Wort im Sinne der formalen Sprachen

Der Einfachheit halber betrachten wir hier nur *endliche* lineare Strukturen, also Wörter

MSO über linearen Strukturen

Definition MSO eines Nachfolgers (S1S)

Die Menge *S1S* der *MSO-Formeln eines Nachfolgers* sind die MSO-Formeln in der folgenden Signatur:

- ein Konstantensymbol „0“
- ein einstelliges Funktionssymbol „s“ (für „successor“, „Nachfolger“)
- ein zweistelliges Relationssymbol „$<$“
- einstellige Relationssymbole $P_1, P_2, P_3, \ldots$

In S1S sind nur Strukturen $\mathfrak{A}$ zugelassen mit $A = \{0, \ldots, n\}$ für ein $n \geq 0$. Die Interpretation aller Symbole außer der P_i ist wie folgt fixiert:

- $0^{\mathfrak{A}} = 0$;
- $s^{\mathfrak{A}}(n) = n + 1$ wenn $n + 1 \in A$, sonst $s^{\mathfrak{A}}(n) = n$
- $<^{\mathfrak{A}} = \{(n, m) \mid n, m \in A \wedge n < m\}$

S1S Beispiel 1

Lineare Strukturen modellieren Läufe von Systemen in der Verifikation

Die Elemente des Universums entsprechen dann Zeitpunkten

Zum Beispiel:

W_i: Prozess i wartet auf kritischen Abschnitt

A_i: Prozess i im kritischen Abschnitt

$i \in \{1, 2\}$

$$\begin{array}{ccccccccccc} 0 & \xrightarrow{s} & 1 & \xrightarrow{s} & 2 & \xrightarrow{s} & 3 & \xrightarrow{s} & 4 & \xrightarrow{s} & 5 \circlearrowright s \\ W_1 & & A_1 & & A_1 & & A_2 & & A_2 & & A_1 \\ & & & & W_2 & & & & W_1 & & \end{array}$$

S1S beschreibt zu verifizierende Eigenschaften, z.B.:

$$\neg\exists x \left(A_1(x) \wedge A_2(x) \right)$$

$$\bigwedge_{i\in\{1,2\}} \forall x \Big(W_i(x) \rightarrow \exists y \left(x < y \wedge A_i(y) \right) \Big)$$

S1S Beispiel 2

Wir beschränken uns auf endlich viele Relationssymbole $P_1, \ldots, P_n$

Für das Alphabet $\Sigma_n := \{0,1\}^n$ gilt die Entsprechung

S1S Struktur ≈ Wort über Σ_n im Sinne der formalen Sprachen

Beispiel für $n = 2$:

$$
\begin{tikzcd}
0 \arrow[r, "s"] & 1 \arrow[r, "s"] & 2 \arrow[r, "s"] & 3 \arrow[r, "s"] & 4 \arrow[loop right, "s"] \\[-20pt]
P_1 & P_1 & & P_2 & P_1 \\[-20pt]
 & P_2 & & & P_2
\end{tikzcd}
$$

entspricht Wort $\binom{1}{0}\binom{1}{1}\binom{0}{0}\binom{0}{1}\binom{1}{1}$

Spezielle Form des Alphabetes keine Einschränkung,

obiges Wort wird z.B. $bdacd$ bei Zuordnung

$$\binom{0}{0} = a \quad \binom{1}{0} = b$$
$$\binom{0}{1} = c \quad \binom{1}{1} = d$$

S1S Beispiel 2

S1S-Strukturen über $P_1, \dots, P_n$ und Wörter über Σ_n sind also genau dasselbe, nur leicht unterschiedlich präsentiert.

Mit dieser Beobachtung werden die S1S-Sätze zum Werkzeug zur Definition von formalen Sprachen

Definition MSO-definierte Sprache

Ein S1S-Satz φ mit Symbolen $P_1, \dots, P_n$ *definiert* folgende Sprache über dem Alphabet $\Sigma_n = \{0, 1\}^n$:

$$L(\varphi) = \{w \in \Sigma_n \mid w \models \varphi\}$$

Welche Klasse von Sprachen lässt sich mittels S1S/MSO definieren?

Büchi-Elgot-Trakhtenbrot

Interessanterweise sind die MSO-definierbaren Sprachen nichts anderes als die regulären Sprachen.

Um zu beweisen, dass jede MSO-definierbare Sprache regulär ist, zeigen wir:

> Jeder S1S-Satz φ kann in einen endlichen Automaten A_φ gewandelt werden, so dass $L(\varphi) = L(A_\varphi)$.

Dies zeigt auch:

> Erfüllbarkeit und Gültigkeit in S1S ist entscheidbar

Denn φ ist erfüllbar gdw. $L(\varphi) \neq \emptyset$,

und das Leerheitsproblem für endliche Automaten ist entscheidbar

Eine technische Anmerkung

In logischen Strukturen darf das Universum per Def. nicht leer sein.

Daher hat das leere Wort *keine* Entsprechung als logische Struktur.

Wir betrachten daher im Folgenden ausschließlich Sprachen, die das leere Wort *nicht* enthalten:

> z.B. ist also mit „reguläre Sprache" gemeint:
> „reguläre Sprache, die nicht das leere Wort enthält".

Aus Sicht der formalen Sprachen ist das Weglassen des leeren Wortes keine wesentliche Änderung.

Büchi-Elgot-Trakhtenbrot

Theorem von Büchi-Elgot-Trakhtenbrot

Für jede formale Sprache L sind äquivalent:

1. L ist regulär
2. $L = L(\varphi)$ für einen S1S-Satz φ

Wir beweisen zunächst 1 $\Rightarrow$ 2:

Übersetzung von endlichen Automaten in „äquivalenten“ S1S-Satz

Büchi-Elgot-Trakhtenbrot

Beweis von $2 \Rightarrow 1$:

zunächst bringen wir den S1S-Satz in eine geeignete Normalform:

- FO-Variablen werden nicht verwendet
- atomare Formeln haben die Form
  - $X \subseteq Y$, mit Semantik $\forall x \, (\, X(x) \rightarrow Y(x) \,)$
  - $\mathsf{succ}(X) = Y$, mit Semantik
    „X und Y sind Einermengen $\{k\}$ und $\{\ell\}$ mit $\ell = k + 1$“

  wobei X und Y Relationsvariablen oder Relationssymbole sind
- (die Symbole 0, $<$, s werden also nicht verwendet)

Lemma

Jeder S1S-Satz kann effektiv in einen äquivalenten Satz in Normalform gewandelt werden.

Büchi-Elgot-Trakhtenbrot

Theorem von Büchi-Elgot-Trakhtenbrot

Für jede formale Sprache L sind äquivalent:

1. L ist regulär
2. $L = L(\varphi)$ für einen S1S-Satz φ

Wir beweisen nun 2 $\Rightarrow$ 1:

Induktive Übersetzung von S1S-Formeln in Normalform
in „äquivalenten" endlichen Automaten (NEA)

Die strukturelle Induktion macht es notwendig, auch Formeln
mit freien Variablen zu betrachten

Diese werden wie die unären Relationssymbole $P_1, P_2, \ldots$ behandelt,

z.B. liefert $\exists Y \big(X \subseteq Y \wedge P_1 \subseteq Y \big)$ Sprache über $\Sigma_2 = \{0,1\}^2$

FO und formale Sprachen

Da MSO-Definierbarkeit genau den regulären Sprachen entspricht,
ist eine natürliche Frage:

Sei F1S die Einschränkung von S1S auf FO
Welche Sprachklasse entspricht den F1S-definierbaren Sprachen?

Definition: sternfreie Sprachen

Die Klasse der *sternfreien Sprachen* über einem Alphabet Σ
ist die kleinste Klasse, so dass:

- $\emptyset$ und $\{a\}$ sind sternfreie Sprachen für alle $a \in \Sigma$.
- wenn L und L' sternfreie Sprachen sind, dann auch
  $\overline{L} = \Sigma^* \setminus L$, $L \cap L'$, $L \cup L'$ und $L \circ L' = \{ww' \mid w \in L, w' \in L'\}$.

Beachte: Im Unterschied zu den regulären Sprachen steht der
Kleene-Stern nicht zur Verfügung, dafür aber Komplement

FO und formale Sprachen

Ohne Beweis:

Theorem

Für jede formale Sprache L sind äquivalent:

1. L ist sternfrei
2. $L = L(\varphi)$ für einen F1S-Satz φ

Dieses Resultat ist erheblich schwieriger zu beweisen als das Theorem von Büchi-Elgot-Trakhtenbrot

Beispiel für eine nicht sternfreie/F1S-definierbare reguläre Sprache:

$$L_{\text{even}} = \{w \in \{0,1\}^* \mid |w| \text{ ist geradzahlig } \}$$

Beweis mittels EF-Spielen möglich

Entscheidbarkeit S1S

Für jeden S1S-Satz φ:

φ erfüllbar gdw. es gibt S1S-Struktur w mit $w \models \varphi$ gdw. $L(A_\varphi) \neq \emptyset$

Da Leerheit von endlichen Automaten entscheidbar:

Theorem

In S1S sind Erfüllbarkeit und Gültigkeit entscheidbar.

Die Komplexität ist jedoch beträchtlich:

- jede Negation: konstruierter NEA wird exponentiell größerer DEA
  jede existentielle Quantifizierung macht aus DEA wieder NEA
- das Entscheidungsverfahren ist daher *nicht-elementar;*
  man kann beweisen, dass es nicht besser geht

Es gibt trotzdem Reasoner für (Varianten von) S1S wie Mona

Entscheidbarkeit S1S

Das Entscheidbarkeitsresultat lässt sich auf einige andere wichtige Strukturklassen übertragen

Theorem

MSO ist über den folgenden Strukturklassen entscheidbar:

- unendliche lineare Strukturen (unendl. Wörter)
- endliche und unendliche Baumstrukturen

Der Beweis ist im Prinzip ähnlich, außer dass

1. man andere Arten von Automaten benötigt, die unendliche Wörter bzw. endliche/unendliche Bäume verarbeiten
2. einige der Konstruktionen erheblich anspruchsvoller werden (insbesondere das Komplementieren von Automaten)

(siehe auch Master-Kurs „Automatentheorie und ihre Anwendungen“)

Kapitel 4.4: Temporallogik

Temporallogik

In der Verifikation verwendet man S1S meist nicht direkt, sondern *spezialisierte Temporallogiken wie LTL, CTL und das µ-Kalkül*

Nochmal das Verifikations-Beispiel:

W_i: Prozess i wartet auf kritischen Abschnitt

A_i: Prozess i im kritischen Abschnitt

$i \in \{1, 2\}$

$$
\begin{tikzcd}
0 \arrow[r, "s"] & 1 \arrow[r, "s"] & 2 \arrow[r, "s"] & 3 \arrow[r, "s"] & 4 \arrow[r, "s"] & 5 \arrow[loop right, "s"] \\
W_1 & A_1 & A_1 & A_2 & A_2 & A_1 \\
 & & W_2 & & W_1 &
\end{tikzcd}
$$

Eigenschaften kann man nun statt in S1S auch z.B. in LTL beschreiben:

$$\neg \exists x \; (\; A_1(x) \wedge A_2(x) \;) \quad \leadsto \quad \neg \Diamond (A_1 \wedge A_2)$$

$$\bigwedge_{i \in \{1,2\}} \forall x \left(W_i(x) \rightarrow \exists y \; (\; x < y \wedge A_i(y) \;) \right) \quad \leadsto \quad \bigwedge_{i \in \{1,2\}} \Box (W_i \rightarrow \Diamond A_i)$$

LTL: Syntax

Wir werfen einen kurzen Blick auf LTL (Linear Temporal Logic)

Fixiere eine abzählbar unendliche Menge TVAR $= \{p_1, p_2, p_3, \dots\}$ von *temporalen Aussagenvariablen*.

Diese Variablen entsprechen den unären Relationssymbolen P_i in S1S.

Sie verhalten sich ähnlich wie Aussagenvariablen der Aussagenlogik, können allerdings zu jedem Zeitpunkt einen unterschiedlichen Wahrheitswert annehmen.

LTL: Syntax

Definition LTL-Syntax

Die Menge der LTL-Formeln ist induktiv definiert wie folgt:

- jede temporale Aussagenvariable ist eine LTL-Formel
- wenn φ und ψ LTL-Formeln sind, dann auch

$\neg\varphi$	
$\varphi \wedge \psi$	
$\varphi \vee \psi$	
$\bigcirc\varphi$	„im nächsten Zeitpunkt φ“
$\Diamond\varphi$	„in Zukunft irgendwann φ“
$\Box\varphi$	„in Zukunft immer φ“
$\varphi\,\mathcal{U}\,\psi$	„φ until ψ“

LTL: Semantik

Die Semantik von LTL basiert üblicherweise auf *unendlichen* linearen Strukturen; wir verwenden hier endliche S1S-Strukturen

LTL Formeln haben einen Wahrheitswert, der *abhängig ist vom Zeitpunkt*

Definition LTL-Semantik

Wir definieren Erfülltheitsrelation $(\mathfrak{A}, n) \models \varphi$ induktiv, für S1S-Struktur $\mathfrak{A}$, Zeitpunkt $n \in A$ und LTL-Formel φ:

- $\mathfrak{A}, n \models p_i$ gdw. $n \in P_i^{\mathfrak{A}}$
- $\mathfrak{A}, n \models \neg\varphi$ gdw. $\mathfrak{A}, n \not\models \varphi$, ähnlich für $\wedge$ und $\vee$
- $\mathfrak{A}, n \models \bigcirc\varphi$ gdw. $n+1 \in A$ und $\mathfrak{A}, n+1 \models \varphi$
- $\mathfrak{A}, n \models \Diamond\varphi$ gdw. $\exists m \in A$ mit $m \geq n$ und $\mathfrak{A}, m \models \varphi$
- $\mathfrak{A}, n \models \Box\varphi$ gdw. $\forall m \in A$ mit $m \geq n$ gilt: $\mathfrak{A}, m \models \varphi$
- $\mathfrak{A}, n \models \varphi \,\mathcal{U}\, \psi$ gdw. $\exists m \in A$, so dass $m \geq n$,
  $\mathfrak{A}, k \models \varphi$ für $n \leq k < m$ und $\mathfrak{A}, m \models \psi$

LTL versus F1S

Beachte:

- Syntaktisch kann man LTL als Erweiterung von Aussagenlogik auffassen
- Semantisch handelt es sich eher um eine Logik erster Stufe, bei der die FO-Variablen aber implizit sind
- Derartige Logiken nennt man auch *Modallogik*

Eine LTL-Formel φ ist *initial äquivalent* zu einer S1S-Formel $\psi(x)$ mit einer freien Variablen wenn $\mathfrak{A}, 0 \models \varphi$ gdw. $\mathfrak{A} \models \psi[0]$ für alle $\mathfrak{A}$.

Lemma

Zu jeder LTL-Formel φ existiert eine initial äquivalente F1S-Formel $\psi(x)$.

LTL versus F1S

Sehr viel überraschender (und schwieriger zu beweisen):

Theorem (Kamp)

Zu jeder F1S-Formel $\varphi(x)$ mit einer freien Variablen existiert eine initial äquivalente LTL-Formel ψ.

Also ist LTL bzgl. Ausdrucksstärke nichts anderes als Logik erster Stufe!

(es folgt auch:
in LTL lassen sich genau die sternfreien Sprachen definieren)

LTL: Komplexität

Bei der Übersetzung F1S $\Rightarrow$ LTL kann die Formel nicht-elementar größer werden.

Diese Differenz in der Knappheit (engl. Succinctness) schlägt sich in der Komplexität von Erfüllbarkeit/Gültigkeit wieder:

Theorem

Das Erfüllbarkeitsproblem und das Gültigkeitsproblem sind

- PSpace-vollständig in LTL
- nicht-elementar in F1S (und S1S)

Das Auswertungsproblem ist für LTL ebenfalls PSpace-vollständig, wie für MSO/S1S und F1S.

Beide Probleme werden einfacher (NP-vollständig), wenn man nicht alle temporalen Operatoren zulässt (z. B. nur $\Diamond$, $\Box$ oder nur $\bigcirc$)

Weitere Verifikationslogiken

Es gibt in der Verifikation noch andere wichtige Logiken, z.B.:

- CTL, CTL* und das μ-Kalkül für Baumstrukturen

  (Branching time: es gibt mehr als eine mögliche Zukunft, z.B. wegen unbekannter Benutzer-/Sensoreingabe)

- das temporale μ-Kalkül für lineare Strukturen (wie LTL)

Insbesondere hat das temporale μ-Kalkül dieselbe Ausdrucksstärke wie S1S

Prädikatenlogik Zweiter Stufe

Kapitel 4.5: Logik und Komplexitätstheorie

Deskriptive Komplexitätstheorie

Es besteht ein enger Zusammenhang zwischen (Fragmenten von) SO und verschiedenen in der Informatik wichtigen Komplexitätsklassen.

Die grundlegende Beobachtung ist:

> Die in existentiellem SO definierbaren Entscheidungsprobleme sind exakt die Probleme in der Komplexitätsklasse NP.

„Existentielles SO" (ESO) meint dabei Formeln der Form

$$\exists X_1 \cdots \exists X_n \, \varphi, \quad X_i \text{ beliebige Arität und } \varphi \text{ FO-Formel}$$

Diese und ähnliche Beobachtungen erlauben ein Studium von komplexitätstheoretischen Fragen wie „P vs NP" mit logischen Mitteln.

Man nennt diese Forschungsrichtung *Deskriptive Komplexitätstheorie.*

Deskriptive Komplexitätstheorie – Beispiel 1

3-Färbbarkeit ist ein wohlbekanntes NP-vollständiges Problem

Definition (3-Färbbarkeit)

Ein ungerichteter Graph $G = (V, E)$ ist *3-färbbar* gdw. es eine Abbildung $f : V \to \{r, g, b\}$ gibt, so dass gilt:

- für alle $\{v_1, v_2\} \in E$ ist $f(v_1) \neq f(v_2)$

3F ist die Klasse aller 3-färbbaren Graphen.

Betrachte
$$\varphi_{\text{3F}} = \exists C_1 \exists C_2 \exists C_3 \Big(\forall x \forall y \big((C_1(x) \lor C_2(x) \lor C_3(x)) \land \bigwedge_{1 \leq i < j \leq 3} \neg(C_i(x) \land C_j(x)) \land E(x,y) \to \bigwedge_{1 \leq i \leq 3} \neg(C_i(x) \land C_i(y)) \big) \Big)$$

Diese ESO-Formel *definiert* 3F:

$G \models \varphi_{\text{3F}}$ gdw. $G \in \text{3F}$ für alle ungerichteten Graphen G

Deskriptive Komplexitätstheorie – Beispiel 2

Das Hamiltonkreis-Problem ist ein weiteres NP-vollständiges Problem

Definition (Hamiltonkreis)

Sei $G = (V, E)$ ein ungerichteter Graph. Ein *Hamiltonkreis* in G ist ein geschlossener Pfad, der jeden Knoten genau einmal enthält.

HK ist die Klasse aller Graphen, die einen Hamiltonkreis enthalten.

Die folgende ESO-Formel definiert HK:

$\varphi_{\text{HK}} = \exists L\, \exists S \Big($ L ist strikte lineare Ordnung $\wedge$

alle Elemente des Universums kommen in L vor $\wedge$

S ist Nachfolgerrelation von L,

verbindet zusätzlich größtes L-Element mit kleinstem $\wedge$

$\forall x \forall y\, \big(\, S(x,y) \rightarrow E(x,y)\, \big) \Big)$

binär

Fagins Theorem

Wenn man mit Turingmaschinen arbeitet, repräsentiert man

- Probleminstanzen als Wörter
- Entscheidungsprobleme als Mengen von Wörtern

Wir stellen hier Logik in den Mittelpunkt, repräsentieren

- Probleminstanzen als endliche relationale Strukturen
- Entscheidungsprobleme als Klassen von solchen Strukturen

Dies stellt keinerlei Einschränkung der Allgemeinheit dar:

jedes Entscheidungsproblem kann sowohl als Menge von Wörtern als auch als Klasse von endlichen Strukturen dargestellt werden

Fagins Theorem

Definition (ESO-definierbare Probleme)

Ein Problem (Klasse von endlichen relationalen Strukturen) K ist *ESO-definierbar*, wenn es einen ESO-Satz φ_K gibt, so dass

$$\mathfrak{A} \in K \text{ gdw. } \mathfrak{A} \models \varphi_K \qquad \text{für alle endlichen Strukturen } \mathfrak{A}$$

Schon gesehen: 3F und HK sind ESO-definierbar

Um präzise machen zu können, was wir damit meinen, dass ein Problem (Klasse von endlichen relationalen Strukturen) in NP ist, müssen wir Probleminstanzen (Strukturen) als Wörter darstellen

Fagins Theorem

Sei $\mathfrak{A}$ eine endliche τ-Struktur mit $A = \{a_1, \dots, a_n\}$.

Wir fixieren eine beliebige lineare Ordnung auf A, z.B. $a_1 \prec \cdots \prec a_n$

Eine einzelne k-stellige Relation $R^{\mathfrak{A}}$ wird wie folgt kodiert:

- betrachte die *lexikographische Ordnung* aller k-Tupel über A:

$$(a_1, \dots, a_1), (a_1, \dots, a_1, a_2), \cdots, (a_n, \dots, a_n, a_{n-1}), (a_n, \dots, a_n)$$

  (entspricht Zählen zur Basis n)

- repräsentiere $R^{\mathfrak{A}}$ als Wort $w_R \in \{0,1\}^*$ der Länge n^k:

$$\text{das } i\text{-te Symbol ist } \begin{cases} 1 & \text{wenn das } i\text{-te } k\text{-Tupel in } R^{\mathfrak{A}} \text{ ist} \\ 0 & \text{sonst} \end{cases}$$

Für $\tau = \{R_1, \dots, R_\ell\}$ und $s_j = \text{Stelligkeit}(R_j)$ repräsentieren wir $\mathfrak{A}$ als Wort

$$w_{\mathfrak{A}} = \underbrace{0^{|A|}1}_{\text{Größe } \mathfrak{A}} \; \underbrace{0^{s_1}1w_{R_1}}_{R_1^{\mathfrak{A}}} \; \underbrace{0^{s_2}1w_{R_2}}_{R_2^{\mathfrak{A}}} \cdots \underbrace{0^{s_\ell}1w_{R_\ell}}_{R_\ell^{\mathfrak{A}}}$$

Fagins Theorem

Beachte:

- das Präfix 0^n1 kodiert die (endliche) Größe von $\mathfrak{A}$
- ohne diese Information kann man aus $w_{R_1} \cdots w_{R_\ell}$ die Relationen $R_1^{\mathfrak{A}}, \dots, R_\ell^{\mathfrak{A}}$ nicht auf eindeutige Weise extrahieren

„Unsere“ Definition von NP ist nun wie folgt:

Ein Problem (Klasse von endlichen relationalen Strukturen) K ist *in NP*, wenn es eine nichtdeterministische Turingmaschine M_K gibt, so dass:

1. $\mathfrak{A} \in K$ gdw. $w_{\mathfrak{A}} \in L(M_K)$
2. bei Eingabe $w_{\mathfrak{A}}$ terminiert M_K in Zeit poly$(|w_{\mathfrak{A}}|)$

Beachte zu Punkt 2: $|w_{\mathfrak{A}}|$ ist polynomiell in der Größe von $\mathfrak{A}$

Fagins Theorem

Theorem (Fagin)

Für jedes Problem K gilt: K ist in NP gdw. K ist ESO-definierbar.

Man sagt auch: ESO *erfasst* (engl. *captures*) NP

Dieses Theorem stellt eine vollkommen maschinenunabhängige Charakterisierung von NP zu Verfügung.

Es reduziert das schwierige Problem der Trennung von Komplexitätsklassen (z. B. P von NP) auf ein rein logisches Problem

Leider ist bis heute im Allgemeinen nicht klar,
welche Logik der Komplexitätsklasse P entspricht.

Fagins Theorem

Theorem (Fagin)

Für jedes Problem K gilt: K ist in NP gdw. K ist ESO-definierbar.

Es folgt leicht aus Fagins Theorem:

Korollar

Für jedes Problem K gilt: K ist in coNP gdw. K ist USO-definierbar.

(USO $=$ universelles SO)

Das „NP vs coNP"-Problem ist also: haben ESO und USO auf endlichen relationalen Strukturen dieselbe Ausdrucksstärke?

Beachte: wenn man NP ≠ coNP zeigen könnte (wie allgemein vermutet), dann folgt daraus auch P ≠ NP!

Fagins Theorem – Beweis

Theorem (Fagin)

Für jedes Problem K gilt: K ist in NP gdw. K ist ESO-definierbar.

Die einfache Richtung des Beweises ist die folgende:

Lemma

Jedes ESO-definierbare Problem K ist in NP.

Mit anderen Worten: ESO-Auswertung ist in NP bzgl. Datenkomplexität.

Strategie für Gegenrichtung:

Zeige, dass es für jede nichtdeterministische Polyzeit-Turingmaschine M einen ESO-Satz φ_M gibt, so dass $\mathfrak{A} \models \varphi_M$ gdw. $w_{\mathfrak{A}} \in L(M)$ für alle $\mathfrak{A}$.

Fagins Theorem – Beweis

Sei $M = (Q, \Sigma, \Gamma, \Delta, q_0, Q_A, Q_R)$, wobei

- $Q = \{q_1, \ldots, q_m\}$ Zustandsmenge, $\Sigma = \{0, 1\}$ Eingabealphabet
- $\Gamma = \{a_1, \ldots, a_p\} \supseteq \Sigma$ Arbeitsalphabet mit Blanksymbol $\perp \in \Gamma$
- $\Delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, R\}$ Übergangsrelation
- $q_0 \in Q$ Startzustand
- $Q_A, Q_R \subseteq Q$ Menge der akzeptierenden bzw. verwerfenden Zustände

Das Arbeitsband ist einseitig unendlich, der Kopf und die Eingabe stehen anfangs ganz links; wir nehmen an, dass M

1. auf jeder Eingabe der Länge n maximal n^k Schritte macht
2. am linken Ende des Bandes niemals einen Schritt nach links versucht
3. bei Zustand in $Q \setminus (Q_A \cup Q_R)$ immer mindestens einen Übergang zur Verfügung hat, bei Zustand in $Q_A \cup Q_R$ jedoch keinen Übergang

Fagins Theorem – Beweis

Die Formel φ_M soll einen akzeptierenden Lauf von M auf der Eingabe simulieren, wenn so ein Lauf existiert

Schwierigkeit:

- uns steht eine Struktur $\mathfrak{A}$ der Größe $|A|$ zur Verfügung
- wir müssen Bandinhalte, Zustände und Kopfpositionen eines Laufes der Länge $|w_{\mathfrak{A}}|^k$ repräsentieren

Lösung:

- wähle ein k' so dass $|A|^{k'} \geq |w_{\mathfrak{A}}|^k$
- repräsentiere Schrittzähler und Bandposition als k'-Tupel über A, verwende beliebige (strikte) lineare Ordnung über $A^{k'}$ zur Repräsentation der „Reihenfolge“

Fagins Theorem – Beweis

Der Satz φ_M hat die Form

$$\exists L \exists T_{a_0} \cdots \exists T_{a_\ell} \exists H_{q_1} \cdots \exists H_{q_m} \, \psi$$

wobei alle Relationsvariablen $(2 \cdot k')$-stellig sind und

- L die strikte lineare Ordnung repräsentiert
- $T_{a_i}(\overline{p}, \overline{t})$ ausdrückt, dass Bandzelle $\overline{p}$ zum Zeitpunkt $\overline{t}$ das Symbol a_i enthält
- $H_{q_i}(\overline{p}, \overline{t})$ ausdrückt, dass M zum Zeitpunkt $\overline{t}$ in Zustand q_i ist und der Kopf sich auf Bandzelle $\overline{p}$ befindet

Die Formel ψ beschreibt nun das Verhalten von M.

Fagins Theorem – Beweis

Die Formel ψ besteht aus folgenden Konjunkten:

- ψ_{lin} drückt aus, dass L lineare Ordnung ist
- ψ_{ok} stellt sicher, dass Bandsymbole und Zustand eindeutig sind
- ψ_{move} erzwingt, dass alle Übergänge Δ entsprechen
- ψ_{acc} fordert, dass ein akzeptierender Zustand erreicht wird
- ψ_{const} stellt sicher, dass sich der Inhalt von Bandzellen, die sich nicht unter dem Kopf befinden, bei einem Übergang nicht ändert
- ψ_{ini} beschreibt die Startkonfiguration: Band beschriftet mit Eingabe gefolgt von Blanks, Kopf ganz links, M in Startzustand

Fagins Theorem – Beweis

Zusammenfassung:

- Die SO-Variablen erlauben es, die Berechnung einer Turing-maschine zu beschreiben
- Die existentielle SO-Quantifizierung von ESO entspricht dem Nichtdeterminismus der TM
- Die Definierbarkeit einer linearen Ordnung erlaubt es, die „Reihenfolge“ der Bandpositionen und Schritte zu repräsentieren
- Es ergeben sich technische Schwierigkeiten aus den unter-schiedlichen Kodierungen von Probleminstanzen als Graphen und Wörter; die lassen sich aber lösen

Deskriptive Komplexitätstheorie

Welche Logik könnte die Komplexitätsklasse P erfassen?

- FO ist viel zu schwach, kann einfache P-Probleme nicht ausdrücken wie EVEN oder Zusammenhang.
- das in diesem Zusammenhang frappierendste Defizit von FO ist: es gibt keinen Rekursionsmechanismus.
- (E)SO erlaubt Rekursion, ist aber offensichtlich zu ausdrucksstark.

Fixpunktoperatoren stellen einen schwächeren Rekursionsmechanismus dar als SO-Quantifikation.

Tatsächlich erfasst *LFP* (die Erweiterung von FO um Fixpunktoperatoren) genau P, wenn man annimmt, dass die Eingabestrukturen mit einer linearen Ordnung $<$ ausgestattet sind.

Deskriptive Komplexitätstheorie

Dumm nur: in vielen natürlichen Problemen wie 3F oder HK gibt es keine „eingebaute Ordnung“

In LFP ist diese Ordnung auch nicht definierbar

Was wie ein kleines technisches Hindernis aussieht, stellt sich als sehr großes Problem heraus

Es wurde sogar vermutet, dass es keine Logik gibt, die P erfasst (basierend auf einer vernünftigen Definition von „Logik“; Gurevich)

Wenn das so ist, ist es schwer zu beweisen, denn es impliziert natürlich $P \neq NP$

Fast fertig für dieses Semester ...

Vielen Dank

für eure Aufmerksamkeit!