

Monadische Logik zweiter Stufe

Sven Linker

24. September 2007

- 1 Motivation
- 2 Monadische Logik zweiter Stufe
- 3 endliche Automaten
- 4 MSO und Automaten
- 5 Zusammenfassung

Motivation

- Verifikation komplexer Systeme durch *model checking*
- Basiert auf Automaten mit unendlicher Eingabe
- Automatenysteme werden groß und für Menschen unübersichtlich
- Formulierung der Systeme durch Logik
- \Rightarrow kürzere, verständlichere Beschreibung

Motivation

Anforderungen an die Logik

- zu jedem Automaten existiert eine äquivalente Formel
 - Übersetzung von Formel in Automat möglich
 - Übersetzung Formel \rightarrow Automat muss algorithmisch durchführbar sein
-
- Durch solche Logik definierte Systeme können durch *model checking* überprüft werden
 - Für monadische Logik zweiter Stufe wurden diese Eigenschaften nachgewiesen [Büchi, 1962].

Monadische Logik zweiter Stufe

Logik erster Stufe über Wörter

- Prädikatenlogische Formeln werden auf Wörtern $w = a_0 \dots a_n$ interpretiert
- Variablen werden mit Wortpositionen, d.h. Zahlen aus $\{0, \dots, n\}$ belegt
- spezielle Prädikate wie Nachfolgerrelation S , etc.

Logik erster Stufe über Wörter: Syntax

$$\varphi ::= x = y \mid S(x, y) \mid x < y \mid Q_a(x) \mid \neg\varphi' \mid \varphi' \wedge \varphi'' \mid \exists\varphi'$$

- φ', φ'' stellen Formeln, x, y Variablen erster Stufe dar.
- S : Nachfolgerrelation
- $<$: natürliche Ordnung der Positionen
- $=$: Gleichheit
- $Q_a(x)$: wahr, wenn an Position x der Buchstabe a steht.

monadische Logik zweiter Stufe (MSO)

- Erweiterung der Logik erster Stufe
- Variablen zweiter Stufe X_1, \dots, X_n werden eingeführt
- werden mit Mengen an Wortpositionen belegt
- Variablen erster Stufe können durch Mengen mit nur einem Element (*singletons*) ersetzt werden.

monadische Logik zweiter Stufe: Wortmodelle

- Wort $w = a_0 \dots a_n \in \mathcal{L} \subset A^*$ wird durch Struktur

$$\underline{w} = (\text{dom}(w), S^w, <^w, (Q_a^w)_{a \in A})$$

repräsentiert

- $\text{dom}(w) = \{0, \dots, n\}$ ist die Menge an Wortpositionen (Domäne)
- S^w ist die Nachfolgerrelation über den Positionen von w
- $<^w$ ist die natürliche Ordnung auf den Positionen
- $(Q_a^w)_{a \in A}$ gibt die Familie an Buchstabenprädikaten an, d.h. $Q_a^w(x)$ ist wahr, wenn $a_x = a$.
- Definition auch für unendlich lange Worte, dann $\text{dom}(w) = \mathbb{N}$.

monadische Logik zweiter Stufe: Wortmodelle

- Belegungen für Interpretation einer Formel nötig
- Belegung der Variablen zweiter Stufe reicht aus, Variablen erster Stufe können als *singletons* aufgefasst werden.
- $(w, P_1, \dots, P_n) \models \varphi(X_1, \dots, X_n), P_1, \dots, P_n \subseteq \text{dom}(w)$
- (w, P_1, \dots, P_n) kann auch als Wort über $\mathcal{L} \times \{0, 1\}^n$ aufgefasst werden.
- (a, c_1, \dots, c_n) an Position p und $c_j = 1$ bedeutet, dass $p \in P_j$

w		a	b	b	a	b	a	\dots
P_1		0	1	0	1	0	1	\dots
P_2		1	0	1	0	1	0	\dots

endliche Automaten

endliche Automaten

Ein endlicher Automat wird durch ein Tupel

$$\mathcal{A} = (Q, A, q_0, \Delta, F)$$

beschrieben

- Q ist die Menge der Zustände
- A ist das Eingabealphabet
- q_0 ist der Startzustand
- Δ ist die Transitionsrelation $\Delta \subseteq Q \times A \times Q$
- F ist eine Menge an Endzuständen

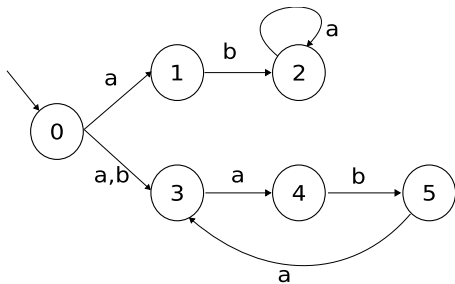
endliche Automaten und endliche Wörter

- Automat akzeptiert ein Wort $w = a_0 \dots a_{n-1}$, wenn es eine Sequenz von Zuständen (*Lauf*) des Automaten ρ gibt, so dass
- $\rho = \rho(0) \dots \rho(n)$ mit $\rho(0) = q_0$, $(\rho(i), a_i, \rho(i+1)) \in \Delta$ für $i < n$ und $\rho(n) \in F$
- Sprache des Automaten $\mathcal{L}(\mathcal{A})$ besteht aus allen von \mathcal{A} akzeptierten Worten

endliche Automaten und unendliche Wörter

- Angegebene Definition gilt nur für endliche Wörter, da ein Endzustand erreicht, d.h. ein letzter Buchstabe gelesen werden muss.
- Änderung der Akzeptanzbedingung lässt Akzeptieren unendlicher Worte zu.
- Menge unendlich oft durchlaufener Zustände:
$$\text{In}(\rho) = \{q \in Q \mid \exists^\omega i: \rho(i) = q\}$$
- Büchi-Bedingung: Menge $F \subseteq Q$, so dass gilt $\text{In}(\rho) \cap F \neq \emptyset$
- Muller-Bedingung: Familie $\mathcal{F} \subseteq \mathcal{P}(Q)$, so dass gilt
$$\bigvee_{F \in \mathcal{F}} \text{In}(\rho) = F$$

Beispiel



- Von Automat zu erkennende Sprache:
 $\mathcal{L}(\mathcal{A}) = (a + b)(aba)^\omega + aba^\omega$
- Büchi-Akzeptanz: $F = \{2, 3\}$ oder $F = \{2, 4\}$, $F = \{2, 5\}$,
 $F = \{1, 2, 3, 4\}$, etc.
- Muller-Akzeptanz: $\mathcal{F} = \{\{2\}, \{3, 4, 5\}\}$

MSO und Automaten

MSO und Automaten auf endlichen Wörtern

J.R. Büchi hat 1960 das folgende Theorem aufgestellt.

Theorem

Eine Sprache mit endlichen Worten ist von einem endlichen Automaten akzeptierbar gdw. sie MSO-definierbar ist. Die Übersetzung von endlichen Automaten nach MSO ist effektiv durchführbar.

Für den Beweis nötig sind

- 1 Entwicklung einer Formel, die Lauf eines Automaten beschreibt
- 2 Konstruktion eines Automaten aus einer MSO-Formel

Entwicklung einer Formel, die Automaten beschreibt

- Idee: für Wort $w = a_0 \dots a_{n-1}$ behauptet Formel φ die Existenz eines Laufs
- Dann gilt: Wenn \mathcal{A} w akzeptiert, existiert ein Lauf, d.h. $\underline{w} \models \varphi$.
- Variablen zweiter Stufe X_i sammeln Wortpositionen, an denen sich \mathcal{A} in Zustand i befindet.

Entwicklung einer Formel, die Automaten beschreibt

Anforderungen an die Formel φ

- 1 Die X_i sollen paarweise disjunkt sein.
- 2 Den Anfang des Laufs bildet der Startzustand
- 3 Nur Übergänge, die durch die Transitionsrelation definiert wurden, sind erlaubt
- 4 Nach dem letzten gelesenen Buchstaben befindet sich \mathcal{A} im Endzustand.

Entwicklung einer Formel, die Automaten beschreibt

- 1 paarweise disjunkt: $\bigwedge_{i \neq j} \forall x \neg (X_i(x) \wedge X_j(x))$
- 2 Startzustand: $\forall x (\text{first}(x) \rightarrow X_0(x))$
- 3 Übergänge: $\forall x \forall y (S(x, y) \rightarrow \bigvee_{(i,a,j) \in \Delta} (X_i(x) \wedge Q_a(x) \wedge X_j(y)))$
- 4 Endzustand: $\forall x (\text{last}(x) \rightarrow \bigvee_{\exists j \in F: (i,a,j) \in \Delta} (X_i(x) \wedge Q_a(x)))$

Konjunktion der Formeln und existentielle Quantifizierung aller Variablen zweiter Stufe ergibt φ .

Konstruktion eines Automaten aus MSO-Formel

Idee der Automatenkonstruktion

- Definition von atomaren Automaten $\mathcal{A}(\varphi)$ für atomare MSO-Formeln
- induktive Kombination der atomaren Automaten zu komplexeren Automaten
- $\neg\varphi$ entspricht Komplementierung des Automaten $\mathcal{A}(\varphi)$
- $\varphi \vee \varphi'$ entspricht Vereinigung der Automaten $\mathcal{A}(\varphi)$ und $\mathcal{A}(\varphi')$
- $\exists X_i(\varphi)$ entspricht Projektion des Automaten $\mathcal{A}(\varphi)$

MSO und Automaten auf unendlichen Wörtern

1962 hat Büchi dieses Ergebnis auf Sprachen mit unendlichen Worten übertragen können.

Theorem (Büchis Theorem)

Eine Sprache mit unendlichen Worten ist Büchi-akzeptierbar gdw. sie MSO-definierbar ist. Die Übersetzung von Büchi-Automaten in MSO-Formeln und umgekehrt ist effektiv.

- Hin- und Rückrichtung fast analog zum Ergebnis auf endlichen Worten
- \Rightarrow : Anpassung der Formel für Akzeptanz eines Wortes
- \Leftarrow : Komplement von Büchi-akzeptierbaren Sprachen abgeschlossen?

Anpassungen des Beweises

- Büchi-Akzeptanz als MSO-Formel:

$$\bigvee_{i \in F} \exists x (X_i(x)) \wedge \forall x (X_i(x) \rightarrow \exists y (x < y \wedge X_i(y)))$$

Anpassungen des Beweises

- Abgeschlossenheit des Komplements über zwei Wege möglich:
 - 1 Form der Sprachen (*ursprünglicher Beweis von Büchi*)
 - 2 Umwandlung nichtdeterministischer Büchi- in deterministische Muller-Automaten (*späterer Ansatz von McNaughton*)
- Hier zweiter Fall betrachten.
- deterministische Muller-Automaten sind bzgl. Komplement abgeschlossen
- nur Modifikation der Akzeptanzbedingung von \mathcal{F} zu $\mathcal{P}(Q) \setminus \mathcal{F}$ nötig

Zusammenfassung

- Klasse der MSO-definierbaren Sprachen gleich Klasse der durch endliche Automaten akzeptierbaren Sprachen
- Umwandlung von MSO-Formeln in äquivalente Automaten möglich
- dadurch: Möglichkeit, durch MSO definierte Systeme mittels *model checking* zu prüfen