Multioperator Weighted Monadic Datalog

Torsten Stüber

Institut Theoretische Informatik
Fakultät Informatik
Technische Universität Dresden
Nöthnitzer Str. 46
01062 Dresden
torsten.stueber@tu-dresden.de

Abstract: In dieser Arbeit stellen wir ein formales Modell zur Verarbeitung von baumstrukturierten Daten vor. Dieses vereint und generalisiert Konzepte von Baumautomaten, Attributgrammatiken, Monadic Datalog und MSO-Logik. Unser Modell verarbeitet Bäume unter Einsatz von *Multioperator-Monoiden* und erhält dadurch eine hohe Ausdrucksstärke und vielseitige Einsetzbarkeit. In unserer Arbeit beschreiben und vergleichen wir vier verschiedene Semantiken sowie zahlreiche Normalformen. Einige der Semantiken sind nur dann einsetzbar, wenn das Modell kein zirkuläres Verhalten aufweist; daher entwickeln wir des Weiteren einen Zirkularitätstests. Wir betrachten beispielhaft einige Instanzen unseres Modells und stellen Bezüge zu existierenden Berechnungsmodellen her.
Diese Kurzfassung ist eine Zusammenfassung der Kapitel der Dissertation.

1 Einführung

Einer der Kernaspekte der Informatik ist die Untersuchung und Entwicklung von Berechnungsmodellen. Berechnung in diesem Sinne ist die Verarbeitung von Eingabe- zu Ausgabedaten. Gewöhnlich sind die zu verarbeitenden Daten nach einem gewissen Schema organisiert, zum Beispiel in einer Baumstruktur. Baumstrukturierte Daten haben viele Anwendungen in der Informatik, beispielsweise im Bereich der semistrukturierten Datenbanken, der formalen Sprachtheorie oder der Übersetzung natürlicher Sprachen.

Anwendungen, bei denen Bäume als Eingabe verwendet werden, lassen sich gemäß der Art der Ausgabe ihrer Verarbeitung einteilen. Die wichtigsten Vertreter einer solchen Einteilung sind:

- *Prüfen einer Eigenschaft*: beispielsweise die Überprüfung darauf, ob der Eingabebaum ein Binärbaum ist,
- *Berechnung eines Zahlenwertes*: beispielsweise das Zählen der Blattknoten eines Baumes,
- *Transformation des Baumes*: beispielsweise die Übersetzung des Ableitungsbaumes

eines englischen Satzes in einen entsprechenden Ableitungsbaum eines deutschen Satzes im Anwendungsbereich der Syntax-basierten maschinellen Übersetzung,

- *Transformation des Baumes bei gleichzeitiger Berechnung eines Zahlenwertes*: beispielsweise die probabilistische Übersetzung eines Ableitungsbaumes im Anwendungsbereich der statistischen maschinellen Übersetzung.

Die Informatik hat eine große Anzahl an Berechnungsmodellen hervorgebracht und untersucht. Gängige Beispiele sind die Turingmaschine, Programmiersprachen, neuronale Netze oder endliche Automaten, also Modelle mit begrenztem Speicher. Forschung im Bereich der Automatentheorie hat zu vielen fruchtbaren Entdeckungen und Anwendungen geführt, da endliche Automaten meist eine einfache, reguläre Struktur besitzen, effizient verarbeiten können, robuste Berechnungsklassen definieren (d.h. vergleichbare Modelle weisen die gleiche Ausdrucksmächtigkeit auf) und oftmals befriedigende Berechnungsmächtigkeiten aufweisen.

Im Folgenden geben wir einen Überblick über endliche Modelle zur Verarbeitung von baumstrukturierten Daten. Wir konzentrieren uns dabei auf zwei Entwicklungszweige solcher Modelle.

Endliche Automaten und Transducer Die Theorie der Baumautomaten [GS97] begann in der Mitte der 1960er Jahre und verallgemeinerte das Konzept der endlichen Zeichenreihenautomaten. Die Semantik eines FTA ist eine Baumsprache; sie besteht aus allen Bäumen, die durch den FTA akzeptiert werden.

Das Konzept des FTA kann auf natürliche Art und Weise durch eine Gewichtung der Transitionen erweitert werden. Das Rechnen mit diesen Gewichten erfordert den Einsatz einer algebraische Struktur; gewöhnlich sind Halbringe [HW98] für diese Anwendung besonders geeignet. Diese Erweiterung führt zu dem Konzept der gewichteten Baumautomaten (*weighted tree automaton* – WTA) [BR82]. Ein gegebener WTA definiert eine Abbildung von der Menge der Eingabebäume in die Trägermenge des Halbrings. Gewichtete Baumautomaten haben viele Anwendungen in der Informatik, zum Beispiel für die Befehlsauswahl in Kompilern, Baummusterabgleich, und die natürliche Sprachverarbeitung [KM09].

WTA können als eine Erweiterung von FTA angesehen werden. Weitere Modelle, die von FTA durch ähnliche Erweiterungen abgeleitet wurden, sind tiefgründig untersucht und fruchtbar angewandt wurden. Wir nennen hier beispielhaft das Modell der Baumübersetzer (*tree transducer* – TT) [Eng75]. Ein TT definiert eine *Baumübersetzung*, also ein Abbildung von Eingabebäumen in Mengen von Ausgabebäumen. Auch TT können durch einen Halbring gewichtet werden. Das führt zu dem Modell der gewichteten Baumübersetzer (*weighted tree transducer* – WTT).

Da FTA, WTA, TT und WTT eine ähnliche Struktur aufweisen, wurde ein vereinheitlichendes Modell entwickelt, welches alle ebengenannten Modelle subsumiert. Dieses Modell, bezeichnet als gewichteter Multioperator-Baumautomat (*weighted multioperator tree automaton* – WMTA), wurde erstmalig von Kuich [Kui99] untersucht und nutzt zur Berechnung eine Algebra namens Multioperator-Monoid (kurz *M-Monoid*). Durch gezielten Einsatz verschiedener M-Monoide kann ein WMTA die Modelle FTA, WTA, TT und WTT

simulieren. Das heißt, dass Untersuchungen und Resultate für WMTA entsprechende Untersuchungen und Resultate für die vier letztgenannten Modelle automatisch beinhalten.

Monadic Datalog *Monadic Datalog* [GK02], ein Fragment von Datalog, ist ein Mittel zum formalen Beschreiben von Baumsprachen und zur Knotenauswahl von Bäumen. Ein Monadic-Datalog-Programm (kurz: MD) besteht im Wesentlichen aus einer Menge von Regeln.

Monadic Datalog kann auf natürliche Art und Weise für Rangbäume als auch für rangfreie Bäume verwendet werden. Damit hat es Anwendungen im Bereich der semistrukturierten Datenbanken und XML. Das große praktische Potential von Monadic Datalog ist wie folgt begründet: (i) die Benutzung von Regeln zur Spezifikation von Eigenschaften von Baumsprachen ist intuitiv und natürlich, (ii) die Zeitkomplexität zur Auswertung eines MD ist linear im Bezug auf die Größe des Eingabebaumes und des betrachteten MD und (iii) die Klasse der Baumsprachen, die mit Monadic Datalog spezifiziert werden können, ist die Klasse der durch FTA erkennbaren Baumsprachen (vgl. [GK02]).

Gewichtetes Monadic Datalog (*weighted monadic datalog* – WMD) ist die durch einen Halbring gewichtete Variante von Monadic Datalog. Es wurde gezeigt, dass WMD strikt ausdrucksstärker als WTA sind und dass es ähnlich wie Monadic Datalog die Eigenschaft hat, in Linearzeit auswertbar zu sein (sowohl in Abhängigkeit von der Größe des Eingabebaums als auch des WMD) (vgl. [SV08]).

Eine weitere Abwandlung von Monadic Datalog, die Monadic-Datalog-Baumübersetzer (kurz MDTT), erlauben die Spezifikation von endlichen Baumübersetzungen (d.h., die Menge der Ausgabebäume für jeden Eingabebaum ist endlich) als auch von unendlichen Baumübersetzungen (d.h., die Menge der Ausgabebäume für einen gegebenen Eingabebaum kann unendlich sein). Es wurde gezeigt, dass MDTT mindestens die Ausdrucksstärke von attributierten Baumübersetzern aufweisen [Fül81] (siehe [BS09]).

Zielstellung dieser Arbeit In dieser Arbeit untersuchen wir eine Generalisierung von MD, WMD und MDTT, die darüber hinaus die Modelle FTA, WTA, TT und WMTA subsumiert. Dieses Modell heißt *Multioperator-gewichtetes Monadic Datalog* (*multioperator weighted monadic datalog* – MWMD). In Tabelle 2 wird dargestellt, wie sich dieses Modell in die Landschaft der wichtigsten obengenannten Modelle eingliedert.

	Baumsprache	Baumreihe	Baumübersetzung	Allgemeine Berechnung
Baumautomaten	FTA	WTA	TT	WMTA
Monadic Datalog	MD	WMD	MDTT	MWMD

Tabelle 1: Eine Übersicht über die wichtigsten genannten Formalismen.

MWMD stellen also ein verallgemeinertes Konzept zum Verarbeiten baumstrukturierter Daten dar. Sie erben dabei die positiven Eigenschaften sowohl von der auf Baumautomaten als auch von der auf Monadic Datalog basierenden Modelle: Auswertbarkeit bei ge-

ringer Zeitkomplexität, einfache Spezifizierbarkeit von Baumsprachen, Baumreihen oder Baumübersetzungen, große Ausdrucksmächtigkeit und ein großes Gebiet von Anwendungen durch die Verwendung von M-Monoiden.

Der universell einsetzbare Formalismus MWMD ermöglicht darüber hinaus eine unifizierte Sichtweise auf die zahlreichen Anwendungen und Eigenschaften konventioneller Modelle zum Verarbeiten von Bäumen und führt diese zu einem Ganzen zusammen. In der Dissertation werden im Speziellen Resultate präsentiert, die belegen, dass eine einheitliche Betrachtung baumbasierter Modelle gewinnbringend ist.

2 Grundlagen

Da MWMD eine Vielzahl an Verarbeitungsmodellen vereinen, basieren sie auf einer großen Zahl von Konzepten der Mathematik und Theoretischen Informatik. In diesem Kapitel besprechen wir die wichtigsten Begriffe, die für das grundlegende Verständnis der nachfolgenden Kapitel notwendig sind. Dabei gehen wir im Speziellen auf Operationen mit unendlicher Stelligkeit ein. Das Kapitel schließt mit einer Einführung in das Gebiet der Hypergraphen und Hyperpfade ab. Dabei richten wir besonderes Augenmerk auf Hyperpfad-Segmente, Abhängigkeitsbeziehungen und Dekomposition sowie Komposition von Hyperpfaden.

3 M-Monoide

Dieses Kapitel behandelt die zentrale algebraische Struktur unseres Formalismus, sogenannte Multioperator-Monoide (kurz *M-Monoide*). Unsere Definition der M-Monoide basiert auf dem Begriff der distributiven M-Monoide (oder distributiven Ω-Monoide), die auf Kuich [Kui99] zurückgehen.

Ein M-Monoid besteht zum einen aus einem kommutativen Monoid und zum anderen aus einer Δ-Algebra (für eine Signatur Δ); das Monoid und die Δ-Algebra sind dabei auf der gleichen Trägermenge definiert. Formal ist ein M-Monoid also ein Tupel $\mathcal{A} = (A, +, \mathbf{0}, \theta)$, wobei $(A, +, \mathbf{0})$ ein kommutatives Monoid und (A, θ) eine Δ-Algebra ist. M-Monoide sind ein essenzieller Bestandteil für die Definition der Semantik eines MWMD-Programms, da deren Auswertung die Monoid-Operation und die Operationen der Δ-Algebra des M-Monoids benutzt.

Es zeigt sich, dass die Operationen eines M-Monoids im Allgemeinen nicht ausreichend sind, um die Semantik eines beliebigen MWMD zu berechnen. Das tritt immer dann ein, wenn der betrachtete MWMD ein zirkuläres Verhalten aufweist. Diesem Problem begegnen wir dadurch, dass wir zwei Erweiterungen von M-Monoiden einführen, mit denen es möglich ist, wohldefinierte Ausgabewerte für zirkuläre MWMD zu berechnen. Diese Erweiterungen heißen *ω-vollständige* und *ω-stetige M-Monoide*. Des Weiteren arbeiten wir Beziehungen zwischen ω-vollständigen und ω-stetigen M-Monoiden heraus.

4 M-gewichtete Monadic-Datalog-Programme

In diesem Kapitel präsentieren wir das Kernmodell der Arbeit und beschreiben dabei detailliert die Syntax und Semantik von MWMD. Die syntaktische Struktur eines MWMD orientiert sich an der Syntax von MDTT [BS09] und WMD [SV08]. Wir beschreiben diese nun im Ansatz.

Sei Σ ein Rangalphabet und Δ eine Signatur. Ein MWMD-Programm *über* Σ *und* Δ ist ein Tripel (P, R, q), wobei P ein Rangalphabet ist, in dem jedes Symbol ein- oder nullstellig ist, R eine endliche Menge und q ein einstelliges Symbol aus P ist. Die Elemente von P heißen *nutzerdefinierte Prädikate*, die Elemente von R *Regeln*, und q ist das *Abfrageprädikat*. Jede Regel r aus R ist von der Form

$$\textit{Kopf} \leftarrow \textit{Rumpf}\,;\,\textit{Guard}\ .$$

Der Kopf der Regel ist dabei ein prädikatenlogisches Atom über einem Prädikat aus P, also von der Form $p()$ für ein nullstelliges p oder $p(x)$ für ein einstelliges p; x ist eine Variable und wird einem vorgegebenen Vorrat an Variablen entnommen. Der Rumpf ist ein Baum mit Symbolen aus Δ, an dessen Blättern zusätzlich prädikatenlogische Atome über P stehen dürfen; diese Atome sind auf gleiche Art und Weise aufgebaut wie der Regelkopf. Der Guard ist eine endliche Menge von Atomen über der Prädikatenmenge sp_Σ:

$$\mathrm{sp}_\Sigma = \{\mathrm{root}^{(1)}, \mathrm{leaf}^{(1)}\} \cup \{\mathrm{label}_\sigma^{(1)} \mid \sigma \in \Sigma\} \cup \{\mathrm{child}_i^{(2)} \mid i \in [\mathrm{maxrk}(\Sigma)]\}\ .$$

Die Definition der Semantik von MWMD ist komplex und stellt einen Schwerpunkt der Dissertation dar. Um eine reichhaltige Theorie der MWMD zu entwickeln, führen wir zwei verschiedene Arten von Semantiken ein, die als *Fixpunktsemantik* und *Hypergraphsemantik* bezeichnet sind. Die Fixpunksemantik hat Ähnlichkeit zur Initial-Algebra-Semantik eines WTA [BR82]; dagegen weist die Hypergraphsemantik Parallelen zur Lauf-Semantik von gewichteten Baumautomaten auf. Die Fixpunktsemantik basiert auf der Semantik von MDTT, WMD und MD, die Hypergraphsemantik ist dagegen ein neuartiges Konzept.

Jede dieser beiden Semantikarten benötigt drei Eingaben: ein MWMD-Programm, einen Eingabebaum und ein M-Monoid. Die Semantiken sind so definiert, dass sie den Eingabebaum gesteuert durch das MWMD-Programm auswerten und dabei die Operationen des M-Monoids anwenden. Die Ausgabe ist schließlich ein Element des M-Monoids. Betrachtet man also ein festes MWMD-Programm und ein festes M-Monoid, dann sind die Semantiken Abbildungen von Eingabebäumen in die Trägermenge des M-Monoids; eine solche Abbildung heißt Baumreihe.

Die Fixpunktsemantik basiert auf der Anwendung eines *Konsequenz-Operators* und geht auf die Definition der Semantik der Logikprogrammierung bzw. Hornklauseln und Monadic Datalog [GK02] zurück. Wir beschreiben nun die grobe Idee der Fixpunktsemantik. Zunächst werden die Variablen jeder Regel r in R durch Knoten des Eingabebaumes auf all solche Weisen instanziiert, dass der Guard von r erfüllt ist; lautet der Guard beispielsweise $\{\mathrm{label}_\sigma(x), \mathrm{child}_2(x, y)\}$, dann darf x nur durch einen mit σ beschrifteten Knoten

und y durch den zweiten Kindknoten von x instanziiert werden. Eine *Interpretation* ordnet jeder Instanz der in den Regeln vorkommenden Atomen ein Element des betrachteten M-Monoids zu. Ausgehend von einer festen Startinterpretation wird durch wiederholtes Anwenden des Konsequenz-Operators schrittweise eine Zielinterpretation berechnet. Der Konsequenz-Operator nimmt dabei Bezug auf die instanziierten Regeln: beispielsweise bedeutet die Regelinstanz

$$a \leftarrow \delta\big(b, c\big)\,;\{\dots\}\,,$$

dass die Konsequenz-Interpretation der Atominstanz a den Wert $\delta\big(I(b), I(c)\big)$ zuordnet, wobei I die aktuelle Interpretation ist und die Operation δ durch die Δ-Algebra des M-Monoids interpretiert wird. Das Ergebnis der Semantik ist dann der Wert, den die Zielinterpretation der Atominstanz zuordnet, die aus dem Abfrageprädikat und dem Wurzelknoten des Eingabebaums besteht.

Die Hypergraphsemantik ordnet einem MWMD-Programm und einem Eingabebaum einen Hypergraphen, den *Abhängigkeitshypergraphen*, zu. Aus diesem Hypergraphen wird eine Menge von Termen über der Signatur Δ extrahiert. Jeder dieser Terme wird anschließend in der Δ-Algebra des betrachteten M-Monoids ausgewertet. Die Ergebnisse der Auswertung für jeden Term werden dann durch das kommutative Monoid verknüpft. Das entstehende Element des M-Monoids ist schließlich das Ergebnis der Hypergraphsemantik.

Für jede der beiden Semantiken definieren wir eine *endliche* und eine *unendliche* Variante. Die endlichen Varianten der Semantiken sind dabei nur auf einer Teilklasse der MWMD-Programme definiert, den *schwach nichtzirkulären* MWMD*-Programmen*. Außerhalb dieser Klasse liegende MWMD-Programme weisen beim Berechnen der Semantik ein zirkuläres Verhalten auf und sind nur mit den unendlichen Varianten der Semantiken anwendbar; diese erfordern dabei allerdings ein ω-stetiges M-Monoid (bei der Fixpunktsemantik) oder ein ω-vollständiges M-Monoid (bei der Hypergraphsemantik) als Eingabe. Insgesamt untersuchen wir also vier verschiedene Semantiken.

Die folgende Tabelle gibt eine Übersicht darüber, welche der vier Semantiken unter welchen Umständen einsetzbar ist.

	Fixpunktsemantik	Hypergraphsemantik
endlich	schwach nichtzirkuläre MWMD beliebige M-Monoide	schwach nichtzirkuläre MWMD beliebige M-Monoide
unendlich	beliebige MWMD ω-stetige M-Monoide	beliebige MWMD ω-vollständige M-Monoide

Tabelle 2: Eine Übersicht über die Semantiken von MWMD-Programmen.

Wir schließen das Kapitel mit einem Vergleich der vier Semantiken ab und erarbeiten hinreichende Bedingungen für ihre Äquivalenz (siehe Theorem 4.53).

5 Normalformen

In diesem Kapitel betrachten wir vier syntaktische Teilklassen der MWMD, sogenannte *eingeschränkte*, *zusammenhängende*, *lokale* und *echte* MWMD. Wir beschreiben diese Klassen nun informell.

Eingeschränkt: die Position der in den Regeln vorkommenden Variablen muss bestimmten strukturellen Eigenschaften genügen.

Zusammenhängend: die Variablen in den Regeln müssen logisch zusammenhängen.

Lokal: die Regeln müssen so definiert sein, dass die Variablen bei der Bildung von Regelinstanzen nur mit direkt benachbarten Knoten des Eingabebaumes instanziiert werden; die Struktur der Regeln ähnelt dabei den Regeln von Attributgrammatiken [Cou84].

Echt: alle nutzerdefinierten Prädikate sind einstellig.

Die Teilklasse der zusammenhängenden MWMD wurde in dieser Form erstmalig von Gottlob und Koch [GK02, Theorem 4.2] für MD eingeführt; sie wurde weiterhin in [SV08, BS09] untersucht. Die anderen drei Klassen wurden in [BS09] für MDTT eingeführt.

Wir untersuchen hinreichende Bedingungen, die eine Übereinstimmung zwischen diesen syntaktischen Klassen garantieren. Es handelt sich dabei also um Bedingungen, unter denen diese Teilklassen (und Schnitte der Teilklassen) Normalformen von MWMD bilden (siehe Theorem 5.8). Dazu analysieren wir, wann ein gegebenes MWMD-Programm in ein semantisch äquivalentes MWMD-Programm transformiert werden kann, der zu einer der genannten Teilklassen gehört. Hierbei ist zu klären, was wir unter „semantischer Äquivalenz“ verstehen. In der Tat gebrauchen wir die stärkste Definition von semantischer Äquivalenz, die in diesem Kontext möglich ist. Genauer gesagt präsentieren wir Konstruktionen, die alle vier im vorherigen Kapitel eingeführten Semantiken erhalten. Dies stellt eine besondere Herausforderung dar und ist im Detail sehr technisch. Die Konstruktionen, die wir in diesem Kapitel vorstellen, basieren auf Konstruktionen in [BS09].

6 Zirkularitätstest

In diesem Kapitel beweisen wir, dass es ein effektives Verfahren gibt, mit dem man entscheiden kann, ob ein MWMD-Programm schwach nichtzirkulär ist (siehe Theorem 6.1). Dieses Resultat ist wichtig, da nur so entschieden werden kann, welche der vier Semantiken für ein MWMD-Programm in einer konkreten Situation anwendbar sind.

Die Definition von schwacher Nichtzirkularität beruht auf dem Begriff der Nichtzirkularität von Attributgrammatiken [Cou84] und WMD [SV08]. Ein Entscheidungsverfahren für die Nichtzirkularität von Attributgrammatiken, bezeichnet als Zirkularitätstest, wurde erstmals durch Knuth [Knu68] untersucht. Es basiert auf einer rekursiven Konstruktion von endlichen Mengen von Graphen, sogenannten IS-Graphen, die auf Zyklen überprüft werden.

In dieser Dissertation verfolgen wir nicht den Ansatz, einen Zirkularitätstest zu entwickeln,

der auf IS-Graphen basiert, da er sich als zu schwierig und komplex erweist. Stattdessen wenden wir die folgende Idee an. Sei M ein MWMD-Programm und L_M die Menge der Eingabebäume, für die M ein zirkuläres Verhalten aufweist. Dann ist M schwach nichtzirkulär genau dann, wenn L_M leer ist. Wir zeigen, dass effektiv eine MSO-Formel [TW68] konstruiert werden kann, die L_M definiert. Das impliziert, dass L_M eine erkennbare Baumsprache ist. Dann folgt die Entscheidbarkeit der schwachen Nichtzirkularität aus der Tatsache, dass das Leerheitsproblem für erkennbare Baumsprachen entscheidbar ist.

7 Gewichtetes Monadic Datalog

In diesem Kapitel zeigen wir, dass MWMD das Konzept der WMD subsumiert. Dazu untersuchen wir die Semantik von MWMD für die Teilklasse von M-Monoiden, die das algebraische Verhalten von Halbringen [HW98] simulieren. Um möglichst starke Resultate zu erhalten, betrachten wir sogar M-Monoide, die starke Bimonoide [DSV10] simulieren. Dieses Kapitel ist eine überarbeitete und erweiterte Version der Arbeit [SV08]; wir merken an, dass in jener Arbeit WMD über Halbringen und rangfreien Bäumen betrachtet wurde; dagegen behandelt diese Dissertation WMD über starken Bimonoiden und Rangbäumen.

Kern unserer Untersuchungen sind die Ausdrucksstärke und die Effizienz der Auswertung von WMD. Im Speziellen vergleichen wir die endliche und unendliche Variante der Semantik von WMD (Lemma 7.15) und zeigen, dass WMD unter Benutzung des Booleschen Halbrings MD simulieren können und dass WMD streng ausdrucksstärker sind als WTA (Theorem 7.18). Wir schließen die Betrachtungen durch einen Beweis dafür ab, dass WMD effizient ausgewertet werden können (Theorem 7.21).

8 Monadic-Datalog-Baumübersetzer

Dieses Kapitel behandelt das Konzept der MDTT. Wir zeigen, dass die Klasse der MDTT in der Klasse der MWMD enthalten ist. Das erreichen wir durch den Einsatz von speziellen M-Monoiden. Diese M-Monoide verhalten sich ähnlich einer Termalgebra und sorgen so zum Beispiel bei der Hypergraphsemantik dafür, dass die Auswertung eines Termes den Term selbst erzeugt. Dies macht deutlich, dass MDTT nichts anderes als MWMD sind, bei denen von einer konkreten semantischen Domäne abstrahiert wurde. Dieses Kapitel ist eine überarbeitete und erweiterte Version der Arbeit [BS09], in der MDTT erstmalig untersucht wurden.

Wir zeigen, dass es eine scharfe Abgrenzung zwischen solchen MDTT gibt, für die die Semantik in linear beschränkter Zeit vollständig berechnet werden kann (diese MDTT sind demnach für praktische Zwecke einsetzbar) und solchen MDTT, für die die Semantik nicht in endlicher Zeit berechnet werden kann. Erstere Sorte von MDTT bezeichnen wir als *ausführbar* und zeigen, dass Ausführbarkeit von MDTT entscheidbar ist (siehe Lemma 8.12).

MDTT und (nichtdeterministische) attributierte Baumübersetzer [Fül81] sind konzeptuell eng verwandt. Wir beweisen, dass attributierte Baumübersetzer und eingeschränkte MDTT die gleiche Ausdrucksmächtigkeit haben (siehe Theorem 8.21).

9 Gewichtete Multioperator-Baumautomaten

Dieses Kapitel behandelt das Konzept der WMTA. Wir zeigen, dass MWMD insbesondere WMTA simulieren können. Dazu definieren wir eine syntaktische Teilklasse von MWMD, so dass sich die MWMD in dieser Klasse exakt wie WMTA verhalten. Dieses Kapitel ist eine überarbeitete Version der wichtigsten Resultate der Arbeiten [SVF09, FSV10]. Wir beschränken uns auf einen Beweis der folgenden beiden Hauptresultate:

1. Wir betrachten M-Monoide, welche gewisse zusätzliche Eigenschaften erfüllen. Wir zeigen, dass die Klasse der Baumreihen, welche durch WMTA über einem solchen M-Monoid erkannt werden, dekomponiert werden kann in (1) die Klasse der Relabelings, gefolgt von (2) der Klasse der charakteristischen Baumtransformationen von erkennbaren Baumsprachen, gefolgt von (3) der Klasse der Baumreihen, die durch Homomorphismus-WMTA erkannt werden. Ein Homomorphismus-WMTA ist ein WMTA mit genau einem Zustand (siehe Theorem 9.17).
2. Wir präsentieren eine alternative Charakterisierung der Klasse der Baumreihen, die durch WMTA erkannt werden. Diese Charakterisierung basiert auf sogenannten *M-Ausdrücken*, einer neuen Art von gewichteter MSO-Logik. Diese Charakterisierung ist ein Büchi-artiges Resultat [Büc60] für die Klasse der Baumreihen, die durch WMTAerkannt werden (siehe Theorem 9.26).

Literatur

[BR82] J. Berstel und C. Reutenauer. Recognizable formal power series on trees. *Theoret. Comput. Sci.*, 18(2):115–148, 1982.

[BS09] M. Büchse und T. Stüber. Monadic Datalog Tree Transducers. In A. H. Dediu, A.-M. Ionescu und C. Martín-Vide, Hrsg., *LATA*, Jgg. 5457 of *Lecture Notes in Computer Science*, Seiten 267–278. Springer, 2009.

[Büc60] J. R. Büchi. Weak Second–order arithmetic and finite automata. *Zeitschr. für math. Logik und Grundl. der Mathem.*, 6:66–92, 1960.

[Cou84] B. Courcelle. Attribute grammars: definitions, analysis of dependencies, proof methods. In B. Lorho, Hrsg., *Methods and tools for compiler construction*, Seiten 81–102. Cambridge University Press, 1984.

[DSV10] M. Droste, T. Stüber und H. Vogler. Weighted automata over strong bimonoids. *Inform. Sci.*, 180:156–166, 2010.

[Eng75] J. Engelfriet. Bottom-up and top-down tree transformations - a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[FSV10] Z. Fülöp, T. Stüber und H. Vogler. A Büchi-Like Theorem for Weighted Tree Automata over Multioperator Monoids. *Theory of Computing Systems*, Seiten 1–38, 2010.

[Fül81] Z. Fülöp. On attributed tree transducers. *Acta Cybernet.*, 5:261–279, 1981.

[GK02] G. Gottlob und C. Koch. Monadic Queries over Tree-Structured Data. In *LICS '02: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, Seiten 189–202, Washington, DC, USA, 2002. IEEE Computer Society.

[GS97] F. Gécseg und M. Steinby. Tree Languages. In G. Rozenberg und A. Salomaa, Hrsg., *Handbook of Formal Languages*, Jgg. 3, Kapitel 1, Seiten 1–68. Springer-Verlag, 1997.

[HW98] U. Hebisch und H.J. Weinert. *Semirings - Algebraic Theory and Applications in Computer Science*. World Scientific, Singapore, 1998.

[KM09] K. Knight und J. May. Applications of Weighted Automata in Natural Language Processing. In M. Droste, W. Kuich und H. Vogler, Hrsg., *Handbook of Weighted Automata*, Kapitel 14. Springer-Verlag, 2009.

[Knu68] D.E. Knuth. Semantics of context–free languages. *Math. Systems Theory*, 2:127–145, 1968.

[Kui99] W. Kuich. Linear systems of equations and automata on distributive multioperator monoids. In *Contributions to General Algebra 12 - Proceedings of the 58th Workshop on General Algebra "58. Arbeitstagung Allgemeine Algebra", Vienna University of Technology. June 3-6, 1999*, Seiten 1–10. Verlag Johannes Heyn, 1999.

[SV08] T. Stüber und H. Vogler. Weighted monadic datalog. *Theor. Comput. Sci.*, 403(2-3):221–238, 2008.

[SVF09] T. Stüber, H. Vogler und Z. Fülöp. Decomposition of weighted multioperator tree automata. *Int. J. Foundations of Computer Sci.*, 20(2):221–245, 2009.

[TW68] J.W. Thatcher und J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Syst. Theory*, 2(1):57–81, 1968.

Torsten Stüber

Von 2001 bis 2006 studierte Torsten Stüber an der Technischen Universität Dresden zunächst Informatik und anschließend im internationalen Masterstudiengang *Computational Logic*. 2005 führte er ein Auslandssemester an der University of Auckland, Neuseeland, durch. Als bester Absolvent seines Jahrgangs an der Fakultät Informatik der TU Dresden wurde er 2007 mit der Lohrmann-Medaille ausgezeichnet.

Seit Ende 2006 ist er wissenschaftlicher Mitarbeiter am Lehrstuhl *Grundlagen der Programmierung* der TU Dresden. In dieser Funktion hat er sich vorrangig mit der Theorie der Baumautomaten und -Logiken beschäftigt und über dieses Thema im Februar 2011 promoviert. Im Verlauf seiner wissenschaftlichen Arbeit wurde sein Interesse für die Bereiche der *natürlichen Sprachverarbeitung* und des *maschinellen Lernens* geweckt und er hat seine Tätigkeit nach seiner Promotion verstärkt auf diese Gebiete konzentriert.