Als Manuskript gedruckt

Institut für Algebra
Technische Universität Dresden

Workshop „Perspektiven der Automaten und formalen Sprachen“ und 6. Theorietag der GI-Fachgruppe „Automaten und formale Sprachen“

Vortragszusammenfassungen

Manfred Droste und Dietrich Kuske (Hrsg.)
18.-20. September 1996 in Cunnersdorf
(Sächsische Schweiz)

MATH-AL-7-1996

September 1996

Inhaltsverzeichnis

Progamm 3

Thomas Buchholz und Martin Kutrib: *Zeitberechenbarkeit und -konstruierbarkeit in Polyautomaten* 21

Volker Diekert: *Über die Elimination unsichtbarer Aktionen in zeitabhängigen Automaten* 15

Manfred Droste: *Asynchrone zelluläre Automaten und logische Definierbarkeit für Pomsets* 54

Henning Fernau, Rudolf Freund und Markus Holzer: *Bounding Resources in Multi-Agent Systems* 29

Henning Fernau und Ralf Stiebe: *L-Systeme mit Valenzen und Bewertungen* 27

Rudolf Freund: *Molecular Computing and Formal Language Theory* 49

Fritz von Haeseler: *Über die Automatizität von Doppelfolgen, die durch rationale Funktionen definiert sind* 31

Dieter Hofbauer, Maria Huber und Gregory Kucherov: *Projektionen in kontextfreien Baumgrammatiken* 43

Maria Huber und Dieter Hofbauer: *Universeller Abschluß regulärer Baumsprachen und minimale Testmengen* 46

Klaus P. Jantke: *The importance of solving formal language problems in knowledge-based process supervision and control* 32

Daniel Kirsten: *Refining Graph Grammars in Therapy Planning using Temporal Logic* 41

Hans-Jörg Kreowski: *Formale Sprachen für Graphen und Bilder* 7

Dietrich Kuske: *Über temporale Logiken linearer Zeit für Spuren* 50

Klaus-Jörn Lange: *Formalsprachliche Aufgaben aus der Komplexitätstheorie* 10

Friedrich Otto: *Endlich dargestellte Monoide mit entscheidbarem Wortproblem, die keine kontext-freien Repräsentatensysteme besitzen* 16

Torsten Rönsch: *Erkennbare Sprachen unendlicher Berechnungen von Spur-Automaten* 51

Helmut Seidl: *Deforestation: Beseitigung temporärer Datenstrukturen funktionaler Programme mit Hilfe von Baumautomaten* 11

Wolfgang Thomas: *Automatentheorie und Theoretische Informatik: Neue Chancen einer alten Verbindung* 14

Klaus Wagner: *Das Blattsprachen-Konzept: Formale Sprachen in der Komplexitätstheorie* 6

Teilnehmerliste 55

6. Theorietag der GI-Fachgruppe „Automaten und formale Sprachen“

Donnerstag, 19. September

9^{00}–9^{40} FRIEDRICH OTTO: *Endlich dargestellte Monoide mit entscheidbarem Wortproblem, die keine kontext-freien Repräsentatensysteme besitzen*

9^{40}–10^{10} THOMAS BUCHHOLZ UND MARTIN KUTRIB: *Zeitberechenbarkeit und -konstruierbarkeit in Polyautomaten*

10^{10}–10^{40} HENNING FERNAU UND RALF STIEBE: *L-Systeme mit Valenzen und Bewertungen*

10^{40}–11^{10} Kaffeepause

11^{10}–11^{40} HENNING FERNAU, RUDOLF FREUND UND MARKUS HOLZER: *Bounding Resources in Multi-Agent Systems*

11^{40}–12^{10} FRITZ VON HAESELER: *Über die Automatizität von Doppelfolgen, die durch rationale Funktionen definiert sind*

12^{10}–14^{00} Mittagspause

14^{00}–14^{30} KLAUS P. JANTKE: *Probleme formaler Sprachen in der wissensbasierten Prozeßführung*

14^{30}–15^{00} DANIEL KIRSTEN: *Refining Graph Grammars in Therapy Planning using Temporal Logic*

15^{00}–15^{30} Kaffeepause

15^{30}–16^{00} DIETER HOFBAUER, MARIA HUBER UND GREGORY KUCHEROV: *Projektionen in kontextfreien Baumgrammatiken*

16^{00}–16^{30} MARIA HUBER UND DIETER HOFBAUER: *Universeller Abschluß regulärer Baumsprachen und minimale Testmengen*

16^{30}–17^{30} Vollversammlung der GI-Fachgruppe

19^{00} „Wild Cat Session“

Freitag, 20. September

$9^{\underline{00}}$ –$9^{\underline{30}}$ RUDOLF FREUND: *Molecular Computing and Formal Language Theory*

$9^{\underline{30}}$ –$10^{\underline{00}}$ ROMAN KÖNIG: *Automaten für Codes*

$10^{\underline{00}}$ –$10^{\underline{30}}$ DIETRICH KUSKE: *Über temporale Logiken linearer Zeit für Spuren*

$10^{\underline{30}}$ –$11^{\underline{00}}$ Kaffeepause

$11^{\underline{00}}$ –$11^{\underline{30}}$ TORSTEN RÖNSCH: *Erkennbare Sprachen unendlicher Berechnungen von Spur-Automaten*

$11^{\underline{30}}$ –$12^{\underline{00}}$ MANFRED DROSTE: *Asynchrone zelluläre Automaten und logische Definierbarkeit für Pomsets*

Das Blattsprachen-Konzept: Formale Sprachen in der Komplexitätstheorie

Klaus W. Wagner
Lehrstuhl für Theoretische Informatik
Institut für Informatik
Julius-Maximilians-Universität Würzburg

Eine nichtdeterministische Polynomialzeitmaschine M berechnet auf jedem Rechenweg einen Wert (z.B. 1 für "Akzeptierung" und 0 für "Ablehnung"; oder auch Werte ohne weitere Bedeutung). Die Folge dieser Werte bei einer Eingabe x wird als das *Blattwort* $\beta_M(x)$ bezeichnet. Nun definiert jede Sprache L eine Komplexitätsklasse L-P wie folgt: $A \in L$-P genau dann, wenn es eine nichtedeterministische Polynomialzeitmaschine M gibt mit $x \in A \leftrightarrow \beta_M(x) \in L$. Viele interessante Komplexitätsklassen lassen sich auf diese Weise charakterisieren: NP, coNP, die Klassen der Booleschen Hierarchie und der Polynomialzeithierarchie, die Klassen $\oplus$P, Mod$_k$P, 1NP, PP, PSPACE u.v.a.m. Es werden Beziehungen zwischen sprachtheoretischen und algebraischen Eigenschaften der Blattsprachen L einerseits und den Eigenschaften der entsprechenden Komplexitätsklassen L-P andererseits untersucht.

Formale Sprachen für Graphen und Bilder

Hans-Jörg Kreowski
Fachbereich Mathematik und Informatik
Universität Bremen
Postfach 33 04 40
D-28334 Bremen
email: kreo@informatik.uni-bremen.de

In der Theorie formaler Sprachen ist in den letzten Jahrzehnten ein wirksames Instrumentarium entwickelt worden, um Mengen von Zeichenketten beschreiben und untersuchen zu können. Eine breite Palette von Ergebnissen ist inzwischen verfügbar mit vielen interessanten Anwendungen wie der Syntaxdefinition von Programmiersprachen, der Spracherkennung und -übersetzung, der Modellierung von Prozessen aller Art u.v.a.m. Motiviert durch verschiedene Anwendungssituationen ist in den letzten 25 Jahren häufig versucht worden, Techniken aus der Theorie formaler Sprachen auch auf Mengen und Sprachen zu übertragen, deren Elemente nicht einfache Zeichenketten, sondern komplex strukturierte Konfigurationen sind wie Bäume, Graphen, Bilder u. ä.

In dem Vortrag soll in die Theorie formaler Sprachen für Graphen und Bilder eingeführt werden, wobei insbesondere ein Überblick über die Arbeiten der Forschungsgruppe Theoretische Informatik der Universität Bremen auf diesem Gebiet gegeben wird.
Im einzelnen ist geplant, auf folgende Themen näher einzugehen:

1. Hyperkantenersetzung ist eine graphgrammatikalische Realisierung von Kontextfreiheit mit interessanten Struktur- und Entscheidbarkeitseigenschaften.

2. Termgraphersetzung ist eine geeignete Grundlage für eine effiziente Auswertung funktionaler und logischer Ausdrücke und deren Analyse.

3. Das Konzept der Transformationseinheit erlaubt, Regelmengen strukturiert aufzubauen. Es bildet einen Kern der graph- und regelzentrierten Spezifikations- und Programmiersprache GRACE, die seit einiger Zeit von Forscherinnen und Forschern aus Aachen, Berlin, Bremen und Leiden entwickelt wird.

4. Kanonische Ableitungen stellen ein graph-grammatikalisches Pendant zu Linksableitungen dar und bilden eine aufschlußreiche Repräsentation äquivalenter paralleler Prozesse.

5. Mit Hilfe von Collagen-Grammatiken können die Ideen der Hyperkantenersetzung auch im Bereich der Bildgenerierung eingesetzt werden. Aufschlußreiche Beziehungen bestehen zur fraktalen Geometrie, zu Pflasterungen und zu zellularen Automaten.

6. Zusammen mit anderen syntaktischen Methoden der Bildgenerierung lassen sich Collagen-Grammatiken auch praktisch bei der Visualisierung und Animation einsetzen.

Es folgen einige Hinweise auf Arbeiten, die in den letzten Jahren in Bremen zu diesen Themen entstanden sind.

Literaturhinweise

Lars Brinkhus, Frank Drewes, Hans-Jörg Kreowski, Ralf Wirdemann (Hrsg.), *Syntaktische Methoden der Bilderzeugung.* Grundlagen eines computergestützten Studios zur Herstellung syntaktischer Filme. Anschlußbericht des studentischen Projekts BIZARR 2, Universität Bremen, 1995

Jürgen Dassow, Annegret Habel, Stefan Taubenberger: Chain-Code Pictures and Collages Generated by Hyperedge Replacement. *Proc. Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science* 1073, 412-427 (1996)

Gnanamalar David, Frank Drewes, Hans-Jörg Kreowski: Hyperedge Replacement with Rendezvous. In J. P. Jouannaud (ed.): TAPSOFT'93:Theory and Practice of Software Development, LNCS 668, 167-181 (1993)

Frank Drewes: The Use of Tree Transducers to Compute Translations Between Graph Algebras. *Proc. Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science*1073, 196-210 (1996)

Frank Drewes: NP-completeness of k-connected Hyperedge-replacement Languages of Order k. Information Processing Letters 45, 89-94 (1993)

Frank Drewes: Recognising k-connected Hypergraphs in Cubic Time. Theoretical Computer Science 109, 83-122 (1993)

Frank Drewes: On the connectedness of pictures defined by iterated function systems. In *Proc. 2nd International Conference on Developments in Language Theory*, 289-298, World Scientific (1996).

Frank Drewes: A lower bound on the growth of functions computed by tree transducers.Fundamenta Informaticae 26, 267-286 (1996).

Frank Drewes: Language theoretic and algorithmic properties of d-dimensional collages and patterns in a grid. *Journal of Computer and System Sciences.* To appear.

Frank Drewes, Annegret Habel, Hans-Jörg Kreowski: Hyperedge Replacement Graph Grammars. In G. Rozenberg, editor, *Handbook of Graph Rewriting. Vol. I: Foundations.* World Scientific. To appear.

Frank Drewes, Annegret Habel, Hans-Jörg Kreowski, Stefan Taubenberger: Generating self-affine fractals by collage grammars. *Theoretical Computer Science* 145, 159-187 (1995)

Frank Drewes, Hans-Jörg Kreowski: (Un)decidability of properties of pictures generated by collage grammars. Fundamenta Informaticae 25, 295-325 (1996)

Hartmut Ehrig, Hans-Jörg Kreowski, Gabriele Taentzer: Canonical derivations for high-level replacement systems. In H. Ehrig, H.-J. Schneider, editors, Proc. Graph Transformations in Computer Science, Lecture Notes in Computer Science 776, 153-169 (1994)

Annegret Habel: Hyperedge Replacement: Grammars and Languages, Lecture Notes in Computer Science 643, Berlin-Heidelberg 1992

Annegret Habel, Reiko Heckel, Gabi Taentzer: Graph grammars with negative application conditions. *Fundamenta Informaticae.* To appear.

Annegret Habel, Hans-Jörg Kreowski, Clemens Lautemann: A Comparison of Compatible, Finite, and Inductive Graph Properties. Theoretical Computer Science 110, 145-168 (1993)

Annegret Habel, Hans-Jörg Kreowski, Stefan Taubenberger: Collages and Patterns Generated by Hyperedge Replacement. Languages of Design 1, 125-145 (1993)

Annegret Habel, Detlef Plump: Unification, Rewriting, and Narrowing on Term Graphs. In A. Corradini, U. Montanari, editors, *Proc. Joint* COMPUGRAPH/SEMAGRAPH WORKSHOP ON GRAPH REWRITING AND COMPUTATION (SEGRAGRA '95), *Electronic Notes in Theoretical Computer Science* 2, 1995

Renate Klempien-Hinrichs: Node Replacement in Hypergraphs: Simulation of Hyperedge Replacement and Decidability of Confluence. *Proc. Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science* 1073, 397-411 (1996)

Hans-Jörg Kreowski: Translations into the Graph Grammar Machine. In M.R. Sleep, M.J. Plasmeijer, M.C.J.D. van Eekelen (eds.): Term Graph Rewriting: Theory and Practice, chapter 13, John Wiley (1993)

Hans-Jörg Kreowski: Five Facets of Hyperedge Replacement Beyond Context-freeness, in *Proc. Fundamentals of Computation Theory, LNCS 710, 69-86 (1993)*

Hans-Jörg Kreowski: An axiomatic approach to canonical derivations. In B. Pehrson, I. Simon, editors, Proc. 13th IFIP World Computer Congress 1994, 348–353. Elsevier Science B.V. (North Holland), 1994

Hans-Jörg Kreowski, Sabine Kuske: On the Interleaving Semantics of Transformation Units – A Step into GRACE. *Proc. Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science* 1073, 89-106 (1996)

Sabine Kuske: A Maximum Path Length Pumping Lemma for Edge-replacement Languages, in Proc. Fundamentals of Computation Theory, LNCS 710, 342-351 (1993)

Detlef Plump: Hypergraph Rewriting: Critical Pairs and Undecidability of Confluence. In M.R. Sleep, M.J. Plasmeijer, M.C.J.D. van Eekelen (eds.): Term Graph Rewriting: Theory and Practice, chapter 15, John Wiley (1993)

Detlef Plump: Collapsed Tree Rewriting: Completeness, Confluence, and Modularity. In M. Rusinowitch, J.L. Remy (eds.): Conditional Term Rewriting Systems, LNCS 656, 97-112 (1993)

Detlef Plump: Critical pairs in term graph rewriting. In P. Ružička, I. Prívara, B. Rovan, editors, Proc. Mathematical Foundations of Computer Science 1994, Lecture Notes in Computer Science 841, 556-566, 1994

Detlef Plump: On Termination of Graph Rewriting. *Proc. Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science 1017*, 88-100, 1995.

Detlef Plump, Annegret Habel: Graph Unification and Matching. *Proc. Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science* 1073, 75-88 (1996)

Formalsprachliche Aufgabenstellungen der strukturellen Komplexitätstheorie

Klaus-Jörn Lange, Universität Tübingen

Das Forschungsgebiet der formalen Sprachen und die strukturelle Komplexitätstheorie weisen vergleichbare Fragestellungen und Vorgehensweisen auf. Sehr unterschiedlich sind allerdings nicht nur die verwendeten Werkzeuge sondern auch die Art der erreichten Ergebnisse. Während im Bereich der formalen Sprachen vergleichsweise wenige Probleme ungelöst blieben, bestehen die Resultate der strukturellen Komplexitätstheorie überwiegend aus indirekten Ergebnissen, in denen offene Fragen in Beziehung gesetzt werden. Insbesondere gibt es im Gegensatz zum Bereich der formalen Sprachen so gut wie keine (nichttrivialen) Trennungen von Sprachklassen.

Inhalt des Vortrages ist ausgehend von den engen Beziehungen zwischen den beiden Gebieten die Herausarbeitung von formalsprachlichen Fragestellungen, deren Beantwortung für die Komplexitätstheorie aufschlußreich sein könnte. Alle diese Fragen behandeln im weiteren Sinne die Darstellung von sequentiellen und parallelen Komplexitätsklassen durch Familien formaler Sprachen. Die einzelnen Probleme lassen sich in folgender Weise gliedern:

a.) Die Frage nach formalen Sprachen für Klassen, die bisher keine derartige Darstellung genießen.

b.) Die Frage nach Reduktionen oder Abschlußmechanismen, die weitergehende Aussagen über die Zusammenhänge zwischen formalen Sprachen und Komplexitätsklassen zulassen.

c.) Die Frage nach der Verallgemeinerbarkeit von Einzelresultaten durch Betrachtung abstrakterer Modelle.

Deforestation: Beseitigung temporärer Datenstrukturen funktionaler Programme mithilfe von Baumautomaten

H. Seidl
FB IV - Informatik Universität Trier
D-54286 Trier, Germany
seidl@uni-trier.de

10. September 1996

Moderne funktionale Programmiersprachen begünstigen einen Programmierstil mit Zwischendatenstrukturen etwa wie in dem folgenden Programm:

$$
\begin{array}{l}
\textbf{letrec} \\
\quad a \;=\; \lambda x, y.\textbf{case}\ x\ \textbf{of} \\
\qquad\qquad\qquad [\,] \;\rightarrow\; y \\
\qquad\qquad\quad (h:t) \rightarrow\; h : a\, t\, y \\
\textbf{in}\ \lambda u, v, w.\ a\,(a\,u\,v)\,w
\end{array}
$$

Der Term $\lambda u, v, w.a\,(a\,u\,v)\,w$ konkateniert the Listen u, v und w. Konkatenation von u und v resultiert in einer Zwischendatenstruktur zu welcher w konkateniert wird. Allokation und Deallokation der Zwischendatenstruktur zur Laufzeit ist teuer. Unter Preisgabe von Klarheit zugunsten von Effizienz würden wir deshalb das folgende Programm vorziehen:

$$
\begin{array}{l}
\textbf{letrec} \\
\quad d \;=\; \lambda x, y, z.\textbf{case}\ x\ \textbf{of} \\
\qquad\qquad\qquad [\,] \;\rightarrow\; a'\, y\, z \\
\qquad\qquad\quad (h:t) \rightarrow\; h : d\, t\, y\, z \\
\quad a' \;=\; \lambda y, z.\textbf{case}\ y\ \textbf{of} \\
\qquad\qquad\qquad [\,] \;\rightarrow\; z \\
\qquad\qquad\quad (h:t) \rightarrow\; h : a'\, t\, z \\
\textbf{in}\ \lambda u, v, w.\ d\, u\, v\, w
\end{array}
$$

Idealerweise sollten wir die erste Version programmieren dürfen, welche dann durch den Compiler in die zweite übersetzt wird. Tatsächlich ist dies mithilfe Phil Wadler's Programmtransformation *Deforestation* möglich. Deforestation terminiert auf sog. *treeless programs*, welche einer bestimmten Klasse deterministischer Topdown Tree-Transducer entspricht, die unter Komposition abgeschlossen ist. Erweiterungen, die Terminierung auf *allen* First-order-Programmen sicherstellen, stammen etwa von Chin und später von Hamilton. Die grundlegende Idee dieser Verfahren besteht darin, alle jene Teil des Programms zu annotieren, die die Syntax von treeless programs verletzen, und dann diese Teile durch den Deforestation-Algorithmus ignorieren zu lassen.

Diese Verfahren betreffen nur First-order-Programme. Moderne funktionale Programmiersprachen wie ML, Haskell, and Miranda bieten jedoch auch die Möglichkeit, Higher-order-Funktionen zu definieren, welche ebenfalls transformiert werden sollten. Einige vorläufige Ansätze reduzieren den Higher-order-Fall auf den wohluntersuchten First-order-Fall. Die erste Formulierung von Deforestation, die direkt auf beliebige Higher-order-Programme anwendbar ist, stammt von Marlow and Wadler (1992), die allerdings die Frage der Terminierung offen lassen. Dies wurde erstmals von Hamilton (1995) behandelt, der eine Formulierung des Higher-order-Deforestation-Algorithmus vorstellt, welche auf annotierte Programme anwendbar ist, und eine Definition von *higher-order treeless programs* präsentiert. Hamilton skizziert einen Beweis, daß Deforestation auf jedem Hindley-Milner typbaren Programm terminiert, falls alle Teile, die gegen die higher-order treelessness verstoßen, annotiert wurden. Diese Ideen wurden von Marlow (1996) ausgearbeitet und in den Glasgow Haskell Compiler eingebaut.

Die Syntax von higher-order treeless Programmen erfordert, daß Argumente von Funktionsanwendungen und Selectoren von **case**-Ausdrücken Variablen sind. Dies hat Annotierungen und damit Auslassungen von Programmteilen zur Folge, die andernfalls hätten verbessert werden können. Betrachten wir etwa das folgende Programm.

$$
\begin{array}{lll}
\textbf{letrec} & & \\
\quad c & = & \lambda x, xs.x : xs \\
\quad \mathit{foldr} & = & \lambda f, a, l.\textbf{case}\ l\ \textbf{of} \\
& & \qquad\qquad [\,] \quad \rightarrow a \\
& & \qquad\quad (z : zs) \rightarrow f\ z\ (\mathit{foldr}\ f\ a\ zs) \\
\textbf{in}\ \lambda u, v, w.\mathit{foldr}\ c\ w\ (\mathit{foldr}\ c\ v\ u) & &
\end{array}
$$

Der Term $\mathit{foldr}\ c\ v\ u$ ist eine higher-order Formulierung des Terms $a\ u\ v$ des ersten Beispiels. Das ganze Programm ist äquivalent zu unserem ersten Programm und sollte darum auch in die effizientere Form transformiert werden können. In der Tat geschieht genau dieses, wenn wir Deforestation auf das Programm anwenden. Die Verfahren von Hamilton und Marlow verlangen jedoch, daß das Argument $\mathit{foldr}\ f\ a\ zs$ in der Definition von foldr annotiert wird – was die erwünschte Transformation gerade verhindert.

Es gibt viele ähnliche Beispiele. In dem Vortrag stellen wir eine neue Technik vor, um Terminierung von Higher-order Deforestation zu garantieren. Wir präsentieren eine vereinfachte Version von Hamilton's Deforestation-Algorithmus auf annotierten Programmen, verlangen aber nicht, daß alle Teile annotiert sind, die die higher-order-treeless Syntax verletzen. Vor der Transformation berechnen wir stattdessen eine Menge von einfachen *Constraints*, die die Menge der Terme approximiert, die während Deforestation auftreten. Gleichzeitig extrahieren wir *quantitative Information*, um Situationen zu entdecken, die zu Nichtterminierung von Deforestation führen können. Dann annotieren wir bestimmte Teilterme, um solche Nichtterminierung zu verhindern. Diese Technik ist eine Verallgemeinerung unserer Vorgehensweise bei First-order-Deforestation.

Der Vortrag basiert auf einer gemeinsamen Arbeit mit M.H. Sørensen, University of Copenhagen.

Automatentheorie und theoretische Informatik: Neue Chancen einer alten Verbindung

Wolfgang Thomas

Automatentheorie und Formale Sprachen sind Kerngebiete der theoretischen Informatik, die nach wie vor und mit gutem Grund einen zentralen Platz im Studium der Informatik einnehmen. Es sind aber (sowohl in der Forschung als auch in der Lehre) Anzeichen von Stagnation festzustellen; sie haben nach Auffassung des Vortragenden ihre Ursache in einer zu engen Sicht der Automatentheorie (im Kontext der Chomsky-Sprachklassen), mit einem recht eingeschränkten Spektrum von Fragen und Methoden. Weder findet man diese Spezialisierung bei den ersten Autoren des Gebiets (wie von Neumann, Kleene, Moore, Church, Trahktenbrot, Rabin), noch ist eine solche Fokussierung durch die aktuellen Anforderungen der praktischen Informatik gerechtfertigt. Es werden einige aus Anwendungen motivierte Fragestellungen diskutiert, die in der aktuellen Diskussion unter theoretischen Informatikern zu kurz kommen.

Über die Elimination unsichtbarer Aktionen in zeitabhängigen Automaten

Volker Diekert
FB Informatik
Universität Stuttgart
70565 Stuttgart

Timed automata (and more general hybrid automata) are among the most widely studied models for real-time systems. Silent transitions, have already been proposed in the original paper on timed automata by Alur and Dill. In last year STACS paper of Bérad et al. it is shown that ε-reset transitions can be removed, if they do not reset clocks; moreover ε-reset transitions strictly increase the power of timed automata, if there is a loop containing ε-reset transitions which reset some clocks. The authors of left open, in particular, the problem about the power of the ε-reset transitions with reset set, if they are arranged such that they do not lie on any cycle.

In the talk we prove that a timed automaton with ε-reset transitions can *be transformed into a timed automaton without ε-transitions, if no ε transition with nonempty reset set lies on any directed cycle of the automaton.* Conversely, we exhibit a particular automaton with a cycle containing a single ε-transition being not equivalent to any automaton without ε-transitions. *We prove the non equivalence using a new technique based on a notion of precise time in a run. Another interesting fact is that our main result holds under the assumption of non-Zenoness and it is false otherwise.*

Endlich dargestellte Monoide mit entscheidbarem Wortproblem, die keine kontext-freien Repräsentantensysteme besitzen

Friedrich Otto *
Fachbereich Mathematik/Informatik
Universität Gh Kassel, D-34109 Kassel, Germany
Internet: otto@theory.informatik.uni-kassel.de

10. September 1996

Wird ein Monoid M durch ein endliches, konvergentes Wortersetzungssystem (WES) S dargestellt, so ist die Menge $\mathrm{IRR}(S)$ der irreduziblen Wörter ein reguläres Repräsentantensystem für M. Dies gilt natürlich auch dann noch, wenn S unendlich ist, sofern die Menge $\mathrm{dom}(S)$ der linken Regelseiten eine reguläre Menge ist. Solche unendlichen Systeme werden wir im folgenden **links-regulär** nennen. Hat nun aber umgekehrt jedes endlich dargestellte Monoid, das ein reguläres Repräsentantensystem hat, auch eine Darstellung durch ein links-reguläres, konvergentes WES? Oder haben gar alle endlich dargestellten Monoide, die ein entscheidbares Wortproblem haben, ein reguläres Repräsentantensystem?

Um diese Fragen anzugehen, stellt es sich als vorteilhaft heraus, zunächst den Zusammenhang zwischen unendlichen, konvergenten WESen, deren linke Regelseiten eine Sprache eines bestimmten Typs bilden, und der Existenz von Repräsentantensystemen, die eine Sprache desselben Typs bilden, zu untersuchen.

Eine Partialordnung $>$ auf Σ^* heißt *min-vollständig für* S, wobei S ein WES über Σ ist, wenn jede Kongruenzklasse $[w]_S$ genau ein kleinstes Element bezüglich $>$ enthält. Mit $\mathrm{MIN}(S, >)$ bezeichnen wir dann die Menge aller kleinster Elemente, d.h., $\mathrm{MIN}(S, >) = \{w \mid w$ ist das kleinste Element in $[w]_S\}$. Es gilt nun die folgende Aussage.

Satz 1 [Ave86] *Sei M das durch die endlich erzeugte Darstellung $(\Sigma; S)$ definierte Monoid, und sei $>$ eine zulässige, wohl-fundierte Partialordnung auf Σ^*, die min-vollständig für S ist. Dann gibt es ein konvergentes WES R über Σ, das zu S äquivalent ist, so daß $\mathrm{IRR}(R) = \mathrm{MIN}(S, >)$ ist.*

Man beachte, daß diese Aussage für alle endlich erzeugten Monoid-Darstellungen gilt, also auch für solche mit unentscheidbarem Wortproblem.

*Die dargestellten Ergebnisse entstanden im Frühjahr 1996 während eines von der DFG geförderten Forschungsaufenthaltes des Authors in Japan in gemeinsamer Arbeit mit Herrn **Professor Masashi Katsura** vom Department of Mathematics der Kyoto Sangyo University und Herrn **Professor Yuji Kobayashi** vom Department of Information Science der Toho University in Funabashi.

Sei nun $\mathbb{C}$ eine Familie formaler Sprachen. Ein WES R über Σ heißt *links-$\mathbb{C}$*, wenn die Menge $\mathrm{dom}(R)$ der linken Regelseiten von R zu $\mathbb{C}$ gehört. Eine Klasse $\mathbb{C}$ heißt *komplementär*, wenn sie die beiden folgenden Bedingungen erfüllt:

- $\mathbb{C}$ ist abgeschlossen bezüglich Komplementbildung;
- $\mathbb{C}$ ist abgeschlossen bezüglich der Produktbildung mit Σ^*, d.h., ist $L \subseteq \Sigma^*$ in $\mathbb{C}$, so ist auch die Menge $\Sigma^* \cdot L \cdot \Sigma^*$ in $\mathbb{C}$.

Die Klassen der regulären Sprachen, der kontext-sensitiven Sprachen und der rekursiven Sprachen sind Beispiele hierfür. Es gilt nun die folgende Aussage.

Satz 2 *Sei $(\Sigma; S)$ eine endlich erzeugte Darstellung eines Monoids M, sei $\mathbb{C}$ eine komplementäre Klasse von Sprachen, und sei $C \subseteq \Sigma^*$ eine teilwort-abgeschlossene Sprache. Dann sind die beiden folgenden Aussagen äquivalent:*

(a) es gibt ein links-$\mathbb{C}$, konvergentes WES R über Σ, das zu S äquivalent ist, und das $\mathrm{IRR}(R) = C$ erfüllt;

(b) C gehört zur Klasse $\mathbb{C}$, und es gibt eine zulässige, wohl-fundierte Partialordnung $>$ auf Σ^*, die min-vollständig für S ist, und die $C = \mathrm{MIN}(S, >)$ erfüllt.

Im allgemeinen kann ein endlich erzeugtes Monoid mit regulärem Repräsentantensystem noch unentscheidbares Wortproblem haben. Für endlich dargestellte Monoide ist dies nicht mehr der Fall, wie der folgende Satz zeigt.

Satz 3 *Sei $(\Sigma; S)$ eine endliche Darstellung eines Monoids M. Die folgenden vier Aussagen sind dann äquivalent:*

(a) M hat entscheidbares Wortproblem;

(b) M hat ein rekursives Repräsentantensystem $C \subseteq \Sigma^*$;

(c) es gibt ein links-rekursives, konvergentes WES R über Σ, das zu S äquivalent ist;

(d) es gibt eine zulässige, wohl-fundierte Partialordnung $>$ auf Σ^*, die min-vollständig für S ist, so daß $\mathrm{MIN}(S, >)$ eine rekursive Menge ist.

Hier stellt sich nun die Frage, ob diese Aussage gültig bleibt, wenn man statt eines links-rekursiven WES ein links-kontext-sensitives oder gar ein links-kontext-freies WES haben will. Diese Frage ist allerdings noch offen.

Wenden wir uns nun der Frage zu, wie man nachweisen kann, daß ein endlich dargestelltes Monoid kein reguläres Repräsentantensystem hat.

Sei dazu M ein Monoid, und sei $\alpha \in M$. Ein Element $\beta \in M$ heißt *torsional* in α, falls es positive ganze Zahlen m, n, $m \neq n$, gibt, so daß $\gamma\beta^m\delta =_M \gamma\beta^n\delta$ gilt, wobei $\alpha =_M \gamma\beta\delta$ ist.

Man kann nun die folgende Aussage zeigen.

Satz 4 *Sei $(\Sigma; S)$ eine endlich erzeugte Darstellung eines Monoids M. Enthält M eine Teilmenge A, die die folgende Bedingung $(*)$ erfüllt, so hat M kein reguläres Repräsentantensystem $C \subseteq \Sigma^*$:*

$(*)$ Für jede Zahl $n \in \mathbb{N}$ gibt es ein Element $\alpha_n \in A$, so daß folgendes gilt: ist $x \in \Sigma^*$ ein Wort mit $[x]_S = \alpha_n$, so enthält x ein Teilwort y der Länge $|y| \geq n$, so daß für jedes Teilwort z von y, $[z]_S$ ein torsionaler Faktor von $[x]_S$ in M ist.

Mit Hilfe dieses Satzes kann man nun beispielsweise zeigen, daß das folgende Monoid kein reguläres Repräsentantensystem hat: $\Sigma = \{a, b, c, d\}$, $t \in \mathbb{N}$, und $S(t) = \{ba \to ab, bc \to aca, a^t cc \to d\} \cup \{de \to d, ed \to d \mid e \in \Sigma\}$. Da $S(t)$ zu dem links-kontext-freien, konvergenten WES $R(t) = S(t) \cup \{a^{n+t}ca^n c \to d \mid n \geq 1\}$ äquivalent ist, folgt sofort, daß das dargestellte Monoid M_t entscheidbares Wortproblem hat. Die Menge $A := \{\alpha_n \mid n \in \mathbb{N}\}$ mit $\alpha_n = [a^n ca^{n+1}ca^{n+2}c]_{S(t)}$ erfüllt nun die Bedingung $(*)$ aus obigem Satz. Also hat M_t kein reguläres Repräsentantensystem.

Eine weitere interessante Gruppe von Beispielen erhalten wir wie folgt.

Beispiel 5 *Sei t eine positive ganze Zahl, sei $\Sigma_t = \{a_1, \ldots, a_t, b, c_1, \ldots, c_t, d\}$, sei $X_t = \{a_i^2, a_i c_i, c_j a_{j+1}, a_j b, b a_j, c_j b, b c_j, b^2 \mid i = 1, \ldots, t,\ j = 1, \ldots, t-1\}$, sei $S(t) = \{x \to d \mid x \in \Sigma_i^2 \setminus X_t\} \cup \{ba_i \to a_i b, bc_i \to a_i c_i a_{i+1} \mid i = 1, \ldots, t-1\} \cup \{de \to d, ed \to d \mid e \in \Sigma_t\}$, und sei N_t das durch $(\Sigma_t; S(t))$ definierte Monoid. $S(t)$ ist zum links-kontext-freien, konvergenten WES $R(t) = S(t) \cup \{a_i^n c_i a_{i+1}^n c_{i+1} \to d \mid i = 1, \ldots, t-1, n \geq 1\}$ äquivalent. Also hat N_t entscheidbares Wortproblem. Andererseits kann man mit Hilfe des obigen Satzes zeigen, daß für $n \geq 3$ das Monoid N_t kein reguläres Repräsentantensystem hat.*

Das Wachstum des Monoids N_t ist polynomial vom Grad t. Also gibt es endlich dargestellte Monoide mit polynomialem Wachstum, die entscheidbares Wortproblem haben, die aber durch kein links-reguläres, konvergentes WES dargestellt werden können. Dies beantwortet eine Frage aus [Kob95].

Kann eine Satz 4 entsprechende Aussage auch für die Eigenschaft, kein kontext-freies Repräsentantensystem zu haben, bewiesen werden?

Sei dazu M ein Monoid, und sei $\alpha \in M$. Ein Dreitupel $(\beta, \gamma, \delta) \in M^3$ heißt *bi-torsional* in α, falls es positive ganze Zahlen m, n, $m \neq n$, gibt, so daß $\varphi\beta^m\gamma\delta^m\eta =_M \varphi\beta^n\gamma\delta^n\eta$ gilt, wobei $\alpha =_M \varphi\beta\gamma\delta\eta$ ist.

Ferner brauchen wir die folgende Fassung von Ogdens Lemma [Ber79].

Lemma 6 *Sei $L \subseteq \Sigma^*$ eine kontext-freie Sprache. Dann gibt es eine positive ganze Zahl $k := k(L) \geq 2$, so daß jedes Wort $w \in L$, das mindestens k markierte Buchstaben enthält, derart als $w = uvxyz$ faktorisiert werden kann, daß die folgenden drei Bedingungen erfüllt werden:*

(1.) u, v und x oder x, y und z enthalten jeweils mindestens einen markierten Buchstaben,

(2.) vy enthält höchstens k markierte Buchstaben, und

(3.) $uv^ixy^iz \in L$ für alle $i \geq 0$.

Es gilt nun der folgende Satz.

Satz 7 *Sei $(\Sigma; S)$ eine endlich erzeugte Darstellung eines Monoids M. Enthält M eine Teilmenge A, welche die folgende Bedingung $(*)$ erfüllt, dann gibt es kein kontext-freies Repräsentantensystem $C \subseteq \Sigma^*$ für M:*

$(*)$ Zu jedem $n \in \mathbb{N}$ gibt es ein Element $\alpha_n \in A$, so daß folgendes gilt: ist $w \in \Sigma^*$ ein Wort mit $[w]_S = \alpha_n$, so gibt es eine Markierung von n Buchstaben in w so, daß für jede Dekomposition $w = uvxyz$ von w, die den Bedingungen (1.) und (2.) von Ogdens Lemma genügt, $([v]_S, [x]_S, [y]_S)$ bi-torsional in α_n ist.

Mit Hilfe dieses Satzes kann man nun zeigen, daß keines der oben erwähnten Monoide M_t ein kontext-freies Repräsentantensystem hat. Dasselbe gilt auch für die Monoide N_t mit $t \geq 5$ aus Beispiel 5. Insbesondere zeigt dies, daß ein endlich dargestelltes Monoid, welches durch ein links-kontext-freies, konvergentes WES beschrieben wird, kein kontext-freies Repräsentantensystem zu haben braucht.

Interessant ist noch der Spezialfall $t = 3$ von Beispiel 5. Während nämlich die Menge $\mathrm{IRR}(R(3))$ nicht kontext-frei ist, kann man doch ein links-kontext-freies, konvergentes WES R' angeben, das zu $S(3)$ äquivalent ist, so daß $\mathrm{IRR}(R')$ eine kontext-freie Sprache, und damit ein kontext-freies Repräsentantensystem, für N_3 ist. Also ist N_3 ein endlich dargestelltes Monoid mit entscheidbarem Wortproblem, das zwar kein reguläres Repräsentantensystem besitzt, das aber wenigstens ein kontext-freies Repräsentantensystem hat. Insbesondere zeigt dies, daß die kontext-freien Repräsentantensysteme eine größere Klasse von endlich dargestellten Monoiden mit entscheidbarem Wortproblem beschreiben als die regulären Repräsentantensysteme.

Schließlich sollte in diesem Zusammenhang noch bemerkt werden, daß selbst ein konvergentes WES, das nicht links-kontext-frei ist, doch eine kontext-freie Menge von irreduziblen Worten besitzen kann. Diese Ergebnisse zeigen insbesondere, daß sich Satz 2 nicht auf die Klasse der kontext-freien Sprachen übertragen läßt. Dasselbe gilt für die Klasse der deterministischen kontext-freien Sprachen.

Andererseits gilt für kontext-sensitive Repräsentantensysteme die folgende Aussage im Kontrast zu den Sätzen 4 und 7.

Satz 8 *Sei M ein endlich erzeugtes Monoid mit entscheidbarem Wortproblem. Dann existiert für M eine endlich erzeugte Darstellung $(\Sigma; S)$, die ein kontext-sensitives Repräsentantensystem $C \subseteq \Sigma^*$ für M enthält.*

Literatur

[Ave86] J. Avenhaus. On the descriptive power of term rewriting systems. *J. Symbolic Computation*, 2:109–122, 1986.

[Ber79] J. Berstel. *Transductions and Context-free Languages.* Teubner Studienbücher. Teubner-Verlag, 1979.

[Kob95] Y. Kobayashi. A finitely presented monoid which has solvable word problem but has no regular complete presentation. *Theoretical Computer Science*, 146:321–329, 1995.

Zeitberechenbarkeit und -konstruierbarkeit von Funktionen in Polyautomaten

Thomas Buchholz und Martin Kutrib

AG Informatik, Universität Gießen
Arndtstr. 2, 35392 Gießen

{buchholz,kutrib}@informatik.uni-giessen.de

September 1996

1 Einleitung

Polyautomaten unterscheiden sich wesentlich in der Verbindungsstruktur der einzelnen Zellen und in der Art und Weise, wie die Eingabe in die Zellen erfolgt. Deterministische Varianten mit paralleler Eingabe werden als *Zellularautomaten* bezeichnet, während für Automaten mit sequentieller Eingabe in eine ausgezeichnete Zelle die Bezeichnung *iterative Arrays* üblich ist.

Einzelne Zellen sind entweder endliche Moore-Automaten oder deterministische Kellerautomaten. Entsprechend werden die Abkürzungen CA, IA und PDCA benutzt.

Im folgenden werden zwei naheliegende Verbindungsstrukturen im eindimensionalen Euklidischen Raum unterschieden. Zum einen ist jede Zelle mit ihren beiden unmittelbaren Nachbarn verbunden, woraus sich ein bidirektionaler Informationsfluß ergibt. Zum anderen ist jede Zelle lediglich mit ihrem jeweils rechten Nachbarn verbunden, woraus sich ein unidirektionaler Informationsfluß von rechts nach links ergibt (Abkürzungen OCA, OIA und OPDCA).

Vergleiche der einzelnen Modelle erfolgen üblicherweise anhand ihrer Fähigkeiten, formale Sprachen verarbeiten zu können, wobei die Zeit als Komplexitätsmaß eine herausragende Rolle spielt.

Eine naheliegende Forderung an entsprechende Komplexitätsschranken ist deren prinzipielle Berechenbarkeit. Zum Nachweis unendlicher Hierarchien in sequentiellen Automaten mittels Diagonalisierung wird darüber hinaus gefordert, daß die Komplexitätsschranken in dem betrachteten Modell selbst in gewisser Weise berechenbar sind. Konkrete Aussagen über die Familie derartiger Schranken sind allerdings selten.

In Polyautomaten gewinnt dieser Umstand an Bedeutung, da diese Modelle konzeptionell nie halten (sondern bestenfalls stabile Konfigurationen erreichen). Legt man nun eine in dem Modell selbst nicht berechenbare Funktion als Komplexitätsschranke zugrunde, so wird die entsprechende Komplexitätsklasse

durch Fähigkeiten „von außen“ – nämlich das Ablesen des Ergebnisses zum Zeitpunkt $f(n)$ – mitbestimmt. Vor diesem Hintergrund erscheinen manche Ergebnisse in der Literatur in ihrer Allgemeinheit eher fraglich.

Gegenstand unserer Untersuchungen ist es nun, verschiedene Arten von Polyautomaten hinsichtlich ihrer Fähigkeiten, Funktionen von der Menge der natürlichen Zahlen in sich zu „handhaben“, zu untersuchen. Die Ergebnisse lassen Rückschlüsse auf in dem jeweiligen Modell „vernünftige“ Komplexitätsschranken zu und liefern darüber hinaus Möglichkeiten, die Fähigkeiten der Modelle jenseits formaler Sprachen miteinander zu vergleichen.

Im folgenden werden zeitberechenbare und zeitkonstruierbare Funktionen unterschieden. Wir nennen eine Funktion f *zeitberechenbar*, falls der zugrundeliegende Automat bei Eingabe von n identischen Zeichen nach genau $f(n)$ Zeitschritten eine akzeptierende Konfiguration einnimmt.
Eine Funktion f heißt *zeitkonstruierbar*, falls der Automat jeweils genau zu den Zeitpunkten $f(1), f(2), \ldots, f(n)$ eine akzeptierende Konfiguration einnimmt.

2 Definitionen

Definition. Ein Zellularautomat (CA) ist ein System $(S, \sigma, \#)$, wobei gilt:

a) S ist eine endliche, nichtleere *Zustandsmenge*,
b) $\# \in S$ ist der *Grenzzustand* und
c) $\sigma : S^3 \to S$ ist die *lokale Überführungsfunktion*, für die gilt:
$\forall\, s_1, s_2, s_3 \in S : \sigma(s_1, s_2, s_3) = \# \iff s_2 = \#$

Die lokale Überführungsfunktion induziert eine längenerhaltende Abbildung $\mathcal{T} : S^+ \longrightarrow S^+$ gemäß dem folgenden: $\forall\, n \in \mathbb{N}, i \in \{1, \ldots, n\} : \forall\, s_i \in S :$

$$\begin{aligned} \mathcal{T}(s_1) &:= \sigma(\#, s_1, \#) \\ \mathcal{T}(s_1 \cdots s_n) &:= \sigma(\#, s_1, s_2)\sigma(s_1, s_2, s_3) \cdots \sigma(s_{n-1}, s_n, \#) \end{aligned}$$

Abbildung 1: Ein bidirektionaler Zellularautomat.

Für unidirektionale Zellularautomaten gilt entsprechend $\sigma : S^2 \to S$ und

$$\begin{aligned} \mathcal{T}(s_1) &:= \sigma(s_1, \#) \\ \mathcal{T}(s_1 \cdots s_n) &:= \sigma(s_1, s_2) \cdots \sigma(s_n, \#) \end{aligned}$$

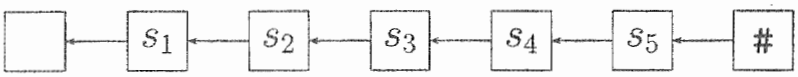


Abbildung 2: Ein unidirektionaler Zellularautomat.

In iterativen Arrays muß die Eingabe berücksichtigt werden:

Definition. Ein iterativer Array (IA) ist ein System $(S, A, \sigma, \#, s_0, a_0)$, wobei gilt:

a) S ist eine endliche, nichtleere *Zustandsmenge*,
b) A ist eine endliche, nichtleere *Menge von Eingabesymbolen*,
c) $a_0 \in A$ ist das *Eingabeendezeichen*, das nach der regulären Eingabe unendlich oft vorhanden ist,
d) $\# \in S$ ist der *Grenzzustand*,
e) $s_0 \in S$ ist der *Ruhezustand* und
f) $\sigma : S^3 \cup (S^3 \times A) \to S$ ist die *lokale Überführungsfunktion*, für die gilt: $\forall\, s_1, s_2, s_3 \in S : \sigma(s_1, s_2, s_3) = \# \iff s_2 = \#$ und für $s_l, s_r \in \{s_0, \#\}$ ist $\sigma(s_l, s_0, s_r) = s_0$.

Die globale Überführungsfunktion ist nun eine Abbildung von A^+S^+ nach A^+S^+ wie folgt:
Es sei $n, m > 0, i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} : \forall\, a_i \in A, s_j \in S :$

$$\begin{aligned} \mathcal{T}(a_m \cdots a_1, s_1) &:= (a_m \cdots a_2, \sigma(\#, s_1, \#, a_1)) \\ \mathcal{T}(a_m \cdots a_1, s_1 \cdots s_n) &:= (a_m \cdots a_2, \sigma(\#, s_1, s_2, a_1) \cdots \sigma(s_{n-1}, s_n, \#)) \end{aligned}$$

Für unidirektionale iterative Arrays gilt $\sigma : S^2 \cup (S^2 \times A) \longrightarrow S$ und

$$\begin{aligned} \mathcal{T}(a_m \cdots a_1, s_1) &:= (a_m \cdots a_2, \sigma(s_1, \#, a_1)) \\ \mathcal{T}(a_m \cdots a_1, s_1 \cdots s_n) &:= (a_m \cdots a_2, \sigma(s_1, s_2) \cdots \sigma(s_n, \#, a_1)) \end{aligned}$$

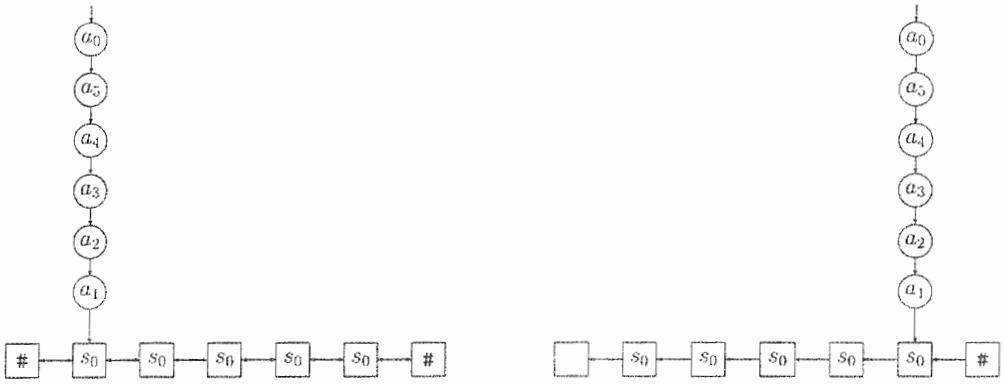


Abbildung 3: Bidirektionaler und unidirektionaler iterativer Array.

Die Definitionen für Keller-Zellularautomaten lassen sich leicht aus denen für Zellularautomaten ableiten und werden hier zugunsten der Übersichtlichkeit vernachlässigt.

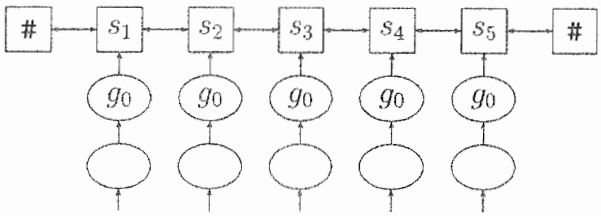


Abbildung 4: Ein bidirektionaler Keller-Zellularautomat.

Im weiteren bezeichne π_i die i-te Projektion eines Wortes oder Vektors.

Definition. Es sei POLY ein i) (O)CA, ii) (O)IA, iii) (O)PDCA.
Eine Funktion $f : \mathbb{N} \longrightarrow \mathbb{N}$ heißt *POLY-zeitberechenbar* genau dann, wenn ein POLY mit Zustandsmenge S, einem ausgezeichneten Zustand $q \in S$ oder einem ausgezeichneten Eingabesymbol $q \in A$ und einer Menge von Endzuständen $F \subseteq S \setminus \{s_0\}$ existiert, so daß $f(n)$ die kleinste natürliche Zahl ist, für die gilt: i) $\pi_1(\mathcal{T}^{f(n)}(q^n)) \in F$, ii) $\pi_1(\mathcal{T}^{f(n)}(q^n, s_0^n)) \in F$, iii) $\pi_{1,1}(\mathcal{T}^{f(n)}((q, g_0)^n)) \in F$.

Die Familie der POLY-zeitberechenbaren Funktionen wird mit C(POLY) bezeichnet.

Definition. Es sei POLY ein i) (O)CA, ii) (O)IA, iii) (O)PDCA.
Eine streng monoton wachsende Funktion $f : \mathbb{N} \longrightarrow \mathbb{N}$ heißt *POLY-zeitkonstruierbar* genau dann, wenn ein POLY mit Zustandsmenge S, einem ausgezeichneten Zustand $q \in S$ oder einem ausgezeichneten Eingabesymbol $q \in A$ und einer Menge von Endzuständen $F \subseteq S\}$ existiert, so daß für alle $t \in \{1, \ldots, f(n)\}$:

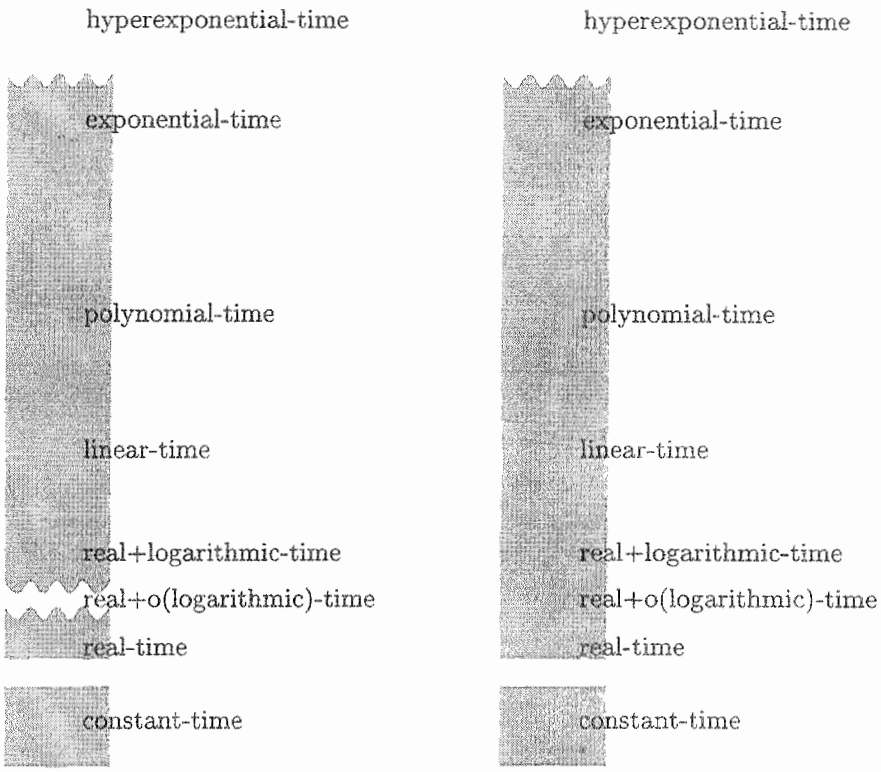
$$\begin{array}{ll} \text{i)} & \pi_1(\mathcal{T}^t(q^n)) \in F \iff t \in \bigcup_{i=1}^n \{f(i)\} \\ \text{ii)} & \pi_1(\mathcal{T}^t(q^n, s_0^n)) \in F \iff t \in \bigcup_{i=1}^n \{f(i)\} \\ \text{iii)} & \pi_{1,1}(\mathcal{T}^t((q, g_0)^n)) \in F \iff t \in \bigcup_{i=1}^n \{f(i)\} \end{array}$$

Die Familie der POLY-zeitkonstruierbaren Funktionen wird mit F(POLY) bezeichnet.

3 Einige Ergebnisse

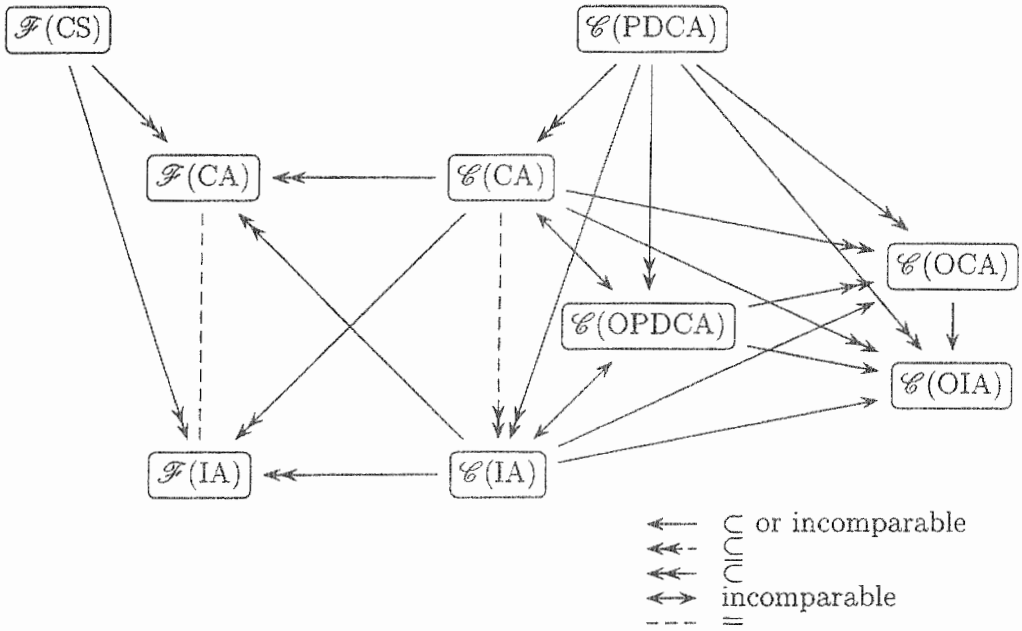
Die hierarchische Struktur der Familien C(OCA) und C(CA) ist in folgendem Diagramm dargestellt. Insbesondere existiert eine Lücke zwischen den konstanten Funktionen k und der Identität. Zum Beispiel ist weder $\lfloor \log \rfloor$ noch $\lfloor \sqrt{\ } \rfloor$ OCA-zeitberechenbar.

Bei den OCAs existiert eine weitere Lücke zwischen $\mathrm{id} + k$ und $\mathrm{id} + \lfloor \log \rfloor$. Die Grenzen sind unscharf dargestellt, da die Basis des Logarithmus beliebig gewählt werden darf.



Hierarchische Struktur von C(OCA) und C(CA).

Bekannte Beziehungen – und implizit offene Probleme – zwischen den verschiedenen untersuchten Familien sind in folgender Abbildung zusammengefaßt.



Bekannte Beziehungen zwischen den Familien.

Literatur

[1] Amoroso, S., Lieblein, E. und Yamada, H. *A unifying framework for the theory of iterative arrays of machines*. 1st Ann. ACM Symposium on Theory of Computing (STOC '69), Marina del Ray, CA, 1969, 259–269.

[2] Buchholz, Th. und Kutrib, M. *On time computability of functions in one-way cellular automata*. Report 9502, Arbeitsgruppe Informatik, Universität Gießen, Gießen, 1995.

[3] Buchholz, Th. und Kutrib, M. *On the computing power of arrays of processing elements*. Report 9603, Arbeitsgruppe Informatik, Universität Gießen, Gießen, 1996.

[4] Buchholz, Th. und Kutrib, M. *Some relations between massively parallel models*. Erscheint in Parallel Computing.

[5] Codd, E. F. *Cellular Automata*. Academic Press, New York, 1968.

[6] Fischer, P. C. *Generation of primes by a one-dimensional real-time iterative array*. Journal of the ACM 12 (1965), 388–394.

[7] Hennie, F. *Iterative arrays of logical circuits*. MIT Press, Cambridge, Mass., 1961.

[8] Kutrib, M. *On stack-augmented polyautomata*. Report 9501, Arbeitsgruppe Informatik, Universität Gießen, Gießen, 1995.

[9] Kutrib, M. und Richstein, J. *Real-time one-way pushdown cellular automata languages*. In Dassow, J., Rozenberg, G. and Salomaa, A. (eds.), *Developments in Language Theory II. At the Crossroads of Mathematics, Computer Science and Biology*. World Scientific Publishing, Singapore, 1996, 420–429.

[10] Kutrib, M. und Worsch, Th. *Investigation of different input modes for cellular automata*. Parcella '94, Akademie Verlag, Berlin 1994, 141–150.

[11] Mazoyer, J. und Terrier, V. *Signals in one dimensional cellular automata*. Research Report RR 94-50, Ecole Normale Supérieure de Lyon, Lyon, 1994.

[12] Vollmar, R. *Algorithmen in Zellularautomaten*. Teubner, Stuttgart, 1979.

L-Systeme mit Valenzen und Bewertungen

Henning Fernau
Universität Tübingen

Ralf Stiebe
Universität Halle

10. September 1996

Der von PĂUN eingeführte Begriff einer Valenzgrammatik wird in diesem Beitrag auf Lindenmayer-Systeme übertragen. Ein *L-System mit Valenzen (valence L system)* (über der Gruppe $(\mathbb{Q}_+, \cdot, 1)$) erhält man, indem man in einem gewöhnlichen L-System jeder Produktion eine positive rationale Zahl als *Valenz* zuordnet.

Formal ist ein 0L-System mit Valenzen ein Konstrukt $G = (\Sigma, P, \omega)$, wobei Σ ein Alphabet, P eine endliche Teilmenge von $\Sigma \times \Sigma^* \times \mathbb{Q}_+$ (die eine endliche Substitution auf dem Alphabet Σ definiert) und ω ein nichtleeres Wort über Σ sind. Aus einem Paar $(w, r) \in \Sigma^* \times \mathbb{Q}_+$ läßt sich das Paar $(w', r') \in \Sigma^* \times \mathbb{Q}_+$ direkt ableiten, $(w, r) \Rightarrow (w', r')$, falls

1. $w = a_1 a_2 \ldots a_m,\ a_i \in \Sigma, 1 \leq i \leq m,$
2. $w' = w_1 w_2 \ldots w_m,\ w_i \in \Sigma^*, 1 \leq i \leq m,$
3. $r' = r \cdot r_1 \cdot r_2 \cdot r_m,$
4. $(a_i \to w_i, r_i) \in P,\ 1 \leq i \leq m.$

Die von G erzeugte Sprache ergibt sich als $L(G) = \{w \in \Sigma^* : (\omega, 1) \overset{*}{\Rightarrow} (w, 1)\}$. Die verschiedenen Varianten von L-Systemen werden analog definiert.

Ein 0L-System mit Valenztest unterscheidet sich von einem 0L-System mit Valenzen dadurch, daß nur Ableitungsschritte der Form $(w, 1) \Rightarrow (w', 1)$ zugelassen sind.

Schließlich betrachten wir noch L-Systeme mit Bewertungen. Ein 0L-System mit Bewertungen ist ein Quadrupel $G = (\Sigma, P, \omega, \varphi)$, wobei $G' = (\Sigma, P, \omega)$ ein gewöhnliches 0L-System ist und φ die Bewertungsabbildung $\varphi : \Sigma \to \mathbb{Q}_+$ ist. Die Abbildung φ wird wie folgt auf Σ^* erweitert: $\varphi(\lambda) = 1$, $\varphi(wa) = \varphi(w) \cdot \varphi(a)$, $w \in \Sigma^*, a \in \Sigma$. Die von G erzeugte Sprache enthält alle Wörter aus $L(G')$ mit der Bewertung 1.

Auf analoge Weise lassen sich [E][D][P][T]0L-Systeme bzw. (uniform) limitierte L-Systeme mit Valenzen (Valenztests, Bewertungen) definieren.

Es stellt sich heraus, daß durch Valenzen, Valenztest und Bewertungen die Erzeugungskraft von L-Systemen beträchtlich erhöht werden kann. Beispielsweise

sind E[P]T0L-Systeme mit Valenztests äquivalent zu [λ-freien] programmierten Grammatiken mit Vorkommensprüfung.

Ein weiteres interessantes Ergebnis ist die Tatsache, daß die Einführung von Terminalsymbolen bei L-Systemen mit Valenzen bzw. Bewertungen keine Auswirkung auf die Erzeugungskraft hat (sehr wohl aber bei Valenztests).

Weiterhin werden die Abschlußeigenschaften der unterschiedlichen Sprachfamilien untersucht.

Als wichtige offene Frage bleibt die Klärung des Entscheidbarkeitsstatus zahlreicher Probleme, z.B. des Mitgliedschafts- und des Leerheitsproblems für ET0L-Systeme mit Valenzen.

sind E[P]T0L-Systeme mit Valenztests äquivalent zu [λ-freien] programmierten Grammatiken mit Vorkommensprüfung.

Ein weiteres interessantes Ergebnis ist die Tatsache, daß die Einführung von Terminalsymbolen bei L-Systemen mit Valenzen bzw. Bewertungen keine Auswirkung auf die Erzeugungskraft hat (sehr wohl aber bei Valenztests).

Weiterhin werden die Abschlußeigenschaften der unterschiedlichen Sprachfamilien untersucht.

Als wichtige offene Frage bleibt die Klärung des Entscheidbarkeitsstatus zahlreicher Probleme, z.B. des Mitgliedschafts- und des Leerheitsproblems für ET0L-Systeme mit Valenzen.

Bounding Resources in Multi-Agent Systems*

Henning Fernau[1] and Markus Holzer[1] and Rudolf Freund[2]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: {fernau/holzer}@informatik.uni-tuebingen.de
[2] Institut für Computersprachen, Technische Universität Wien
Resselgasse 3, A-1040 Wien, Austria
email: rudi@csdec1.tuwien.ac.at

We consider natural complexity measures for multi-agent systems, i.e., systems of agents (or experts) cooperating in a distributed manner in order to achieve a common goal (without a centralized control), which has recently become one of the major fields in AI, see [12]. Natural such resources are: (1) the number of agents in a system, (2) the number of steps an agent may perform, (3) the amount of (static) memory each agent has corresponding to the basic abilities of the agents, (4) the amount of (dynamic) memory each agent has, and (5) the information different agents may exchange.

A grammatical model for multi-agent systems called cooperating distributed grammar systems (CDGS) has been proposed in order to investigate the concepts behind multi-agent systems in a well-defined formal manner, see [3, 8] for an overview on this topic. In a CDGS, different context-free grammar forms—called components—(corresponding to agents) try to work on a common sentential form (corresponding to the workpiece) in a cooperating distributed manner in order to derive a terminal string (corresponding to the common goal).

As regards the resources discussed above, the number of components (1) and the number of productions of the components (3) has thoroughly been investigated. In short, there is an expected trade-off between these resources, what may be paraphrased as "a few clever guys can do the work of many dull ones." This also shows that results obtained in formal language theory models can be meaningfully interpreted within the modelled world.

We discuss the resources (1), (2), (4), and (5) in detail. What is our interpretation of (2), (4), and (5) in the CDGS model? As regards (2), we may require that a component does at most (exactly, or at least, respectively) k derivation steps, leading to the $\leq k$- ($= k$-, or $\geq k$-, respectively) mode. Combining these modes with the t-mode requirement—work as long as you are able to do something, we are led to the modes $(t \wedge \leq k)$, $(t \wedge = k)$, and $(t \wedge \geq k)$ which have not been considered before in the literature. These combinations can be interpreted as: work as long as you are able to do something but finish the work within a given time bound. As measure of the dynamic memory (4) we take the number of nonterminals which occur in the sentential form. Observe that this restriction is just the finite index restriction well-known in the literature on formal languages [4, 10, 11]. Interestingly, this measure originated from two different sources: informational complexity [1] and linguistics [2, 5]. Since nonterminals are the only way to pass information from one context-free component to another in CDGS, this informational complexity interpretation is also valid here (5).

In case of the $(t \wedge \leq k)$- and $(t \wedge = k)$-modes, we can find a trade-off between the complexity measures (1) and (2), which may be paraphrased as "a few hard workers can do the work of many lazy ones." Moreover, CDGS working in the

*First author supported by Deutsche Forschungsgemeinschaft grant DFG La 618/3-1.

$(t \wedge \leq k)$- or $(t \wedge = k)$-mode can be used to characterize programmed languages of finite index. This is (besides the somewhat artificially looking case of absolutely parallel grammars [9]) the first purely syntactical characterization of this language class, which is a really appealing language family, both from a linguistical and a mathematical point of view

In CDGS, all agents (components) may have different abilities (rule sets), but they all work according to the same protocol (mode). It is of course natural to alleviate this second requirement, which leads us to the notion of so-called hybrid CDGS (HCDGS) [6, 7]. We can show that HCDGS with the new modes $(t \wedge \leq k)$, $(t \wedge = k)$, and $(t \wedge \geq k)$ characterize the programmed languages with appearance checking, a fact which is not known for using only modes $*$, t, $\leq k$, $= k$, and $\geq k$. Moreover, we prove that HCDGS with modes t and $\geq k$ characterize CDGS with mode $(t \wedge \geq k)$ which in turn characterize ET0L languages with permitting random context. Therefore, the old question whether ET0L languages with permitting random context characterize programmed languages with appearance checking or whether they are a strict subset gains new interest.

References

[1] R. B. Banerji. Phrase structure languages, finite machines, and channel capacity. *Information and Control*, 6:153–162, 1963.

[2] B. Brainerd. An analog of a theorem about context-free languages. *Information and Control*, 11:561–567, 1968.

[3] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and Gh. Păun. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach, 1994.

[4] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*. Springer, 1989.

[5] S. Ginsburg and E. H. Spanier. Derivation-bounded languages. *Journal of Computer and System Sciences*, 2:228–250, 1968.

[6] V. Mitrana. Hybrid cooperating/distributed grammar systems. *Computers and Artificial Intelligence*, 12(1):83–88, 1993.

[7] Gh. Păun. On the generative capacity of hybrid CD grammar systems. *J. Inf. Process. Cybern. EIK*, 30(4):231–244, 1994.

[8] Gh. Păun. Grammar systems: a grammatical approach to distribution and cooperation. In *Proceedings of ICALP'95, LNCS* 944, pages 429–443. Springer, 1995.

[9] V. Rajlich. Absolutely parallel grammars and two-way deterministic finite state transducers. *Journal of Computer and System Sciences*, 6:324–342, 1972.

[10] G. Rozenberg and D. Vermeir. On the effect of the finite index restriction on several families of grammars. *Information and Control*, 39:284–302, 1978.

[11] G. Rozenberg and D. Vermeir. On the effect of the finite index restriction on several families of grammars; Part 2: context dependent systems and grammars. *Foundations of Control Engineering*, 3(3):126–142, 1978.

[12] M. J. Wooldridge and N. R. Jennings, editors. *Intelligent Agents; ECAI-94 Workshop on Agent Theories, Architectures, and Languages (Amsterdam 1994), LNCS (LNAI)* 890. Springer, 1994.

Über die Automatizität von Doppelfolgen, die durch rationale Funktionen definiert sind

Fritz von Haeseler
CeVis Universität Bremen
Postfach 330440
28334 Bremen
email: fritz@mathematik.uni-bremen.de

Wir betrachten rationale Funktionen $f(X,Y) = p(X,Y)/q(X,Y)$, dabei sind $p(X,Y)$, $q(X,Y)$ Polynome mit Koeffizienten in einem endlichen Ring. Die rationale Funktion definiert durch $f(X,Y) = \sum f(n,t)X^nY^t$ die Doppelfolge $(f(n,t))$. Wir geben ein notwendiges und hinreichendes Kriterium dafür an, wann diese Doppelfolge k-automatisch ist. Weiterhin geben wir eine Anwendung des Resultats auf die Orbits zellulärer Automaten.

The Importance of Solving Formal Language Problems in Knowledge-Based Process Supervision and Control

(Extended Abstract)

Klaus P. Jantke
Hochschule für Technik, Wirtschaft und Kultur Leipzig
FB Informatik, Mathematik und Naturwissenschaften
Postfach 30066
04251 Leipzig

Abstract

From the perspective of the topics of main interest in automata and formal language theory, this is a marginal paper. It deals with the origin of certain problems of formal language theory in an application area called knowledge-based process supervision and control. The focus of the presentation is on motivating the specific formal language theory problems and on the exploitation of their perspective solutions to advance problem solving in the application area.

The key concepts are graph-theoretic, i.e. we are interested in formal languages of graphs. These formal languages are generated by guarded rewriting. Guards are particular formulae of some underlying modal temporal logic. The underlying graph grammar concept will be described on some detail. This investigation will be completed by a survey of the invoked logical reasoning.

In the application domain, plans which are generated for driving a disturbed complex dynamic process back to a working regime are formalized as finite, directed acyclic graphs. Planning knowledge may be understood as some graph grammar. From this perspective, certain key questions of knowledge representation and processing are language-theoretic in spirit. The aim of this paper is to extract theoretical questions for the benefit of knowledge-based process supervision and control.

1 Motivation and Introduction

Recent analyses (cf. [SU94], [SU94] and [KRSU95]) exhibit that the majority of reported accidents in German industries happen in chemical installations. In most cases, human operators turn out to be unable to interpret the flood of incoming alarm indications when an unrestrained disturbance occurs. [SU94], [SU94] and [KRSU95] contain paramount illustrative examples described in detail. As clearly formulated in [Row89] for chemical processes, "in these cases, great expertise and patience are required to control the process. The long time constants and large inertial loads of these processes make operating them much like navigating a supertanker ..." It is not the need of a timely response, but mainly the complexity of the process, its dynamics, and the complexity of message bursts which overcharge human operators.

This bears abundant evidence of the need for computer support in process supervision and control. The most ambitious task is the automatic generation of therapy plans to drive a seriously disturbed process back into a normal working regime, thereby minimizing both the danger of a complete damage of the installation and the loss of production.

Several endeavours have been undertaken to introduce intelligent computer support into supervision and control of complex dynamic processes (cf. [BB87], [FGH+88], [Row89], [May90], [BBMS90], [WIS94], [Arn96a], e.g.). There are several essential inference tasks like fault diagnosis, simulation, and therapy plan generation. Among them, generating therapy plans is the most ambitious problem which is firmly based on diagnosis and simulation (cf. [Arn96a]).

Therapy control knowledge may be reasonably understood as knowledge for generating finite, directed, and acyclic graphs, i.e. as some graph grammar. The aim of the present paper is to extract language-theoretic problems for the benefit of knowledge-based process supervision and control. Finally, there will be developed some scenario illustrating the exploitation of theoretical results to advance future knowledge-based process supervision and control systems.

2 Notions and Notations

For therapy plan generation as investigated here, [Arn96a] is the key reference. Plans are finite, directed, and acyclic graphs (cf. [AJ94a], [AJ94b] and [JA95a]). Graph rewriting is guarded by constraints over some modal temporal logics (cf. [Arn94]). In the present chapter, we will confine ourselves to an illustration of the key concepts.

For every particular application process, the available therapy control knowledge (cf. [Zsc95] for a sample knowledge base of some chemical installation) forms a particular graph grammar. Individual graphs are described as in the following example.

```
(CREATE-ACTION
   :NAME                 Vakuumzusammenbruch-D-Kolonne-infolge-
                         Brüdenüberzug-beseitigen
   :OBJECT               VE-Reaktor
   :PREFERENCE           100
   :SUBACTIONS           ((Brüdenüberzug-beseitigen OBJECT I1)
                          (Reinigen (OUTPUT-TO OBJECT Brüdenstrom) I2)
                          (i.B.-nehmen (OUTPUT-TO OBJECT Brüdenstrom) I3)
                          (Abfahren (COMPONENT-OF* OBJECT Anlage) I4)
                          (Anfahren (COMPONENT-OF* OBJECT Anlage) I5))
   :TIME-RELATIONS       ((b= I1 I4)
                          (=s I4 I2)
                          (b= I2 I3)
                          (b= I3 I5)) )
```

In a frame-like notation, the example above presents some action script to fulfill a specific goal specification of planing named **Vakuumzusammenbruch-D-Kolonne-infolge-Brüdenüberzug-beseitigen** with respect to the technological unit **VE-Reaktor**. Subgoals are listed in the slot :SUBACTIONS. They may be understood as vertices of the given graph. The existing edges are determined by the entries of the slot :TIME-RELATIONS. They are attached to the nodes via some labels. Addittionally, there is specified some semantics of the edges by naming certain interval relations. But this exceeds our present investigation.

For subgoals, there may exist alternative actions of a different preference. Two examples for the subgoal Reinigen are given here:

```
(CREATE-ACTION
   .NAME                 Reinigen
   :OBJECT               Destillationskolonne
   :PREFERENCE           100
   :SUBACTIONS           ((Isolieren-Ausg OBJECT I1)
                          (Auskochen OBJECT I2))
   :TIME-RELATIONS       ((b= I1 I2)) )
```

```
(CREATE-ACTION
   :NAME                 Reinigen
   :OBJECT               Destillationskolonne
   :PREFERENCE           10
   :SUBACTIONS           ((Isolieren-Ausg OBJECT I1)
                          (Isolieren-Eing OBJECT I2)
                          (Ausputzen OBJECT I3))
   :TIME-RELATIONS       ((b= I1 I2)
                          (b= I2 I3)) )
```

From a theoretical point of view, planning is graph rewriting by node expansion. The alternatives are explicitly given as illustrated above. Which individual rewriting step can be performed during planning, depends both on the a priori given preferences and on certain process conditions to be checked at planning

time. The following example contains a typical interval constraint which has to be valid during the whole perspective execution time of its action.
We refrain from going into technological details as well as from discussing peculiarities of the planning approach.

```
(CREATE-ACTION
   :NAME                    Überbrückungs-Regime-fahren
   :OBJECT                  VE-Reaktor
   :PREFERENCE              50
   :INTERVAL-CONDITIONS     (and (> (WTH Primärdiphyltemperatur :VALUE) 240)
                                 (== (OBJECT Zustand :VALUE) in-Betrieb) )
   :SUBACTIONS              ((Arbeitspunkt-normalisieren OBJECT I1)
                             (Schonend-lagern OBJECT I2))
   :TIME-RELATIONS          ((b= I1 I2 )) )
```

The following figure illuminates that generated plans really appear to be graphs. The window is printed from a session using the plan tracer described in [Arn93].

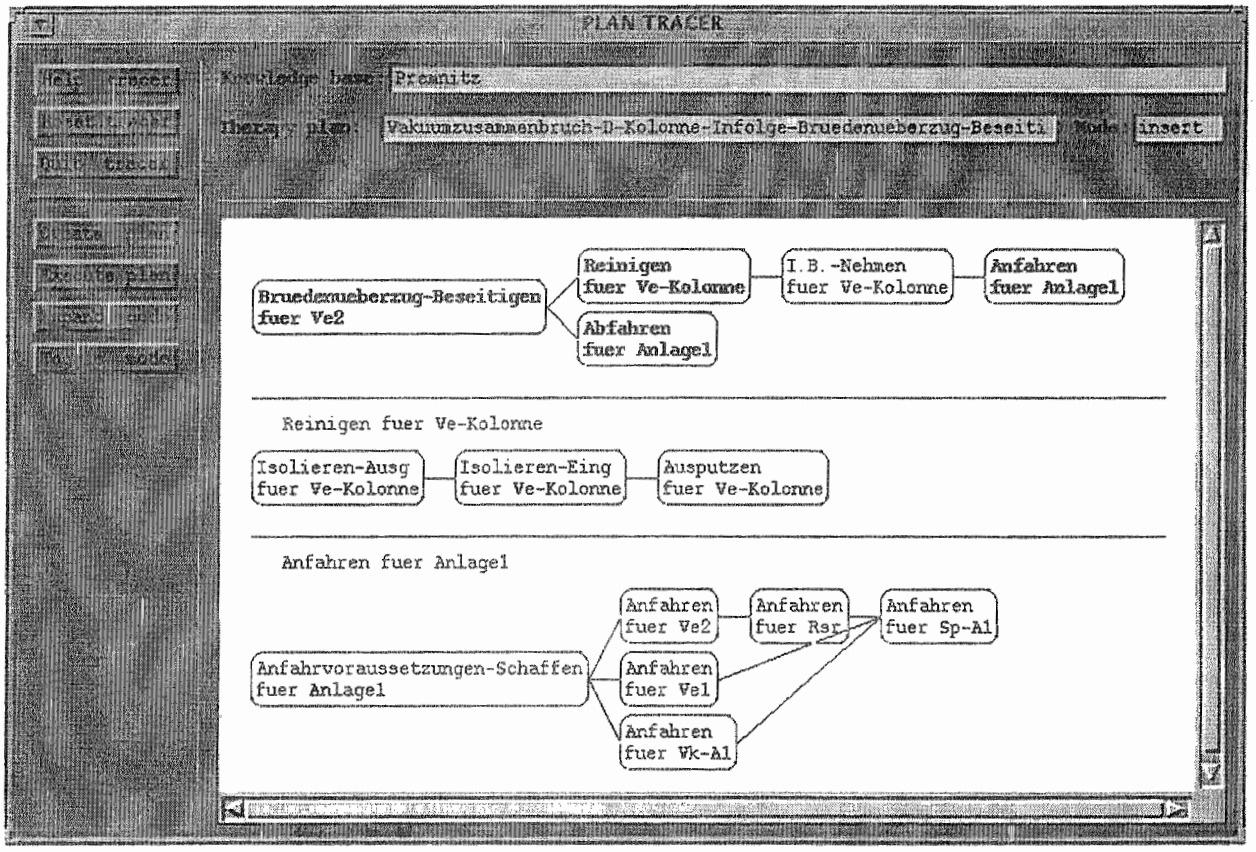


Figure 1: Hierarchically Structured Plan during Generation

The uppermost graph depicts the first example discussed above. Vertices with bold inscriptions indicate complex nodes for which further node expansions are possible. Two of those graphs available for substitution are shown.
The following two subsections are dedicated to a little closer look. They are intended to introduce informally what is described more precisely in [AJ94a], [AJ94b], [JA95a], [Kir95a], [Kir95b], [AJ96c], [AJ96a], e.g., and in [Arn96a], especially. As the aim of the present paper is less technical, we keep the following technicalities as informal and brief as possible.

2.1 Graph Rewriting by Node Expansion

The graph concepts in use are the result of an adaptation of hierarchically structured cellular graphs and pin graphs as investigated in computational complexity (cf. [Len82], [Wan86], [Len87], [LW87], [LW88b], [LW88a], [Wan89], [Len89], and [HLW92]). Whereas in computational complexity the focus is clearly on representing a single graph by some hierarchical description, our emphasis is put on representing large classes, i.e. graph languages, by some grammar-like description. For given knowledge about therapy planning, the corresponding language describes the space of all potentially available therapy plans.
Basically, there is a collection of graphs. The nodes of every graph are classified into elementary nodes and compound nodes. For every compound node, there is a collection of graphs which may be used for node expansion. Here, we concentrate on explaining the technical essence of the graph-grammatical approach: the substitution rule.

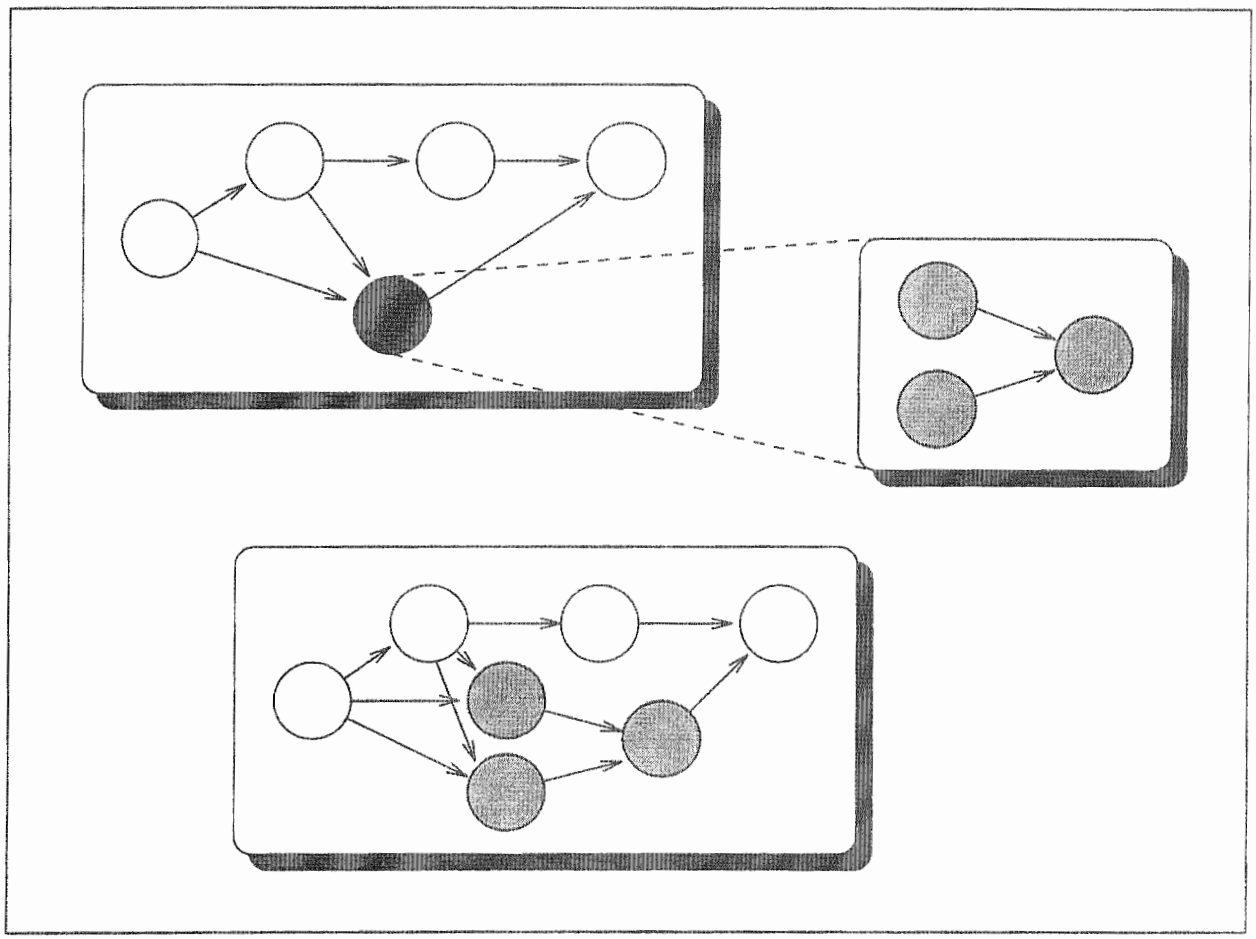


Figure 2: Node Expansion Illustrated

The illustrated substitution rule reflect the interpretation of edges meaning ordering in time. Informally speaking, it works as follows. Every graph has a well-defined set of input nodes, i.e. vertices without incoming edges, and another well-defined set of output nodes, i.e. vertices without outgoing edges. The substitution of an internal node c by some Graph $\mathcal{G}$ proceeds as illustrated above. The node c is replaced. All edges formely reaching c are redirected to the input nodes of $\mathcal{G}$. This may result in splitting edges. Similarly, all edges from c to any other node d result in edges from all the output nodes of $\mathcal{G}$ to d. The substitution for input resp. output nodes is defined analogously. The key difference is that either the set of input nodes or the set of output nodes of the resulting graph changes, as the original set is enriched correspondingly.
Node expansion as explained so far is guarded by constraint checking. The essentials of this issue are introduced in the following section.

2.2 Constraint Monitoring as Constraint Refutation

Constraint checking needs, in formal terms, some underlying logic, i.e. some formal language of formulae, some validity concept, and some algorithmic way of proving or disproving validity. All this has been provided by [Arn94]. The logic has been explicitly exploited and discussed in some more detail in [AJ94a], [AJ94b], [MJA95], [AJ96d], [AJ96c], [JAL96], [JH96], [JLA96], [JM96], [Leh96], and [AJ96a], e.g. [JA96] is – as far as we know – the first publication extracting the role of some logic as parameter of knowledge-based process supervision and control. In the present section, we confine ourselves to a sketch of some essentials, only.

Due to the unavoidable vagueness and incompleteness of information about disturbed complex dynamic processes, generating plans is more inductive than deductive in spirit. This basic insight leads to careful decisions about appropriate knowledge representation (cf. [AJ96d], for a discussion of suitably representing process dynamics, e.g.) and about appropriate knowledge processing (cf. [JLA96], for derived innovative communication scenarios, e.g.). First, we sketch the concepts underlying the representation of vague and incomplete knowledge. Second, we describe the essentials of some modal temporal logic based on this knowledge representation.

We refrain from a comprehensive discussion of modal logics including specific problems like choosing a model of time. The interested reader is directed to [GHR94] and [Sho88], in this respect.

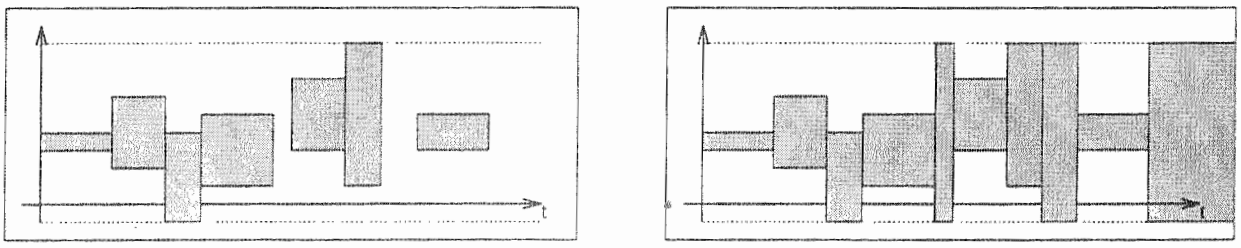


Figure 3: Incomplete and Completed History of some Process Parameter

The key semantic concepts are called episode and history. An episode is a rectangle spanned by some time interval and by some interval of values taken from the range of some given process parameter. A finite set of episodes which do not overlap in time is called a history. Histories are chosen as interpretations of process parameters. Note that in technological processes, where process parameters have determined upper and lower bounds, there are some defaults to complete histories as depicted above.

Even so simple conditions like (> (WTH Primärdiphyltemperatur :VALUE) 240) taken from the example above allow for a variety of interpretations. It turned out that some modal validity concept is appropriate which determines the validity of some constraint over some time interval via the classical modal logic concept of possibility.

For illustration, consider the history displayed by figure 4 describing the uncertain knowledge about some process parameter x. Compared to the two constants c and d, what about the validity of constraints like (> x c) and (> x d) ? Reasonably, the validity of constraints depends on additional time information, i.e. constraints may be valid only with respect to some time interval. In the present example, (> x d) is valid during the time interval $[r, s)$ whereas (> x c) is not.

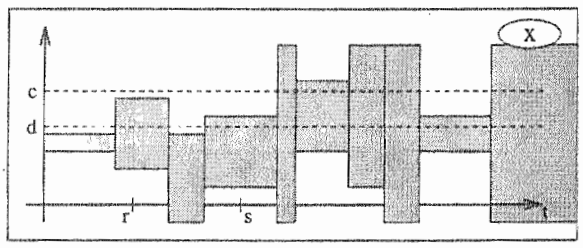


Figure 4: Comparing a Parameter with Constants

More formally, we assume elementary time intervals determining the finest granularity of episodes. There is the usual subinterval relation. Over a given collection of histories taken as models in the classical sense, any constraint φ is valid over some time interval i exactly if there exists at least one elementary subinterval j of i such that φ is valid over j as usual. Intuitively, constraints are considered valid as long as there is any hope that they may really become valid during the perspective time period under consideration.

This modal temporal logic (cf. [Arn94] and [Arn96a], for a comprehensive treatment) is based on a standard interpretation of time which is isomorphic to the natural numbers, although we are aware of the complexity-theoretic implications of this decision (cf. [GHR94]). It is upward hereditary as investigated in [Sho88].

3 Application Scenarios

The present chapter is devoted to knowledge processing scenarios which are essentially related to the graph rewriting mechanisms introduced. We will focus our investigation on plan generation. This knowledge processing mechanism makes intensive and extensive use both of our graph-theoretic and of our logic concepts introduced. For our own contributions to integrating diagnosis and simulation, the reader is directed to [JH96] and [JAL96], respectively. The relevance of action scripts representing the underlying planning knowledge will become quite clear. It turns out to be a key question whether or not this given planning knowledge is both conceptually and representationally adequate for efficient plan generation.

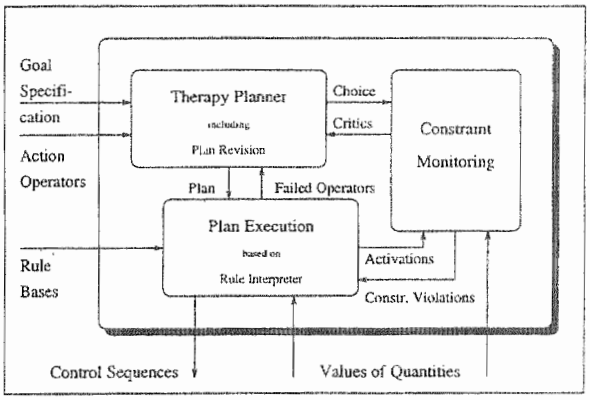


Figure 5: The Architecture of Therapy Control

For briefty, the embedding knowledge processing approach is dropped here. It is described in several of our other publications, from different perspectives. [Arn96a] and [Arn96b], resp., are the most comprehensive recent sources.

Consistent plan generation (cf. [JA95b]) is based on checking constraints with respect to some underlying knowledge (a set of episodes as described above). Because of the inductive character of plan generation (cf. [AJ96b]), the crucial deductive task is constraint refutation. Actions are used for plan generation as long as they are not refuted.

Planning starts over some initially given model. As time proceeds during plan generation, the underlying dynamic knowledge, i.e. the model, is also changing. In fact, there is a sequence of models containing changing histories of process parameters. This dynamics of the knowledge base exhibits the difficulty of designing consistent planning algorithms. Here, we refrain from going into further details.

The key practical problem addressed within the present paper is: *Faced to any application domain, how to qualify the planning knowledge such that it supports successful therapy plan generation?*

Knowledge about therapy plan generation usually consists of dozens or of hundreds of action scripts, i.e. graphs, as exemplified above (cf. [Zsc95], for just one example collection). There is no hope to maintain therapy knowledge without substantial computer support.

The ultimate goal of the programme presented here is to develop some methodology and, furthermore, to develop and implement some tool based on this methodology to check and modify planning knowledge as it occurs in knowledge-based process supervision and control. Systematically, this type of endeavour belongs to the research and development area of knowledge acquisition. Nevertheless, we are approaching it from a quite theoretic point of view.

Roughly speaking, one may imagine a large variety of variants using such a tool. Here, we focus on questions which possess some language-theoretic aspects:

- During knowledge acquisition, new action scripts are checked for their impact on the overall knowledge base. To give a few extreme examples, it may be undesirable
  - that adding some action script does not change the overall class of plans which can be generated,
  - that adding some action script causes the class of possibly generated plans to collaps,
  - that adding some action script results in an "explosion" of the class of possible plans.
- Given complex sets of action scripts are checked for fundamental properties. A particularly strange case may be a collection of action scripts defining the empty language.
- Given complex sets of action scripts are off-line transformed into equivalent sets, i.e. sets generating the same language, which meet certain efficiency criteria.
- During plan execution, the planning knowledge is incrementally reduced to avoid provably useless plans, i.e. the language of possibly generated plans is narrowed in a supervised manner.

4 Formal Language Theory Problems

For dealing with problems like those listed at the end of the preceding chapter, a considerable amount of knowledge about guarded graph rewriting as introduced above seems desirable. We devide the extracted problems (**A**) into those which have a clear formal representation in terms of formal language theory, graph rewriting or logic, and (**B**) into those which are more methodological in spirit.

(**A**) Assume any graph grammar as informally introduced before. This includes for every graph some constraint to be checked prior to substitution.

- Is emptyness of the given language of graphs decidable?
  - Does this depend on certain parameters or on properties of the underlying logic?
  - In case emptyness is decidable, how difficult is this problem?
- Are there "maximal" languages of graphs which can be generated?
  - How to make such a concept of maximality reasonably precise?
- Is the equivalence of guarded graph grammars decidable?
  - If so, how complex is the problem?
  - If not, are there reasonable approximations?
- Is the (proper) inclusion of guarded graph grammars decidable?
  - If so, how complex is the problem?
  - If not, are there reasonable approximations?
- Are there non-trivial monotonbicity properties of guarded graph grammars relating the admissible substitutions (including preferences) to the languages generated?
- Are there non-trivial equivalent transformations of graph grammars?
  - How to compare equivalent grammars?

Questions like the last one lead to less formalized problems. We keep this part quite condensed.

(**B**) It is an informally specified goal of the present investigation to contribute to theoretical research and to practical developments which may help to qualify knowledge underlying therapy plan generation.

- Which characteristica make a collection of graphs appropriate for therapy plan generation?
- Is there any practically relevant partial ordering of grammars? In particular, given any two equivalent grammars, what are reasons to prefer one of them?
- Is there any algorithmically describable way to trade logical properties for graph-theoretic properties?

5 Conclusions

The author feels the urgent need to thank Oksana Arnold for her paramount contribution to the overall research area of knowledge-based process supervision and control, in general, and to her particular research of therapy plan generation, in particular. She has layed the corner stone of the present investigations. The publications listed below, [Arn96a] and [Arn96b] in particular, reflect part of this work. She has a remarkable impact on the work of a couple of students and friends.
It is also worth to be mentioned that recently some students at Leipzig University of Technology provided substantial contributions to the area. These students in 1996 are/were Jörg Herrmann, Daniel Kirsten, Torsten Lehmann, and Daniel Matuschek. Daniel Kirsten's work as exemplified by [Kir95a] and [Kir95b] is particularly relevant for the present investigation.

References

[AJ94a] Oksana Arnold and Klaus P. Jantke. Therapy plan generation in complex dynamic environments. ICSI Report TR-94-054, International Computer Science Institute, Berkeley, California, October 1994.

[AJ94b] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. In *Fifth International Workshop on Graph Grammars and their Application to Computer Science, Williamsburg, Virginia, USA*, November 1994.

[AJ96a] Oksana Arnold and Klaus P. Jantke. Inductive program synthesis for therapy plan generation. *New Generation Computing*, 14(4), 1996.

[AJ96b] Oksana Arnold and Klaus P. Jantke. Planning is learning. In Werner Dilger, Michael Schlosser, Jens Zeidler, and Andreas Ittner, editors, *Machine Learning, 1996 Annual Meeting of the Special Interest Group of Machine Learning of the German Computer Science Society (GI), Chemnitzer Informatik-Berichte CSR-96-06*, pages 12–17. TU Chemnitz, 1996.

[AJ96c] Oksana Arnold and Klaus P. Jantke. Related monotonicity phenomena in complex dynamic processes and in plan generation and simulation. In Susanne Biundo, Thomas Dean, and Richard Waldinger, editors, *12th European Conference on Artificial Intelligence, ECAI'96 Workshop on Cross-Fertilization in Planning, August 12, 1996, Budapest, Hungary*, pages 41–48. European Coordinating Committee for Artificial Intelligence (ECCAI), 1996.

[AJ96d] Oksana Arnold and Klaus P. Jantke. Representing and processing dynamic knowledge in complex dynamic systems. In E. Alpaydın, U. Çilingiroğlu, F. Gürgen, and C. Güzeliş, editors, *Fifth Turkish Symposium on Artificial Intelligence and Neural Networks, TAINN'96, Istanbul, June 27–28, 1996*, pages 351–356. Boğaziçi University, 1996.

[Arn93] Oksana Arnold. Entwicklungswerkzeug zur Analyse generierter Pläne "Plan-Tracer". WISCON Report 03/93, HTWK Leipzig (FH), Fachbereich IMN, December 1993.

[Arn94] Oksana Arnold. A logic of constraints for dynamic process control. WISCON Report 09/94, HTWK Leipzig (FH), Fachbereich IMN, December 1994.

[Arn96a] Oksana Arnold. *Die Therapiesteuerungskomponente einer wissensbasierten Systemarchitektur für Aufgaben der Prozeßführung.* PhD thesis, TH Leipzig, Juli 1996.

[Arn96b] Oksana Arnold. *Die Therapiesteuerungskomponente einer wissensbasierten Systemarchitektur für Aufgaben der Prozeßführung*, volume 130 of *DISKI, Dissertationen zur Künstlichen Intelligenz*. infix, 1996.

[BB87] Dietrich Balzer and Berndt Böhme. Expertensysteme als integraler Bestandteil mikroprozessorgestützter Prozeßleittechnik. *msr*, 30(10):443–448, 1987.

[BBMS90] Dietrich Balzer, Berndt Böhme, Volker May, and Uwe Starke. PROCON–Eine Expertensystemshell für Entwicklung und Einsatz wissensbasierter Beratungssysteme zur Prozeßführung. *KI*, 3(3):12–18, 1990.

[FGH+88] H.W. Fruechtenicht, H.W. Guesgen, T. Hrycej, G. Mörler, and P. Struss. *Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren.* Oldenbourg Verlag, München, 1988.

[GHR94] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic. Mathematical Foundations and Computational Aspects.* Oxford Logic Guides, Vol. 28. Clarendon Press, Oxford, 1994.

[HLW92] Franz Höfting, Thomas Lengauer, and Egon Wanke. Processing of hierarchally defined graphs and graph families. In Burkhard Monien and Thomas Ottmann, editors, *Data Structures and Efficient Algorithms*, volume 594 of *Lecture Notes in Computer Science*, pages 44–69. Springer-Verlag, 1992.

[JA95a] Klaus P. Jantke and Oksana Arnold. Graphgrammatik-Konzepte in der Therapieplanung für komplexe dynamische Prozesse. In Markus Holzer, editor, *4. GI Theorietag "Automaten und Formale Sprachen", 1994, Proc., Herrsching*, pages 42–47. Universität Tübingen, Wilhelm-Schickard-Institut, 1995.

[JA95b] Klaus P. Jantke and Oksana Arnold. Variants of plan generation for complex dynamic systems. In Xin Rao, editor, *Proc. 8th Australian Conference on Artificial Intelligence (AI'95), November 13-17, 1995, Canberra, Australia*, pages 531–538. World Scientific Publ. Co., 1995.

[JA96] Klaus P. Jantke and Oksana Arnold. Probleme und Perspektiven der Wissensbasierten Prozeßführung. In Klaus P. Jantke and Gunter Grieser, editors, *4. Leipziger Informatik-Tage, 29./30. August 1996*, pages 65–70. FORSCHUNGSINSTITUT FÜR INFORMATIONSTECHNOLOGIEN Leipzig e.V., 1996.

[JAL96] Klaus P. Jantke, Oksana Arnold, and Torsten Lehmann. Flexible simulation scenarios for real-time planning in dynamic environments. In John H. Stewman, editor, *Proc. Florida AI Research Symposium (FLAIRS-96), Special Track on Real-Time Planning and Reacting, Key West, FL, USA, May 20–22, 1996*, pages 90–95. Florida AI Research Society, 1996.

[JH96] Klaus P. Jantke and Jörg Herrmann. Logik als Parameter der Diagnose in Kausalnetzen. In Klaus P. Jantke and Gunter Grieser, editors, *4. Leipziger Informatik-Tage, 29./30. August 1996*, pages 71–76. FORSCHUNGSINSTITUT FÜR INFORMATIONSTECHNOLOGIEN Leipzig e.V., 1996.

[JLA96] Klaus P. Jantke, Torsten Lehmann, and Oksana Arnold. Meeting real-time constraints in therapy plan generation for complex dynamic systems by flexible simulation scenarios. In *Proc.* ECHTZEIT'96, *Karlsruhe, June 18–20, 1996*, pages 40–46. Franzis-Verlag, 1996.

[JM96] Klaus P. Jantke and Daniel Matuschek. Developing and implementing planning heuristics in Prolog. In John H. Stewman, editor, *Proc. Florida AI Research Symposium (FLAIRS-96), Key West, FL, USA, May 20–22, 1996*, pages 40–44. Florida AI Research Society, 1996.

[Kir95a] Daniel Kirsten. Properties of formal languages of therapy plans created by graph grammars. Communications of the Algorithmic Learning Group CALG–04/95, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, September 1995.

[Kir95b] Daniel Kirsten. Redundant edges in therapy planning. Communications of the Algorithmic Learning Group CALG–06/95, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, December 1995.

[KRSU95] Michael Kleiber, Sabine Ramm, Susanne Säger, and Hans-Joachim Uth. *Jahresbericht 1994*. Umweltbundesamt Berlin, 1995.

[Leh96] Torsten Lehmann. Probleme und Lösungsansätze bei der Integration von Simulation und Planung. In Klaus P. Jantke and Gunter Grieser, editors, *4. Leipziger Informatik-Tage, 29./30. August 1996*, pages 77–82. FORSCHUNGSINSTITUT FÜR INFORMATIONSTECHNOLOGIEN Leipzig e.V., 1996.

[Len82] Thomas Lengauer. The complexity of hierarchically specified layouts of integrated circuits. In *Proceedings of FOCS'82, the Annual ACM Symposium on Foundations of Computer Science*, pages 358–368, 1982.

[Len87] Thomas Lengauer. Efficient algorithms for finding minimum spanning forests of hierarchically defined graphs. *Journal of Algorithms*, 8:260–284, 1987.

[Len89] Thomas Lengauer. Hierarchical planarity testing algorithms. *Journal of the ACM*, 36(3):474–509, 1989.

[LW87] Thomas Lengauer and Klaus Wagner. The correlation between the complexities of the non-hierarchical and hierarchical version of graph problems. In *Proceedings of STACS'87, Symposium on Theorecital Aspects of Computer Science*, volume 247 of *Lecture Notes in Computer Science*, pages 100–113, 1987.

[LW88a] Thomas Lengauer and Egon Wanke. Efficient analysis of graph properties on context-free graph languages. In *Proceedings of ICALP, Intern. Colloquium on Automata, Languages, and Programming*, pages 379–393, 1988.

[LW88b] Thomas Lengauer and Egon Wanke. Efficient solution of connectivity problems on hierarchically defined graphs. *SIAM Journal of Computing*, 17(6):1063–1080, 1988.

[May90] Volker May. *Grundlagen und Entwicklung von Rahmenexpertensystemen für Aufgaben der Prozeßführung – das Expertensystemshell PROCON*. PhD thesis, TH Leipzig, 1990.

[MJA95] Daniel Matuschek, Klaus P. Jantke, and Oksana Arnold. Generierung von Therapieplänen mit Mitteln der logischen Programmierung. In Peter H. Schmitt, editor, *Logik in der Informatik, 3. Jahrestagung der GI-Fachgruppe 0.1.6*, pages 75–78. Universität Karlsruhe, Fakultät für Informatik, Bericht 23/95, Juni 1995.

[Row89] Duncan A. Rowan. *On-line Expert Systems in Process Industries*. Miller Freeman Publications, 1989.

[Sho88] Yoav Shoham. *Reasoning about Change*. MIT Press, Cambridge, MA, 1988.

[SU94] Susanne Säger and Hans-Joachim Uth. *Bericht. Meldepflichtige Ereignisse 1980–1992*. Umweltbundesamt Berlin, 1994.

[Wan86] Egon Wanke. Resultate und Implementierungen hierarchischer Graphenalgorithmen. Diplomarbeit, Universität-Gesamthochschule Paderborn, 1986.

[Wan89] Egon Wanke. Algorithms for graph problems on BNLC structured graphs. *Information and Computation*, 94(1):93–122, 1989.

[WIS94] WISCON-Team (ohne explizite Autorenangabe). WISCON – Methodenentwicklung für Intelligentes Monitoring and Control. In Gottfried Wolf, editor, *Statusseminar des BMFT Künstliche Intelligenz*, pages 111–135. Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V., 1994.

[Zsc95] Christoph Zschiesche. Erstellung und Analyse von Wissensbasen der Prozeßsicherung und -optimierung zur Verbesserung und Erweiterung geeigneter Wissensrepräsentationen. Master's thesis, Leipzig University of Technology, August 1995.

Verfeinerung von Graphgrammatiken in der Therapieplanung mittels Temporallogik

Daniel Kirsten*
Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fachbereich IMN
Postfach 30066, 04251 Leipzig

1 Motivation/Einführung

Graphgrammatiken und die Generierung von Plänen sind untrennbar verbunden, wenn Pläne als Graphen oder als komplexe Gebilde, die auf Graphen aufbauen, formalisiert werden. Jeder Algorithmus, jedes Programm zur Planung ist somit auf formaler Ebene eine Graphgrammatik. Zur Entwicklung von Planungsalgorithmen sind solide theoretische Untersuchungen der zugrunde liegenden Graphgrammatiken erforderlich. Theoretische Erkenntnisse über Graphgrammatiken können z.B. verborgene Mängel in Planungsalgorithmen aufdecken und Ideen zu deren Behebung liefern.
Im Rahmen des BMFT-Verbundprojektes WISCON wurden Methoden zur Therapieplanung entwickelt. Dabei werden Therapiepläne als endliche, gerichtete, zyklenfreie Graphen, sog. Pingraphen formalisiert. Dabei haben Knoten die Semantik von Aktionen. Die Semantik der Kante $a \rightarrow b$ ist die Aussage, daß die Aktion des Knotens b nach der Ausführung der Aktion von Knoten a stattfindet. Verzweigungen bedeuten keine Alternativen, sondern Nebenläufigkeit. Man unterscheidet zwei Arten von Knoten, Elementarknoten und Komplexknoten. Elementarknoten sind bereits geplant. Komplexknoten enthalten eine Art Zielspezifikation, es ist jedoch noch unklar, wie diese erreicht werden soll. Während der Planung werden diese Komplexknoten durch Pingraphen ersetzt, wobei die eingesetzten Pingraphen wiederum Komplexknoten enthalten können. Dies wird fortgeführt bis alle Komplexknoten ersetzt sind, d.h. bis alle Aktionen geplant sind. Gibt es keine Möglichkeit einen Komplexknoten zu ersetzen, so überwindet der Planer die Sackgasse durch Backtracking.
Die Semantik ist erst von Bedeutung, wenn die Erkenntnisse über die Graphgrammatik in die Planung projiziert werden.
Das grundlegende Szenario kann man sich wie folgt vorstellen: Zunächst wurde als Grundtyp eine Graphgrammatik speziell für die Anforderungen der Therapieplanung definiert. Dieser Grundtyp mit der Bezeichnung "rooted family" und die mit ihm erzeugbaren Sprachen wurden in [Kir95a] und [Kir95b] ausführlich untersucht. Der wesentliche Mangel der "rooted family" ist die Ignoranz von dynamischem Prozeßwissen, d.h. aktuelle Daten können die Ableitungen der Graphgrammatik nicht beeinflussen.
Bei einer Erweiterung dieses Grundtyps besteht die Möglichkeit, daß aktuelles Wissen die Ableitungen der Grammatik beeinflussen kann. Dieser Typ wird mit "rooted family refined by temporal logic" bezeichnet. Die Beeinflussung der Grammatik geschieht durch Überprüfen und eventuelles Verbieten von Ableitungsschritten. Hieraus erklärt sich der Begriff "Verfeinerung" (refinement).
"Überprüfen" bedeuted, es wird mit Hilfe eines Theorembeweisers für eine Temporallogik versucht,

*email: kirsten@imn.htwk-leipzig.de

Formeln zu widerlegen. Diese Formeln sind Paare aus vorab festgelegten Constraints und Zeitintervallen, die während der Planung propagiert werden. Die Temporallogik wurde speziell für die operative Prozeßführung entwickelt [Arn94].
Im Rahmen meiner Diplomarbeit [Kir96] analysiere ich die Unterschiede(?) zwischen beiden Typen. Hier möchte ich Ergebnisse dieser Analyse vorstellen.

2 Ergebnisse

Jede Grammatik vom Typ "rooted family refined by temporal logic" $\mathcal{R}$ kann mit Hilfe einer Transformation Γ in eine Grammatik vom Typ "rooted family" $\mathcal{R}'$ überführt werden. Die Größe der Grammatik $\mathcal{R}'$ (Anzahl der Regeln) nimmt dabei exponentiell in Abhängigkeit von den Bestandteilen von $\mathcal{R}$ zu. (Für eine präzisere Darstellung müßten zu viele Grundlagen formal behandelt werden.) Es ist unbekannt, ob man dieses exponentielle Wachstum verhindern kann.
Eine Umkehrung zur Transformation Γ ist trivial. Daher sind die Grammatikkonzepte "rooted family" und "rooted family refined by temporal logic" gleichwertig. Somit gelten alle Erkenntnisse für die bereits untersuchte "rooted family" auch für den neuen Typ "rooted family refined by temporal logic". Dies bedeutet für die Planung, daß theoretisch kein Problem der "rooted family" durch die "rooted family refined by temporal logic" behoben werden kann.
Betrachtet man jedoch das exponentielle Wachstum bei der Transformation Γ, so ergeben sich durch die neue Graphgrammatik vielleicht bessere praktische Möglichkeiten, da man mit "kleineren" Grammatiken arbeiten kann.

Literatur

[AJ94] Oksana Arnold and Klaus P. Jantke. Therapy plan generation as program synthesis. In Setsuo Arikawa and Klaus P. Jantke, editors, *LNAI*, volume 872, pages 40–55. Springer-Verlag, 1994.

[Arn94] Oksana Arnold. A logic of constraints for dynamic process control. WISCON Report 09/94, HTWK Leipzig (FH), Fachbereich IMN, December 1994.

[Kir95a] Daniel Kirsten. Properties of formal languages of therapy plans created by graph grammars. Communications of the Algorithmic Learning Group CALG–04/95, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, September 1995.

[Kir95b] Daniel Kirsten. Redundant edges in therapy planning. Communications of the Algorithmic Learning Group CALG–06/95, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, December 1995.

[Kir96] *Geplant* Daniel Kirsten. Refining graph grammars in therapy planning using temporal logic. Communications of the Algorithmic Learning Group CALG–??/96, Hochschule für Technik, Wirtschaft und Kultur Leipzig, FB Informatik, Mathematik und Naturwissenschaften, October 1996.

Projektionen in kontextfreien Baumgrammatiken*

Dieter Hofbauer †
Technische Universität Berlin
email: dieter@cs.tu-berlin.de

Maria Huber ‡
Universität Gesamthochschule Kassel
email: maria@theory.informatik.uni-kassel.de

Gregory Kucherov §
CRIN & INRIA-Lorraine, Nancy
e-mail: kucherov@loria.fr

Zusammenfassung

Im Gegensatz zu kontextfreien Wortgrammatiken können in kontextfreien Baumgrammatiken Projektionsregeln nicht immer eliminiert werden. Für solche Baumgrammatiken gibt es also kein direktes Analogon zur Greibach-Normalform. Wir zeigen, daß Projektionsfreiheit dennoch bis zu jedem gegebenen Grad approximiert werden kann: Für jede endliche Menge von Positionen kann eine kontextfreie Baumgrammatik in eine dazu äquivalente Grammatik transformiert werden, die nur Ableitungen zuläßt, wo Projektionen an den gegebenen Positionen unmöglich sind. Als direkte Folgerung erhalten wir einen neuen direkten Beweis für die Entscheidbarkeit des Elementproblems für kontextfreie Baumsprachen.

Eine kontextfreie Baumsprache wird durch eine endlich Menge kontextfreier Termersetzungsregeln aus einem (nullstelligen) Startsymbol erzeugt. Solche Regeln haben die Form

$$A(x_1, \ldots, x_n) \to t.$$

Darin ist A ein (n-stelliges) Nichtterminalsymbol, die x_i sind paarweise verschiedene Variablen, und t ist ein Term, der aus Terminal- und Nichtterminalsymbolen sowie Variablen aus $\{x_1, \ldots, x_n\}$ aufgebaut ist. Das schließt *nichtlineare* Regeln (alle Regeln sind nach Definition zwar linkslinear, nicht aber

*Ein Extended Abstract erscheint in den Proceedings zum *Tbilisi Symposium on Language, Logic and Computation* [9]. Die Arbeit wurde finanziell unterstützt durch PROCOPE (DAAD und MAE).

†Fachbereich Informatik, TU Berlin, FR 6-2, Franklinstraße 28/29, D-10587 Berlin
‡Fachbereich Mathematik/Informatik, GhK Kassel, D-34109 Kassel
§CRIN & INRIA-Lorraine, 615, rue du Jardin Botanique, F-54602 Villers-lès-Nancy, France

notwendig rechtslinear) ebenso mit ein wie *Projektionsregeln*. Bei Projektionsregeln (oder kurz: Projektionen) ist t eine Variable:

$$A(x_1, \ldots, x_n) \to x_i.$$

Im allgemeinen sind für kontextfreie Baumgrammatiken Projektionen unvermeidbar in dem Sinne, daß nicht zu jeder Grammatik eine äquivalente Grammatik ohne Projektionen existiert ([10], siehe auch [4]). Damit kann man nicht auf verkleinernde Regeln verzichten, wodurch das Elementproblem für die erzeugte Sprache zu einer nichttrivialen Aufgabe wird. Dieses Phänomen tritt bei kontextfreien Wortgrammatiken (das ist im wesentlichen der Spezialfall mit ausschließlich monadischen Symbolen) oder allgemeiner bei linearen kontextfreien Baumgrammatiken nicht auf, ebensowenig bei *top-kontextfreien* Grammatiken, wo Nichtterminale in rechten Regelseiten höchstens an der Wurzel stehen dürfen (siehe [2, 8]).

Wir zeigen, wie Projektionen an einer vorgegebenen endlichen Menge von Positionen vermieden werden können, das heißt wir zeigen, wie eine Grammatik so transformiert werden kann, daß danach Projektionen in Satzformen nur noch an Positionen unterhalb einer vorher festgelegten Tiefe k im Termbaum angewendet werden können. Die Ableitungen mit der resultierenden Grammatik haben eine interessante Struktur. Jede solche Ableitung kann nämlich zerlegt werden in

- ein „top-kontextfreies" Anfangsstück, wo Regeln nur an den Wurzeln der Termbäume angewendet werden und Projektionen nicht vorkommen und
- eine Restableitung, wo Regeln nur unterhalb der Tiefe k greifen; es zeigt sich, daß hierfür die Regeln aus der ursprünglichen Grammatik ausreichen.

Wie schon bei kontextfreien Baumgrammatiken generell können wir uns auch hier auf OI-Ableitungen (*outside-in*) beschränken, also auf Ableitungen, wo niemals eine Regel an einer Position oberhalb einer bereits vorher benutzten Position angewendet wird.

Als eine Anwendung dieser Konstruktion erhalten wir einen neuen direkten Beweis für die Entscheidbarkeit des Elementproblems für kontextfreie Baumsprachen. Dieser basiert auf der einfachen Beobachtung, daß alle Terme der Sprache, deren Tiefe k nicht übersteigt, bereits als rechte Regelseiten in der entstehenden Grammatik vorkommen. Deshalb läßt sich damit das Elementproblem auf ein einfaches Erreichbarkeitsproblem reduzieren. Üblicherweise wird zunächst die Entscheidbarkeit des Leerheitsproblems gezeigt, das dazu auf das Leerheitsproblems für die zugehörige Yield-Sprache reduziert wird ([1, 7]). Zusammen mit dem effektiven Abschluß der kontextfreien Baumsprachen unter Schnitt mit regulären ([11, 12]) ergibt sich die Entscheidbarkeit des Elementproblems. In [3] und [6] finden sich solche Ergebnisse für allgemeinere Sprachklassen, in [5] werden Komplexitätsschranken behandelt.

Literatur

[1] A. V. Aho. Indexed Grammars–An Extension of Context-Free-Grammars. *Journal of the ACM*, 15(4):641–671, 1968.

[2] A. Arnold and M. Dauchet. Un Théorème de Duplication pour les Forêts Algébriques. *Journal of Computer and System Sciences*, 13:223–244, 1976.

[3] W. Damm. The IO- and OI-Hierarchies. *Theoretical Computer Science*, 20:95–207, 1982.

[4] M. Dauchet and S. Tison. Structural Complexity of Classes of Tree Languages. In *Tree Automata and Languages*, S. 327–353. Elsevier Science Publishers (North-Holland), 1992.

[5] J. Engelfriet. Iterated Stack Automata and Complexity Classes. *Information and Computation*, 95:21–75, 1991.

[6] J. Engelfriet and H. Vogler. Macro Tree Transducers. *Journal of Computer and System Sciences*, 31:71–146, 1985.

[7] M. Fischer. Grammars with Macro-Like Productions. In *9th Symposium on Switching and Automata Theory*, S. 131–142, 1968. Long version in the PhD–thesis, Harvard University, 1968.

[8] D. Hofbauer, M. Huber und G. Kucherov. Some results on top-context-free tree languages. *Proc. 19th Conf. on Trees in Algebra and Programming*, LNCS Vol. 787, S. 157–171, 1994.

[9] D. Hofbauer, M. Huber und G. Kucherov. How to get rid of projection rules in context-free tree grammars. *Proc. Tbilisi Symposium on Language, Logic and Computation*, CLSI (Stanford), erscheint 1996.

[10] B. Leguy. Réductions, Transformations et Classification des Grammaires Algébriques d'Arbres. *Thèse de Doctorat de Troisième Cycle, Université des Sciences et Techniques de Lille, France*, 1980.

[11] W. C. Rounds. Mappings and Grammars on Trees. *Mathematical Systems Theory*, 4(3):257–287, 1970.

[12] W. C. Rounds. Tree-Oriented Proofs of some Theorems on Context-Free and Indexed Languages. In *Proceedings of 2nd Annual Symposium on Theory of Computing*, S. 109–116, 1970.

Universeller Abschluß regulärer Baumsprachen und minimale Testmengen

Maria Huber *
Universität Gesamthochschule Kassel
email: maria@theory.informatik.uni-kassel.de

Dieter Hofbauer †
Technische Universität Berlin
email: dieter@cs.tu-berlin.de

Der universelle Abschluß $L^\forall$ einer Baumsprache L ist die Menge derjenigen Terme mit Variablen, deren Grundinstanzen alle in L sind. Analog läßt sich der existenzielle Abschluß $L^\exists$ definieren als die Sprache aller Terme, die mindestens eine Grundinstanz in L besitzen. Wir beschränken uns zunächst auf reguläre Sprachen L und auf lineare Terme im Abschluß, das heißt auf Terme ohne Mehrfachvorkommen von Variablen.

Unser Interesse gilt vor allem der Frage, wie das Elementproblem für $L^\forall$ und $L^\exists$ zu lösen ist. Eine mögliche Anwendung – und gleichzeitig unsere vorrangige Motivation – ist die Frage nach der Grundreduzierbarkeit von Termen bezüglich eines Termersetzungssystems: Bezeichnet $Red(R)$ die Sprache der mit R reduzierbaren Grundterme, dann ist $Red(R)^\forall$ gerade die Sprache der grundreduzierbaren Terme.

Für reguläres L sind auch $L^\forall$ und $L^\exists$ regulär; aus einem bottom-up Automat für L läßt sich durch eine modifizierte Potenzmengenkonstruktion leicht ein entsprechender Automat für $L^\forall$ bzw. für $L^\exists$ gewinnen. Deshalb ist das Elementproblem für den universellen und den existenziellen Abschluß einer regulären Sprache immer entscheidbar.

Ein anderer, jedoch sehr eng damit verknüpfter Ansatz verwendet *Testmengen*: Für die Zugehörigkeit eines Terms zu $L^\forall$ reicht es hierbei aus, daß alle Testmengeninstanzen dieses Terms in L sind. Speziell für den Grundreduzierbarkeitstest wurde eine Vielzahl von Methoden zur Konstruktion von Testmengen vorgestellt (siehe unter anderem [3, 4, 1, 5, 2]). Oft werden dabei jedoch unnötig große Mengen berechnet.

Wir zeigen allgemein, daß sich endliche Testmengen für $L^\forall$ und $L^\exists$ immer konstruieren lassen, falls L regulär ist. Insbesondere geben wir Lösungen für folgende Probleme an:

*Fachbereich Mathematik/Informatik, GhK Kassel, D-34109 Kassel
†Fachbereich Informatik, TU Berlin, FR 6-2, Franklinstraße 28/29, D-10587 Berlin

- Charakterisierung (minimaler) Testmengen
- Berechnung minimaler Testmengen
- Minimierung gegebener Testmengen

Um die Grundidee für unseren Ansatz kurz zu skizzieren, geben wir die zentralen Begriffe noch einmal formaler an. Lineare Terme mit Variablen fassen wir als „Kontexte" auf: Variablen werden durch das zusätzliche Symbol $\Box$ dargestellt. Ist Σ die zugrundeliegende Signatur, dann nennen wir $\mathcal{C}_\Sigma = \mathcal{T}_\Sigma(\{\Box\})$ die Menge der Σ-Kontexte. Für $c \in \mathcal{C}_\Sigma$ und $T \subseteq \mathcal{C}_\Sigma$ ist $c \circ T$ die Menge all der Kontexte, die aus c entstehen, indem jedes Vorkommen von $\Box$ durch einen Term aus T ersetzt wird (für verschiedenen Vorkommen von $\Box$ können verschiedene Terme aus T gewählt werden). Der universelle und der existenzielle Abschluß von $L \subseteq \mathcal{T}_\Sigma$ ist nun

$$L^\forall = \{\, c \in \mathcal{C}_\Sigma \mid \forall t \in c \circ \mathcal{T}_\Sigma : t \in L \,\}$$
$$L^\exists = \{\, c \in \mathcal{C}_\Sigma \mid \exists t \in c \circ \mathcal{T}_\Sigma : t \in L \,\}.$$

Für $c \in \mathcal{C}_\Sigma$ ist dann $c \in L^\forall$ äquivalent zu $c \circ \mathcal{T}_\Sigma \subseteq L$. Eine Menge $T \subseteq \mathcal{T}_\Sigma$ ist also eine (Grund-)Testmenge für $L^\forall$, falls für alle $c \in \mathcal{C}_\Sigma$ gilt:

$$c \circ T \subseteq L \quad \text{gdw} \quad c \in L^\forall.$$

Eine zentrale Rolle bei der Charakterisierung von Testmengen spielen die syntaktische Links- und Rechtskongruenz: (Für $c_1, c_2 \in \mathcal{C}_\Sigma$, $T_1, T_2 \subseteq \mathcal{T}_\Sigma$)

$$T_1 \frown_L T_2 \quad gdw \quad \forall c \in \mathcal{C}_\Sigma : \; c \circ T_1 \subseteq L \Leftrightarrow c \circ T_2 \subseteq L$$
$$c_1 \smile_L c_2 \quad gdw \quad \forall T \subseteq \mathcal{T}_\Sigma : c_1 \circ T \subseteq L \Leftrightarrow c_2 \circ T \subseteq L$$

Dies verallgemeinert den Ansatz von Nivat und Podelski ([6, 7]) und ermöglicht insbesondere, Kontexte verschiedener Stelligkeit zu vergleichen. Eine Menge ist damit genau dann eine Testmenge für $L^\forall$, wenn

$$T \frown_L \mathcal{T}_\Sigma.$$

Falls sich eine „einfache" Menge $L_\Box$ mit $L \subset L_\Box \subseteq L^\forall$ angeben läßt ($L_\Box \subseteq \mathcal{C}_\Sigma$, $L_\Box \not\subseteq \mathcal{T}_\Sigma$), dann entstehen meist kleinere Testmengen ($\not\subseteq \mathcal{T}_\Sigma$) als im Grundfall, indem nun $c \circ T \subseteq L_\Box$ getestet wird. Für den Grundreduzierbarkeitstest sind Testmengen mit Variablen dann unerläßlich, wenn die Testmenge zusätzliche Eigenschaften besitzen soll, wie etwas die Eigenschaft, alle irrreduzierbaren Grundterme abzudecken. Unser Ansatz liefert auch für diese verallgemeinerte Problemstellung eine Charakterisierung minimaler Testmengen.

Literatur

[1] R. Bündgen, W. Küchlin. Computing ground reducibility and inductively complete positions. *Proc. 3rd RTA*, LNCS Vol. 355, S. 59–95, 1989.

[2] D. Hofbauer und M. Huber. Linearizing term rewriting systems using test sets. *Journal of Symbolic Computation* 17:91–129, 1994.

[3] J. -P. Jouannaud, E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation* 82(1):1–33, 1989.

[4] D. Kapur, P. Narendran, H. Zhang. Automating inductionless induction using test sets. *Journal of Symbolic Computation* 11:83–111, 1991.

[5] E. Kounalis. Testing for the ground (co-)reducibility property in term-rewriting systems. *Theoretical Computer Science* 106:87–117, 1992.

[6] M. Nivat und A. Podelski. Tree monoids and recognizability of sets of finite trees. In *Resolution of Equations in Algebraic Structures* Vol. 1, Aït-Kaci und Nivat (Hrsg.), S. 351–367. Academic Press, 1989.

[7] A. Podelski. A monoid approach to tree automata. In *Tree Automata and Languages*, Nivat und Podelski (Hrsg.), S. 327–353. Elsevier Science Publishers (North-Holland), 1992.

Molekulare Computer und Formale Sprachen

Rudolf FREUND
Institut für Computersprachen, Technische Universität Wien,
Resselgasse 3, A-1040 Wien, Austria
email: rudi@logic.tuwien.ac.at, Tel. und fax: ++43 1 5041589

Die Interpretation bestimmter biologischer Mechanismen als Operationen auf Wörtern hat in den letzten Jahren auch zu neuen interessanten Ergebnissen für die Theorie der Formalen Sprachen geführt. Die Operation des *Splicings* von DNA- oder RNA-Molekülen kann formalsprachlich derart interpretiert werden [8], daß zwei Wörter an bestimmten Stellen auseinandergeschnitten werden und die Teile sofort kreuzweise wieder miteinander verbunden werden. Betrachtet man Auseinanderschneiden (mit Entstehen von Endmarkierungen) und Zusammenfügen (bestimmter Markierungen) als unabhängig voneinander ablaufende Prozesse, so führt dies zur Definition von *C(utting)/R(ecombination)-Systemen* [6]. Während Splicing- (oderC/R-) Systeme mit endlich vielen Axiomen und Regeln nur reguläre Sprachen erzeugen können, genügt das Hinzunehmen geeigneter Kontrollmechanismen (*multisets, permitting contexts, forbidden contexts* [5]) oder die Verwendung einer unendlichen regulären Menge von Regeln [9], um alle rekursiv aufzählbaren Sprachen erzeugen zu können.

Test-Tube-Systeme stellen ebenfalls eine neue Variante molekularer Computer dar [1], [2]. Die Verwendung von Splicing- [3] oder C/R-Regeln [7] oder aber auch von kontextfreien Produktionen [4] in den *Test-Tubes* sowie von einfachen speziellen (regulären) *Filtern*, über welche die Inhalte der einzelnen Test-Tubes immer wieder auf alle Test-Tubes verteilt werden, ermöglicht die Darstellung jeder rekursiv aufzählbaren Sprache.

References

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems, *Science*, 226 (Nov. 1994), 1021 – 1024.

[2] L. M. Adleman, On constructing a molecular computer, manuscript, January 1995.

[3] E. Csuhaj-Varjú, L. Kari, and Gh. Păun, Test tube distributed systems based on splicing, *Computers and Artificial Intelligence*, Vol. 15 (2) (1996), 211-232.

[4] R. Freund and F. Freund, Test tube systems or How to bake a DNA cake, *Proceedings Workshop Grammar Systems: Recent Results and Perspectives*, Budapest, July 26 -27, 1996.

[5] R. Freund, L. Kari, and Gh. Păun, DNA computing based on splicing: The existence of universal computers, Techn. Report 185-2/FR-2/95, TU Wien, Institute for Computer Languages, 1995.

[6] R. Freund and F. Wachtler, Universal systems with operations related to splicing, *Computers and Artificial Intelligence*, Vol. 15 (4) (1996), 273-294.

[7] R. Freund, E. Csuhaj-Varjú, and F. Wachtler, Test tube systems with cutting/recombination operations, *Proceedings Second Pacific Symposium on Biocomputing "Computing with Biomolecules"*, Maui, Hawaii, January 6 - 9, 1997.

[8] T. Head, Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology*, 49 (1987), 737 – 759.

[9] Gh. Păun, Regular extended H systems are computationally universal, *J. of Automata, Languages and Combinatorics*, Vol. 1, Nr. 1 (1996), 27 – 37.

Temporale Logiken linearer Zeit für Spuren

Dietrich Kuske
Institut für Algebra
Technische Universität Dresden
D-01062 Dresden
kuske@math.tu-dresden.de

Kamp hat 1968 gezeigt, daß die temporale Logik PTL für (endliche und unendliche) Wörter über einem endlichen Alphabet gleich ausdrucksstark ist wie die Logik erster Stufe. Fünfzehn Jahre später schlug Wolper vor, die Ausdrucksstärke der Logik PTL durch die Quantifikation über Mengen zu erweitern. Er zeigte, daß die so konstruierte Logik ebenso ausdrucksstark ist wie die monadische Logik. Mit den klassischen Ergebnissen von Büchi bzw. McNaughton und Papert ergibt sich damit, daß eine Sprache genau dann in PTL definierbar ist, wenn sie aperiodisch (sternfrei) ist, und daß man sie genau dann in der durch Quantifikation erweiterten temporalen Logik definieren kann, wenn sie erkennbar ist.

In seiner Dissertation beschreibt Werner Ebinger eine temporale Logik für Spuren TLPO, die gleich ausdrucksstark ist wie die Logik erster Stufe für Spuren. In Analogie zum Fall der Wörter erweitern wir diese Logik um die Quantifikation über beliebige Mengen. Überraschenderweise ist die Erfüllbarkeit einer Formel dann nicht mehr entscheidbar. Dies wird gezeigt, indem wir das Recurring Tiling Problem in dieser Logik codieren. Damit lassen sich also nicht erkennbare Sprachen in dieser Logik definieren. Aber auch die Frage, ob die definierte Sprache erkennbar ist, ist nicht entscheidbar, was durch die Zurückführung auf das Post'sche Korrespondenz-Problem gezeigt wird.

Beschränkt man die Quantifikation auf gewisse Mengen, so ergibt sich wahrscheinlich[1] eine Logik, die äquivalent ist zur monadischen Logik zweiter Stufe für Spuren.

[1] Ich hoffe, beim Theorietag mit Sicherheit sagen zu können, ob dies gilt oder nicht bzw. wie die Mengen eingeschränkt werden müssen - die Beweise sind noch nicht endgültig sicher.

Erkennbare Sprachen unendlicher Berechnungen von Spur-Automaten

T. Rönsch
Dresden

Die Theorie der Spuren, als Theorie der freien teilweise kommutativen Monoide zunächst für rein kombinatorische Zwecke entstanden, ist in den letzten Jahren zu einem mächtigen Werkzeug für die Modellierung nebenläufiger Prozesse geworden [Ma, Di, DR]. Spuren sind Kongruenzklassen von Wörtern über einem Alphabet A, das zusätzlich mit einer irreflexiven, symmetrischen Relation, der Unabhängigkeit, versehen ist. Dabei betrachtet man die kleinste Kongruenz, die alle Paare (ab, ba) mit $a, b \in A$, $a \parallel b$ enthält. Zwei endliche Wörter sind also äquivalent, wenn sie durch eine Folge von Vertauschungen benachbarter unabhängiger Buchstaben auseinander hervorgehen. Intuitiv heißt das, daß man Ereignisfolgen identifiziert, wenn sie sich nur durch die (lokale) Reihenfolge unabhängiger, d. h. potentiell simultan auftretender, Ereignisse unterscheiden. Bei den Spuren werden also parallele Prozesse durch die Menge aller möglichen äquivalenten sequentiellen Abläufe simuliert, und Nebenläufigkeit reduziert sich auf Vertauschbarkeit.

Einen allgemeineren Zugang bilden die Spur-Automaten und Automaten mit Nebenläufigkeitsrelationen [St, BCS, DK]. Ein *Spur-Automat* ist ein Quadrupel $(A, Q, T, \parallel)$, wobei A ein Alphabet, Q eine Menge (von Zuständen), $T \subseteq Q \times A \times Q$ eine rechtseindeutige Relation (die Transitionsmenge) und $\parallel$ wie oben eine irreflexive, symmetrische Relation über A ist. Transitionsfolgen, bei denen der Startzustand jeder Transition (außer der ersten) mit dem Endzustand der Vorgängertransition übereinstimmt, heißen Berechnungsfolgen. Jedenfalls sollen zwei Berechnungsfolgen $(p, a, q)(q, b, r)$ und $(p, b, q')(q', a, r)$ äquivalent sein, wenn a und b unabhängig sind. Da Ereignisse hier also den Zustand des Systems ändern können, sollte Nebenläufigkeit bedeuten, daß das System durch äquivalente Berechnungsfolgen in

denselben Zustand überführt wird. Bei den Automaten mit Nebenläufigkeitsrelationen darf die Unabhängigkeit von Ereignissen sogar noch vom Zustand des Systems abhängen.

In [GPZ] werden erkennbare Sprachen von unendlichen Spuren untersucht. Einige der Ergebnisse können auf Sprachen unendlicher Berechnungen eines Spur-Automaten verallgemeinert. Dabei muß man allerdings voraussetzen, daß der zugrundeliegende Spur-Automat *voll* ist, d. h. die Spur-Struktur in gewissem Sinne respektiert. Zunächst wird gezeigt, daß Erkennbarkeit und schwache Erkennbarkeit (jeweils durch Morphismen) für Sprachen unendlicher Berechnungen zusammenfallen. Hilfsmittel dafür ist das *Diamantprodukt* von Halbgruppen bzw. Morphismen, eine weitere Verallgemeinerung der in [GPZ] aus dem SCHÜTZENBERGER-Produkt von Monoiden entwickelten Konstruktion. Desweiteren wird bewiesen, daß das Produkt von erkennbaren Berechnungssprachen wieder erkennbar ist. Schließlich werden die erkennbaren Sprachen unendlicher Berechnungen eines vollen Spur-Automaten als endliche Vereinigungen von erkennbaren Sprachen charakterisiert, die sämtlich von einer bestimmten Normalform sind.

[BCS] BOLDI, P., F. CARDONE und N. SABADINI: *Concurrent automata, prime event structures and universal domains.* In: *Semantics of Programming Languages and Model Theory* (M. DROSTE, Y. GUREVICH, eds.), Gordon and Breach Science Publ., OPA, Amsterdam, 1993, 89–108.

[Di] DIEKERT, V.: *Combinatorics on Traces.* Lecture Notes in Computer Science vol. 454, Springer, 1990.

[DR] DIEKERT, V. und G. ROZENBERG: *The Book of Traces.* World Scientific Publ. Co., 1990.

[DK] DROSTE, M. und D. KUSKE: *Recognizable and logically definable languages of infinite computations in concurrent automata.* Internal report of TU Dresden, 1995.

[GPZ] GASTIN, P., A. PETIT und W. ZIELONKA: *An extension of* KLEENE*'s and* OCHMAŃSKI*'s theorems to infinite traces.* Theoretical Computer Science 125 (1994), 167–204.

[Ma] MAZURKIEWICZ, A.: *Trace theory.* In: *Advanced Course on Petri Nets*, Lecture Notes in Computer Science, Springer, 1986, 279–324.

[St] STARK, E. W.: *Connections between a concrete and an abstract model of concurrent systems.* In: *Proceedings of the 5th Conf. on the Mathematical Foundations of Programming Semantics*, Lecture Notes in Computer Science vol. 389, Springer, 1989, 52–74.

Asynchrone zelluläre Automaten und logische Definierbarkeit für Pomsets

Manfred Droste
Institut für Algebra
Technische Universität Dresden
D-01062 Dresden
droste@math.tu-dresden.de

Paul Gastin
LITP, IBP
Universite Paris 7
F-75251 Paris Cedex 05
Paul.Gastin@litp.ibp.fr

Zielonka untersuchte asynchrone zelluläre Automaten für Spuren und zeigte, dass diese ausreichen, alle erkennbaren Spur-Sprachen zu akzeptieren. Nach einem Resultat von W. Thomas sind dies andererseits genau die in der monadischen Logik 2. Stufe definierbaren Sprachen. Wir untersuchen nun deterministische und nichtdeterministische asynchrone zelluläre Automaten für eine grosse Klasse von Pomsets (bewerteten teilweisen Ordnungen), die die Spuren und auch eine grosse Familie von Abhängigkeitsordnungen von Nebenläufigkeitsmonoiden enthalten, und zeigen, dass die durch diese Automaten akzeptierten Sprachen von Pomsets jeweils genau mit den in der monadischen Logik definierbaren Sprachen zusammenfallen.

1. M. Droste and P. Gastin: Asynchronous cellular automata for pomsets without auto-concurrency, in: CONCUR '96, Lecture Notes in Comp. Science, vol. 1119, Springer, 1996, pp. 627-638.

2. W. Thomas: On logical definability of trace languages, in: Proceedings of a workshop of the ESPRIT BRA No 3166: Algebraic and Syntactic Methods in Computer Science Report TUM-I9002 (V. Diekert, ed.), TU München, 1990, pp. 172-182.

3. W. Zielonka: Notes on finite asynchronous automata, R.A.I.R.O. - Informatique Theorique et Applications 21 (1987), 99-135.

Mark Bienk
Friedrichstr. 63
45468 Mülheim
Tel.: 0208/380239

Oliver Boldt
Otto-von-Guericke-Universität
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
email: Oliver.Boldt@student.uni-magdeburg.de

Bernd Borchert
Universität Heidelberg
Mathematisches Institut
Im Neuenheimer Feld 294
69120 Heidelberg
Tel.: 06221/54 6282
email: bb@math.uni-heidelberg.de

Henning Bordihn
Otto-von-Guericke-Universität
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
Tel.: 0391/67 12851
email: bordihn@cs.uni-magdeburg.de

Thomas Buchholz
Universität Giessen
AG Informatik
Arndtstr. 2
35392 Giessen
Tel.: 0641/702 2583
email: buchholz@informatik.uni-giessen.de

Jürgen Dassow
Otto-von-Guericke-Universität
Fakultät für Informatik
Postfach 4120
39016 Magdeburg
Tel.: 0391/67 18829
email: dassow@irb.cs.uni-magdeburg.de

Volker Diekert
FB Informatik
Uni Stuttgart
70565 Stuttgart
Tel.: 0711/7816329
email: diekert@informatik.uni-stuttgart.de

Manfred Droste
Institut für Algebra
TU Dresden
01062 Dresden
Tel.: 0351/463 3908
email: droste@math.tu-dresden.de

Henning Fernau
Wilhelm-Schickard-Institut für Informatik
Uni Tübingen
Sand 13
72076 Tübingen
Tel.: 07071/2977569
email. fernau@informatik.uni-tuebingen.de

Rudolf Freund
Institut für Computersprachen E 185/2
TU Wien
Resselgasse 3
A-1040 Wien
Österreich
email: rudi@logic.tuwien.ac.at

Fritz von Haeseler
CeVis Universität Bremen
Postfach 330440
28334 Bremen
Tel.: 0421/218 2923
email: fritz@mathematik.uni-bremen.de

Dieter Hofbauer
TU Berlin
FB Informatik
FR 6-2
Franklinstr. 28/29
10587 Berlin
Tel.: 030/314 22624
email: dieter@cs.tu-berlin.de

Markus Holzer
Wilhelm-Schickard-Institut für Informatik
Uni Tübingen
Sand 13
72076 Tübingen
Tel.: 07071/297568
email: holzer@informatik.uni-tuebingen.de

Maria Huber
Uni GH Kassel
Fb 17
34109 Kassel
Tel.: 0561/804 4307
email: maria@theory.informatik.uni-kassel.de

Klaus P. Jantke
Hochschule für Technik
Wirtschaft und Kultur
Fb IMN
Postfach 30066
04251 Leipzig
Tel.: 0341/307 6426
email: jantke@informatik.th-leipzig.de

Bernd Jokubeit
Fakultät Informatik
TU Dresden
01062 Dresden
email: bj2@inf.tu-dresden.de

Daniel Kirsten
Hochschule für Technik
Wirtschaft und Kultur
Fb IMN
Postfach 30066
04251 Leipzig
Tel.: 03462/84121
email: kirsten@imn.htwk-leipzig.de

Roman König
Informatik I
Martensstr. 3
91058 Erlangen
Tel.: 09131/857921
email: koenig@informatik.uni-erlangen.de

Armin Kühnemann
Fakultät Informatik
TU Dresden
01062 Dresden
Tel.: 0351/463 8374
email: kuehne@orchid.inf.tu-dresden.de

Dietrich Kuske
Institut für Algebra
TU Dresden
01062 Dresden
Tel.: 0351/463 3642
email: kuske@math.tu-dresden.de

Martin Kutrib
Universität Giessen
AG Informatik
Arndtstr. 2
35392 Giessen
Tel.: 0641/702 2540
email: kutrib@informatik.uni-giessen.de

Hans-Jörg Kreowski
Uni Bremen
FB Mathematik und Informatik
PF 33 04 40
28334 Bremen
Tel.: 0421/218 2948 (3697 - Sekr.)
email: kreo@informatik.uni-bremen.de

Klaus-Jörn Lange
Wilhelm-Schickard-Institut für Informatik
Uni Tübingen
Sand 13
72076 Tübingen
Tel.: 07071/297567
email: lange@informatik.uni-tuebingen.de

Sebastian Maneth
Fakultät Informatik
TU Dresden
01062 Dresden
Tel.: 0351/463 8374
email: maneth@inf.tu-dresden.de

Dirk Nowotka
Fakultät Informatik
TU Dresden
01062 Dresden
email: dn2h@inf.tu-dresden.de

Friedrich Otto
Fachbereich Mathematik/Informatik
Uni GH Kassel
34109 Kassel
Tel.: 0631/804 4573
email: otto@theory.informatik.uni-kassel.de

Holger Petersen
FB Informatik
Uni Stuttgart
70049 Stuttgart
Tel.: 0711/7816451
email: petersen@informatik.uni-stuttgart.de

Torsten Rönsch
Institut für Allgemeine Psychologie
TU Dresden
01062 Dresden
Tel.: 0351/463 5927
email: roensch@rcs.urz.tu-dresden.de

Ina Schiering
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Olshausenstr. 40
24098 Kiel
Tel.: 0431/880 4389
email: is@informatik.uni-kiel.de

Helmut Seidl
FB IV - Informatik
Uni Trier
54286 Trier
Tel.: 0651/201 2853
email: seidl@uni-trier.de

Ralf Stiebe
Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Kurt-Mothes-Straße 1
06120 Halle (Saale)
Tel.: 0345/5524736
email: stiebe@informatik.uni-halle.de

Nicola Traub
(bei W. Huhn)
45478 Mülheim/Ruhr
Heerstr. 74
Tel.: 0208/591304

Wolfgang Thomas
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität Kiel
Olshausenstr. 40
24098 Kiel
Tel.: 0341/880 4480
email: wt@informatik.uni-kiel.de

Heiko Vogler
Fakultät Informatik
TU Dresden
01062 Dresden
Tel.: 0351/463 8232
email: vogler@inf.tu-dresden.de

Klaus Wagner
Institut für Informatik
Universität Würzburg
97072 Würzburg
Tel.: 0931/7962111
email: wagner@informatik.uni-wuerzburg.de

Egon Wanke
Abteilung für Informatik
Mathematisches Institut
Universität Düsseldorf
Universitätsstraße 1
40225 Düsseldorf
Tel.: 0211/811 3730
email: wanke@cs.uni-duesseldorf.de