UNIVERSITÄT TRIER
Mathematik/Informatik

Forschungsbericht Nr. 98-17

8. Theorietag der GI-Fachgruppe 0.1.5 „Automaten und Formale Sprachen"

Helmut Seidl (Hrsg)

Vorwort

Dieser technische Report enthält die Zusammenfassungen der Vorträge des 8. Theorietags der GI-Fachgruppe 0.1.5 *Automaten und Formale Sprachen*. Der 8. Theorietag setzt die Serie der Theorietage der GI-Fachgruppe 0.1.5 in Magdeburg, Kiel, Dagstuhl, Herrsching, Schloss Rauischholzhausen, Cunnersdorf und Barnsdorf fort. Dieses Jahr findet er am 29./30. September 1998 in Riveris bei Trier statt. Wie in den vergangenen zwei Jahren geht dem eigentlichen Theorietag ein eintägiger Workshop voraus, dieses Jahr zu dem Thema:

Prozesse und Formale Sprachen

Als eingeladene Vortragende konnten wir:

- Olaf Burkart,
- Javier Esparza,
- Anca Muscholl,
- Geraud Senizergues und
- Thomas Wilke

gewinnen.

Der Theorietag zusammen mit den eingeladenen Vorträgen wendet sich an alle, die an Fragestellungen aus dem weiteren Umfeld der Theorie der Automaten und Formalen Sprachen interessiert sind. Neben dem Austausch über neue Resultate soll er insbesondere auch Nachwuchswissenschaftlern Gelegenheit geben, Anregungen für eigenständige Forschungen zu erhalten.

Danksagungen: Der GI möchten wir für ihre finanzielle Unterstützung danken; Frau Weiland kämpfte sich durch die Spezialmakros der gesendeten LATEX-Dokumente; und Andreas Neumann half bei technischen Problemen.

Helmut Seidl

Teilnehmerliste

Oliver Boldt
Institut für Informatik
Universität Potsdam
Am Neuen Palais 10
14469 Potsdam
Telefon: 0331/977 1686
boldt@cs.uni-potsdam.de

Henning Bordihn
Fakultät für Informatik
Otto-von-Güricke-Universität
Postfach 4120
39016 Magdeburg
Telefon : 0391-67-12851
bordihn@iws.cs.uni-magdeburg.de

Anne Brüggemann-Klein
TU München
Institut für Informatik
Arcisstraße 21
80290 München
Telefon: 089/45055230
brueggem@informatik.tu-muenchen.de

Thomas Buchholz
Institut für Informatik
Universität Gießen
Arndtstraße 2
35392 Gießen
Telefon: 0641/99 32143
buchholz@informatik.uni-giessen.de

Olaf Burkart
Fachbereich Informatik
Universität Dortmund
Baroper Straße 301
44221 Dortmund
Telefon: 0231-755-5807
burkart@ls5.informatik.uni-dortmund.de

Frank Drewes
Fachbereich 3
Universität Bremen
Postfach 33 04 40
28334 Bremen
Telefon: 0421/218-4335
drewes@informatik.uni-bremen.de

Jürgen Dassow
Fakultät für Informatik
Otto-von-Guericke-Universität
Postfach 4120
39016 Magdeburg
Telefon: 0391-67-18853
dassow@iws.cs.uni-magdeburg.de

Javier Esparza
Institut für Informatik
TU München
Arcisstraße 21
80290 München
Telefon: 089 2892-2405/2/4
esparza@informatik.tu-muenchen.de

Rudolf Freund
Technische Universität Wien
Institut für Computersprachen
Karlsplatz 13
A-1040 Wien, Austria
Telefon: 0043 663019017
rudi@logic.at

Andreas Klein
Institut für Informatik
Universität Gießen
Arndtstraße 2
35392 Gießen
Telefon: 0641/82621
buchholz@informatik.uni-giessen.de

Annegret Habel
Universität Hildesheim
Institut für Informatik
Marienburger Platz 22
33141 Hildesheim
Telefon: 05121 883-728
habel@informatik.uni-hildesheim.de

Roman König
Universität Erlangen
Informatik I
Martensstraße 3
1058 Erlangen
Telefon: 09131-857921
koenig@informatik.uni-erlangen.de

Dieter Hofbauer
Fachbereich Informatik
TU Berlin
Franklinstraße 28/29
10587 Berlin
dieter@cs.tu-berlin.de

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

Markus Holzer
Wilhelm-Schickard Institut für Informatik Universität Tübingen
Sand 13
72076 Tübingen
Telefon: 07071 29-77565
holzer@informatik.uni-tuebingen.de

Martin Kutrib
Institut für Informatik
Universität Gießen
Arndtstraße 2
35392 Gießen
Telefon: 0641/99 32144
kutrib@informatik.uni-giessen.de

Daniel Kirsten
Fakultät für Informatik
TU Dresden
Institut für Softwaretechnik
01062 Dresden
Telefon: 351 463-8293
Daniel.Kirsten@inf.tu-dresden.de

Klaus-Jörn Lange
Universität Tübingen
WSI
Sand 13
72076 Tübingen
Telefon: 07071 29 77567
lange@informatik.uni-tuebingen.de

Martin Leucker
Informatik II
RWTH-Aachen
Ahornstraße 55
Telefon: 0241/80-21210
leucker@i2.informatik.rwth-aachen.de

Anca Muscholl
Universität Stuttgart
Institut für Informatik
Breitwiesenstr. 20-22
70565 Stuttgart
Telefon : (0711) 7816 327
muscholl@informatik.uni-stuttgart.de

Eric Lilienblum
Faculty of Computer Science
Otto-von-Güricke-University
P.O. Box 41 20
39016 Magdeburg
Telefon : 391-67-11309
lili@iws.cs.uni-magdeburg.de

Andreas Neumann
FB 4 - Informatik
Universität Trier
54286 Trier
Telefon: 0651 201-2823
neumann@uni-trier.de

Christof Löding
Margaretenweg 26
24536 Neumünster
Telefon: 04321/32586
chl@informatik.uni-kiel.de

Thomas Noll
Lehrstuhl für Informatik II
RWTH Aachen
Ahornstraße 55
52056 Aachen
Telefon: 241/8888-217
noll@informatik.rwth-aachen.de

Jan-Thomas Löwe
Institut für Informatik
Universität Gießen
Arndtstraße 2
35392 Gießen
Telefon: 0641/99 32143 (Buchholz)
loewe@informatik.uni-giessen.de

Friedrich Otto
Fachbereich Informatik
Universität Kassel
34109 Kassel
Telefon: 0561-804 4573
otto@theory.informatik.uni-kassel.de

Markus Müller-Olm
Lehrstuhl Informatik
Universität Dortmund
Baroper Str. 301
44221 Dortmund
Telefon: 0231/755-5811
mmo@ls5.informatik.uni-dortmund.de

Holger Petersen
Institut für Informatik
Universität Stuttgart
Breitwiesenstraße 20-22
D-70565 Stuttgart
Telefon: 0711-7816-451
petersen@informatik.uni-stuttgart.de

Klaus Reinhardt
Institut für Informatik
Universität Tübingen
Sand 13
72076 Tübingen
Telefon: 07071 29-78964
reinhard@informatik.uni-tuebingen.de

Wolfgang Thomas
Informatik VII
RWTH Aachen
Ahornstraße 55
52056 Aachen
Telefon : 0241-8021700
thomas@informatik.rwth-aachen.de

Markus Seemann
Technische Universität Braunschweig
Institut für Theoretische Informatik
Postfach 3329
38023 Braunschweig
Telefon: 0531-391-9524
m.seemann@tu-bs.de

Erich Valkema
Institut für Informatik
Universität Kiel
Olshausenstr. 40
24105 Kiel
Telefon : 0431-880-4482
ev@informatik.uni-kiel.de

Helmut Seidl
FB IV - Informatik
Universität Trier
54286 Trier
Telefon: 0651 201-2835
seidl@uni-trier.de

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

Geraud Sénizergues
Labri Université Bordeaux 1
351 Cours de la Libération
33401 Talence Cedex/France
Telefon: 05-56-84-60-48
ges@labri.u-bordeaux.fr

Walter Vogler
Institut für Informatik
Universität Augsburg
86135 Augsburg
Telefon: 49-821-598-2120
vogler@informatik.uni-augsburg.de

Ralf Stiebe
Martin-Luther-Universität Halle
Universitätsplatz 10
06099 Halle (Saale)
Telefon : 0345/5524736
stiebe@informatik.uni-halle.de

Dietmar Wätjen
Institut für Theoretische Informatik
TU Braunschweig
Postfach 3329
38023 Braunschweig
Telefon: 0531 3919520
waetjen@iti.cs.tu-bs.de

Johannes Waldmann
Institut für Informatik
Universität Leipzig
Augustusplatz 10-11
04109 Leipzig
Telefon : 0341 97 32 204
joe@informatik.uni-leipzig.de

Klaus Wich
Fachbereich Mathematik/Informatik
Universität Gesamthochschule Kassel
Heinrich-Plett-Str. 40
34109 Kassel
Telefon: 0561 804 4307
wich@theory.informatik.uni-kassel.de

Thomas Wilke
Institut für Informatik
Christian-Albrechts-Universität Kiel
24098 Kiel
Telefon : 0431 880 4479
tw@informatik.uni-kiel.de

Prozesse als Verallgemeinerung von formalen Sprachen

Olaf Burkart
Universität Dortmund

Die Theorie der formalen Sprachen, die ihren Ursprung in der Mathematik, der Logik, und der Linguistik hat, kann als einer der ältesten Zweige der Informatik angesehen werden.

Hauptgegenstand der Untersuchung war bisher der Begriff der formalen Sprache verstanden als Menge von endlichen Worten über einem endlichen Alphabet. Insbesondere die Betrachtung von unendlichen Sprachen erfordert hierbei endliche Formalismen zur Sprachbeschreibung, wie Grammatiken, Ersetzungssysteme, oder Automaten, die in den letzten Jahrzehnten zu einer reichhaltigen Theorie mit imposanten theoretischen Ergebnissen geführt haben.

Prozeßtheorie ist dagegen ein relativ neuer Bereich der Informatik, mit oft abweichenden, eigenen Begriffen und Formalismen, bei dem es um die Beschreibung von reaktiven Systemen geht. Grammatiken, Ersetzungssysteme, und Automaten spielen jedoch auch hier eine zentrale Rolle als Beschreibungsmethode.

In diesem Vortrag soll nun aufgezeigt werden, wie Prozesse als eine Erweiterung von formalen Sprachen angesehen werden können. Anhand von Beispielen wird demonstriert, wie diese verallgemeinerte Sicht zum einen zu einer Verfeinerung und Verbesserung bekannter Ergebnisse aus dem Bereich der formalen Sprachen führt, und zum anderen auch interessante neue Fragestellungen in der Theorie der formalen Sprachen aufwirft.

Process Rewrite Systems

Javier Esparza
Institut für Informatik, TU München

In the last years there has been an increasing interest in the analysis of infinite state automata generated by finite machines like Petri nets, pushdown automata, and various fragments of process algebras. In the talk I introduce Process Rewrite Systems (PRS) and show how a Chomsky-like hierarchy of PRSs can be used to give a unifying description of all these models. Then, I study the decidability and complexity of different model-checking problems for the classes in this hierarchy. Finally, I sketch how these results can be applied to dataflow analysis of concurrent procedural programs with dynamic process creation.

The talk is based on work by many people, but mostly on the Ph. D. Thesis of Richard Mayr.

Deciding Properties for Message Sequence Charts

Anca Muscholl
Universität Stuttgart

Message sequence charts (MSC) are commonly used in designing communication systems. They allow describing the communication skeleton of a system and can be used for finding design errors. Protocols can be specified using MSC graphs, corresponding to rational expressions over finite MSCs. As semantics we consider synchronous fifo communication, which leads precisely to semi-traces (generalizing Mazurkiewicz traces, which represent particular partial orders).

We consider in this talk the complexity of checking properties for specifications based on MSC graphs. Using an exact semantics for the properties which have to be checked easily leads to undecidable problems. The questions become however tractable for a semantics with gaps. In this case we characterize the exact complexity of the verification problems as being complete for NP resp. PSPACE.

$\Gamma(A) \sim \Gamma(B)$

Géraud Sénizergues
Labri Université Bordeaux 1

Keywords: bisimulation; equational graphs; deterministic pushdown automata; rational languages; finite dimensional vector spaces, matrix semi-groups; complete formal system.

The *bisumulation* Problem for equational graphs of finite out-degree is shown to be decidable. We reduce this problem to the η-bisimulation problem for deterministic rational (vectors of) boolean series on the alphabet of a deterministic pushdown automaton M. We then exhibit a *complete formal system* for deducing equivalent pairs of such vectors. This result generalizes the decidability of the equivalence problem for dpda's ([Sén97b], a full proof is givem in [([Sén98, p.12-85]). It generalizes also the decidability of the bisimulation problem for normed pushdown processes ([Sti96]) and for one-counter processes ([Jan97]). The full proofs of this generalization can be found in ([Sén97a]).

References:

[Jan97] P. Jancar. Bisimulation is decidable for one-counter processes. In *Proceedings ICALP 97*, pages 549-559. Springer Verlag, LNCS 1256, 1997.

[Sén97a] G. Sénizergues. $\Gamma(A) \sim \Gamma(B)$?. Technical report, nr 1183-97, LaBRI, Université; Bordeaux I, can be accessed at URL: `http://www.labri.u-bordeaux.fr/~ges`, 1997.

[Sén97b] G. Sénizergues. $L(A) = L(B)$? In *Proceedings INFINITY 97*, pages 1-26. Electronic Notes in Theoretical Computer Science 9, URL: `http://www.elsevier.nl/locate/entcs/volume9.html`, 1997. [Very short version in Proceedings ICALP 97, LNCS 1256 p. 671-681].

[Sén98] G. Sénizergues. $L(A) = L(B)$? Technical report, corrected and extended version of nr 1161-97, LaBRI, Université; Bordeaux I, can be accessed at URL: `http://www.labri.u-bordeaux.fr/~ges`, 1998.

[Sti96] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. In *Proceedings CONCUR 96*, pages 217-232. Springer-Verlag, LNCS 1119, 1996.

Lineare temporale Logik und ihre Fragmente

Thomas Wilke
Christian-Albrechts-Universität Kiel

Seit der Einführung in die Informatik in den siebziger Jahren findet die temporale Logik zunehmend weitere Verbreitung und Anwendung bei der Spezifikation von reaktiven Systemen. In diesem Zusammenhang werden Formeln der linearen temporalen Logik benutzt, um Eigenschaften von Berechnungsfolgen zu beschreiben. Neben ihrer häufigen Verwendung ist die Bedeutung der linearen temporalen Logik aber auch auf ihre engen Bezüge zu anderen Konzepten der Informatik zurückzuführen.

Im Vortrag werden Ergebnisse der vergangenen drei Jahre vorgestellt, die über die bekannten Resultate von Kamp, Stockmeyer, McNaughton, Papert und Schützenberger hinaus eine klare und enge Verbindung zwischen syntaktischen Aspekten der temporalen Logik einerseits und strukturellen Eigenschaften von endlichen Automaten und Halbgruppen andererseits aufdecken. Diese Ergebnisse können dann benutzt werden, um Fragen hinsichtlich der Ausdrucksstärke der linearen temporalen Logik zu klären. So werden im Vortrag u.a. die Fragen beantwortet: Welche temporalen Operatoren sind notwendig, um eine gegebene Systemeigenschaft auszudrücken? Welchen Beitrag zur Ausdrucksstärke leisten der Until- und der Since-Operator?

Neues über die Mächtigkeit verschiedener Varianten von Collagen-Grammatiken

Frank Drewes

Universität Bremen

Es werden einige neue Ergebnisse zum Vergleich der Mächtigkeit kontextfreier, tabellengesteuerter und kontextsensitiver Collagen-Grammatiken vorgestellt.

Zum Verständnis des Folgenden hier zunächst eine kurze Erläuterung der wesentlichen Begriffe. Eine Collage im $\mathbb{R}^n$ ist eine endliche Menge von Teilmengen des $\mathbb{R}^d$. Letztere werden Teile genannt. Zusätzlich können Collagen-Grammatiken nichtterminale Hyperkanten enthalten, die an einer endlichen Sequenz von Punkten aufgehängt sind und eine Markierung tragen. Schließlich enthält jede Collage aus technischen Gründen eine endliche Sequenz von Pinnpunkten.

Eine (kontextfreie) Produktion besteht aus einer Markierung als linker und einer Collage als rechter Seite. Eine solche Produktion wird auf eine Hyperkante mit entsprechender Markierung angewendet, indem die rechte Seite affin transformiert wird, so dass ihre Pinnpunkte auf die Punkte der Hyperkante fallen, und dann die Hyperkante durch die so passend gemachte rechte Seite ersetzt wird.

In tabellengesteuerten Collagen-Grammatiken ist die Menge der Produktionen in Tabellen aufgeteilt. Die Anwendung von Produktionen erfolgt maximal parallel und darf nur jeweils eine Tabelle verwenden. Bei kontextsensitiven Collagen-Grammatiken kann die Anwendbarkeit einer Produktion vom Vorhandensein bestimmter Teile im Kontext der zu ersetzenden Hyperkante abhängig gemacht werden.

Folgende Resultate werden vorgestellt:

(1) Wählt man zur Kodierung von Zeichen eines n-elementigen Alphabets n nicht kongruente Teile und stellt dann Wörter als Sequenzen solcher Teile, angeordnet auf einer Achse und in gleichem Abstand, dar, so sind (unter dieser Kodierung!) von kontextfreien Collagen-Grammatiken genau die regulären Wortsprachen erzeugbar.

    Dies bestätigt eine früher geäußerte Vermutung, die besagte, dass $a^n b^n$ auf diese Weise von kontextfreien Collagen-Grammatiken nicht erzeugbar sei.

(2) Bezüglich der Teileanzahl generierter Collagen ist das mittels tabellengesteuerter Collagen-Grammatiken erzielbare Wachstum höchstens exponentiell.

(3) Kontextsensitive Collagen-Grammatiken können bis auf Kodierung alle rekursiv aufzählbaren Wortsprachen erzeugen, und das gilt selbst im eindimensionalen Fall. Der Beweis besteht im Wesentlichen aus einer Simulation von Zwei-Zähler-Maschinen.

(4) Umgekehrt gibt es aber für alle Dimensionen ≥ 1 Collagensprachen, die von tabellengesteuerten Collagen-Grammatiken erzeugbar sind, nicht jedoch von kontextsensitiven Collagen-Grammatiken. Ein Beispiel einer solchen Sprache ist die Menge aller zweielementigen Collagen der Form $\{[0,2^{-n}],[0,2^n]\}$ mit $n \in \mathbb{N}$ (wobei $[x,x']$ das abgeschlossene Intervall mit den Grenzen x und x' bezeichnet).

Insbesondere erweisen sich also mit (2)–(4) die von tabellengesteuerten bzw. kontextsensitiven Collagen-Grammatiken erzeugten Sprachklassen als unvergleichbar (während, wie bereits bekannt, beide echt größer sind als die Klasse der kontextfreien Collagensprachen).

Sticker Systems

Rudolf Freund
Vienna University of Technology

DNA sequences are double stranded (helicoidal) structures composed of four nucleotides, A (adenine), C (cytosine), G (guanine), and T (thymine); the pairs $\{A, T\}$ and $\{C, G\}$ are called Watson-Crick pairs according to their complementary occurrence in double-stranded DNA molecules. If we have two DNA molecules with sticky ends of single-stranded sequences of A, C, G, T nucleotides that are matching according to the Watson-Crick complementarity, the two sequences may be glued together (by hydrogen bonds).

This matching of complementary nucleotides now is the biological mechanism we have in mind when formally defining the sticker operation for (bidirectional) *sticker systems*, where we prolong to the left and to the right a given domino of (single or double) symbols by using given single stranded strings or even more complex dominoes with sticky ends, gluing these ends together with the sticky ends of the current sequence according to a complementarity relation. Iterating these prolongations we get "computations" of possibly arbitrary length. A codification procedure (a projection) finally gives "meaning" to blocks of symbols, in this way yielding the "decoded" language.

The generative power of several variants of sticker systems yields the following results [2]: Unidirectional sticker systems show up to represent the regular languages. Bidirectional sticker systems of *bounded delay* (i.e., the length of the sticky ends during a derivation is bounded) represent the linear languages. Finally, any recursively enumerable language can be represented as the projection of a language generated by a bidirectional sticker system of unbounded delay, which can be proved by using the characterization of recursively enumerable languages by equality sets [4]:

For each recursively enumerable language L, $L \subseteq T^*$, there exist two non-erasing morphisms $h_1, h_2 : \Sigma_2^* \to \Sigma_1^*$, a regular language $R \subseteq \Sigma_1^*$, and a projection $h_T : \Sigma_1^* \to T^*$ defined by $h_T(a) = \begin{cases} a, & \text{if } a \in T, \\ \lambda, & \text{if } a \in \Sigma_1 \setminus T, \end{cases}$ such that $L = h_T(h_1(E(h_1, h_2)) \cap R)$, where $E(h_1, h_2) = \{w \in \Sigma_2^* \mid h_1(w) = h_2(w)\}$ is the equality set of h_1 and h_2.

Moreover, this universality result for bidirectional sticker systems also allows us to establish an optimal representation result for recursively enumerable languages: Any recursively enumerable language L, $L \subseteq \Sigma^*$, can be written as $L = h(L_1 \cap L_2)$, where L_1 and L_2, $L_1, L_2 \subseteq \Sigma_1^*$, are minimal linear languages (a *minimal* linear language is generated by a linear grammar with only one non-terminal symbol S; if, moreover, the only terminal rule is $S \to \lambda$, then the linear grammar and the generated linear language are called *strictly minimal*) and h is a projection from Σ_1 on Σ.

With a small refinement of this proof given in [2], in [3] it was shown that a strictly minimal language L' can be written as $h'(L_0)$, where L_0 is the reversed copy language $rcp(V^+) = \{w\bar{w}^R \mid w \in V^+\}$ for some alphabet V. ($\bar{V} = \{\bar{a} \mid a \in V\}$ contains the barred versions of the letters in V.) From these representations we immediately infer another representation of L that is based on one single generator language L_0 as $L = h_T(g_1(L_0) \cap g_2(L_0))$, where g_1, g_2 are two deterministic generalized sequential machines and $L_0 = rcp(\{0,1\}^+)$.

Literatur

[1] J. Engelfriet, G. Rozenberg, Fixed point languages, equality languages, and representations of recursively enumerable languages, *Journal of the ACM*, 27 (1980), 499 – 518.

[2] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, Bidirectional sticker systems, *Third Annual Pacific Symposium on Biocomputing*, Hawaii, 1998.

[3] R. Freund, Bidirectional Sticker Systems and Representations of Recursively Enumerable Languages by Copy Languages, 1998.

[4] L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa, S. Yu, DNA computing, sticker systems, and universality, *Acta Informatica*, 1998.

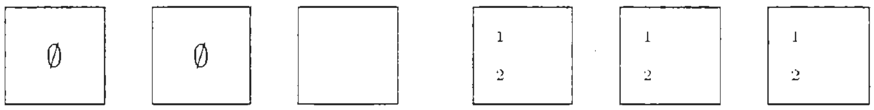
Graph Transformation with Injective Matching

Annegret Habel
Universität Hildesheim

We investigate three variations of the classical double-pushout approach to graph transformation. By considering injective matching and/or arbitrary right-hand morphisms in rules, three approaches are obtained that have greater expressive power than the classical approach. For each of them, we clarify whether the well-known commutativity and parallelism theorems are still valid and—where this is not the case—give modified results. In particular, for the most general approach with injective matching and arbitrary right-hand morphisms, we establish sequential and parallel commutativity by appropriately strengthening sequential and parallel independence.

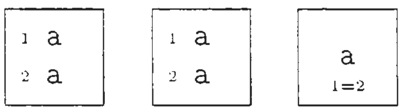
Introduction

Traditionally, in the double-pushout approach to graph transformation, left-hand sides of rules are matched by arbitrary graph morphisms satisfying the so-called gluing condition (see [Ehr79, CMR+97]). However, sometimes it is more natural and convenient to require that matching morphisms must be injective. For example, the set of all (directed, unlabelled) loop-free graphs can be generated from the empty graph by the following two rules if injective matching is assumed:



It seems impossible to generate this graph class in the classical double-pushout approach without using non-terminal labels and a larger number of rules.

To give a second, non-grammatical example, consider the problem of identifying in a graph all nodes labelled with some fixed label a. This is easily achieved—assuming injective matching—by applying as long as possible the following rule:



Without injective matching, however, there is no finite, terminating set of rules solving this problem for arbitrary input graphs. (A set of rules is *terminating* if it does not admit any infinite rewrite sequence.)

In what follows, we investigate two variations of the double-pushout approach in which matching morphisms are required to be injective. We denote these approaches by $\mathrm{DPO}^{\mathrm{i/i}}$ and $\mathrm{DPO}^{\mathrm{a/i}}$, where the first component of the exponent indicates whether right-hand morphisms in rules are injective or arbitrary, and where the second component refers to the requirement for matching morphisms. (So our second example belongs to $\mathrm{DPO}^{\mathrm{a/i}}$.)

Besides the classical approach $\mathrm{DPO}^{\mathrm{i/a}}$, we will also consider $\mathrm{DPO}^{\mathrm{a/a}}$. Obviously, $\mathrm{DPO}^{\mathrm{a/a}}$ and $\mathrm{DPO}^{\mathrm{a/i}}$ have greater expressive power than $\mathrm{DPO}^{\mathrm{i/a}}$ and $\mathrm{DPO}^{\mathrm{i/i}}$, respectively, as the latter are included in the former. Moreover, using a quotient construction for rules, we will show that $\mathrm{DPO}^{\mathrm{i/i}}$ and $\mathrm{DPO}^{\mathrm{a/i}}$ can simulate $\mathrm{DPO}^{\mathrm{i/a}}$ and $\mathrm{DPO}^{\mathrm{a/a}}$, respectively, in a precise and strong way. Thus, the relationships between the approaches can be depicted as follows:

$$\begin{array}{ccc} & \mathrm{DPO}^{\mathrm{i/a}} & \\ \mathrm{DPO}^{\mathrm{a/a}} & & \mathrm{DPO}^{\mathrm{i/i}} \\ & \mathrm{DPO}^{\mathrm{a/i}} & \end{array}$$

The question, then, arises to what extent the theory of $\mathrm{DPO}^{\mathrm{i/a}}$ carries over to the stronger approaches. We answer this question for the classical commutativity and parallelism theorems by either establishing their validity or by giving counterexamples and providing modified results. We summarize our results in the following table, where "yesïndicates a positive result and "nomeans that there exists a counterexample.

	$\mathrm{DPO}^{\mathrm{i/a}}$	$\mathrm{DPO}^{\mathrm{a/a}}$	$\mathrm{DPO}^{\mathrm{i/i}}$	$\mathrm{DPO}^{\mathrm{a/i}}$
char. of parallel independence	yes	yes	yes	yes
char. of seqential independence	yes	no	yes	no
parallel commutativity	yes	yes	yes	no
weak parallel commutativity I	yes	yes	yes	yes
weak parallel commutativity II	yes	yes	yes	yes
sequential commutativity	yes	yes	yes	no
weak sequential commutativity I	yes	yes	yes	yes
weak sequential commutativity II	yes	yes	yes	yes
parallelism	yes	yes	no	no
weak parallelism	yes	yes	yes	yes

Literatur

[CMR+97] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, Michael Löwe. Algebraic approaches to graph transformation. Part I: Basic concepts and double pushout approach. In Grzegorz Rozenberg, ed., Handbook of Graph Grammars and Computing by Graph Transformations, volume 1: Foundations, 163–245. World Scientific, 1997.

[Ehr79] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, G. Rozenberg, eds., Graph-Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science 73, 1–69, 1979.

Reguläre kanonische Systeme und das Leerheitsproblem für indizierte Grammatiken

Dieter Hofbauer
TU Berlin

Die von Aho [1] eingeführten indizierten Grammatiken definieren eine Sprachklasse, die die Klasse der kontextfreien Sprachen weit umfaßt, die aber dennoch deren schöne Abschluß- und Entscheidbarkeitseigenschaften bewahrt. Das Leerheitsproblem spielt dabei eine zentrale Rolle; so läßt sich das Elementproblem für indizierte Grammatiken ebenso darauf reduzieren wie entsprechende Fragen für Makrogrammatiken [5] oder für kontextfreie Baumgrammatiken. Obwohl oft zitiert, ist der bei Aho angegebene Entscheidbarkeitsbeweis mangelhaft; Maibaum [7] etwa schreibt: „It has been brought to my attention by A. Salomaa through K. Culik that Aho's proof of the decidability of the emptiness problem for indexed grammars is incorrect and no correct proof is known."

Als Alternative zu bereits existierenden Verbesserungen (etwa von Albert [2]) wird gezeigt, wie sich das Leerheitsproblem für indizierte Grammatiken ganz direkt auf das Elementproblem für reguläre kanonische Systeme (mit beliebiger Zahl von Prämissen) reduzieren läßt. Diese speziellen Postsysteme [8] erzeugen nur reguläre Sprachen, wie Büchi bereits 1964 behauptet [3] und später bewiesen hat [4], siehe auch Kratko [6].

Literatur

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

[2] J. Albert. Über indizierte und m-Block-indizierte Grammatiken. *Dissertation, Universität Karlsruhe*, 1976.

[3] J. R. Büchi. Regular Canonical Systems. *Archiv für Math. Logik und Grundlagenforschung*, 6:91–111, 1964.

[4] J. R. Büchi und W. H. Hosken. Canonical Systems which Produce Periodic Sets. *MST*, 4:81–90, 1970.

[5] M. Fischer. Grammars with Macro-Like Productions. *9th Symposium on Switching and Automata Theory*, 131–142, 1968.

[6] M. I. Kratko. A Class of Post Calculi. *Soviet Mathematics Doklady*, 6(6):1544–1545, 1965.

[7] T. S. E. Maibaum. Pumping Lemmas for Term Languages. *JCSS*. 17:319–330. 1978.

[8] E. Post. Formal Reductions of the General Combinatorial Decision Problem. *Am. J. Math.*, 65:197–215, 1943.

On the Computational Complexities of Contextual Languages

Markus Holzer
Universität Tübingen

Contextual grammars and languages were investigated by Marcus [3] in 1969 in order to model basic phenomenon in descriptive linguistics. The generative process in a contextual grammar is based on two dual linguistic operations, namely insertion of a word in a give context and adding a context to a given word.

Let $k \geq 1$. A *contextual grammar* is a construct $G = (\Sigma, A, (S_1, C_1), \ldots, (S_k, C_k))$, where Σ is a finite alphabet, $A \subseteq \Sigma^*$ is a finite set of *axioms*, $S_i \subseteq \Sigma^*$ are the *selector sets*, and $C_i \subseteq \Sigma^* \times \Sigma^*$ are the finite *context sets*. There are two basic modes of derivations: For two words $x, y \in \Sigma^*$ we define

1. $x \Rightarrow_{ex} y$ if and only if $x \in S_i$, $y = uxv$, and $(u, v) \in C_i$ for some $1 \leq i \leq k$ (external mode) and
2. $x \Rightarrow_{in} y$ if and only if $x = x_1x_2x_3$, $x_2 \in S_i$, $y = x_1ux_2vx_3$, and $(u, v) \in C_i$ for some $1 \leq i \leq k$ (internal mode).

The language generated by G with respect to the derivation mode $\alpha \in \{ex, in\}$ is defined as $L_\alpha(G) := \{\, w \in \Sigma^* \mid \text{there is an } x \in A \text{ such that } x \Rightarrow_\alpha^* w \,\}$, where $\Rightarrow_\alpha^*$ denotes the reflexive transitive closure of $\Rightarrow_\alpha$. If the sets $S_1, \ldots, S_k$ are languages in a given family F, then G is said to be with F *choice.*

Although the field of contextual languages is very old only very few results on the computational complexity of contextual languages are known (see, e.g., [2]). We try to fill this gap addressing the problems of fixed and general membership for contextual grammars with finite, regular, linear context-free, and context-free choice working in external and internal mode of derivation in more detail [1].

In the external derivation mode it turns out that in most cases the membership problem for contextual grammars with F choice is equivalent to the corresponding membership question for F. For instance, if F is the family of linear context-free languages, then the fixed membership problem for contextual grammars with F choice is *NL*-complete. On the other hand, in case of the internal mode, the membership problem becomes much harder, because even in case of linear context-free choice the fixed membership problem is shown to be already *NP*-complete.

Literatur

[1] M. Holzer. On the complexities of contexutal languages—the external and internal mode. Unpublished manuscript, 1998.

[2] L. Ilie. On computational complexity of contextual languages. *Theoretical Computer Science*, 183(1):33–44, 1997.

[3] S. Marcus. Contextual grammars. *Rev. Roum. Math. Pures Appl.*, 14(10):1525–1534, 1969.

Einige dem Sternproblem in Spurmonoiden verwandte Unentscheidbarkeitsresultate

Daniel Kirsten
Technische Universität Dresden

Im Vortrag beschäftigen wir uns mit einigen Entscheidbarkeitsproblemen bei erkennbaren Spursprachen. Spurmonoide, d.h. freie, partiell kommutative Monoide wurden von Cartier und Foata 1969 eingeführt, doch erst seit Mazurkiewicz sie 1977 als potentielles Modell für nebenläufige Prozesse beschrieb, werden sie von der Informatik und Mathematik systematisch untersucht [1, 2]. Ein Aspekt der Spurtheorie ist die Untersuchung erkennbarer Spurspachen. Erkennbare Spursprachen kann man als eine Verallgemeinerung des Konzeptes der regulären Sprachen auf Spurmonoide auffassen.

Im Gegensatz zu regulären Sprachen im freien Monoid sind erkennbare Spursprachen nicht gegen Iteration (d.h. Kleene *) abgeschlossen. Dadurch ergibt sich das Sternproblem: Ist es entscheidbar, ob die Iteration einer erkennbaren Spursprache eine erkennbare Spursprache ergibt? Obwohl sich seit Mitte der 80er Jahre zahlreiche Publikationen mit diesem Problem beschäftigen, wurden nur Teillösungen erreicht. Der jüngste Erfolg wurde 1994 von Richomme erzielt: Das Sternproblem ist entscheidbar in Spurmonoiden, die kein Untermonoid C4 enthalten [4]. Das Spurmonoid C4 ist (bis auf Isomorphie) das Kartesische Produkt $\{a, b\}^* \times \{c, d\}^*$. Es ist nicht bekannt, ob das Sternproblem im C4 bzw. in Spurmonoiden, die ein C4 als Untermonoid enthalten, entscheidbar ist.

Im Vortrag zeige ich die Unentscheidbarkeit einiger verwandter Probleme: Man setze ein Spurmonoid voraus, das ein C4 als Untermonoid enthält. Dann ist unentscheidbar, ob für zwei erkennbare Sprachen K und L die Sprache K in L^* enthalten ist. Andererseits ist entscheidbar, ob K^* in L enthalten ist. Weiterhin ist unentscheidbar, ob $K \cap L^*$ und $K \cup L^*$ erkennbar sind. Es ist auch unentscheidbar, ob $K \cup L^*$ das gesamte Monoid ergibt.

Die Beweise basieren auf einer Reduktion auf das Post'sche Korrespondenzproblem und sind in [3] verfügbar.

Diese Arbeit wurde unterstützt vom Graduiertenkolleg „Spezifikation diskreter Prozesse und Prozeßsysteme durch operationelle Modelle und Logiken" der Deutschen Forschungsgemeinschaft.

Literatur

[1] V. Diekert and Y. Métivier. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3, Beyond Words*, pages 457–534. Springer-Verlag, Berlin Heidelberg, 1997.

[2] V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.

[3] D. Kirsten. Some undecidability results related to the star problem in trace monoids. Technical Report ISSN 1430-211X, TUD/FI98/07, Dresden University of Technology, Department of Computer Science, May 1998.

[4] G. Richomme. Some trace monoids where both the star problem and the finite power property problem are decidable. In I. Privara et al., editors, *MFCS'94 Proceedings*, volume 841 of *Lecture Notes in Computer Science*, pages 577–586. Springer-Verlag, Berlin, 1994.

Iterative Arrays mit eingeschränktem Nichtdeterminismus

Andreas Klein
Justus-Liebig-Universität Gießen

Ein Iteratives Array (IA) kann als Modell für massiv parallele Rechner mit sequentieller Eingabe aufgefaßt werden [2]. Dabei werden endliche Automaten in einer Reihe miteinander verbunden, so daß jeder Automat den eigenen Zustand und den seiner beiden Nachbarn erkennen kann. Ein Automat wird als Ursprung ausgezeichnet und kann zusätzlich die Eingabe lesen. In jedem Zeitschritt ändern die Automaten gleichzeitig ihre Zustände.

Wir betrachten eine Variante eines Iterativen Arrays (GIA, guessing IA), bei dem die Ursprungszelle zu nichtdeterministischen Übergängen fähig ist. Eine Verallgemeinerung dieses Modells besteht darin, daß man fordert, daß sämtliche nichtdeterministischen Übergänge in den ersten $f(n)$ Zeit-Schritten stattfinden. Dabei bezeichnet n die Länge der Eingabe. Wir sprechen dann von einem f-GIA.

Wir stellen ein notwendiges Kriterium für $L \in \mathrm{L}_{rt}(f - GIA)$ vor. Als Anwendung dieses Kriteriums beweisen wir:

Hierarchie-Satz Seien $f(n)$, $g(n)$ und $h(n)$ Funktionen für die

1. $f(n) \geq g(n)$ und $\{a^{g(n)}b^{f(n)-g(n)} | n \in \mathbb{N}\} \in \mathrm{L}_{lt}(CA)$
2. $\{a^{h(n)}b^n | n \in \mathbb{N}\} \in \mathrm{L}_{lt}(CA)$
3. $f(9n+(n+1)h(n)) \geq c\log(h(n))$ und $g(9n+(n+1)h(n)) = o(\log(h(n)))$

gilt.

Dann ist

$$\mathrm{L}_{rt}(g - GIA) \subsetneq \mathrm{L}_{rt}(f - GIA) \quad .$$

Außerdem untersuchen wir die Abschlußeigenschaften der betrachteten Sprachfamilien. Unter anderem zeigen wir, daß $\mathrm{L}_{rt}(GIA)$ eine AFL und $\mathrm{L}_{rt}(f-GIA)$ im allgemeinen nicht unter Komplementbildung abgeschlossen ist.

Literatur

[1] Thomas Buchholz, Andreas Klein und Martin Kutrib. One guess oneway cellular arrays. *Mathematical Foundations of Computer Science 1998.* LNCS 1450, 807–815.

[2] S. N. Cole. Real-time computation by n-dimensional iterative arrays of finite-state machines. In 7^{th} *Ann. IEEE Symposium on Switching and Automata Theory*, 53–77. IEEE, 1966.

Erkennbare Sprachen in Teilbarkeitsmonoiden

Dietrich Kuske (gemeinsame Arbeit mit Manfred Droste)

Institut für Algebra, TU Dresden

Der bekannte Satz von Kleene besagt, daß in endlich erzeugten freien Monoiden die rationalen und die erkennbaren Sprachen zusammenfallen. Ochmański hat dieses Ergebnis auf Spurmonoide erweitert und gezeigt, daß in einem endlich erzeugten Spurmonoid eine Sprache genau dann erkennbar ist, wenn sie c-rational ist, d.h. wenn sie sich aus den endlichen Sprachen durch Vereinigung, Produkt und Iteration erzeugen läßt, wobei die Iteration nur auf Sprachen angewandt werden darf, deren sämtliche Elemente zusammenhängend sind. Ein ähnliches Ergebnis wurde von Droste für sog. Nebenläufigkeitsmonoide gezeigt.

Unser Ziel ist es, allgemeine algebraische Bedingungen an ein Monoid M zu finden, so daß eine ähnliche Charakterisierung der erkennbaren Sprachen $L \subseteq M$ gezeigt werden kann.

Unsere Bedingungen sind die folgenden:

1. M ist von seinen irreduziblen Elementen endlich erzeugt und kürzbar.

2. Für je zwei Elemente von M existiert der größte gemeinsame Linksteiler.

3. Für jedes $m \in M$ ist die Menge $\{x \in M \mid x \text{ ist ein Linksteiler von } m\}$ bezüglich der Linksteilbarkeitsrelation ein distributiver Verband.

4. M „hat endlich viele Kommutationsverhalten".

Es ist leicht zu zeigen, daß diese Bedingungen in Spur- und in Nebenläufigkeitsmonoiden erfüllt sind. Darüber hinaus können aber auch Gleichungen der Form $ab = cd$ oder $ab = cc$ in M gelten, wobei a, b, c und d paarweise verschiedene irreduzible Monoidelemente sind.

Man erhält, daß alle endlichen Sprachen in M erkennbar sind, und daß die erkennbaren Sprachen unter Produkten abgeschlossen sind. Unter einer zusätzlichen Bedingung an M kann man zeigen, daß auch die Iteration einer erkennbaren „zusammenhängenden" und „unter Residuen abgeschlossenen" Sprache erkennbar ist. Also sind in diesem Falle die c-rationalen Sprachen alle erkennbar.

Für die umgekehrte Implikation benötigen wir eine Beschriftungsfunktion der irreduziblen Elemente von M. Dann kann man das Alphabet eines Monoidelements definieren als die Menge der Beschriftungen seiner irreduziblen

Teiler. Eine Sprache ist mc-rational, falls sie aus den endlichen Sprachen in M durch Produkt, Vereinigung und Iteration erzeugt werden kann, wobei die Iteration nur auf zusammenhängende und monoalphabetische Sprachen angewandt werden darf. Wir zeigen, daß in einem Teilbarkeitsmonoid mit einer solchen Beschriftungsfunktion jede erkennbare Sprache mc-rational ist.

Damit fallen die Klassen der erkennbaren, der c-rationalen und der mc-rationalen Sprachen zusammen.

Bemerkenswert ist, daß hiermit ein Zusammenhang zwischen ordnungstheoretischen Eigenschaften (distributive Verbände) und erkennbaren Sprachen hergestellt wird.

Eingeschränkter Nichtdeterminismus in Zellularautomaten

Martin Kutrib
Institut für Informatik, Universität Gießen

Zellulare Räume sind seit ihrer Einführung durch J. von Neumann (1966) [4] hinsichtlich verschiedener Aspekte untersucht worden. Standen in den frühen Jahren strukturelle Fragen (z.B. Standardisierungen, Fähigkeit zur Selbstreproduktion) im Vordergrund, so konzentrierte sich das Interesse in der Folgezeit mehr und mehr auf Probleme im Zusammenhang mit Berechnungen (z.B. Sprachverarbeitung, Mustertransformation, Simulation, Modellierung).

Wir betrachten die linear raumbeschränkte Variante, die sogenannten *zellularen Automaten*, mit zwei verschiedenen Verbindungsstrukturen. Im eindimensionalen Euklidischen Raum ist zum einen jede Zelle mit ihren beiden unmittelbaren Nachbarn verbunden, woraus sich ein bidirektionaler Informationsfluß ergibt (CA), zum anderen ist jede Zelle lediglich mit ihrem unmittelbaren rechten Nachbarn verbunden, woraus sich ein unidirektionaler Informationsfluß von rechts nach links ergibt (OCA).

Nichtdeterministische Zellularautomaten wurden erstmalig in [3] (NCA) bzw. in [2] (NOCA) hinsichtlich ihrer Fähigkeiten, Formale Sprachen zu erkennen, untersucht. Dabei wurde gezeigt, daß die Familien L (NCA), L (NOCA) und L_1 zusammenfallen, wobei für die Simulation eines NCA durch einen NOCA zusätzlich n (n bezeichnet die Länge der Eingabe) Zeitschritte notwendig waren. Wir werden zeigen, daß eine solche Simulation ohne Zeitverlust möglich ist.

Darüber hinaus werden Klassen betrachtet, bei denen sowohl die Zeit als auch der Nichtdeterminismus als einschränkbare Ressource aufgefaßt werden. Nichtdeterminismus läßt sich dabei selbst in Zeit und/oder Raum beschränken. Es wird gezeigt, daß die Sprachfamilien von Zellularautomaten, die nur in einem, dem ersten, Zeitschritt nichtdeterministische Eigenschaften haben [1], sowohl unter Realzeit- und Linearzeitbeschränkung als auch unter Beschränkung auf unidirektionalen Informationsfluß zusammenfallen. Diese Sprachfamilien charakterisieren zudem den ϵ-freien homomorphen Abschluß entsprechender deterministischer Modelle.

Wird der Nichtdeterminismus im Raum, d.h. auf einige Zellen beschränkt, läßt sich unter gewissen Voraussetzungen eine Austauschbarkeit mit der zeitlichen Beschränkung zeigen. So sind etwa Modelle mit einer nichtdeterministischen Zelle, die n nichtdeterministische Transformationen ausführen darf, gleichmächtig mit Modellen, in denen n Zellen jeweils genau eine nichtdeterministische Transformation ausführen.

Bei gleichzeitiger Beschränkung beider Parameter ergibt sich im sublogarithmischen Raum eine unendliche Hierarchie von Komplexitätsklassen.

Literatur

[1] Buchholz, Th., Klein, A., Kutrib, M. *One guess one-way cellular arrays.* Mathematical Foundations of Computer Science 1998, LNCS 1450, 807–815.

[2] Dyer, C. R. *One-way bounded cellular automata.* Information and Control 44 (1980), 261–281.

[3] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata.* Journal of Computer and System Sciences 6 (1972), 233–253.

[4] von Neumann, J. *Theory of Self-Reproducing Automata.* edited and completed by Arthur W. Burks. University of Illinois Press, 1966.

Dichte Vollständigkeit

Klaus-Jörn Lange
Universität Tübingen

Die Theorie der formalen Sprachen und die Strukturelle Komplexitätstheorie weisen enge Beziehungen zueinander auf. Obwohl beide Theorien vergleichbare Fragestellungen verfolgen wie beispielsweise den Vergleich von Determinismus und Nichtdeterminismus, weisen beide sehr unterschiedliche Erfolge in der Beantwortung dieser Fragestellungen auf. Während die Komplexitätstheorie zu großen Teilen eine Ansammlung offener Fragen ist und die Mehrzahl ihrer Ergebnisse offene Fragen in Verbindung setzt, ist das Phänomen offener Fragen in der Theorie formaler Sprachen nahezu unbekannt. Es stellt sich also die Frage nach der Enge ihrer wechselseitigen Beziehungen.

Nahezu alle formalen Sprachen, die in irgendeiner Weise kontextfrei sind, weisen Beziehungen zu den Klassen NP, $NAuxPDA_{pt}$, $NSPACE(\log n)$, oder NC^1 auf. Entsprechende Beziehungen gibt es zwischen den deterministischen Versionen dieser Familien und den Klassen P, $DAuxPDA_{pt}$, $DSPACE(\log n)$, und NC^1. Die Art dieser Beziehungen ist dabei derart, daß eine Familie formaler Sprachen $\mathcal{A}$ in einer Komplexitätsklasse $\mathcal{B}$ enthalten ist und daß $\mathcal{A}$ eine $\mathcal{B}$–vollständige Sprache enthält. Durch eine Beobachtung von Richard Beigel ist es nun möglich, systematisch zu einer formalen Sprachfamilie $\mathcal{A}$, die durch sequentielle einweg–Automaten definiert ist und für die die Klasse $\mathcal{B}$ unter logspace–Reduktionen abgeschlossen ist, einen nichtdeterministischen einweg–Automatentyp derart zu konstruieren, daß für die entsprechende Sprachfamilie $\mathcal{A}'$ gilt $\mathcal{A} \subseteq \mathcal{A}' \subset \mathcal{B}$ sowie

$$\forall_{B \in \mathcal{B}} \exists_{A \in \mathcal{A}'} : LOG(A) = LOG(B).$$

Die Familie $\mathcal{A}'$ liegt also *dicht* in der Klasse $\mathcal{B}$. Die Konstruktion und einige ihrer Eigenschaften werden im Vortrag vorgestellt. Es ist offen, ob eine vergleichbare Konstruktion auch für deterministische Familien oder für Klassen unterhalb von $DSPACE(\log n)$ gefunden werden kann.

Effizientes Spielbasiertes Model Checking für den Alternierungsfreien μ-Kalkül

Martin Leucker
RWTH Aachen

Es wird ein spielbasierter Model Checking Algorithmus für den Alternierungsfreien Mu-Kalkül vorgestellt. Spielbasierte Algorithmen eignen sich besonders für den Einsatz in Verifikationstools, da diese neben einer einfachen ja-/nein Antwort eine Strategie zum Beweis bzw. zur Widerlegung einer Formel bestimmen, die verwendet werden kann, um dem Benutzer die Gültigkeit einer Formel zu erklären. Dies ermöglicht den Einsatz von Model Checking Tools zum Debuggen verteilter Systeme.

Der Algorithmus ist linear sowohl in der Größe des Transitionssystems als auch in der Größe der zu untersuchenden Formel. Ferner ist er lokal, was eine bedarfsgesteuerte Generierung des Transitionssystems ermöglicht.

Realzeit Spracherkennung in defekten Zellularautomaten

Jan-Thomas Löwe
Institut für Informatik, Universität Gießen

Ein defekter Zellularautomat ist ein Zellularautomat, bei dem einige Zellen nicht in der Lage sind, Berechnungen durchzuführen. Stattdessen können sie nur Informationen von ihrem linken bzw. rechten Nachbarn weiterleiten. Dabei können wir zunächst zwischen zwei verschiedenen Arten von Defekttypen, den statischen (von Anfang an) SD-CA und den dynamischen Defekten (während der Laufzeit) DD-CA unterscheiden.

In Bezug auf die Sprachverarbeitung interessieren wir uns nun besonders für solche Algorithmen, die auch bei vorhandenen bzw. auftretenden defekten Zellen weiterhin funktionieren. Die Frage ist daher, welche Sprachfamilien in Realzeit und unabhängig von den defekten Zellen akzeptiert werden können.

Frühere Untersuchungen von fehlertoleranten Zellularautomaten und zellularen Räumen wurden bereits in [1, 4] durchgeführt. Das Problem der Synchronisation (FSSP) in defekten Zellularautomaten wurde in [2, 3, 6, 7] betrachtet.

Bei der Sprachverarbeitung ist die Art der Eingabeinitialisierung ein wesentlicher Unterschied der beiden Defekttypen. Im dynamischen Fall gehen wir davon aus, daß zu Beginn alle Zellen funktionsfähig sind. Jede Zelle erhält ein Eingabezeichen, wohingegen im statischen Fall die defekten Zellen von Anfang an feststehen, so daß nur funktionsfähige Zellen ein Eingabezeichen enthalten. Die Gesamtzahl der Zellen ist somit größer als die Länge der Eingabe.

Satz 1. Gibt es keine Einschränkungen der Anzahl defekter Zellen, so gilt

1. $\mathrm{L}_{rt}(\text{DD-CA}) = \mathrm{L}_3$
2. $\mathrm{L}_{rt}(\text{SD-CA}) = \mathrm{L}_{rt}(\text{OCA})$

D.h. wir verlieren im Extremfall die Bidirektionalität. Diese Beobachtung liefert die Idee zu einer neuen, davon abgeleiteten Form von defekten Automaten, nämlich Zellularautomaten mit Defekten, derart daß defekte Zellen OCA-Zellen sind (nO-CA, n Anzahl der OCA-Zellen). Ein Erkennungs-Algorithmus soll nun unabhängig von der Position solcher spezieller Zellen eine Sprache akzeptieren.

Satz 2. $\mathrm{L}_{rt}(1\text{O-CA}) \subsetneq \mathrm{L}_{rt}(\text{CA})$

Bereits bei einer OCA-Zelle gibt es also Sprachen, die nicht mehr erkannt werden können. Z.B. eine Sprache aus [5]

$$L = \{uvu : u, v \in \{0, 1\}^*, |u| > 1\}$$

Offen ist bisher noch die Frage

$$\mathrm{L}_{rt}(\mathrm{OCA}) \subsetneq \mathrm{L}_{rt}(\mathrm{1O\text{-}CA})?$$

Literatur

[1] Masateru Harao, Shoichi Noguchi. *Fault tolerant cellular automata.* Journal of computer and system sciences, 11:171-185, 1975

[2] Martin Kutrib, Roland Vollmar. *The firing squad synchronization problem in defective cellular automata.* IEICE Transations on Information and Systems, E78-D:895-900, Juli 1995

[3] Jacques Mazoyer. *Synchronization of a line of finite automata with non-uniform delays.* Research Report N$^{\mathrm{O}}$ 94-49, Ecole Normale Supérieure de Lyon. Dezember 1994

[4] Hidenosuke Nishio, Youichi Kobuchi. *Fault tolerant cellular spaces.* Journal of computer and system sciences, 11:150-170, 1975

[5] Véronique Terrier. *Language not recognizable in real time by one-way cellular automata.* Theoretical Computer Science, 156:281-287, 1995

[6] Hiroshi Umeo. *A fault-tolerant scheme for optimum-time firing squad synchronization.* Osaka Electro-Communication Univ., 1993

[7] Jean-Baptiste Yunès. *Fault tolerant solutions to the firing squad synchronization problem.* Université Paris 7, LITP-IBP, 2, Januar 1996.

Modaler mu-Kalkül und sequentielle Komposition

Markus Müller-Olm
Universität Dortmund

Der modale mu-Kalkül ist eine ausdrucksstarke Logik, die auch in der Praxis eingesetzt wird, um wünschenswerte Eigenschaften von finite-state Prozessen zu spezifizieren und durch model-checking automatisch zu überprüfen. In diesem Vortrag betrachten wir eine Erweiterung des modalen mu-Kalküls um einen sequentiellen Kompositionsoperator. Logiken mit sequentiellen Kompositionsoperatoren kennt man zwar aus dem Bereich der temporalen Intervall-Logiken, nicht aber aus dem Bereich punktbasierter branching-time temporaler Logiken, zu denen der modale Mu-Kalkül gehört. Die Erweiterung der Logik wird möglich, indem Formeln nicht wie üblich durch Prädikate sondern durch Prädikattransformer interpretiert werden.

Die entstehende Logik ist echt ausdruckskräftiger als der übliche modale Mu-Kalkül, bleibt aber für finite-state Prozesse entscheidbar. Insbesondere existieren in der erweiterten Logik, anders als im modalen mu-Kalkül, Formeln, die kontextfreie Prozesse (BPA-Prozesse) modulo Bisimulation und modulo Simulation charakterisieren. Diese Beobachtung hat eine Reihe interessanter Konsequenzen bezüglich der (Un-)Entscheidbarkeits-Eigenschaften der Logik und ihrer Ausdruckskraft.

Lokalisierung von Baum-Mustern in Wäldern

Andreas Neumann
Universität Trier

Moderne Textverarbeitungs-Systeme (z.B. SGML) repräsentieren Dokumente als Bäume, Folgen von Dokumenten als Wälder. Eine der wichtigsten und häufigsten Aufgaben in der Dokumentenverarbeitung ist die Lokalisierung von Teilbäumen, die ein bestimmtes Muster erfüllen. Ein Muster besteht aus einer strukturellen Bedingung auf den gesuchten Teilbäumen selbst sowie einer kontextuellen Bedingung bezüglich ihrer Position im Dokument. Beide Arten von Bedingungen sind mithilfe von Constraint-Systemen als reguläre Baum/Waldsprachen gegeben.

Für die Lokalisierung der Muster definieren wir Keller-Wald-Automaten (PFAs). Ein PFA durchläuft den Eingabe-Wald in einem Tiefendurchlauf von links nach rechts und besucht dabei jeden Knoten zweimal: einmal vor und einmal nach dem Durchlaufen seiner Nachfolger. Wir weisen nach, daß PFAs genau die Klasse der regulären Waldsprachen akzeptieren können und dabei ggf. mit exponentiell weniger Zuständen auskommen als konventionelle (bottom-up) Automaten. Wir zeigen dann für eine Unterklasse von kontextuellen Bedingungen, daß mithilfe von PFAs in einem einzigen Durchlauf alle Teilbäume gefunden werden können, die ein Muster erfüllen.

Eine ausführliche Version dieser Arbeit wird bei FST&TCS'98 erscheinen. Als technischer Report Nr. 98-08 der Universität Trier ist sie verfügbar unter `http://www.informatik.uni-trier.de/~neumann/Papers`.

Rewriting Logic als Grundlage für einen Prozeßalgebra-Compiler

Thomas Noll
RWTH Aachen

In vielen Modellen für Parallelität wie etwa CCS und Petri-Netzen werden verteilte Systeme durch Zustände und Übergänge zwischen diesen beschrieben. Sie weichen allerdings in der Behandlung der dynamischen Struktur voneinander ab, wie die Unterscheidung zwischen "interleaving" und "true concurrency" zeigt. Meseguer macht sich diese Beobachtungen zunutze, um mit der Rewriting Logic einen einheitlichen formalen Rahmen zur Beschreibung derartiger Modelle zu entwickeln. Bei diesem Ansatz werden die Zustände eines verteilten Systems durch Terme (genauer: durch gleichungsdefinierte Äquivalenzklassen von Termen) und sein operationelles Verhalten durch Termersetzungsregeln beschrieben.

In diesem Vortrag soll gezeigt werden, daß die Rewriting Logic als formale Grundlage für einen Compiler geeignet ist, der aus einer geeigneten Spezifikation einer Prozeßalgebra ein Programm generiert, welches automatisch das Transitionssystem zu einer Prozeßbeschreibung erzeugt und damit als Frontend für ein Verifikationstool verwendet werden kann.

Automatic monoids

Friedrich Otto
Universität Kassel

In recent years the computational aspect has become more and more prominent in combinatorial group and semigroup theory. Although in general not much information on the algebraic structure presented can be extracted from a finite presentation, various methods have been used successfully in certain instances.

One approach is based on the notion of *convergent presentations*. A monoid-presentation $(\Sigma; R)$ is called *convergent* if the reduction relation induced by R is both noetherian and confluent.

As an alternative to the rewrite approach Epstein et al developed the notion of *groups with automatic structure* during the 1980's (Epstein 92). A finitely generated group G has an *automatic structure* if it has a finite set of generators Σ and a regular set $C \subset \Sigma^*$ of representatives. which, however, need not be unique, such that the following tasks can be performed by finite state acceptors:

(1.) Given two elements $u, v \in C$, decide whether or not they both represent the same element of the group G.

(2.) For $a \in \Sigma$, if two elements $u, v \in C$ are given, decide whether or not ua and v represent the same element of the group G.

It turns out that the class of *automatic groups* has many nice properties. For example, they are finitely presented, their word problems are solvable in quadratic time, and their Dehn functions are bounded from above by polynomials of degree two. However, it is still not known whether each automatic group admits a finite convergent presentation.

Recently, the notion of automatic structure has been generalized to monoids (Hudson 96). In this talk the class of *automatic monoids* is discussed. and it is shown that only few of the nice results on automatic groups carry over.

In particular, we will see that the class of automatic monoids and the class of monoids that have a finite convergent presentation are incomparable under inclusion. For the class of monoids this provides a negative anwer to the open problem mentioned above. Further, it is shown that there exist finitely presented automatic monoids the Dehn functions of which are not bounded from above by any primitive-recursive function.

Beschreibungskomplexität endlicher Mengen

Holger Petersen
Universität Stuttgart

Wir untersuchen die Größe regulärer Ausdrücke und verschiedener Varianten endlicher Automaten, die eine gegebene endliche Wortmenge akzeptieren, im Sinne der klassischen Arbeit von Meyer und Fischer [MF71].

Im Fall eines einelementigen Eingabealphabetes schränken wir uns sogar auf einelementige Wortmengen der Form $\{0^n\}$ ein. Tatsächlich ist bei einelementigem Eingabealphabet für deterministische und nichtdeterministische endliche Automaten ohne Erweiterungen das Akzeptieren eines einzelnen Wortes bereits der schwierigste Fall, denn sie benötigen hierfür $n+2$ bzw. $n+1$ Zustände. Kürzere Wörter können durch Einführung neuer Endzustände in die Sprache aufgenommen werden.

Dagegen können einelementige Mengen über einem einelementigen Eingabealphabet durch erweiterte reguläre Ausdrücke, die Komplementierung als Operation erlauben, und durch Pebble-Automaten der Größe $O((\log n)^2/\log\log n)$ beschrieben werden, womit gegenteilige Vermutungen aus [MF71] widerlegt werden. Die asymptotisch optimale Schranke $O(\log n)$ läßt sich mit regulären kanonischen Systemen erreichen [Büc64]. Da eine Kompression der Eingabe die Komplexität von Entscheidungsproblemen erhöht, läßt sich die Technik zur Erzeugung von Wörtern durch kleine reguläre Ausdrücke in Beweisen für untere Schranken einsetzen.

Literatur

[Büc64] J. Richard Büchi. Regular canonical systems. *Archiv Math. Logik und Grundlagenforschung*, 6:91–111, 1964.

[MF71] Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *Conference Record 12th Annual Symposium on Switching and Automata Theory*, pages 188–191, 1971.

Eine kontextfreie Ableitungsbaumhöhenhierarchie

Klaus Reinhardt
Eberhard-Karls-Universität Tübingen

Betrachtet man Ableitungen von Wörtern aus dem Startsymbol mittels kontextfreier Regeln, bei denen Variablen parallel ersetzt werden * m ü s s e n * (L-Systeme...), so können nicht-kontextfreie Sprachen generiert werden. Wenn die Variablen parallel ersetzt werden * d ü r f e n *, so ändert sich dahingegen nichts an der generierten Sprache bzw. Sprachklasse. Wir wollen untersuchen, welche kontextfreien Sprachen sich durch Parallelisierung beschleunigt ableiten lassen und definieren für eine Funktion f die Klasse CFLth($f(n)$) als die Menge der kontextfreien Sprachen, für die es eine kontextfreie Grammatik gibt, mit der jedes Wort der Sprache in $f(n)$ parallelen Schritten, d.h. mit $f(n)$ beschränkter Baumhöhe abgeleitet werden kann. Chulik und Maurer zeigten, dass die regulären Sprachen in CFLth($\log n$) liegen. Wir wiederlegen deren umgekehrt lautende Vermutung und beschreiben Sprachen in einer Hierarchie zwischen CFLth($\log n$) und CFLth(lin)=CFL.

Definition CFLth$(f(n) := \{L \subseteq \Sigma^* | \exists G, L = L(G), \forall x \in L$ gilt x kann in $\leq f(|x|)$ parallelen Schritten abgeleitet werden $\}$.

Entsprechend dem CYK-Algorithmus gilt:

Korollar Sprachen in CFLth($f(n)$) können mit n^2 Prozessoren in $f(n)$ Schritten erkannt werden.

Proposition [CM78] REG$\subseteq$CFLth($\log n$).

Die Frage ob es Sprachen in CFLth$(o(n))\setminus$REG gibt, wird durch folgendes Beispiel [Boa97] beantwortet: Sei L die Menge der Nicht-Präfixe des unendlichen Wortes $w = baba^2ba^3b.....a^nb...$, so ist $L \in$CFLth($\sqrt{n}$).

Theorem $\exists L \in$ (CFL$\setminus$REG)$\cap$CFLth($\log n$)

Hierbei verwenden wir einen Binärzähler

$$w = b0a1b10a11b100a101b110a111b...a111100001b100010000a....$$

Allgemein stellt sich die Frage, welche Funktionen als Grösse eines Zählers vorkommen können, was zu folgender Definition führt:

Definition Eine Funktion $f : \mathcal{N} \to \mathcal{N}$ nennen wir *contextfree complement constructable* (ccc), wenn es eine unendliche Folge $c_0, c_1, ...$ von Wörtern über dem Alphabet Σ mit $|c_k| = f(k)$ gibt, so dass das Komplement von $\{c_k\$c_{k+1}^R | n \in \mathcal{N}\}$ kontextfrei ist.

Theorem Ist f ccc, so ist für ein g mit $g(\sum_{i=1}^{n} f(i)) = f(n)$ ($g \geq \log$) die Sprache der Wörter, die mit a enden, aber kein Präfix von $bc_0ac_1^Rbc_2ac_3^R...ac_{n-1}^Rbc_na...$ sind, in CFLth$(g(n))$.

Theorem 1 und log sind ccc. Wenn f und g ccc sind, so auch $f+g$ und $f*g$ und Funktionen e, d mit $e(n) = \sum_{i=1}^{n} f(i)$ und $d^{-1}(n) = \sum_{i=1}^{n} f^{-1}(i)$.

Beweisidee: $c_{1,i} := 1$, $c_{\log,i} := bin(i)$,
Komme b in $c_{f,i}$ und $c_{g,i}$ nicht vor;
setze $c_{f+g,i} := c_{f,i}bc_{g,i}$, $c_{f*g,i} := c_{f,i}bb^{(|c_{f,i}|-1)*(|c_{g,i}|-1)}c_{g,i}$,

$$c_{e,i} := b^{\sum_{j=1}^{i-1} |c_{f,j}|} c_{f,i}.$$

Falls $f(i) > d(i-1)$
dann $c_{d,i} := b^{|c_{d,i-1}|+1-|c_{f,0}|}c_{f,0}$
sonst $c_{d,i} := b^{|c_{d,i-1}|-|c_{f,j+1}|}c_{f,j+1}$ für $c_{d,i-1} := c^kc_{f,j}$.

Theorem CFLth$(f(n))$ ist eine AFL.

Definition [Gol72] Eine Sprache L ist *bounded* wenn $L \subseteq w_1^*w_2^*...w_m^*$ für Wörter $w_1, ..., w_m \in \Sigma^*$, $m \geq 1$.

Theorem CFLth$(o(n))$ enthält keine nicht-regulären Sprachen, die bounded sind.

Literatur

[Boa97] Luc Boasson. personal communication. 1997.

[Bra81] F. J. Brandenburg. On the height of syntactical graphs. In Peter Deussen, editor, *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, volume 104 of *LNCS*, pages 13–21, Karlsruhe, FRG, March 1981. Springer.

[CM78] K. Culik II and H.A. Maurer. On the height of derivation trees. *Forschungsbericht Nr. 18, Inst. für Informationsverarbeitung TU Graz*, 1978.

[Gol72] Jonathan Goldstine. Substitution and bounded languages. *Journal of Computer and System Sciences*, 6(1):9–29, February 1972.

[Gol76] J. Goldstine. Bounded AFLs. *Journal of Computer and System Sciences*, 12(3):399–419, June 1976.

Mehrfach-limitierte T0L-Sprachen

Markus Seemann
Technische Universität Braunschweig

Lindenmayer-Systeme sind von A. Lindenmayer in [3, 4] eingeführt worden als ein Modell zur Beschreibung von unbeschränktem und parallelem Zellwachstum. Eine Zelle wird hierbei als Zeichen und ein Organismus als Zeichenkette dargestellt. Mehrfach-limitierte Lindenmayer-Systeme [1, 2, 6] sind eine Verallgemeinerung der k-limitierten Lindenmayer-Systeme, die von D. Wätjen [7] eingeführt worden sind. Im Gegensatz zum k-limitierten Fall steht hierbei jedem Zelltyp ein *individueller* limitierter Nahrungsvorrat zur Verfügung, der die Anzahl sich verändernder Zellen dieses Typs in jedem Wachstumsschritt bestimmt.

Im folgenden bezeichne $\wp(S)$ die Potenzmenge von S und $\mathbb{N}$ die Menge aller positiven natürlichen Zahlen. Des weiteren sei $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Es werden einige Grundlagen der formalen Sprachen vorausgesetzt [5].

Es sei Σ ein Alphabet. Eine Abbildung $h_k : \Sigma^* \to \wp(\Sigma^*)$ heißt *k-mehrfach-limitiert*, wenn $h : \Sigma \to \wp(\Sigma^*)$ eine endliche Substitution ist, $k : \Sigma \to \mathbb{N}_0$ eine Abbildung ist und die Menge $h_k(w)$ für jedes Wort $w \in \Sigma^*$ gleich der Menge der Wörter ist, die aus w durch Ersetzen aller, höchstens jedoch $k(a)$ Vorkommen eines jeden Symbols $a \in \Sigma$ hervorgehen.

Ein Quintupel $G = (\Sigma, H, \omega, k)$ heißt *k-mehrfach-limitiertes* T0L-*System* oder kurz $k$mlT0L-*System*, falls $k : \Sigma \to \mathbb{N}_0$ eine Abbildung ist, die sogenannte *Limitierungsabbildung*, und (Σ, H, ω) ein T0L-System ist, d.h., Σ ist ein Alphabet, $\omega \in \Sigma^*$ das *Axiom* und H eine endliche Menge von *Tafeln*, d.h. eine endliche Menge von endlichen Substitutionen über Σ. Besteht H aus lediglich einer Tafel, so heißt G ein $k$ml0L-System. Es heißt propagierend bzw. deterministisch, wenn alle Tafeln ε-frei sind bzw. ausschließlich auf einelementige Mengen abbilden. G wird dann auch als $k$mlP(D)(T)0L-System bzw. $k$ml(P)D(T)0L-System bezeichnet. Die von G erzeugte Sprache lautet

$$L(G) = \{w \mid w = \omega \text{ oder } w \in h_k^1 \circ \cdots \circ h_k^n(\omega) \text{ und } h^1, \ldots, h^n \in H, n \in \mathbb{N}\}$$

mit $h_k(W) = \bigcup_{w \in W} h_k(w)$ für alle $W \subseteq \Sigma^*$. Jede von einem $k$mlT0L-System erzeugte Sprache heißt $k$mlT0L-Sprache. Sie werden auch KmlT0L-System bzw. KmlT0L-Sprache genannt, falls $\mathrm{Im}(k) \subseteq K$. Für festes $K \subseteq \mathbb{N}_0$ wird die Familie aller KmlT0L-Sprachen mit $\mathcal{L}(K\text{mlT0L})$ bezeichnet. Für Familien, die von propagierenden, deterministischen oder Systemen mit lediglich einer Tafel erzeugt werden, werden analoge Bezeichnungen verwendet.

Der folgende Satz zeigt, daß die Menge der zur Verfügung stehenden Limitierungen einen zentralen Parameter der erzeugten Sprachfamilien darstellt.

Satz Es gilt $\mathcal{L}(N\mathrm{ml(P)(D)(T)0L}) \subseteq \mathcal{L}(M\mathrm{ml(P)(D)(T)0L}) \Longrightarrow N \subseteq M$ für alle nichtleeren Mengen $M, N \subseteq \mathbb{N}$.

Es ist offen, ob die Umkehrung des Satzes allgemein richtig ist. Sie gilt zumindest in den folgenden Fällen.

Korollar Für alle nichtleeren Mengen $M, N \subseteq \mathbb{N}$ gilt

$$
\begin{array}{l}
\mathcal{L}(N\mathrm{ml(P)(D)\ \ 0L}) \\
\mathcal{L}(N\mathrm{ml\ P\ (D)\ \ 0L}) \\
\mathcal{L}(N\mathrm{ml(P)\ D\ \ \ 0L}) \\
\mathcal{L}(N\mathrm{ml\ P\ \ D\ \ \ 0L}) \\
\mathcal{L}(N\mathrm{ml(P)(D)T0L}) \\
\mathcal{L}(N\mathrm{ml\ P\ (D)T0L}) \\
\mathcal{L}(N\mathrm{ml(P)\ D\ T0L}) \\
\mathcal{L}(N\mathrm{ml\ P\ \ D\ T0L})
\end{array}
\left\{ \begin{array}{c} \subseteq \\ = \\ \subsetneq \end{array} \right\}
\begin{array}{l}
\mathcal{L}(M\mathrm{ml\ \ \ \ \ (T)0L}) \\
\mathcal{L}(M\mathrm{mlP\ \ (T)0L}) \\
\mathcal{L}(M\mathrm{ml\ \ D(T)0L}) \\
\mathcal{L}(M\mathrm{mlPD(T)0L}) \\
\mathcal{L}(M\mathrm{ml\ \ \ \ \ \ T\ 0L}) \\
\mathcal{L}(M\mathrm{mlP\ \ \ T\ 0L}) \\
\mathcal{L}(M\mathrm{ml\ \ D\ T\ 0L}) \\
\mathcal{L}(M\mathrm{mlPD\ T\ 0L})
\end{array}
\Longleftrightarrow N \left\{ \begin{array}{c} \subseteq \\ = \\ \subsetneq \end{array} \right\} M
$$

In diesen Fällen induzieren die Mengen der erlaubten Limitierungen also eine echte Hierarchie der Sprachfamilien. Allgemein impliziert der Satz das folgende Korollar.

Korollar Für alle nichtleeren Mengen $M, N \subseteq \mathbb{N}$ ist jede der acht Familien $\mathcal{L}(N\mathrm{ml(P)(D)(T)0L})$ mit jeder der Familien $\mathcal{L}(M\mathrm{ml(P)(D)(T)0L})$ unvergleichbar, falls N und M unvergleichbar sind.

Literatur

[1] S. Gärtner: *Partitions-limitierte Lindenmayer-Systeme.* Dissertation, TU Braunschweig. Berichte aus der Informatik. Shaker-Verlag, Aachen 1995.

[2] S. Gärtner: *On Partition Limited 0L Systems.* Developments in Language Theory II. World Scientific, Singapore 1996, 230–236.

[3] A. Lindenmayer: *Mathematical Models for Cellular Interactions in Development I. Filaments with One-sided Inputs.* J. Theoretical Biology **18**(1968), 280–299.

[4] A. Lindenmayer: *Developmental Systems without Cellular Interactions, their Languages and Grammars.* J. Theoretical Biology **30**(1971), 455–484.

[5] A. Salomaa: *Formal Languages.* Academic Press, New York 1973.

[6] M. Seemann: *Multiple-limited ET0L Systems*. Developments in Language Theory III, Thessaloniki, 20.–23. Juli 1997, 377–385.

[7] D. Wätjen: *k-limited 0L Systems and Languages*. J. Inform. Process. Cybern. EIK **24**(1988), 267–285.

Characterization of the Class of Attributed Tree Transformations

Zoltan Fülöp and Heiko Vogler
TU Dresden

It is known that absolutely noncircular attributed tree transducers induce the same class of tree transformations as well-presented macro tree transducers. A characterization for the class of noncircular attributed tree transformations was not yet known. We present such a characterization in terms of attributed-like macro tree transducers.

Distributed Implementation of Asynchronous Transition Systems

Walter Vogler
Institut für Informatik, Universität Augsburg

One methodology for the design of asynchronous circuits takes a transition system (whose arcs are labelled with what we call events) as specification of the desired behaviour, gives it a distributed implementation as a safe Petri net and transforms the latter stepwise into a circuit, see e.g. [CKLY95]; [Yak98] gives in detail a practical example for such a development. In the synthesis of the net, it is desirable that each event of the transition system corresponds to a unique transition of the net: as [Yak98] shows, the transformation of the net may involve event refinement, which is much easier when each event is represented by one transition; also some existing procedures for efficient direct compilation of a net into an asynchronous circuit rely on this assumption. [1] It is natural in this situation that attention is restricted to deterministic transition systems.

An important contribution to the synthesis problem is the theory of regions, see e.g. [NRT92]; it allows a characterization of those transition systems TS for which an elementary Petri net exists whose reachability graph is isomorphic to TS. Elementary nets are (almost) the same as safe nets without loops; a nice feature is that for such nets independence of transitions corresponds exactly to 'diamonds' in the transition system. But [PKY95] points out that loops are very natural in the context of circuits and allows to implement additional transition systems; the theory of regions is extended accordingly (almost) to general safe nets. On the other hand, [CKLY95] weakens the requirements of [NRT92] and shows that for each transition system TS satisfying the weaker requirements there exists an elementary net whose reachability graph is bisimilar to TS. Finally, [Yak98] is confronted with a transition system that cannot be realized by a net in any of these approaches; since the reaction to such a situation can hardly be simply to give up, he first inserts an internal event into the transition system 'in a harmless way', i.e. he achieves in his example an implementation with a net that has an additional internal transition. This implementation strategy is the topic of the present contribution.

Thus, we are given a deterministic transition system TS as a specification of some behaviour and we want to find a (general) safe net whose transitions are the events of TS and possibly some additional internal events, and whose

[1]Thanks go to Alex Yakovlev for pointing this out to me.

reachability graph is (weakly) bisimilar to TS. This ensures (but see below) that the net has essentially the desired behaviour. Internal events could lead to a new and usually unwanted behaviour that is ignored by bisimulation, namely divergence, i.e. infinite internal computation; hence, we additionally require the net to be divergence-free. We will show that it is not too difficult to find such an implementation for each transition system, but it turns out that it is completely sequential. This is undesirable e.g. for performance reasons, hence an additional requirement is to preserve concurrency – for which concurrency must be specified in the first place. We will therefore in fact start from an asynchronous transition system (ATS) which is a deterministic transition system with an additional independence relation on the events.

Preservation of concurrency could mean that the net should have the same step sequences or the same partial order semantics as the ATS, where the latter could be defined via Petri net processes or equivalently as (Mazurkiewicz) traces, see e.g. [NW95] – also for ATS. Or one could combine this with bisimilarity and require step or history-preserving bisimilarity; for our first result, we will consider something in between: ST-bisimilarity, which combines bisimulation with a partial order semantics based on so-called interval orders, see e.g. [Vog92]. We will show that each ATS (with a weak requirement for independence) can be implemented by an ST-bisimilar divergence-free safe net.

Work in progress: we will consider ATS with the usual strong requirement for independence; we intend to show how to decide whether for such an ATS there exists a safe net without internal events with a bisimilar reachability graph, and that one such net is in fact history-preserving bisimilar to the ATS. Finally, we will look at a construction that sometimes helps to find a history-preserving bisimilar divergence-free safe net with internal events.

Another beautiful contribution to the synthesis problem is [Sun98, Chapter 5]; it considers the case that the specification of the desired behaviour is given as a temporal logic formula and presents an effective decision procedure whether a safe net with possibly some additional internal events exists that meets the specification.

Literatur

[CKLY95] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. In *Int. IEEE/ACM Conf. on CAD, ICCAD 95*, pages 164–171, 1995. (full version).

[NRT92] M. Nielsen, G. Rozenberg, and P.S. Thiagarajan. Elementary transition systems. *Theor. Comput. Sci.*, 96:3–33, 1992.

[NW95] M. Nielsen and G. Winskel. Models for concurrency. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science Vol. 4*, pages 1–148. Oxford Univ. Press, 1995.

[PKY95] M. Pietkiwicz-Koutny and A. Yakovlev. Non-pure nets and their transition systems. Technical Report TR 528, Dept. Comp. Sci., Univ. of Newcastle upon Tyne, 1995.

[Sun98] K. Sunesen. *Reasoning about Reactive Systems*. PhD thesis, Univ. Aarhus, 1998.

[Vog92] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets*. Lect. Notes Comp. Sci. 625. Springer, 1992.

[Yak98] A. Yakovlev. Designing control logic for counterflow pipeline processor using Petri nets. *Formal Methods in System Design*, 12:39–71, 1998.

Teams von limitierten und uniform-limitierten 0L-Systemen

Dietmar Wätjen
Technische Universität Braunschweig

In [1] sind Teams von kooperierenden/distributiven Grammatiksystemen eingeführt worden. Die Erzeugungsmächtigkeit der Systeme wird dabei vergrößert, indem mehrere Komponentengrammatiken zu einem Team zusammengefaßt werden, die dann einen gemeinsamen Ableitungsschritt auf derselben sententiellen Form w ausführen. Aus jeder Komponente des Teams wird eine Produktion ausgewählt, und diese Produktionen werden parallel auf w angewendet. Auf diese Weise wird der sequentielle Ableitungsmechanismus von Grammatiken erweitert. Wir wissen jedoch, daß bei Lindenmayersystemen die Ableitungen vollständig parallel erfolgen. Daher können wir keine Teams von verschiedenen Lindenmayersystemen betrachten. Im Falle von limitierten und uniform-limitierten 0L-Systemen können dagegen aufgrund der Limitierung verschiedene Systeme gleichzeitig auf demselben Wort arbeiten. Daher kann das Teamkonzept auf kooperierende/distributive limitierte oder uniform-limitierte 0L-Systeme (CDl0L- oder CDul0L-Systeme, siehe [2], [3]) angewendet werden.

Es seien $k_1, \ldots, k_r \in \mathbb{N}$. Ein CD$(k_1, \ldots, k_r)$-*(uniform-)limitiertes* 0L-*System* (kurz auch CD$(k_1, \ldots, k_r)$(u)l0L-*System*) *mit Teams*

$$G = (\Sigma, (h_1, k_1), \ldots, (h_r, k_r), \omega, R_1, \ldots, R_n), \; r, n \in \mathbb{N},$$

besteht aus einem CD$(k_1, \ldots, k_r)$(u)l0L-System $(\Sigma, (h_1, k_1), \ldots, (h_r, k_r), \omega)$, und die Teams $R_1, \ldots, R_n$ sind nichtleere Teilmengen von $\{(h_1, k_1, 1), \ldots, (h_r, k_r, r)\}$. Die dritte Komponente in den Elementen dieser Menge dient nur dazu, daß zwei gleiche Komponenten des gegebenen CD$(k_1, \ldots, k_r)$(u)l0L-Systems beide in einem Team auftreten können. Zur Vereinfachung der Notation lassen wir diese dritte Komponente im folgenden wieder weg. Ein *direkter Ableitungsschritt* $w_1 \Longrightarrow_{R_i} w_2$ gemäß einem Team $R_i = \{(h_{j_1}, k_{j_1}), \ldots, (h_{j_s}, k_{j_s})\}$ wird im Fall von limitierten 0L-Systemen dadurch gegeben, daß genau $\min\{k_{j_1} + \ldots + k_{j_s}, \#_a w_1\}$ Vorkommen eines jeden Symbols a in w_1 (im Falle von uniform-limitierten 0L-Systemen genau $\min\{k_{j_1} + \ldots + k_{j_s}, |w|\}$ Vorkommen von Symbolen in w_1) mit Hilfe einiger Tafeln h_{j_σ}, $\sigma = 1, \ldots, s$, von R_i ersetzt werden, wobei jedoch höchstens k_{j_σ} dieser Ersetzungen gemäß einem festen h_{j_σ} durchgeführt werden dürfen.

In CD-Grammatiksystemen mit Teams wird in einem Ableitungsschritt jedes Mitglied des Teams genau einmal angewendet. Im Gegensatz dazu ist es

(a) $\mathcal{L}(\mathrm{CD(u)lOL}) \subsetneq \mathcal{L}(\mathrm{TCD(u)lOL}) \subsetneq \mathcal{L}(\mathrm{HTCD(u)lOL})$,
(b) $\mathcal{L}(\mathrm{HTCD}(k_1, \ldots, k_r)\mathrm{(u)lOL}) \subsetneq \mathcal{L}(\mathrm{HTCD}(k_1, \ldots, k_{r+1})\mathrm{(u)lOL})$,
(c) $\mathcal{L}(\mathrm{HT}_{\leq c}\mathrm{CD(u)lOL}) \subsetneq \mathcal{L}(\mathrm{HT}_{\leq c+1}\mathrm{CD(u)lOL})$ und für $r > c$
$\mathcal{L}(\mathrm{HT}_{\leq c}\mathrm{CD}(k_1, \ldots, k_r)\mathrm{(u)lOL}) \subsetneq \mathcal{L}(\mathrm{HT}_{\leq c+1}\mathrm{CD}(k_1, \ldots, k_r)\mathrm{(u)lOL})$,
(d) $\mathcal{L}(\mathrm{H}n\mathrm{TCD(u)lOL}) \subsetneq \mathcal{L}(\mathrm{H}(n+1)\mathrm{TCD(u)lOL})$.

Die Aussagen (b), (c) und (d) sind auch für die entsprechenden Teilfamilien von $\mathcal{L}(\mathrm{TCD(u)lOL})$ erfüllt, (c) und (d) auch für die von $\mathcal{L}_m(\mathrm{TCD(u)lOL})$ für alle Modi $m \in \{t, \geq t \mid t \in \mathbb{N}\}$, und mit wenigen Ausnahmen gilt dies auch für (b). (a) gilt mit einigen Einschränkungen auch für die Teilfamilien von $\mathcal{L}(\mathrm{HTCD}(k_1, \ldots, k_r)\mathrm{(u)lOL})$.

Wir sehen also, daß die Einführung von Teams und weiter von hybriden Teams zu umfassenderen Sprachfamilien führt. Durch Hinzunahme weiterer Komponenten erhalten wir durch (b) eine echte Hierarchie von Sprachfamilien. Eine analoge Aussage folgt aus (c) bezüglich der jeweils maximalen Größe der Teams und in (d) bezüglich der Anzahl der Teams.

In einem gewissen Gegensatz zur Aussage (c) sind die Familien von Sprachen mit Teams genau der Größe c oder c' für $c \neq c'$ bezüglich Inklusion unvergleichbar. Weiter sind die (nicht-uniformen) limitierten Sprachfamilien unvergleichbar mit den uniform-limitierten Familien.

Literatur

[1] L. Kari, A. Mateescu, Gh. Păun and A. Salomaa, Teams in cooperating grammar systems, *J. Exper. Th. AI* **7**(1995), 347–359.

[2] D. Wätjen, On Cooperating/Distributed Limited 0L Systems, *J. Inform. Process. Cybern. EIK* **29**(1993), 129–142.

[3] D. Wätjen, On Cooperating/Distributed Uniformly Limited 0L Systems. In: Gh. Păun and A. Salomaa, eds., *New Trends in Formal Languages. Control, Cooperation, and Combinatorics*, Lecture Notes in Computer Science 1218 (Springer, Berlin 1997), 178–196.

Kriterien für exponentielle Mehrdeutigkeit Kontextfreier Grammatiken

Klaus Wich
Universität Kassel

Es ist bekannt, daß die Mehrdeutigkeit Kontextfreier Grammatiken ($CFGs$) im allgemeinen unentscheidbar ist. Um sich trotzdem dem Wesen der Mehrdeutigkeit zu nähern, kann man untersuchen, wie sich die Mehrdeutigkeit der Wörter einer CFG in Abhängigkeit von ihrer Wortlänge entwickelt.

Im folgenden werden nur zykelfreie $CFGs$ ohne nutzlose Symbole und ohne Nichtterminale, die ausschließlich ins leere Wort ableitbar sind, betrachtet.

Angenommen es gibt zwei unterschiedliche Ableitungsbäume T_1 und T_2 mit gleicher Wurzel und gleichem Blattwort, so daß das Wurzelsymbol im Blattwort vorkommt, dann ist die Grammatik exponentiell mehrdeutig ($\Omega(1+\epsilon)^n$ für ein $\epsilon \in \mathbb{R}_+$). Man zeigt dies, indem man eine n–fache "Verkettung" der Bäume T_1 und T_2 auf 2^n verschiedene Arten bildet, die zu paarweise unterschiedlichen Bäumen führt, während stets dasselbe Blattwort einer Länge von $\mathcal{O}(n)$ erzeugt wird.

Untersucht man andererseits die Mehrdeutigkeit einer Grammatik G, die das obige Kriterium nicht erfüllt, so ergibt sich, daß diese durch ein Polynom beschränkt ist, das sich aus der Beschreibung von G konstruieren läßt. Damit folgt nicht nur die Notwendigkeit des obigen Kriteriums für exponentielle Mehrdeutigkeit, sondern auch, daß jede CFG entweder polynomiell ($\mathcal{O}(n^k)$ für ein $k \in \mathbb{N}$) oder exponentiell mehrdeutig ist.

Obwohl das genannte notwendige und hinreichende Kriterium für exponentielle Mehrdeutigkeit im allgemeinen nicht entscheidbar ist, erleichtert es für viele Grammatiken den Nachweis ihrer exponentiellen Mehrdeutigkeit erheblich. Darüberhinaus kann man, als entscheidbaren Spezialfall, nachweisen, daß die Existenz eines Nichtterminals, das sowohl links– als auch rechtsrekursiv ist, exponentielle Mehrdeutigkeit impliziert. Dieses hinreichende Kriterium ist eine echte Erweiterung eines bekannten Mehrdeutigkeitskriteriums.

Schließlich kann man zu einem beliebigen $k \in N$ eine Grammatik angegeben, deren Mehrdeutigkeitsgrad $\Theta(n^k)$ beträgt.

Trierer Forschungsberichte Mathematik / Informatik

Trierer Forschungsberichte
Fachbereich IV - Mathematik/Informatik
Universität Trier
D-54286 Trier

ISSN 0944-0488