Universität Stuttgart
Fakultät Informatik

Institut für Informatik
Breitwiesenstraße 20-22
D-70565 Stuttgart

Workshop on Formal Languages, Automata and Petri-Nets

Holger Petersen
(Editor)

Report Nr. 1998/01

January 1998

Foreword

This report contains abstracts of the lectures presented at the workshop “Formal Languages, Automata and Petri-Nets” held at the University of Stuttgart on January 16–17, 1998. The workshop brought together partners of the German-Hungarian project No. 233.6, Forschungszentrum Karlsruhe, Germany, and No. D/102, TéT Foundation, Budapest, Hungary. It provided an opportunity to present work supported by this project as well as related topics.

The editor wishes to thank the University of Stuttgart and the “Vereinigung von Freunden der Universität Stuttgart e.V.” for support which made the participation of the Hungarian colleagues possible. Thanks are also due to the speakers for their contributions and to the members of the group “Theoretische Informatik” for their assistance.

Stuttgart, January 1998 Holger Petersen

Speakers

Cooperating/Distributed Grammar Systems – Various Different Points of View

Henning Bordihn
Institut für Wissens- und Sprachverarbeitung
Otto-von-Guericke-Universität
Universitätsplatz 2, 39106 Magdeburg

Cooperating/distributed grammar systems (CD grammar systems, for short) have been introduced by E. Csuhaj-Varjú and J. Dassow in 1988 for describing multi-agent systems by means of formal grammars and languages, based on blackboard architectures. A CD grammar system consists of a finite set of grammars that cooperate in deriving words of a common language, where the grammars work on the sentential form in turns according to some cooperation protocol (the mode of derivation).

By the original motivation, competence and completeness of the agents are important features that influence the behaviour of blackboard-type problem solving systems which can be formalized and interpreted in the syntactic framework provided by CD grammar systems. Moreover, they can form a basis of the cooperation protocol. We compare several formalizations of these concepts, mainly variants of the well-known competence-based t-mode of derivation, concerning the induced derivational power. The results are given for both forward deduction systems with generating and backward deduction systems with accepting (analyzing) context-free grammar components.

We note that grammar systems with a cooperation strategy based on the concept of completeness / incompleteness of the components had already been considered by R. Meersman and G. Rozenberg in 1978, as a generalization of two-level substitution grammars to a multi-level approach.

Another point of view is given by seeing CD grammar systems working in t-mode of derivation as a sequential analogue of extended tabled Lindenmayer (ET0L) systems (as context-free grammars can be seen as sequential counterpart of extended non-tabled L systems). We consider CD grammar systems which correspond to the usual variants of ET0L systems, in particular we define CD grammar systems with pure context-free components and we order the generated language classes in an almost complete hierarchy. Furthermore, the influence of restrictions in the number of components is studied.

Networks of Language Processors

Erzsébet Csuhaj-Varjú
Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende u. 13-17, H-1111 Budapest, Hungary
email: `csuhaj@sztaki.hu`

Networks of language processors is a collective term ([6]) introduced as a formal language theoretic framework for describing several architectures for highly (massively) parallel and distributed symbolic processing. The notion is strongly motivated by some recent models and paradigms ([8], [14], [15], [12], [3], [4], [2], [10], [9], [5]). There are a lot of arguments for formulating such a concept: among other things, the need for reliable language theoretic support of communication networks, understanding the nature of massively parallel and distributed architectures, including ones with biological or other nature-motivated background.

A network of language processors (an NLP system) consists of several language identifying devices (or mechanisms computing multisets of strings), called language processors, which are located at the nodes of a (virtual) network. The language processors operate on strings (on sets of strings or multisets of strings) by performing rewriting steps and communication steps, usually alternately. The strings can represent data and/or programs (the latter ones correspond to operations in coded form or sets of rewriting rules). Both kinds of them can be rewritten and communicated among the nodes, providing dynamism and introducing adaptation and evolution in the mechanism. The same string can be interpreted at different nodes in different manners: it can play the role of a piece of data at some node and the role of a rewriting rule at some another one. Moreover, creation of new nodes and deletion of existing ones is allowed, mainly as a result of communication, leading to a flexible, self-organizing topology of the network.

At the beginning, each node of the network is initialized by a language processor and a set (multiset) of data strings which, together, form the initial configuration (initial state) of the system. The network is functioning by changing its configurations (this term is used in the case when rewriting rules are communicated) or its states (when only strings representing data are transmitted). The change of the configuration (state) can take place either by a rewriting step or by a communication step. By a rewriting step, some strings present at some nodes are rewritten according to the rewriting rule set and the rewriting mode of the corresponding nodes (by the metarules in the case of changing the rewriting rules). By a communication step, some strings (or copies of some strings) which are present at some node and satisfy some criteria (a trigger) are communicated to one or more other nodes. The language processors at the nodes of the network can work either in synchronous or in asynchronous way.

The triggers, the conditions for communication, can be defined in various manners. The most frequently discussed variants are those ones where filter languages are associated with the nodes for controlling the input/output string flow. A string can be communicated from one node to another one if it is an element of the output filter language of the sender node and it is in the input filter language of the target node.

Variants of communication open a wide variety of classifications of networks of language processors. If each rewriting step is followed by a communication step, then we speak of networks of language processors with nodes communicating by command. If the rewriting continues until a previously prescribed state is reached (a state with a request for communication) and the communication step takes place afterwards, then we speak of NLP systems with components communicating by request.

NLP systems are both computational and language identifying devices. Languages can be associated to networks of language processors in several manners. For example, we can distinguish a master node and take, as the determined language, all strings which appear at this node during all computations.

State sequences of networks of language processors are of particular interest: is there any periodicity in the behaviour of the system, are there deadlock situations, what can we say about the reachability of the states, about the safety of the system?

In this talk we provide the framework and discuss some variants of NLP systems, including networks of Watson-Crick reactive systems ([7]), a model motivated by DNA computing, and WAVE rewriting systems ([2]), grammatical models of the WAVE paradigm ([14],[15]).

References

[1] E. Csuhaj-Varjú, Networks of language processors. (A survey). In: Lenguajes Naturales Y Lenguajes Formales XII, (C. Martin-Vide, ed.), PPU, Barcelona, 1996, 169-189.

[2] E. Csuhaj-Varjú and H. Fernau, Grammatical models of the WAVE paradigm. Manuscript, 1997.

[3] E. Csuhaj-Varjú, L. Kari and Gh. Păun, Test Tube Distributed Systems Based on Splicing. Computers and Artif. Intelligence 15(2-3) (1996), 211-232.

[4] E. Csuhaj-Varjú, J. Kelemen and Gh. Păun, Grammar Systems with WAVE-like Communication. Computers and Artif. Intelligence 15(5) (1996), 419-436.

[5] E. Csuhaj-Varjú and G. Rozenberg, Actor Rewriting Systems. In preparation.

[6] E. Csuhaj-Varjú and A. Salomaa, Networks of Parallel Language Processors. In: New Trends in Formal Languages. Control, Cooperation and Combinatorics, (Gh. Păun, A. Salomaa, eds.), LNCS 1218, Springer Verlag, Berlin-Heidelberg-New York, 1997, 299-318.

[7] E. Csuhaj-Varjú and A. Salomaa, Networks of Watson-Crick reactive systems. In preparation.

[8] L. Errico and C. Jesshope, Towards a new architecture for symbolic processing. In: Proc. Conf. Artificial Intelligence and Information-Control Systems of Robots' 94, (I. Plander, ed.), World Scientific, Singapore, 1994, 31-40.

[9] R. Freund, E. Csuhaj-Varjú and F. Wachtler, Test Tube Systems with Cutting/Recombination Operations. In: Pacific Symposium on BIOCOMPUTING'97, (R.B. Altman, et.al eds.), World Scientific, Singapore, 1997, 163-174.

[10] R. Freund and F. Freund, Test Tube Systems or How to Bake a DNA Cake. Acta Cybernetica 12 (1996), 445-459.

[11] S. E. Fahlman, G. E. Hinton, T. J. Seijnowski, Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In: Proc. AAAI- Natl. Conf. on AI., William Kaufman, Los Altos, 1983, 109-113.

[12] C. Hewitt, Viewing Control Structures as Patterns of Passing Messages. J. of Artificial Intelligence 8 (1977), 323-364.

[13] W. D. Hillis, The Connection Machine, MIT Press, Cambridge, 1985.

[14] P.S. Sapaty, The WAVE model for advanced knowledge processing. In: CAD Accelerators, (A. P. Ambler, P. Agrawal, W.R. Moore, eds.), Elsevier Science Publ. B. V., Amsterdam, 1990.

[15] P.S. Sapaty, The WAVE paradigm. Internal Report 17/92, Dept. of Informatics, University of Karlsruhe, 1992.

Some Identities Related to Automata, Determinants, and Möbius Functions

Volker Diekert
Institut für Informatik
Universität Stuttgart
Breitwiesenstr. 20–22, D-70565 Stuttgart, Germany

based on joint work with Yuji Kobayashi
Faculty of Science of Toho University, Japan

Möbius functions play an important role in combinatorics. For monoids they were introduced by Cartier and Foata leading to a generalization of Mac Mahon's Master Theorem. The spirit of this theorem is an expression of certain series as the formal inverse of some determinant. In the same spirit Choffrut and Goldwurm have shown recently that (an unambiguous lifting of) the Möbius function of a free partially commutative monoid can be expressed as the determinant of the minimal automaton recognizing the set of lexicographic normal forms, if and only if the independence relation has a transitive orientation. Taking a slightly different viewpoint this result is in fact a statement about languages defined by forbidden factors of length exactly two. The aim of the present talk is a generalization to other classes of languages defined by forbidden factors. The main result states:

Let $F \subseteq X^*$ be a finite set of forbidden factors, $1 \notin F$, and let $l = \max\{|u| \mid u \in F\}$, $l \geq 2$, the maximal length. Let $S = X^* \setminus X^*FX^*$ and M be an $n \times n$-matrix associated to the minimal finite automaton recognizing the language S. Then we find explicitly a polynomial $D(S, l-1)$ of degree at most $n(l-1) \cdot \max\{1, (l-2)\}$ such that we have the following identity in commuting variables:

$$S \cdot Det(1 - M) \equiv_c D(S, l-1).$$

On Graph-Controlled Grammars with Leftmost Derivation

Henning Fernau
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: `fernau@informatik.uni-tuebingen.de`

We try to study leftmost derivations in regulated rewriting in a systematic way by investigating this notion in the framework of graph-controlled grammars. Since many seemingly different ways of regulation can be seen as special cases of graph-controlled grammars [4], we obtain all known and many new results on leftmost derivation in regulated rewriting (compare the Section 1.4 in the monograph [2]). This work can be seen as continuation of previous works on leftmost derivation, see [1, 5].

Moreover, we want to prove the versatility of graph-controlled grammars as unifying framework for presenting results in regulated rewriting. Taking this approach, some published proofs can be simplified considerably. Moreover, it is useful to detect unexplored sub-areas (which are sometimes surprising findings in view of the maturity of the whole field). For example, there seems to be no previous study of leftmost derivation in time-variant grammars.

Although graph-controlled grammars are not as general as for example selective substitution grammars, see Section 10 in [2], they may provide new insights and view on regulated grammars, so that we promote their use as basic mechanism in regulated rewriting. There is another, pedagogical reason for this promotion: graph-controlled grammars are very intuitive and easy to explain. So, any student or working professional who likes to learn something about regulated rewriting has the chance to get the basics including many of its ramifications quickly. Such things are quite important in our eyes for a rather matured field like regulated rewriting. The need for such systematic presentation is exemplified by the papers [6] and [3], where ideas from regulated rewriting are applied to parsing theory and database theory, respectively, although the papers indicate that the knowledge of basic facts in regulated rewriting is not too widespread.

References

[1] J. Dassow, H. Fernau, and Gh. Păun. On the leftmost derivation in matrix grammars. Work in progress, 1997.

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

[3] G. Dong. Grammar tools and characterizations. In *Proceedings of the 11th ACM Symposium on Principles of Database Systems (PODS'92)*, pages 81–90, 1992.

[4] H. Fernau. Unconditional transfer in regulated rewriting. *Acta Inform.*, 34:837–857, 1997.

[5] H. Fernau and F. Stephan. How powerful is unconditional transfer? —When UT meets AC.—. Technical Report WSI–97–7, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 1997. Short version to be published in the proceedings of DLT'97.

[6] A. Rußmann. Dynamic LL(k) parsing. *Acta Inform.*, 34:267–289, 1997.

Structured Formal Verification of a Fragment of the IBM/390 Clock Chip

Alfons Geser
Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Sand 13, D-72076 Tübingen

We present a simple and powerful method for formal verification of hardware. First the hardware netlist is translated into a system of term graph rewriting rules that model the state transition behaviour symbolically. This term graph rewriting system is then used to rewrite a specification into normal form, which we can show is a term graph formed exclusively by propositional variables and connectives. In the third phase the propositional term graph is evaluated as a functional decision diagram. From it, one can read off whether verification was successful and if not, efficiently derive a counterexample.

The choice of term graph rewriting, rather than term rewriting, is motivated by the adequate translation that is possible in linear time. Term graph rewriting, as opposed to modal calculi, offers neither quantifiers nor fixed points nor modal operators. However, its expressive power is sufficient to prove bisimulation or reachability of states.

Using a prototype implementation we have successfully verified a fragment of an IBM chip which contains a few symmetries and is complex enough to withstand a naive verification attempt.

Minimality and Decidability Results Concerning Generating Systems of Primitive Recursive Functions

Sándor Horváth
Department of Computer Science
Eötvös Loránd University
Múzeum körút 6-8., II. 2-3, Budapest
H-1088, Hungary
email: horvath@cs.elte.hu

Joint work with Holger Petersen

We consider the class of primitive recursive functions (the definition can be found in textbooks on computability, e.g. [3, 7, 8, 9]). As this class includes most of the functions usually encountered in elementary arithmetic it is of some interest to find simple characterizations for it. Significant simplifications in the recursion scheme employed in the definition of the primitive recursive functions have been found by Gladstone [4, 5]. Further research in this direction is reported in [2].

We are interested in the initial functions underlying the class of primitive recursive functions. We reduce the set of initial functions to a minimum that cannot be simplified further without losing some functions.

We also deal with the class of partial recursive functions and eliminate some initial functions from a known generating system for this class.

Finally, we consider some decidability questions concerning program length and recursion-depth.

Lemma 1 *If from a set of initial functions all primitive recursive functions can be obtained by composition and primitive recursion then it necessarily contains an n- or $(n+1)$-place function for every $n \geq 1$.*

Theorem 1 *Every primitive recursive function can be obtained by a finite number of applications of composition and primitive recursion from the following functions:*

1. *The successor function S.*
2. *The projections U_1^3 and U_2^{2n+5}, $n \geq 0$.*

Moreover this set of initial functions is minimal in the sense that not all primitive recursive functions can be obtained from a proper subset of these functions.

If the 0-place functions (constants) are also considered as primitive recursive we get the following statement:

Theorem 2 *Every primitive recursive function including constants can be obtained by a finite number of applications of composition and primitive recursion from the initial functions given in the preceding theorem and any 0-place function. This set of initial functions is minimal.*

It is possible to show results analogous to the preceding theorem if the role of the projection U_1^3 is interchanged with U_1^{2m+1} for some $m \geq 2$ (U_1^3 is replaced by U_2^3 and U_2^{2m+1} is replaced by U_1^{2m+1}). If μ is added to the operations in Theorem 1 all partial recursive functions can be derived. However Lemma 1 does not remain valid. We have:

Theorem 3 *Every partial recursive function can be obtained by a finite number of applications of composition, primitive recursion, and unbounded minimization from the initial functions S, U_1^3, and any infinite subset of $\{U_2^k \mid k \geq 2\}$.*

In [6] a generating system for the set of partial recursive functions is given equipped with the initial functions N, U_1^1, and S. Its operations are composition, primitive recursion, minimization, and attachments of new variables to the left or to the right. The function N can be omitted by applying μ to U_1^2. From Theorem 1 we get another reduction of the set of initial functions:

Theorem 4 *Every partial recursive function can be obtained from the initial functions S and N by composition, primitive recursion, minimization, and attachments of new variables.*

By a standard numerical encoding of the representations of the one-place partial recursive functions (in Kleene's classical generating system), regarding the code numbers as "abstract programs", and assigning to every abstract program, as its "length", the length of the original representation (of the corresponding partial recursive function), we clearly get an (axiomatic) Blum program size measure. Within the above encoding, the (abstract) programs of the one-place primitive recursive functions form a subrecursive enumeration, see [1]. So by a general result we have that the set $SP(primrec)$ of the shortest programs of the one-place primitive recursive functions is recursively enumerable. On the other hand it is also known that the analogous set $SP(partrec)$ (of the shortest programs of all the one-place partial recursive functions) is immune (i.e., infinite, and contains no infinite r.e. subset). Now we formulate the following

Problem 1 *Is $SP(primrec)$* not *recursive?*

Meyer and Ritchie's loop programs compute exactly the primitive recursive functions. It is known (see [1]) that the "complexity problem" of loop programs is undecidable for loop-depth ≥ 3. (This means that if an arbitrary loop program π has loop-depth $n \geq 3$ then it is undecidable whether there exists an equivalent loop program π' with loop-depth $n-1$.) According to loop-depth we obtain Meyer and Ritchie's well-known hierarchy $\{L_n\}_{n\geq 0}$ (within the class of primitive recursive functions). A related hierarchy is Axt's hierarchy $\{K_n\}_{n\geq 0}$, in which the role of loop-depth is played by the depth of primitive recursion. If we allow the more general simultaneous primitive recursion, too, we get a modification $\{K_n^{sim}\}_{n\geq 0}$ of Axt's hierarchy. It is known that the hierarchies $\{L_n\}_{n\geq 0}$ and $\{K_n^{sim}\}_{n\geq 0}$ coincide, and $L_n = K_n$ for $n \geq 4$. Clearly we can construct an effective translation of the loop programs into the representations of the primitive recursive functions in which simultaneous primitive recursion is allowed, too, such that, under this translation, loop-depth n is always mapped into recursion-depth n, for every $n \geq 0$. We can get an analogous effective translation of the loop programs into the "simple" representations of the primitive recursive functions (in which simultaneous primitive recursion is not allowed), for loop-depth $n \geq 4$. So, in view of the above we have:

Proposition 1 *The problem of reducibility of recursion-depth n in the hierarchy $\{K_n^{sim}\}_{n\geq 0}$ is undecidable for $n \geq 3$, and the analogous problem in the Axt hierarchy is undecidable for $n \geq 5$.*

Problem 2 *Is the reducibility of recursion-depth n in the Axt hierarchy undecidable for $n = 3, 4$, too?*

References

[1] Giorgio Ausiello. *Complessità di calcolo delle funzioni.* Editore Boringhieri, Torino, 1975.

[2] Christian Calude and Lila Sântean. On a theorem of Günter Asser. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:143–147, 1990.

[3] Martin Davis. *Computability & Unsolvability*. McGraw-Hill Series in Information Processing and Computers. McGraw-Hill Book Company, New York, Toronto, London, 1958.

[4] M. D. Gladstone. A reduction of the recursion scheme. *Journal of Symbolic Logic*, 32:505–508, 1967.

[5] M. D. Gladstone. Simplifications of the recursion scheme. *Journal of Symbolic Logic*, 36:653–665, 1971.

[6] Sándor Horváth. Finite, simple generating systems for partial recursive functions—a short note. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 28:30–32, 1986.

[7] Michael Machtey and Paul Young. *An Introduction to the General Theory of Algorithms*. Theory of Computation Series. North Holland, New York, 1978.

[8] Claus-Peter Schnorr. *Rekursive Funktionen und ihre Komplexität*. B. G. Teubner, Stuttgart, 1974.

[9] George J. Tourlakis. *Computability*. Reston Publ. Co., Inc., A Prentice-Hall Company, Reston, Virginia, 1984.

Once More on the Context-Freeness of $Q \cap (ab^*)^n$

László Kászonyi
Department of Mathematics
Berzsenyi College
H-9700 Szombathely

Let Q be the language consisting of primitive words over the alphabet $\{a, b\}$ and consider the language $L_n = Q \cap (ab^*)^n$. It is conjectured that L_n is context-free for all $n \in N$. The validity of the conjecture is proved for n-s having special number-theoretic character. The proofs of these results are based on the well-known theorem of Ginsburg, which asserts — applied to our case — that L_n is context-free iff the set

$$E(L_n) = \{(e_0, \ldots, e_{n-1}) \in N^n \mid ab^{e_0} \cdots ab^{e_{n-1}} \in L_n\}$$

is a stratified semilinear set. Here we give a decomposition of $E(L_n)$ in special components and conjecture that each of these components is a stratified semilinear set. We prove the validity of the conjecture in a special case.
(Joint work with M. Katsura)

On General Iteration Lemmata

Manfred Kudlek
Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Straße 30, 22527 Hamburg
email: kudlek@informatik.uni-hamburg.de

1 Introduction

The iteration lemmata for regular, linear and context-free languages are well known. They are based on the catenation operation (with unit element λ) on the free monoid V^* over some alphabet V, and the norm $|w|$ (length of words).

In this paper other binary operations $\circ$ on the power set on underlying monoids M are introduced, as well as other general norms μ. The operations have to be associative with zero element $\emptyset$ and unit element $\{\lambda\}$, and distributive with $\cup$ such that the resulting structure is an ω-complete semiring. The norm μ has to be monotone with respect to $\circ$ and $\cup$, with some minimal norms for $\emptyset$ and $\{\lambda\}$, and be defined for all finite sets.

If rational, linear and algebraic languages are defined as fixed points of corresponding systems of equations on ω-complete semirings, it can be shown that iteration lemmata similar to the classical ones hold for such languages.

2 Definitions

Let M be a monoid with binary operation $\circ$ and unit element λ. Extend $\circ$ to an associative binary operation $\circ : \mathcal{P}(M) \times \mathcal{P}(M) \rightarrow \mathcal{P}(M)$, distributive with $\cup$ ($A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$ and $(A \cup B) \circ C = (A \circ B) \cup (B \circ C)$), with unit element $\{\lambda\}$ ($\{\lambda\} \circ A = A \circ \{\lambda\} = A$), and zero element $\emptyset$ ($\emptyset \circ A = A \circ \emptyset = \emptyset$).

Then $\mathcal{S} = (\mathcal{P}(M), \cup, \circ, \emptyset, \{\lambda\})$ is an ω-complete semiring, i.e. if $A_i \subseteq A_{i+1}$ for $0 \leq i$ then $B \circ \bigcup_{i \geq 0} A_i = \bigcup_{i \geq 0} (B \circ A_i)$ and $(\bigcup_{i \geq 0} A_i) \circ B = \bigcup_{i \geq 0} (A_i \circ B)$.

Define also $A^{\circ(0)} = \{\lambda\},\ A^{\circ(1)} = A, A^{\circ(k+1)} = A \circ A^{\circ(k)},\ A^{\circ} = \bigcup_{k \geq 0} A^{\circ(k)}$.

Let $\mu : \mathcal{P}(M) \rightarrow I\!N$ be a (partial) function (norm) defined for all finite sets, with the following properties :

$\mu(\emptyset) \leq 1\ ,\ \mu(\{\lambda\}) \leq 1\ ,\ A \subseteq B \Rightarrow \mu(A) \leq \mu(B)\ ,\ \mu(A), \mu(B) \leq \mu(A \cup B) \leq max\{\mu(A), \mu(B)\}$,
$\mu(A), \mu(B) \leq \mu(A \circ B) \leq \mu(A) + \mu(B)\ ,\ (A \neq \emptyset \wedge A \neq \{\lambda\}) \Rightarrow \mu(A^{\circ}) = \infty$.

Example 1 : Let $\circ = \cdot$, the usual catenation (being associative with unit element λ on $M = V^*$), and μ be defined by

$\mu(\emptyset) = \mu(\{\lambda\}) = 0\ ,\ w \in V^* \Rightarrow \mu(\{w\}) = |w|\ ,\ \mu(A \circ B) = \mu(A) + \mu(B)\ ,$
$\mu(A \cup B) = max\{\mu(A), \mu(B)\}\ ,\ \mu(A^{\circ}) = \infty$.

Example 2 : Let $\circ = ⧢$, the shuffle operation (being associative and commutative on $M = V^*$ with unit element λ), and μ be defined as in Example 1.

Example 3 : Let V be an alphabet and $I \subseteq V \times V$ a symmetric, not necessarily reflexive, relation, called an independence relation. Define $uabv \sim ubav$ if $(a, b) \in I$ and consider its reflexive and transitive closure $\sim^*$. Then $\sim^*$ is an equivalence relation, and the set $M = V^*/\sim^*$, also written V^*/I, being a monoid, is called the trace monoid of V with respect to I.

For $t_1 = [u], t_2 = [v] \in M$ the binary operation on traces is defined by $t_1 \circ t_2 = [uv]$, and the neutral element is $[\lambda]$. For $t = [w] \in M$ a norm can be defined by $\mu(t) = |w|$.

Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of variables such that $X \cap M = \emptyset$.

A *monomial* over $\mathcal{S}$ with variables in X is a finite string of the form : $A_1 \circ A_2 \circ \ldots \circ A_k$, where $A_i \in \mathcal{X}$ or $A_i \subseteq M, |A_i| < \infty, i = 1, \ldots, k$. (without loss of generality, $A_i = \{\alpha_i\}$ with $\alpha_i \in M$ suffices). A *polynomial* $p(X)$ over $\mathcal{S}$ is a finite union of monomials where $X = (X_1, \cdots, X_n)$.

A *system of equations* over $\mathcal{S}$ is a finite set of equations : $E = \{X_i = p_i(X) \mid i = 1, \ldots, n\}$, where $p_i(X)$ are polynomials.

The *solution* of E is a n-tuple $(L_1, \ldots, L_n)$ of languages over M, with $L_i = p_i(L_1, \ldots, L_n)$ and the n-tuple is minimal with this property, i.e. if $(L'_1, \ldots, L'_n)$ is another n-tuple that satisfies E, then $(L_1, \ldots, L_n) \leq (L'_1, \ldots, L'_n)$ (where the order is defined componentwise with respect to inclusion).

From the theory of semirings follows that any system of equations over $\mathcal{S}$ has a unique solution, and this is the least fixed point starting with $(X_1, \cdots, X_n) = (\emptyset, \cdots, \emptyset)$.

A system of equations is called *linear* if all monomials are of the form $A \circ X \circ B$ or A, and *rational* if they are of the form $X \circ A$ or A, with $A \subseteq M$ and $B \subseteq M$. Corresponding families of languages (solutions of such systems of equations) are denoted by $\underline{ALG(\circ)}$, $\underline{LIN(\circ)}$, and $\underline{RAT(\circ)}$. If $\circ$ is commutative then $\underline{ALG(\circ)} = \underline{LIN(\circ)} = \underline{RAT(\circ)}$.

3 Results

The following theorems can be proven in a way analogous to the classical iteration lemmata.

Theorem 1 : Let $L \in \underline{RAT(\circ)}$ with $L \subseteq M$. Then there exist $n(L) > 0$ such that, for any $w \in L$ with $\mu(\{w\}) \geq n(L)$, there exist $x_1, x_2, x_3 \in M$ such that :
(i) $w \in \{x_1\} \circ \{x_2\} \circ \{x_3\}$.
(ii) $0 < \mu(\{x_1\} \circ \{x_2\}) \leq n(L)$.
(iii) $\{x_1\} \circ \{x_2\}^{\circ} \circ \{x_3\} \subseteq L$. □

Theorem 2 : Let $L \in \underline{LIN(\circ)}$ with $L \subseteq M$. Then there exist $n(L) > 0$ such that, for any $w \in L$ with $\mu(\{w\}) \geq n(L)$, there exist $x_1, x_2, x_3, x_4, x_5 \in M$ such that :
(i) $w \in \{x_1\} \circ \{x_2\} \circ \{x_3\} \circ \{x_4\} \circ \{x_5\}$.
(ii) $\mu(\{x_1\} \circ \{x_2\} \circ \{x_4\} \circ \{x_5\}) \leq n(L)$.
(iii) $0 < \mu(\{x_2\} \circ \{x_4\})$
(iv) $\forall k \geq 0 : \{x_1\} \circ \{x_2\}^{\circ(k)} \circ \{x_3\} \circ \{x_4\}^{\circ(k)} \circ \{x_5\} \subseteq L$. □

Theorem 3 : Let $L \in \underline{ALG(\circ)}$ with $L \subseteq M$. Then there exist $n(L) > 0$ such that, for any $w \in L$ with $\mu(\{w\}) \geq n(L)$, there exist $x_1, x_2, x_3, x_4, x_5 \in M$ such that :
(i) $w \in \{x_1\} \circ \{x_2\} \circ \{x_3\} \circ \{x_4\} \circ \{x_5\}$.
(ii) $\mu(\{x_2\} \circ \{x_3\} \circ \{x_4\}) \leq n(L)$.
(iii) $0 < \mu(\{x_2\} \circ \{x_4\})$
(iv) $\forall k \geq 0 : \{x_1\} \circ \{x_2\}^{\circ(k)} \circ \{x_3\} \circ \{x_4\}^{\circ(k)} \circ \{x_5\} \subseteq L$. □

To prove these theorems the systems of equations are first converted into equivalent systems of equations (with additional variables) where all monomials are in normal form ($X \circ Y$ or α for algebraic, $\alpha \circ X$ or $X \circ \alpha$ or α for linear, and $X \circ \alpha$ or α for rational systems).

Any $w \in L$ can be generated as $w \in \{\beta_1\} \circ \cdots \circ \{\beta_k\}$ where the $\beta_j \in M$ are the leaves of a binary derivation tree with respect to $\circ$, and the children of each node correspond to monomials. Note that μ is monotone with respect to $\cup$ and $\circ$, but bounded by the sum.

References

1 J. S. Golan : The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science. Longman Scientific and Technical, 1992.

2 W. Kuich, A. Salomaa : Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science, Springer, Berlin, 1986.

Dense Completeness

Klaus-Jörn Lange
Theoretische Informatik / Formale Sprachen
Wilhelm-Schickard-Institut für Informatik
Sand 13, D-72076 Tübingen

Formal Language Theory and Structural Complexity Theory exhibit close relationships. In spite of the fact that both theories pursue comparable questions like, e.g., the comparison of determinism and nondeterminism, they both succeed quite differently in answering these questions. While Complexity Theory is to a large extent a collection of open problems and most of its results relate open problems, the phenomenon of open problems is almost unknown in Formal Language Theory. This raises the question, how close their mutual relationship is.

Almost all formal languages that are in some way context-free, are related to the classes *NP*, $NAuxPDA_{pt}$, $NSPACE(\log n)$, or NC^1. Analogous relations hold between deterministic versions of these families and the classes *P*, $DAuxPDA_{pt}$, $DSPACE(\log n)$, and NC^1. The type of these relations is that a family of formal languages $\mathcal{A}$ is contained in a complexity class $\mathcal{B}$ and that $\mathcal{A}$ contains a $\mathcal{B}$-complete language.

Based on an observation by Richard Beigel it is now possible to construct in a systematic way for a class of formal languages $\mathcal{A}$, defined by sequential one-way automata and for which the class $\mathcal{B}$ is closed under logspace reductions, a type of nondeterministic one-way automata, such that for a corresponding class of languages $\mathcal{A}'$ it holds that $\mathcal{A} \subseteq \mathcal{A}' \subset \mathcal{B}$ and

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

The family $\mathcal{A}'$ thus lies *dense* in the class $\mathcal{B}$. The construction and some of its properties will be presented in the lecture. It is open whether a comparable construction can be found for deterministic families or classes below $DSPACE(\log n)$.

Confluence of Terminating 2–Dimensional String Rewriting Systems is Undecidable

Markus Lohrey
Institut für Informatik
Universität Stuttgart
Breitwiesenstr. 20–22, D-70565 Stuttgart

2–dimensional string rewriting systems are a straight forward generalization of ordinary string rewriting systems. Given two finite alphabets Σ_1 and Σ_2, a 2–dimensional string rewriting system over (Σ_1, Σ_2) is a finite set $\mathcal{R}$ of rules of the form $(l_1, l_2) \to (r_1, r_2)$, where $l_1, r_1 \in \Sigma_1$ and $l_2, r_2 \in \Sigma_2$. As expected, $\mathcal{R}$ defines a 1–step rewrite relation $\to_{\mathcal{R}}$ by $(u_1, u_2) \to_{\mathcal{R}} (v_1, v_2)$ iff there exist a rule $(l_1, l_2) \to (r_1, r_2)$ in $\mathcal{R}$ and strings $u_1', u_2'' \in \Sigma_1$, $u_2', u_2'' \in \Sigma_2$ such that $u_1 = u_1' l_1 u_1''$, $u_2 = u_2' l_2 u_2''$ and $v_1 = u_1' r_1 u_1''$, $v_2 = u_2' r_2 u_2''$. Thus, 2–dimensional string rewriting is a special case of trace rewriting.

For ordinary string rewriting systems it is known to be decidable whether a given *terminating* string rewriting system is confluent. In contrast to this result, I will prove that the question whether a terminating 2–dimensional string rewriting system over $(\{1\}, \Sigma)$ is confluent is undecidable. The proof is based on a technique developed by Narendran and Otto.

The Code Problem for Trace Monoids

Anca Muscholl
Institut für Informatik
Universität Stuttgart
Breitwiesenstr. 20–22, D-70565 Stuttgart

The topic of codes in the framework of trace monoids leads to interesting and challenging decision problems of combinatorial flavour. We consider in this talk the unique decipherability problem (code problem), i.e. the question whether a (finite) subset X of a trace monoid generates a free monoid.

The code problem is in general undecidable, yet there is no exact characterization of the trace monoids where this problem is decidable. We establish in this talk the decidability resp. undecidability of the code problem for some particular families of graphs (i.e., commutation relations I). These results are tightly connected to the emptiness problem for certain types of multicounter automata.

On Some Recognizable Picture-Languages *

Klaus Reinhardt
Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13, D-72076 Tübingen, Germany
email: reinhard@informatik.uni-tuebingen.de

In [GRST94] pictures are defined as two-dimensional rectangular arrays of symbols of a given alphabet. A set (language) of pictures is called recognizable if it is recognized by a finite tiling system. It was shown in [GRST94] that a picture language is recognizable iff it is definable in existential monadic second-order logic. In [Wil97] it was shown that star-free picture expressions are strictly weaker than first-order logic. A comparison to other regular and contextfree formalisms to describe picture languages can be found in [Mat97b, Mat97a].

We show that the language of pictures over $\{a, b\}$, where all occurring b's are connected is recognizable, which solves an open problem in [Mat97a]. (Connectedness is not recognizable in general [FSV95].)

Furthermore we show that the language of pictures over $\{a, b\}$, where the number of a's is equal to the number of b's is nonuniformly recognizable. Hereby we use counters similar to those used in [Für82].

References

FSV95 Ronald Fagin, Larry J. Stockmeyer, and Moshe Y. Vardi. On monadic NP vs. monadic co-NP. *Information and Computation*, 120(1):78–92, July 1995.

Für82 Martin Fürer. The tight deterministic time hierarchy. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 8–16, San Francisco, California, 5–7 May 1982.

GRST94 Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over pictures and recognizability by tiling systems. Proceedings of the 11th STACS 94, LNCS 775, pages 365–375, 1994.

Mat97a Oliver Matz. *On Piecewise Testable, Starfree, amd Recignizable Picture Languages*, 1997.

Mat97b Oliver Matz. Regular expressions and context-free grammars for picture languages. In *14th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *lncs*, pages 283–294, Lübeck, Germany, 27 February– March 1 1997. Springer.

Wil97 Thomas Wilke. Star-free picture expressions are strictly weaker than first-order logic. In *Automata, Languages and Programming*, LNCS 1256 , pages 347–357, 1997.

*This research has been supported by the DFG Project La 618/3-1 KOMET.

On the Power of Parallel Communicating Grammar Systems

György Vaszil
Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende u. 13-17, H-1111 Budapest, Hungary

Parallel communicating grammar systems, introduced in [5], are formal language theoretic models of parallel and distributed computation. In these systems several grammars are working on their own sentential forms in parallel and their work is organized in a communicating system to generate a single language.

PC grammar systems have been the subject of detailed study over the past few years, see [1],[4]. Here we report on new results from [3],[2] and [6], concerning the generative power of PC grammar systems with context-free rules. We show that these systems generate all recursively enumerable languages, by demonstrating how they can simulate two-counter machines, a restricted but computationally complete class of Turing machines.

References

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

2 E. Csuhaj-Varjú, Gy. Vaszil, On synchronization in context-free PC grammar systems, submitted, 1997.

3 E. Csuhaj-Varjú, Gy. Vaszil, Context-free PC grammar systems are computationally complete, submitted, 1997.

4 J. Dassow, Gh. Păun, G. Rozenberg, Grammar Systems. Chapter 4. *Handbook of Formal Languages*, ed. by G. Rozenberg and A. Salomaa, Springer-Verlag, Berlin, 1997, 155-213.

5 Gh. Păun, L. Santean, Parallel communicating grammar systems: the regular case, *Ann. Univ. Bucharest, Ser. Matem.-Inform.* 38, 2 (1989), 55-63.

6 Gy. Vaszil, On parallel communicating Lindenmayer systems, In: *Grammatical Models of Multi-Agent Systems* (Gh. Păun, A. Salomaa, eds.), Gordon and Breach Science Publishers, to appear.