MATHEMATISCHES FORSCHUNGSINSTITUT OBERWOLFACH

T a g u n g s b e r i c h t 43/1972

Automatentheorie und Formale Sprachen

Die diesjährige Tagung über Automatentheorie und Formale Sprachen im Forschungsinstitut Oberwolfach stand unter der Leitung von G.Hotz und H.Langmaack (beide Saarbrücken). Aus Platzgründen war die Teilnehmerzahl auf 38 beschränkt.

Der Schwerpunkt lag mit insgesamt 9 Vorträgen auf dem Gebiet der formalen Sprachen. Es wurden Eigenschaften spezieller Grammatiktypen untersucht und insbesondere die Unentscheidbarkeit des Äquivalenzproblems für die Satzformen kontext-freier Sprachen gezeigt.

Die algebraische Behandlung formaler Sprachen kam vor allem in den Vorträgen der französischen Teilnehmer zur Geltung. Weiter wurde eine interessante verbandstheoretische Charakterisierung von Sprachen angegeben.

Auf dem Gebiet der Automatentheorie wurden universelle Automaten für die Klasse der endlichen Automaten, Automaten zur Erkennung mehrdimensionaler Sprachen und Automaten auf unendlichen Strukturen behandelt.

Die Komplexitätstheorie wurde in Vorträgen behandelt, die den Mindestaufwand beim Umgruppieren von Daten, die Effizienz von Speicherstrukturen und die Kosten Boole'scher Ausdrücke untersuchten. Weiter wurde ein Zusammenhang zwischen dem Syntheseproblem von Schaltkreisen und Analyseproblem kontextfreier Sprachen hergestellt.

Die Untersuchungen über Programmiersprachen bezogen sich auf die Analyse für kontext-freie Sprachen. Es wurde die Beziehung zwischen ALGOL-Programmen und Makrogrammatiken sowie die Bedeutung von Prozedurvariablen in ALGOL-Programmen untersucht und über eine Theorie der Semantik von Programmiersprachen vorgetragen. Ein Vortrag beschäftigte sich mit einer Sprache für Melodien.

Die traditionelle Wanderung führte uns bei schönem Herbstwetter von Oberwolfach nach St. Roman, wo es köstliche Schwarzwälder Kirschtorte gab.

Teilnehmer

L.S.v.Bethem-Jutting, Eindhoven
J.Berstel, Straßburg
A.Blikle, Warschau
K.H.Böhling, Bonn
W.Brauer, Hamburg
B.Brosowsky, Göttingen
V.Claus, Dortmund
A.Cremers, Karlsruhe
O.J.Dahl, Oslo
P.Deussen, Karlsruhe
H.-D.Ehrich, Kiel
H.Ehrig, Berlin
J.E.Engelfriet, Enschede
H.Feldmann, Hamburg
G.Hotz, Saarbrücken
H.Jürgensen, Kiel
P.Kandzia, Saarbrücken
R.Kemp, Saarbrücken
H.Kopp, Saarbrücken
W.Kuich, Wien
H.Lagaly, München
J.Loeckx, Saarbrücken
H.Maurer, Karlsruhe
O.Mayer, Karlsruhe
M.Nivat, Paris
W.Oberschelp, Aachen
Th.Ottmann, Münster
J.F.Perrot, Paris
J.Riguet, Paris
H.Ring, Oberwolfach
M.Rosendahl, Bochum
A.Salomaa, Turka
U.Schampel, Karlsruhe
C.P.Schnorr, Frankfurt
D.Siefkes, Heidelberg
H.J.Stoß, Konstanz
R.Vollmar, Erlangen

Vortragsauszüge

Eric Birgert (vorgetragen von J. Berstel):

Über einige arithmetische Eigenschaften rationaler formaler Sprachen

Es werden Ergebnisse von Untersuchungen zu den folgenden beiden Problemen vorgetragen:

1) Wann ist eine $\mathbb{Z}$-rationale Reihe mit nicht-negativen Koeffizienten $\mathbb{N}$-rational.
2) Wann ist die Hadamard-inverse Reihe einer rationalen Reihe selbst wieder rational.

Es wird gezeigt, wie diese beiden Probleme miteinander verknüpft sind und eine Antwort gegeben für den Fall, daß die vorgelegte Reihe austauschbar oder ihr Träger eine beschränkte Sprache ist.

A. Blikle:

Fixed Point Approach to the Theory of Formal Languages

In this presentation some ideas and arguments are given towards an algebraic (more precisely: lattice-theoretical) approach to formal language theory. A startpoint of this approach is an observation that the family of all languages over an alphabet is a complete lattice and that concatenation is a continuous operation in this lattice. Hence, some languages usually defined by grammars can be defined also as fixpoints of certain functions in a lattice of languages.

As it turns out there are other lattices similar in a sense

to the lattice of languages that are also of interest for computer science: lattices of binary relations and lattices of formal power series. In order to develop a general theory of the three types of lattices mentioned above the concept of a net is developed. A net is a system

$$\mathcal{N} = (U, \leq, o, e)$$

where U is a set, $(U, \leq)$ is a complete lattice, (U, o, e) is a monoid and o is left and right distributive over finite or countable joints.

Now one can investigate equations of the form

$$x = \phi(x)$$

where $\phi : U^n \to U^n$ for some positive n. A theory for such equations is developed. This theory is then applied to the theory of formal languages and to problems of the theory of programs.

A. B. Cremers:

Über Ordnungseigenschaften kontextfreier Sprachen

Mit Hilfe der Abhängigkeitsrelation von Čulik werden O(n)-Sprachen definiert. O(o)-Sprachen sind identisch mit den bekannten sequentiellen CF-Sprachen. Eine Verallgemeinerung eines Satzes von Shamir liefert dann die Darstellung der Familie aller CF Sprachen als unendliche Vereinigung einer aufsteigenden Folge von O(n)-Sprachen.

O(n)-Sprachen werden anhand der Konstruktionskomplexität der

zugehörigen CF-Ausdrücke charakterisiert: Eine unendliche CF-Sprache ist O(n) genau dann, wenn zu ihrer Darstellung mindestens n + 1 Iterationssymbole benötigt werden. Dieses Resultat führt zur Lösung eines offenen Problems von Gruska.

Die Unentscheidbarkeit einiger grundlegender Reduktionsprobleme für O(n)-Sprachen wird bewiesen.

O.-J. Dahl:

Top Down Parsers Expressed in a High Level Language

A simple ALGOL-like recursive procedure may be formulated to scan the relevant part of the left derivation tree D with respect to a given context-free grammars G and thereby parse an arbitrary word Z. The procedure requires an amount of storage space of the order $|Z|^2$ for parameter transmission. Space requirement is reduced to order $|Z|$ by replacing the parameter Z by a non-local storage device and by decomposing the algorithm with respect to the nodes of D. The latter requires the introduction of coroutine-like sequencing which in addition precides arbitrary interleaving of actions associated with the nodes. For a strategy corresponding to the original algorithm Z may be held on a two-way back-tracking tape.

P. Deussen:

Kostenfunktionen auf Boole'schen Ausdrücken

Für reellwertige Funktionen auf der Menge aller Boole'schen Ausdrücke wird ein Axiomensystem motiviert, das die Auswirkung von Substitutionen in den Ausdrücken auf den Wert der Funktion in der Weise regelt, daß die reellwertige Funktion als Kostenfunktion angesehen werden kann.

Die Sicherung der Existenz kostenminimaler Ausdrücke ist eine Voraussetzung für die effektive Berechnung solcher Ausdrücke.

Neben einigen Resultaten zur Berechnung kostenminimaler Ausdrücke wird eine Aufwandsabschätzung für die durch die Resultate nahegelegten Algorithmen angegeben.

H. Ehrig, M. Pfendner, H.-J. Schneider: (vorgetragen v.H.Ehrig)

Kategorielle Formulierung mehrdimensionaler formaler Sprachen

Unter einer mehrdimensionalen formalen Sprache wird hier eine Teilklasse aller durch ein Alphabetpaar $\Omega = (\Omega_E, \Omega_V)$ markierter Graphen verstanden. Hierbei kann man die Kardinalität von Ω_E als Dimension der Sprache auffassen, wenn man die Kanten zur Festlegung der gegenseitigen Lage oder Abhängigkeit der Knoten verwendet.

Beispiele: Formalisierung von Flußdiagrammen oder Netzwerken durch markierte Graphen bzw. "n-Diagramme".

Direkte Ableitungen zwischen markierten Graphen werden hier über Verklebungen beider Seiten einer "Produktion von markierten Graphen" mit der gleichen "Erweiterung" konstruiert. Eine solche Verklebung ist ein Pushout in der Kategorie der durch Ω markierten Graphen. Mit diesem symmetrischen Ableitungsprozeß werden Graph Ω-Grammatiken und -Sprachen definiert. Für diese Grammatiken wird eine 2-dimensionale Typenhierarchie angegeben, die die Chomsky-Hierarchie und die Graph-Grammatiken von T. W. Pratt umfaßt.

Darüberhinaus wird eine Ableitungskategorie als Verallgemeinerung von freien X-Kategorien konstruiert und es wird untersucht, unter welchen Voraussetzungen Ableitungen gegenüber Verklebungen abgeschlossen sind.

J. Engelfriet:

Programmschemata mit Hilfsvariablen

Sei V eine Menge von Variablen. Jede Variable v aus V hat einen endlichen Bereich B_v, und einen Anfangswert b_v aus B_v. Wir betrachten nun Programmschemata ("mit Hilfsvariablen"), die Tests der Form "v=b" und Operationen der Form "v:=b" enthalten (v aus V, b aus B_v). Die übrigen Tests und Operationen sind uninterpretierte Symbole. Bei einer Interpretation werden die Variablen am Anfang auf ihre Anfangswerte gesetzt; die Resultatwerte der Variablen werden negiert.

Sei RSI^V die Menge der rekursiven Programmschemata mit Hilfsvariablen. Wir zeigen:

(i) RSI^V ist "kräftiger" als $RSI^\emptyset$, aber noch nicht "kräftig genug".

(ii) Übersetzbarkeit eines Programmschemas aus RSI^V in ein nicht-rekursives Programmschema ist entscheidbar.

(iii) Das Äquivalenzproblem in RSI^V ist gleichwertig mit dem Äquivalenzproblem für determinierte kontextfreie Sprachen.

H. Feldmann:

Erprobung verschiedener Grammatiken an einer Sprache für Melodien hinsichtlich adäquater Darstellung von Aufruf und Vereinbarung

Es wird eine allgemeine Sprache für Melodien und ein Übersetzer zur Umsetzung in Rechenwerksschwingungen angegeben, die u.a. (aus ALGOL bekannt) "Vereinbarung eines Programmteils" und "Aufruf eines Programmteils" sowie Festlegung deren Relation verlangt. Erprobt werden u.a. "Scattered-Context-Grammars" (Greibach, Hopcroft 1968), "Indexed Grammars" (Aho 1968), "Grammars with Makrolike Productions" (Fischer 1968) und "Zweischichtige Grammatiken" (von Wijngaarden 1968).

G. Hotz:

Über das Syntheseproblem von Schaltkreisen

Es wird ein einfacher Zusammenhang hergestellt zwischen dem Syntheseproblem von Schaltkreisen und dem Analyseproblem formaler Sprachen. Es überträgt sich unmittelbar der Satz über generelle Unlösbarkeit des Syntheseproblems. Weiter gelten folgende Sätze:

1. Das Syntheseproblem ist generell lösbar für endliche semantische Bereiche, Bausteine mit einem Ausgang und Bausteine, die die Vertauschung von Eingängen und die Ausspaltung in mehrere Ausgänge repräsentieren.
2. Der Berechnungsaufwand für die Lösung des Syntheseproblems unter 1. steigt mindestens exponentiell mit der Zahl der Eingänge.

P. Kandzia:

Zur Bedeutung von Programmen mit Prozedurvariablen

Programme höherer Programmiersprachen lassen sich i.a. als Bezeichnungen von gewissen Zustandstransformationen ansehen, die sich mittels Konkatenation und Fallunterscheidung aus einigen elementaren Transformationen aufbauen lassen. Die Bedeutung eines derartigen Programms ergibt sich dann induktiv über den Aufbau des Programms aus den Bezeichnungen der elementaren Transformationen.
Bei Programmen mit Prozedurvariablen (siehe etwa EULER ALGOL 68) treten bei der Festlegung der Bedeutung in der angegebenen Weise Probleme auf. Es wird eine allgemeine Aufruffunktion eingeführt, mit deren Hilfe sich ein großer Teil der Probleme lösen läßt.

R. Kemp:

LR(k)-Akzeptoren für k > 0

An einem Beispiel wird ein Algorithmus, der sog. LR(0)-Algorithmus, geschildert, welcher den LR(0)-Akzeptor in Form eines endlichen Automaten liefert. Es wird zunächst

gezeigt, daß dieser Automat minimal ist. Anschließend wird ein Verfahren angegeben, welches den reduzierten LR(k)-Automaten aus dem LR(0)-Automaten liefert, falls die vorgegebene kontextfreie Grammatik G vom Typ LR(k) war. Anschließend wird eine Diskussion dieses LR(k)-Algorithmus durchgeführt und daraus einige Kriterien für das Nicht-vorliegen einer LR(k)-Grammatik abgeleitet. Diese Kriterien beziehen sich auf die Gestalt der auftretenden Produktionsregeln.

H. Kopp:

Eine Theorie der Semantik von Programmiersprachen

Die Semantik von Programmiersprachen läßt sich formal beschreiben als eine "mathematische Maschine" $\Delta : K \to K$. In der Konfigurationsmenge K stehen dabei Programm, Daten und die Ablauforganisation.

Programme werden aus elementaren Grundelementen, den "Zeilen" aufgebaut. Die syntaktische Struktur von Anweisungen, z.B. von arithmetischen Ausdrücken wird durch eine freie X-Kategorie F beschrieben und die Semantik wird darauf durch einen Funktor $\phi : F \to \mathfrak{A}$ verankert. Die Formulierung der Programmorganisation führt zu Ergebnissen über die Minimalität von Programmen, aus denen insbesondere die Notwendigkeit von runtime-Kontrollen folgt.

H. Langmaack:

Über eine Beziehung zwischen ALGOL-Programmen und Makrogrammatiken

Für syntaktisch korrekte ALGOL-Programme mit Prozeduren ist es generell nicht entscheidbar, ob alle Parameterübergaben bei Prozeduraufrufen korrekt sind oder nicht. Diese Unentscheidbarkeit gilt sogar dann, wenn Daten und Bedingungen völlig unberücksichtigt bleiben und wenn alle Prozeduraufrufe parallel ausgeführt werden. Unter den gleichen Voraussetzungen ist es auch unentscheidbar, ob eine Prozedur erreichbar ist oder nicht. Wenn dagegen die Prozeduren keine globalen formalen Parameter haben, dann liegt Entscheidbarkeit vor.

Das Unentscheidbarkeitsresultat scheint im Widerspruch zu einem Resultat von M. Fischer über Makrogrammatiken zu stehen die als ALGOL 60-Programme mit Prozeduren gedruckt werden können. Man kann aber zeigen, daß die beiden entscheidbaren Fälle lediglich Randfälle im Rahmen allgemeiner ALGOL-Programme sind.

J. Loeckx:

Die Relation zwischen "top-down", "bottom-up" und "left-corner" - Analyse

Es wird ein Automat beschrieben, der für eine gegebene beliebige kontext-freie Grammatik die syntaktische Analyse entweder nach dem "top-down"-Verfahren, dem "bottom-up"-

Verfahren, dem "left-corner"-Verfahren oder sogar nach einem gemischten Verfahren ausführen kann.

Auf diese Weise wird die Relation zwischen den drei klassischen Verfahren für die syntaktische Analyse illustriert. Außerdem stellt dieser Automat ein theoretisches Modell für einen Übersetzer dar, der nach dem gemischten Verfahren arbeitet.

O. Mayer:

Über matrixähnliche Sprachen

Die Matrix-Grammatik ist ein bekanntes Beispiel für eine kontext-freie Grammatik mit Einschränkungen bei der Anwendung der Produktionen. Durch verschiedene Interpretationen des zugrunde liegenden Prinzips der Anwendung ganzer Matrizen erhält man einige neue Grammatiktypen, die matrixähnlich genannt werden. Sie werden bezüglich der von ihnen erzeugten Sprachfamilien verglichen. Bei jeder der betrachteten matrixähnlichen Grammatiken genügen die in Ableitungen möglichen Anzahlen der Anwendungen der einzelnen Matrizen einem zugeordneten linearen Gleichungssystem. Für die sogenannten verallgemeinerten ungeordneten Vektorgrammatiken, einem der eingeführten matrixähnlichen Grammatikkonzepte, werden gewisse Zusammenhänge zwischen der Struktur der erzeugten Sprache und den Lösungen des zugeordneten linearen Gleichungssystems festgestellt aus denen folgt:

In der Familie der von diesen verallgemeinerten ungeordneten Vektorgrammatiken erzeugbaren Sprachen, einer echten Obermenge der kontext-freien Sprachen sind die Leerheit und die Endlichkeit entscheidbar.
Weiter ist das kommutative Bild einer jeden Sprache im Sinne von Parikh eine semi-lineare Menge.

M. Nivat:

L'AFL des non-générations de l'ensemble des langages algébriques est non-principal

Une famille de langages $\mathcal{L}$ est un cône ssi pour tout $L \in \mathcal{L}$ et pour toute transduction rationelle τ, $\tau(L) \in \mathcal{L}$. C'est un AFL si de plus elle est fermée par $\cup$, produit, étoile. Notant $\mathcal{C}(L)$ l'ensemble des images d'un élément de L dans une transduction rationelle quelconque (qui est le plus petit cône contenant $\mathcal{L}$) et $\mathcal{F}(\mathcal{L})$ le plus petit AFL contenant $\mathcal{L}$ on a $\mathcal{F}(\mathcal{L}) = \text{Rat} \circ \mathcal{C}(\mathcal{L})$. En d'autres termes l'AFL engendré par $\mathcal{L}$ est la fermeture rationelle du cône engendré par $\mathcal{L}$.

Nous nous intéressons aux sous-AFL de l'ensemble des langages algébriques (noté Alg.). La plupart de ceux qui jouent un rôle dans la théorie sont coniques c'est à dire vérifient la condition:

A cônique <=> $\forall m \quad \mathcal{F}(m) = A \Rightarrow \mathcal{C}(m) = A$, et ils le sont en vertu du théorème suivant.

Théorème 1: Tout AFL translatable est conique.

La famille $\mathcal{L}$ est translatable ssi pour tout $L \in \mathcal{L}$ et a, b lettres quelconques

$$[(a,b)^*,L] = \{a^n l\, b^n \mid n \in \mathbb{N},\ l \in L\} \quad .$$

On donne un certain nombre de corollaires de ce théorème de Boasson et Nivat à paraitre dans Acta Informatica. La démarche suit de très près celle de S. Greibach dans "Chains of full AFL's". En possession de cet outil, plus facile à manier que la substitution, il est naturel de se pencher sur l'ensemble des non-générateurs de Alg c'est à dire l'ensemble des langages algébriques L tels que $\mathcal{L}(L) \subsetneq$ Alg. Du lemme de S. Greibach sur la substitution découle aisement que cet ensemble noté NG est un AFL clos par substitution. S'il était principal, engendré par L, il contiendrait le langage

$$[(a,b)^*,LL] = \{a^n l\, l'\, b^n \mid n \in \mathbb{N},\ l,l' \in L\}$$

et la non principalité de NG découle alors du Théorème:

Théorème 2: $L \in$ Alg et $[(a,b)^n, LL] \in \mathcal{L}(l) \Rightarrow \mathcal{L}(L) =$ Alg.

Un corollaire immédiat permet d'énoncer

Corollaire: Alg est le plus petit AFL translatable et principal.

Th. Ottmann:

Simulation endlicher Automaten durch sequentielle Netzwerke aus einfachen Bausteinen

Es werden endliche, unvollständig definierte deterministische Automaten vom Mealy-Typ betrachtet. Dafür werden zwei Operationen "Parallelschalten" und "Rückkoppeln" definiert.

$\mathfrak{N}(M)$ bezeichne die Klasse der Automaten ("Netzwerke"), die alle Automaten der Menge M enthält und gegen die beiden Operationen abgeschlossen ist. Automaten sollen "einfach" heißen, wenn sie höchstens einen Zustand, einen Eingang oder einen Ausgang haben.

Satz 1:

Es gibt keine Klasse einfacher Automaten M, so daß jeder endliche Automat durch ein Netzwerk aus $\mathfrak{N}(M)$ simuliert werden kann.

Es werden ferner zwei konkrete einfache Automaten K,F sowie nicht-einfache Automaten D, H angegeben, so daß gilt:

Satz 2:

Jeder endliche Automat ist simulierbar durch ein Netzwerk aus $\mathfrak{N}(K,F,D)$ derart, daß die Zahl der verwendeten nicht-einfachen Bausteine höchstens linear von der Zahl der Ausgänge des gegebenen Automaten abhängt.

Satz 3:

Jeder Akzeptor ist simulierbar durch ein Netzwerk aus $\mathfrak{N}(K,F,D)$ mit höchstens 3 Exemplaren von H.

J.-F. Perrot:

Groupes de Suschkewitsch des Codes Préfixes

Un code préfixe sur un alphabet X est une partie C du monoide libre X^* engendré par X qui vérifie $C \cap C \cdot XX^* = \emptyset$.

Le code préfixe C est dit complet lorsque pour tout $w \in X^*$ on a $w \cdot X^* \cap CX = \emptyset$.

Le groupe de Suschkewitsch G(C) du code (préfixe, complet, rationel) est par définition la classe d'équivalence des sous-groupes maximaux de l'idéal minimal du monoide syntactique de C^*, le sous-monoide engendré par C dans X^* - ce monoide syntactique étant fini en vertu de l'hypothèse que C est rationel. G(C) est obtenu sous forme d'un groupe transitif de permutations.

On montre que tout groupe transitif est groupe de Suschkewitsch d'un code préfixe rationel, lequel est en général infini, et fini si le code C est fini. G(C) ne peut pas être quelquonque.

Differents critères sont donnés permettants d'établir que certains familles de groupes abstraits ne peuvent pas être réalisées comme groupes de Suschkewitsch de codes finis.

Par example le centre de G(C) est cyclique si C est fini.

J. Riguet

Categorical Algebra and Automata Theory

The main use of categorical algebra is to explain similarities between structures. Thus Automata Theory is now worked out in many places. We shall discuss here a special aspect of this theory: An automaton can be considered as an "action" and programming will be nothing but the art of realizing an action of computing by sequencing and repeating elementary actions. We get action categories and isomorphism diagrams between them. The formal details are rather complicated and will be discussed in a later publication.

H. Ring:

Universelle Automaten für die Klasse der endlichen Automaten

Ein Automat U heiße universell für eine Automatenklasse $\mathcal{A}$, falls zwei Codes $\phi_1 : \mathcal{A} \to \Sigma_u^*$ und

$$\phi_2 : \bigcup\{\Sigma_A^* : A \in \mathcal{A}\} \longrightarrow \Sigma_u^*$$

derart existieren, daß für alle $A \in \mathcal{A}$ und alle $x \in \Sigma_a^*$ gilt:

U akzeptiert $\phi_1(A)$ $\phi_2(x)$ genau dann, wenn A die Eingabe x akzeptiert.

Es wird ein determinierter Zweiweg-Kellerautomat angegeben, der universell ist für die Klasse der endlichen Automaten. Durch Abwandlung dieses Automaten läßt sich das Ergebnis noch verstärken.

M. Rosendahl:

Automaten zur Bearbeitung mehrdimensionaler formaler Sprachen

Die konfliktfreien mehrdimensionalen Grammatiken (siehe z.B. Vortrag Oberwolfach 1970) werden erweitert durch control-sets. Die control-sets steuern die Anwendung der Regeln des konfliktfreien Regelsystems. Über den Chomsky-Typ der Grammatik, die den control-set erzeugt, wird ein Zusammenhang zu den Automaten hergestellt. So werden u.a. mehrdimensionale (S)-Systeme und Kellerautomaten definiert und diesen mehrdimensionalen Grammatiken zugeordnet.

Sowohl die 1-dimensionalen Chomsky-Grammatiken und ihre zugeordneten Automaten, als auch einige Ansätze zur Erweiterung 1-dimensionaler Sprachen (z.B. parallel grammars, n-grammars, tuple-grammars) ergeben sich als Spezialfälle.

A. Salomaa:

Decidable and Undecidable Problems Concerning Lindenmeyer Systems

A language is termed an SF-language iff it equals the set of sentential forms of a context-free grammar. The equivalence problem for SF-languages (even those generated by a linear grammar) is shown undecidable by a direct reduction to Post's correspondence problem. Undecidability results for Lindenmayer systems follow, eg., it is undecidable whether two

propagating OL-systems generate the same language. SF-languages display resistance to ordinary closure operations, in fact, they form an "anti-AFL". Although there are non-context-free OL-languages, some decidability results for context-free languages (such as the decidability of the finiteness problem) can be extended to OL-languages. If growth functions of Lindenmayer systems are considered rather than languages, then many problems become decidable.

C.P. Schnorr:

Effizienz von Speicherstrukturen

Eine Speicherstruktur $S = \langle S,O,T \rangle$ ist ein Tripel, bestehend aus einer Menge S, einer endlichen Menge $O \subseteq S$ von Speichertransformationen und einer endlichen Menge von Prädikaten auf S. Man erklärt Homomorphismen und Simulationen zwischen Speicherstrukturen. Die Speicherstrukturen kann man nach ihrer "Effizienz" wie folgt ordnen:

$S_1 < S_2$ <=> Es gibt eine Simulation $S : S_1 \to S_2$, so daß S jeden Befehl von S_1 in ein schleifenloses Programm auf S_2 überführt. Wir zeigen, daß Graphenspeicher und Registerspeicher mit indirekter Adressierung gleich effizient sind. Diese Speicher sind effizienter als jede Mehr-Band-Turingmaschine.

D. Siefkes:

Endliche Automaten auf gewissen unendlichen Strukturen

Die monadische Theorie zweiter Stufe irgendwelcher Strukturen ist geeignet, das Verhalten endlicher Automaten auf diesen Strukturen zu beschreiben. Im Fall der abzählbaren Ordinalzahlen als Strukturen treten neben den Ordinalzahlen Modelle der Theorie auf, deren Grundbereich nicht wohlgeordnet ist. Am Beispiel von $\omega + \omega^*$ wird gezeigt, wie sich endliche Automaten auf solchen nicht wohlgeordneten Strukturen verhalten. Daraus ergibt sich eine Charakterisierung der vollständigen Erweiterungen der monadischen Theorien zweiter Stufe der endlichen und der abzählbaren Ordinalzahlen.

H.-J. Stoß:

Über den Mindestaufwand beim Umgruppieren von Daten

Ausgehend von der Situation beim Umgruppieren von Daten in einer EDV-Anlage mit großem Hintergrundspeicher wird folgendes Problem untersucht:
Gegeben sei eine Menge E von Blöcken zu je höchstens r Daten. Aus diesen Blöcken sollen neue gebildet werden, wobei in einem Schritt jeweils die Daten von höchstens k Blöcken neu angeordnet werden können. Gesucht ist eine untere Abschätzung für die Mindestzahl L(E,F) von Schritten, die notwendig ist, um aus einer gegebenen Blockmenge E eine neue Blockmenge F zu erzeugen.

Hierzu wird zunächst der Begriff "Komplexitätsmaß" eingeführt und gezeigt, daß jedes Komplexitätsmaß eine untere Abschätzung für L liefert.
Anschließend wird ausgehend von den Größen (e f), (eεE, fεF) ein numerisch einfach zu bestimmendes Komplexitätsmaß und damit eine konkrete untere Abschätzung für L angegeben.

R. Vollmar:

Über Turingmaschinen mit variabler Struktur

Es wird hier eine modifizierte Art von Turing-Maschinen betrachtet:

1) Die auf dem Speicherband stehende Information "verschwindet", wenn sie während einer gewissen Zeit nicht berührt, d.h. geschrieben oder wieder gelesen wurde.
2) Zustände der endlichen Steuereinheit, die während einer gewissen Zeit nicht angenommen werden, sind nicht mehr erreichbar.

Damit soll ein Ansatz gemacht werden, Untersuchungen von Salomaa über endliche Automaten mit variabler Struktur auf Turing-Maschinen auszudehnen. Zum anderen soll geprüft werden, ob es Möglichkeiten gibt, gewisse einfache Ausfallerscheinungen durch entsprechende "Selbstreparatur - Strategien" zu überspielen. Dazu werden die Fähigkeiten solcher Automaten bzgl. des Annehmens von formalen Sprachen untersucht und auch ein entsprechender Vergleich zwischen Turing-Maschinen mit variabler Struktur bei linearer Zeitbeschränkung vorgenommen.

H. Kopp (Saarbrücken)