MATHEMATISCHES FORSCHUNGSINSTITUT OBERWOLFACH

T a g u n g s b e r i c h t 17/1971

Formale Sprachen und Programmiersprachen

18.4. bis 24.4.1971

Die Tagung "Formale Sprachen und Programmiersprachen" fand unter Leitung von Herrn Prof.Dr.W.Händler (Erlangen), Herrn Prof.Dr.G.Hotz und Herrn Prof.Dr.H.Lanmaack (beide Saarbrücken) statt. Wie die beidne früheren Informatiktagungen im Forschungsinstitut Oberwolfach war auch diese Tagung zahlreich besucht (45 Teilnehmer).
Das Thema dieser Tagung war etwa weiter gesteckt als das der letzten; neben den formalen Sprachen wurde auch deren Hauptanwendungsgebiet behandelt: die Programmiersprachen. So war ein weiter Raum zwischen Theorie und Praxis gegeben, was sich auch in den Vorträgen widerspiegelte. Teilweise beschäftigten sie sich mit rein theoretischen Fragestellungen, wie asymptotische Dichte formaler Sprachen und Darstellung von Sprachen mit Hilfe gekoppelter Ersetzungen und kontextfreien Grammatiken mit Einschränkungen. Speziell wurden die Lindemayersprachen behandelt sowie ein Beweis angegeben, daß es zu einem festen $k \in N$ entscheidbar ist, ob eine Grammatik LR(k) Grammatik ist.
Weiter wurden allgemeine Schemata betrachtet, wie abstrakte Programme, verallgemeinerte Chomskysysteme und Ianovschemata. Andere Vorträge befaßten sich mit Problemen der Syntax und Semantik von Programmiersprachen (Darstellung der Syntax durch Fahnendiagramme, Definition der Semantik von Programmiersprachen, semantikerhaltende Abbildungen und formale Umformungen von Programmen) und mit einer neuen Theorie der Programmiersprachen. Schließlich wurde noch ein spezielles Problem einer im wesentlichen zu "Automath" äquivalenten Sprache behandelt (s. Tagungsbericht 31/1970).

Neben den Vorträgen über formale Sprachen und Programmiersprachen wurde noch je ein Vortrag aus der Komplexitäts- und Automatentheorie gehalten. Der erste befaßte sich mit optimalen Gödelisierungen, der zweite mit Schieberegisterrealisierungen. Zahlreiche Diskussionen und eine schon zur Tradition gewordene Wanderung gaben die Möglichkeit, sich näher kennenzulernen. Die Tagung verlief insgesamt sehr erfolgreich, wozu die gute Organisation des Forschungsinstituts wesentlich beigetragen hat.

Teilnehmer

K.Alber, Böblingen
H.Bauer, Konstanz
D.B.Benson, USA
J.Berstel, Straßburg
G.Beyer, Konstanz
K.H.Böhling, Bonn
W.Brauer, Birlinghoven
W.Carstengerdes, Wolfsburg
V.Claus, Saarbrücken
J.Engelfriet, Enschede
H.Feldmann, Hamburg
W.Händler, Erlangen
F.W.v.Henke, Birlinghoven
G.Hotz, Saarbrücken
K.Indermark, Birlinghoven
H.Jürgensen, Kiel
L.S.Jutting, Eindhoven
L.Kalmar, Szeged
H.Kamp, Münster
P.Kandzia, Saarbrücken
G.Kaufholz, Saarbrücken
W.Knödel, Stuttgart
I.Kupka, Hamburg
W.Kuich, Wien
K.Lagally, München
H.Langmaack, Saarbrücken
O.Mayer, Karlsruhe
W.Menzel, Karlsruhe
B.Monien, Hamburg
H.Noltemeier, Karlsruhe
U.Peters, Saarbrücken
T.Postelnicu, Bukarest
H.Pudlatz, Münster
J.Riguet, Paris
G.Rozenberg, Utrecht
H.J.Schneider, Berlin
C.-P.Schnorr, Saarbrücken
J.Spieß, Göttingen
Strosser, Straßburg
H.-J.Stoß, Konstanz
E.Valkema, Kiel
L.A.M.Verbeek, Delft
K.Zeller, Tübingen
B.Ziegler, Stuttgart
R.Zumkeller, Saarbrücken

Vortragsauszüge

K.ALBER: Zur Definition der Semantik von Programmiersprachen

Es wurde die von einer Gruppe des IBM-Labors Wien (K. Walk, P. Lucas, u.a.) entwickelte Methode zur formalen Definition von Programmiersprachen vorgetragen. Die Definition erfolgt durch Angabe einer abstrakten Maschine, die Programme der Sprache interpretiert. Sowohl die Programme als auch die Zustände der Maschine werden als Baumstrukturen dargestellt. Zu diesem Zweck wird zunächst ein Kalkül zur Beschreibung und Manipulation von Bäumen eingeführt.

D.B.BENSON: Semantic-preserving Translations

We review the notion of the interpretation of derivation systems, i.e., certain x-categories generated by rewriting systems. The semantics under the interpretation is a system of sets and functions between them.

It is possible to factor these interpretations through an algebraic theory in certain cases. Algebraic theories are shown to be a certain type of x-category. We generalize the notion of algebraic theory to $\underset{\sim}{\Omega}^*$ -theory and show that each interpretation to sets and functions can be factored through any appropriate $\underset{\sim}{\Omega}^*$ -theory. Therefore, in considering translations and their semantic-preserving properties, the interpretations of the translated derivation systems are interpretations into a common $\underset{\sim}{\Omega}^*$ -theory.

Thatcher's g^2sm maps, where the interpretations of the Σ-expressions are in an algebraic theory, are shown to be semantic-preserving under reasonable conditions relating the output functions to the interpretations. The syntax directed translation schemes of Aho & Ullman are shown to be semantic-preserving, again under reasonable conditions, which amount to the existence of a certain natural equivalence between a pair of contructed functors.

J.BERSTEL: Die asymptotische Dichte formaler Sprachen

Es sei X ein endliches Alphabet. Die asymptotische Dichte d(L) einer Sprache $L \subset X^*$ ist

$$(1) \quad d(L) = \lim_{n\to\infty} \frac{\mathrm{Card}\{f\in L: |f|\leq n\}}{\mathrm{Card}\{f\in X^*: |f|\leq n\}}$$

falls dieser Grenzwert existiert. Ist L eindeutig algebraisch (resp. rational) und existiert d(L), so ist die Dichte eine algebraische (rationale) Zahl. Für den Fall einer rationalen Sprache L wird eine obere Schranke für die Zahl der Häufungspunkte der Folge (1) als Funktion der Anzahl der Zustände des minimalen, L erkennenden Automaten angegeben. Dazu wird folgender Satz benutzt: Ist a(t) eine $\mathbb{N}$-rationale Reihe in der Variablen t, so ist für jeden Pol α auf dem Konvergenzkreis von a(t) die Zahl $\alpha/|\alpha|$ eine Einheitswurzel. Die $\mathbb{N}$-rationalen Reihen a(t) mit a(0) = 0 sind genau die erzeugenden Funktionen der rationalen Sprachen $L \subset XX^*$ über einem unendlichen Alphabet X. Es wurden Beziehungen zur Theorie akzeptabler Zahlenmengen aufgezeigt.

V.CLAUS: Formale Umformungen von Programmen

Programmschemata, bzw. Programme kann man als Morphismen freier X-Kategorien darstellen. Wir betrachten Programme, die keine Rücksprünge enthalten. Das Ziel ist es, solche Programme in andere Programme umzuformen, ohne daß die vom Programm realisierte Funktion geändert wird. Man kann leicht Regeln angeben, mit deren Hilfe man ALGOL-Programme in dieser Weise abändern kann, z.B. die Regel $\sin^2(x)+\cos^2(x)=1$, aber man kann bekanntlich kein endliches Regelsystem angeben, um zu einem vorgegebenen ALGOL-Programm alle Programme zu erzeugen, die dieselbe Funktion realisieren (Unvollständigkeit). Um eine möglichst allgemeine Theorie zu erhalten, nehmen wir nur an, daß von den in einem Programm vorkommenden Symbolen einige (z.B. Zuordnung, Nachfolgerfunktion Konstante 0) bekannt sind, während alle übrigen noch frei interpretiert werden können. Unter dieser Voraussetzung gibt es ein endliches Regelsystem R, so daß zwei Programm (ohne Rücksprünge) genau dann für jede Interpretation die gleiche Funktion realisieren, wenn sie bezüglich R ineinander transformiert werden können (Korrektheit und Vollständigkeit von R).

J.ENGELFRIET: Generalisierte Ianovschemata

Sei Σ ein Alphabet, das Funktionssymbole, Prädikatsymbole und negierte Prädikatsymbole enthält (einer Variablen).
Ein "abstraktes Programmschema" ist eine Untermenge von Σ^* (die Menge der Wörter über Σ). Wir beweisen einen Normalformsatz für abstrakte Programmschemata: $A \equiv B \leftrightarrow S(A) = S(B)$. ($S(A)$ ist die

Normalform von A; A ≡ B heißt, daß A und B unter jeder Interpretation dasselbe leisten). Aus diesem Satz folgen leicht (mit Hilfe der Theorie der formalen Sprachen) die wohlbekannten Resultate von Ianov über Programmschemata. Auch wird bewiesen, daß die Äquivalenz von Mengen von schematischen rekursiven Gleichungen einer Variablen entscheidbar ist, wenn man sich auf Gleichungen, die nicht "auf Prädikatsymbole enden", beschränkt.

F.W.v.HENKE: Abstrakte Programme als Grundlage einer allgemeinen Automatentheorie

Unter einem abstrakten Programm wird ein interpretierbares Programm in der üblichen Definition (s. Ianov u.a.) verstanden. Als System zweier aufeinander wirkender Strukturen entspricht es anderen Modellen von diskreten Systemen der Informationsumwandlung, wie dem "diskreten Prozessor" von Glushkov und Letichevskii. Den Ideen von Scott folgend, werden mit Hilfe von Familien abstrakter Programme (mit derselben Interpretation) die klassischen Automatentypen beschrieben. Unter anderem erreicht man so eine durchsichtigere Darstellung von AFDA, AFA, closed classes of Salloon automata, usw. und eine einfache Ableitung ihrer Eigenschaften. Es bieten sich über die klassische Theorie hinausgehende Verallgemeinerungen an. Zum Schluß wird auf ein allgemeines Symmetrieprinzip in Aussagen über erkannte Wortmengen und Transduktionen hingewiesen.

G.Hotz: Grundzüge einer Theorie der Programmiersprachen

Es werden verschiedene Klassen von Programmiersprachen streng definiert und bezüglich ihrer Leistungsfähigkeit unter Verwendung des Begriffes der Simmulation verglichen. Es wird plausibel gemacht, daß man mittels dieser Sprachen die existierenden problemorientierten Programmiersprachen beliebig gut approximieren kann.

L.S. JUTTING: A Normal Form Theorem in a λ-Calculus with Types

A λ-calculus with types will be presented which is essentially equivalent to the language Automath. This language was developed by N.G. de Bruijn in Eindhoven, Holland. Its purpose is to serve as a language for expressing large parts of mathematics in such a way that they may be checked by a computer.
A proof will be sketched that every well-formed formula in this calculus has a normal form. The proof is mainly due to L. Fleischhacker, Enschede and R. Nederpelt, Eindhoven.

L. KALMAR: "Fahnendiagramme" - ein anschauliches Hilfsmittel zur Angabe von Programmiersprachen

Ein geordnetes Tripel P = (S, B, h) mit elementfremden nichtleeren endlichen Mengen S und B (Menge der Grundsymbole, bzw. der syntaktischen Begriffe genannt) und einer Abbildung $h:B \xrightarrow{in} 2^{S^*}$ wird eine Programmiersprache genannt. Die Begriffe von Sprachen mit endlich vielen Zuständen, von kontextfreien Sprachen, usw. können ohne weiteres auf Programmiersprachen übertragen werden. Anschaulicher ist es jedoch, sie durch Fahnendiagramme - eine Art von syntaktischen Diagrammen - anzugeben. Ein Fahnendiagramm wird als ein geordnetes Siebentupel $F = (S, B, G, f_1, f_2, g_1, g_2)$ mit elementfremden nichtleeren endlichen Mengen S und B, mit einem endlichen gerichteten Graphen G und mit Abbildungen

$f_1: E_1 \to B$, $f_2: E_2 \to B$, $g_1: K_1 \to S$, $g_2: K_2 \overset{in}{\to} B$ definiert, wobei E_1 und E_2 elementfremde nichtleere Teilmengen der Eckenmenge, ferner K_1 und K_2 elementfremde Teilmengen der Kantenmenge von G sind und $K_1 \neq \emptyset$. Es wird die durch ein Fahnendiagramm angegebene Programmiersprache definiert und die Nützlichkeit der Fahnendiagramme zur Veranschaulichung von Syntaxen an Hand einiger Beispiele (ALGOL 60, ein Teil der Metasprache von ALGOL 68) erläutert.

P.KANDZIA: Verallgemeinerte Schieberegister-Realisierungen endlicher Automaten

Bezeichnet man mit W^n die Menge aller n-tupel über einem Wertebereich W, mit $(\phi_x)_{x\in X}$ eine Teilmenge von $Abb(W^n \to W^n)$, der Menge der Abbildungen von W^n in sich, und mit $(\phi_1^{(x)},\ldots,\phi_n^{(x)})$ die Komponenten-Darstellung von ϕ_x, dann läßt sich eine eigentliche Schieberegister-Familie über W beschreiben als Paar $\langle W^n, (\phi_x)_{x\in X}\rangle$ mit der Eigenschaft: $\exists\, m\in\{1,\ldots,n-1\}: \forall i\in\{1,\ldots,m\}: \phi_i^{(x)} = p_{ji}$; dabei ist p_{ji} die j_i-te Projektion aus $Abb(W^n \to W^1)$. Die Wahl der p_{ji} unterliegt einigen Einschränkungen.
Für $W = \{0,1\}$ wird untersucht, welche Familien von "Registern" neben den Schieberegister-Familien entstehen, wenn für die $\phi_i^{(x)}$, $i = 1,\ldots,m$, 1) beliebige Projektionen, 2) beliebige, evtl. negierte Projektionen, 3) beliebige lineare Funktionen - jeweils aus $Abb(W^n \to W^1)$ - zugelassen werden. Es wird eine übersichtliche

Charakterisierung sowohl der Schieberegister-Familien als auch aller unter 1), 2) und 3) eingeführten Familien angegeben. Aufgrund dieser Charakterisierung lassen sich günstige Algorithmen zur Realisierung abstrakter Automaten mit Hilfe der genannten Familien aufstellen.

H.LANGMAACK: Reguläre kanonische Systeme und LR(k)-Grammatiken

1965 hat D.E. Knuth die Klasse der von links nach rechts übersetzbaren Grammatiken G mit der Schranke $k \geqslant 0$ eingeführt ($G \in LR(k)$). Knuth hat gezeigt, daß man bei fester Schranke k in endlich vielen Schritten entscheiden kann, ob eine gegebene kontexfreie Grammatik G LR(k)-Grammatik ist. Mit Hilfe eines Resultats von J.R. Büchi (1964) über reguläre kanonische Systeme kann die Entscheidbarkeit sehr einfach bewiesen werden. Man wird auf diesen Beweis durch eine Charakterisierung der LR(k)-Grammatiken mittels der sog. k-Kellerklassen geführt, die sich nach Büchi als reguläre Mengen herausstellen.

O. MAYER: Über kontextfreie Grammatiken mit Einschränkungen bei der Anwendung der Produktionen

Es werden verschiedene Typen von Grammatiken betrachtet, die aus kontextfreien Grammatiken durch geeignete Einschränkungen bei der Anwendung der Produktionen hervorgehen, nämlich: Die "programmierten Grammatiken" (Rosenkrantz), die "Matrix-Grammatiken" (Abraham),

die "Zufalls-Kontext-Grammatiken" (van der Walt) und sogenannte "System-Grammatiken" (hier sind gewisse endliche Folgen von Produktionen - genannt "Regelsysteme" - vorgegeben, wobei ein Ableitungsschritt gerade darin besteht, daß alle Produktionen eines Regelsystems gleichzeitig angewendet werden). Die erwähnten Grammatiktypen werden bezüglich ihrer Erzeugungskraft untereinander und mit den Grammatiken der Chomsky-Hierarchie verglichen. Dadurch wird u.a. ein von van der Walt gestelltes Problem gelöst; weiter erhält man zwei neue Charakterisierungen der Familie der rekursiv aufzählbaren Sprachen. Jede rekursiv aufzählbare Sprache läßt sich erzeugen durch eine System-Grammatik und eine Zufalls-Kontext-Grammatik. Außerdem wird für System-Grammatiken und Zufalls-Kontext-Grammatiken je eine Normal form angegeben.

G.ROZENBERG: Some Results related to Lindemayer Languages

L-languages were introduced by A. Lindemayer for description of development of some biological organisms.

Roughly speaking L-systems differ from phrase structure grammars of Chomsky in the way they generate languages:

i rules are applied simultaneously to all occurrances of all letters in a string,

ii there is no distinction between terminal and non-terminal symbols,

iii an axiom can be a non-empty word.

A natural from both biological and formal point of view generalization of L-languages is to include in one generating system several sets of rules and to allow to apply only one of these, but arbitrary, at a time. These systems are called T-L-systems.

In our talk we present properties of o-L-systems and languages and T-o-L-systems and languages, where "o" stands for "context-free" substitution.

H.J.SCHNEIDER: Listenmanipulation mit verallgemeinerten Chomskysystemen

Es gibt verschiedene Ansätze zur formalen Beschreibung von mehrdimensionalen Zeichenanordnungen, darunter auch einige Verallgemeinerungen der Chomsky-Systeme (z.B. Dacey, Shaw). Bei der Verwendung von Relationalsystemen zur Beschreibung solcher n-Diagramme (n verschiedene Richtungen der "Konkatenation", wobei jedoch nur partielle Ordnung vorausgesetzt wird) stellt sich die Frage, wie sich die Verankerung eines beim Ableitungsprozeß ersetzten Teildiagramms im generierten Diagramm widerspiegelt. Es ergibt sich eine enge Beziehung zu den Operationen bei verketteten Listen.

C.P.SCHNORR: Optimale Gödelisierungen

Sei P^k (R^k) die Menge der k-stelligen partiell rekursiven (total rekursiven) Funktionen. $g:N \to N$ heiße linear beschränkt, wenn $\overline{\lim_n} g(n)/n<\infty$. Dann heißt $\phi \varepsilon P^2$ eine optimale Gödelisierung, wenn

$$\bigwedge_{\psi \varepsilon P^2} \bigvee_{g \varepsilon R^1} (\psi = \phi\,(g \times id))$$

linear beschränkt.

Grundlegende Sätze der rekursiven Funktionstheorie, wie das Rekursionstheorem und der Isomorphiesatz für Gödelisierungen, kann man für optimale Gödelisierungen verschärfen. Sei

$$K_\phi(h) = \min\{i \mid h = \phi_i\}$$

die Programmkomplexität von $h \varepsilon P^1$ bzgl. $\phi \varepsilon P^2$. (Es bezeichnet $\phi_i = \phi(i,*)$). Dann gibt es zu einer optimalen Gödelisierung ϕ und einer Funktion $\bar{\phi} \varepsilon P^2$ stets ein $c \varepsilon N$ mit $K_\phi(h)<c\, K_{\bar{\phi}}(h)$ $(h \varepsilon P^1)$. D.h. optimale Gödelisierungen haben für alle $h\ P^1$ im wesentlichen die kleinsten Gödelnummern. Es wird ein Zusammenhang zwischen $K_\phi(h)$ und der relativen Häufigkeit des Auftretens von Gödelnummern zu h in einer optimalen Gödelisierung ϕ hergestellt. Es wird ein neuer Beweis eines Satzes von M. Blum gegeben, der inhaltlich aussagt, daß die kleinsten Rechenprogramme zu einer Funktion nicht immer die schnellsten sind. Ferner wird auf Anwendung des Begriffs der optimalen Gödelisierung für die Theorie der Programmierung, insbesondere Compilerbau hingewiesen.

R. ZUMKELLER: Klassifikation der durch gekoppelte Ersetzung erzeugbaren Sprachen

Es wird eine Technik der gekoppelten Ersetzung zur Erzeugung von Sprachen definiert; das mathematische Werkzeug dieser Technik ist eine endlich erzeugte Kategorie von Abbildungen der Form

$$f: T\, x_1 T\, x_2 T \ldots T\, x_m T \longrightarrow T\, y_1 T \ldots T\, y_n T$$

$$f(u_1 x_1 u_2 \ldots u_m x_m u_{m+1}) = u_1 w_1 u_2 \ldots u_m w_m u_{m+1}$$

wobei

$$x_i, y_i \in X \; (X \cap T = \emptyset),\; u_i \in T^*,\; w_i \in (X \cup T)^*;\; m \geq 1;, n \geq 0.$$

Der Ableitungsbegriff für derartige Grammatiken (bzw. Sprachformen) wird auf kanonische Weise durch freie X-Kategorien vermittelt. Die Klassifikation der Sprachformen nach den erzeugenden Abbildungen, sowie die Wahl von Verankerungsmengen liefert eine Klasseneinteilung von Sprachen, die die Chomsky-Hierarchie echt enthält. Die Definition von Operationen zwischen Sprachformen ermöglicht eine weitere Verfeinerung dieser Klassifikation. Einige Beispiele demonstrieren die Praktikabilität der gekoppelten Ersetzung zur Generierung kontext-sensitiver Sprachen.

G. Kaufholz
Saarbrücken

Gefördert durch
DFG Deutsche Forschungsgemeinschaft