Manfred Droste, Yuri Gurevich (editors):

Semantics of Programming Languages and Algebra

Dagstuhl-Seminar-Report; 65
07.06.-11.06.93 (9323)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloss Dagstuhl
Universität des Saarlandes
Postfach 1150
D-66041 Saarbrücken, Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: office@dag.uni-sb.de

Semantics of Programming Languages and Algebra

Organized by:

MANFRED DROSTE (Universität Essen)
YURI GUREVICH (University of Michigan)

June 6-11, 1993

The theme of the conference was the interplay between various approaches to the semantics of programming languages and the mathematical areas of logic (especially model theory) and algebra. The topics of the 26 talks included

1. questions of definability in first-order logic and its various extensions (fixed-point logic, generalized quantifiers, etc.)
2. evolving algebras and various applications of them,
3. denotational semantics and domain theory, and
4. generalizations of automata theory and network theory.

There were many fruitful interactions and collaboration between specialists from different areas as well as wide-ranging and vigorous discussion sessions. The conference was attended by 33 participants from 11 countries.

On behalf of all participants, the organizers would like to thank the staff of Schloß Dagstuhl for providing an excellent environment for the conference.

Abstracts

Bakery Algorithms

URI ABRAHAM MENACHEM MAGIDOR

Lamport's Bakery Algorithm and some variants are studied (these are all mutual exclusion protocols). The main contribution is a protocol that is similar to the Bakery Algorithm but only uses bounded values. Another variant uses local clocks to coordinate the protocol's execution, and conditions on the clocks are determined that suffice for the protocol's correctness.

In what sense is the bounded algorithm similar to the unbounded Bakery Algorithm? Using the notion of definability in models of system executions, a new property, the "Bakery Definability property", which is a strong form of the First Come First Serve (FCFS) property is defined. It is shown that Lamport's Bakery Algorithm and its variants have this property.

Expressiveness of Datalog

FOTO AFRATI STAVROS COSMADAKIS

Datalog is the language of negation-free, function-free Horn clauses; it mayviewed as a fragment of fixpoint logic. The computational and expressive capabilities of Datalog have been the object of considerable research recently, because it provides a simplified model of the data sublanguage of logic programming.

Any query expressible in Datalog can be expressed in polynomial time. If the database includes a total order of the domain and negation in the input relations is allowed, then Datalog expresses all polynomial time queries. However, without this extension, every query expressible in Datalog satisfies certain well-known monotonicity properties, which exclude even some very simple queries; an example is the query which asks if the domain has even cardinality.

In this work, we give monotonic queries computable in polynomial time, which are not expressible in Datalog. We show that a query which asks if a given directed graph has a path with length a perfect square, is not expressible in Datalog. We develop new tools to prove the results. The known proof techniques for proving non-expresibility in Datalog make use of the fact that Datalog queries can be expressed by infinitary logic with finitely many variables. The perfect square query , however is expressible in this logic. To prove that perfect square query is not expressible in Datalog, we show that such queries satisfy a "pumping lemma";this result exploits the "regular" structure of the infinitary formulas which express Datalog programs.

We, also, give NC computable Datalog queries that cannot be expressed by any linear Datalog program. To prove the latter result, we show that queries expressible in linear Datalog satisfy a more restricted pumping lemma.

A Unifying View of Imperative and Functional Languages

EGIDIO ASTESIANO
(joint work with ELENA ZUCCA)

D–oids are a model for systems of dynamic entities, based on the following features:

- States are modelled by structures (algebras, usually) in some category; we assume that every structure (called instant structure) has an underlying support consisting in an S–sorted family of sets;
- Variations of state are modelled by dynamic operations (also called methods) calls, whose semantics is the most typical features of the approach: the effect of a call is a transformation of the source state into a new state, consisting in a map between the underlying carriers; this map is called 'tracking map', since intuitively it keeps track of the identities of the evolving dynamic entities.

An appropriate rather natural notion of morphism is given such that the d–oids over a category of instant structures are a category.

The emphasis of this talk is on a recent new result, consisting in the construction of the d–oid of dynamic terms over a dynamic signature and a family of variables. In the case that the instant structures are provided of a free construction of terms over families of generators, then the d–oid of dynamic terms is a free construction. A most notable feature of the construction is the unique representation ;in other words a canonical representation is provided (no identification) as it happens in the classical static case. It can be shown that usual imperative method expressions can be translated into the language of dynamic terms, which then may be seen as the basis of a kernel language for defining methods with the typical flavour of an applicative language.

Designing an Abstract Machine and Proving Compiler Correctness for Constraint Logic Programming

CHRISTOPH BEIERLE

The constraint logic programming scheme CLP(X) provides a means of extending logic programming by constraints over a domain X. Given a constraint solver for X, one obtains a constraint solver for the freely generated terms over X. By viewing the required term unification part as interacting with the actual constraint solver over X, it is argued that a general scheme for an extension of Warren's Abstract Machine for executing Prolog can be designed for CLP(X).

The thus obtained abstract machine WAM(X) not only uses the WAM's original AND/OR structure, but also most of the WAM's term representation can be kept and is extended in an orthogonal way. This permits us to introduce modular extensions of Börger and Rosenzweig's formal derivation of the WAM and their correctness proof which uses the concept of evolving algebras. As in the original WAM case, our correctness proof also covers the sophisticated WAM optimizations like environment trimming, last call optimization, switching, etc. which are still present in our WAM(X) extension. Our development is generic in the sense that it allows applications to different constraint formalisms like Prolog III, the type constraints of PROTOS-L (Beierle and Börger 92), or the arithmetic constraints of CLP(R) (Börger and Salamone 93).

Semantics for Linear Logic

ANDREAS BLASS

This survey talk began with an introduction to linear logic, emphasizing its semantical origins in coherence spaces. After a short summary of Lorenzen's dialog interpretation of intuitionistic logic, I described game semantics for linear logic and stated its soundness and completeness theorems. Finally, I described the modifications of game semantics introduced by Abramsky and Jagadeesan to obtain their full completeness theorem for multiplicative linear logic with the mix rule. (Time did not permit a planned discussion of de Paiva's "relational" semantics or the observation that there seem to be two rather different semantical intuitions attached to the "of course" modality.)

Occam: Specification and Compiler Correctness

EGON BÖRGER DEAN ROSENZWEIG

Aiming at a transparent mathematical correctness proof for a general compilation scheme of Occam programs on the Transputer, we develop a simple and natural abstract machine for (formalization of) operational semantics of Occam, at several levels of abstraction. Starting from a truly concurrent model of the language, we refine communication, parallelism and alternation to an abstract notion of processor, running a queue of processes, still close to the abstraction level of atomic Occam commands. The specification is effected within the framework of evolving algebras of Gurevich, relying on the theory of concurrency developed recently within that framework by Glavan and Rosenzweig. The model lends itself naturally to refinement down to the abstraction level of Transputer Instruction Set.

Efficient Reduction of NP–Problems to Satisfiability by Interpretations

ELIAS DAHLHAUS NATHAN SINCLAIR

We deal with the problem to find reductions to satisfiability given a logical description. Our aim is to solve NP-problems with the help of an efficient satisfiability solver. We assume that NP-problems are expressed by existential second order formulas (see Fagin). Using the ideas of Lovasz and Gacs, we can find reductions to satisfiability efficiently. These reductions are not only polynomial time but have nice logic properties. Moreover, we can take care that the size of the corresponding satisfiability problem can be minimized.

A Kleene Theorem for Recognizable Languages over Concurrency Monoids

MANFRED DROSTE

We investigate an operational model for concurrent systems, automata with concurrency relations. These are classical finite automata A, endowed with a system of symmetric binary relations indicating when two events in a particular state of A can occur independently of each other. These concurrency relations are assumed to depend locally, but not globally of each other. On the set of all finite computation sequences of A they induce, similarly as in trace theory, an equivalence, and via composition of computation sequences we obtain a monoid $M(A)$.

In previous work, we investigated the relationship between these automata and Petri nets resp. Scott-domains. Here, we obtain, under suitable assumptions on A, a Kleene-type characterization of all recognizable languages over concurrency monoids $M(A)$: These are precisely the languages which can be constructed from finite languages of $M(A)$ by finitely many applications of the operations union, product, concurrent iteration. This contains Kleene's theorem as a special case. The proof uses methods of Ochmanski and Cori & Metivier from trace theory, Kleene's theorem and the combinatorics/algebra of concurrent algebra.

Inductive Definability with Counting

ERICH GRÄDEL
(joint work with MARTIN OTTO)

We study the properties and the expressive power of logical languages that include both a mechanism for inductive definitions and the ability to count. The most important of these languages is fixpoint logic with counting terms, denoted (FP + C).

Although it is known that these languages do not quite capture all polynomial-time computable queries, they provide a natural and important level of expressiveness. Besides (FP + C) we also investigate several other logical and algorithmical versions of inductive definability with counting and show that they all have the same expressive power. In particular we consider:

Datalog with counting. By adding counting terms to Datalog, we obtain (Datalog + C) which we show to be equivalent to (FP +C). In particular, (Datalog + C) is closed under negation and therefore, in the presence of counting, the usual extensions of Datalog, notably Stratified Datalog are equivalent to Datalog. This is not at all the case without counting.

A functional logic It is known that the usual schemes, that define — over **N** — the recursive functions, characterize, when interpreted over ordered finite structures, precisely the functions computable in polynomial time. We define a class of global functions, defined by a similar scheme, which has precisely the power of (FP + C).

A machine theoretic characterization. An algorithmic definition of (FP + C) is provided by a "relational machine". The queries computed in polynomial time by these machines are precisely the queries in (FP + C), those computed in polynomial space characterize partial fixpoint logic with counting (PFP + C).

We also investigate the relationship of fixpoint logic and infinitary logic in the presence of counting. In particular, we show that the types can be uniformly ordered by a formula in (FP +C). As a consequence, there exists for every structure an (ordered) arithmetical invariant, which is uniformly (FP + C)-definable and which characterizes the original structure up to equivalence. Furthermore the (FC + C) definable properties of the original structures exactly correspond to the PTIME-properties of the associated invariants.

Abiteboul and Vianu recently proved that partial fixpoint logic collapses to fixpoint logic if and only if PTIME= PSPACE. We show that the analogous result in the presence of counting is also true: PTIME = PSPACE iff (FP + C) = (PFP + C). On the other side, Abiteboul and Vianu also proved that already a collapse of $PFP|_P$ to fixpoint logic is equivalent to PTIME = PSPACE. (The logic $PFP|_P$ is partial fixpoint logic restricted to PFP operators that close after at most polynomially many iterations on every structure.) In the presence of counting we can prove instead that (FP + C) = ($PFP|_P$ + C).

Distributed Evolving Algebras

YURI GUREVICH

This is a joint work with my student Jim Huggins. In order to test the expressive power of DE algebras, we construct a DE algebra for Kermit, the well known communication protocol.

Lower Bag Domains

REINHOLD HECKMANN

In the classical power domain constructions, multiplicities of results are not taken into account. Together with the order structure, this implies that certain finite sets are identified. In contrast, we propose to count results, thus considering bags instead of sets. Since the lower construction is the simplest of the three classical power constructions, we investigate its non-idempotent analogue, the *lower bag domain construction.*

According to the general theory of power constructions, it can be introduced in two ways: as 'initial' construction $\mathcal{L}_i^*$, where $\mathcal{L}_i^* X$ is the free *lower monoid* over X in DCPO, or as 'final' construction $\mathcal{L}_f^*$, where $\mathcal{L}_f^* X = [[X \to \mathbf{N}_0^\infty] \xrightarrow{add} \mathbf{N}_0^\infty]$ is the dcpo of continuous additive functions from $[X \to \mathbf{N}_0^\infty]$ to $\mathbf{N}_0^\infty$. The latter can be simplified to $\mathcal{L}_m^* X = [\Omega X \xrightarrow{mod} \mathbf{N}_0^\infty]$, the dcpo of continuous modular functions from the lattice ΩX of opens of X to $\mathbf{N}_0^\infty$.

We show that for sober dcpo's X, each element of $\mathcal{L}_m^* X$ can be represented as a (possibly infinite) sum of singletons. For algebraic X, $\mathcal{L}_m^* X$ is again algebraic; its basis consists of finite sums of singletons from the basis of X. Equivalently, it consists of all finite bags of finite elements of X ordered by an analogue of the lower order on sets. In contrast to the set case, all finite bags can be distinguished. From this description, one concludes that $\mathcal{L}_m^* X$ and $\mathcal{L}_i^* X$ are isomorphic for algebraic X and hence for continuous X.

The construction $\mathcal{L}_i^* = \mathcal{L}_m^*$ does not preserve the property to be a Scott domain, nor to be bifinite. This negative result can be seen at a small finite example. Moreover, we can show that there is no power domain construction with characteristic semiring $\mathbf{N}_0^\infty$ that preserves these domain classes.

Defining the Behaviour of a System

WILFRID HODGES

We ask how far one can give a mathematical sense to the statement that two specifications (possibly in different languages) specify the 'same behaviour' of a system. It is possible to give a uniform denotational semantics for the standard algebraic and set-theoretic specification languages; this is a theorem, and its proof shows how to give translations between these various languages. Two problems are noted.

(a) Specifications in set-theoretical languages (notably Z) are incompletely formalised, and to give their semantics in the style discussed above, we need to understand the author's intentions. This gap seems not to be serious.

(b) More seriously, the notion of 'behaviour' used above is crude; in particular it does not include the notion of behaviour being expressed in time as a response to earlier input. We can hope to extend the theorem above to more refined notions of behaviour (as for example in CCS), but the result is likely to be less canonical.

Representation of Domains by Residuum Systems

DIETRICH KUSKE

Several models of concurrent systems and the domains generated by them have been investigated. Monoids of computations introduced in this talk can be used to represent all domains in a uniform way. As a generalization of classical residuum operations (on automata, on term rewriting systems, on concurrent transition systems) we endow the monoids of computations by such an operation which gives the minimal amount of computation necessary to obtain the result of one computation after performing another one. These extended monoids of computations are called residuum systems. By additional axioms, the classes of those residuum systems which generate L–domains and Scott–domains, respectively, are determined. The classes of residuum systems generating coherent Scott–domains, bifinite domains, dI–domains and several other distributive domains can be described as well.

Additionally it is possible to define morphisms between residuum systems to get categories which are equivalent to wellknown categories of domains with continuous or with stable functions as morphisms.

Logical Reducibility, Lindstrom Quantifiers and Oracles

J.A. MAKOWSKY
(joint work with Y.B. PNUELI)

We describe a general way of building logics with Lindstrom quantifiers which capture complexity classes based on oracle Turing machines. We first introduce a notion of reducibility based on extensions of first order logic rather than resource bounded Turing machines or first order logic. Its main property is that if a logic $\mathcal{L}$ captures a complexity class D, then a problem A is D reducible to a problem B iff A is $\mathcal{L}$ reducible B. This allows us to build logics capturing complexity classes incrementally, rather than by resorting to first order reductions. Our approach is sensitive to the oracle computation model. Our results hold for the unbounded model introduced by Buss in support of his 'relativization thesis'. They do not hold for the nondeterministic and alternating oracle computations studied by Ladner and Lynch, Simon and Ruzzo, Simon and Tompa. Our results generalize and extend previous results of Stewart and Makowsky and Pnueli.

Eventual Periodicity and Monadic Least Fixed Points

GREGORY L. MCCOLM

This talk concerned an application of the monotonicity of positive elementary formulas. The important point was that this monotonicity could be used in the studying expressibility of (fragments of) Least Fixed Point logic. A system $\varphi_0(\bar{x}_0, \bar{S}), \ldots, \varphi_\nu(\bar{x}_\nu, \bar{S})$ of $\bar{S}$-positive formulas is *operative* if, for each i, $\bar{x}_i$ and S_i are of the same arity. Usually, we use such systems to get "least fixed points" as follows: $\varphi_i^{n+1}(\bar{x}) \equiv \varphi_i(\bar{x}, \varphi_0^n, \ldots, \varphi_\nu^n)$ (where $\varphi_i^{-1} = \emptyset$), and then set the least fixed point to be $\varphi_i^\infty = \bigcup_n \varphi_i^n$. Given some relations $\bar{R} = R_0, \ldots, R_\nu$, we want the unique system of relations $\bar{X} = X_0, \ldots, X_\nu$ such that:

1. For each i, $\bar{x}$, $X_i(\bar{x}) \equiv \varphi(\bar{x}, \bar{X})$, and,
2. for each i, $R_i \subseteq X_i$, and,
3. for any $\bar{X}'$ also satisfying (1) and (2), $X_i \subseteq X_i'$ for each i.

Such a system $\bar{X}$ can be obtained by induction on the system $\bar{\varphi}$, only starting at $R_0, \ldots, R_\nu$ instead of $\emptyset, \ldots, \emptyset$.

We use this method of building fixed points from someplace other than the beginning to impose restrictions on fixed points, and to prove that least fixed points on some structures look like (are r-Fraïssé equivalent to) least fixed points on other structures. We apply these techniques to the problem of determining whether given boolean LFP-definable queries are 1-dimensional, i.e., whether they can be defined from systems $\varphi_i(x_i, \bar{S})$, $i = 0, \ldots, \nu$, where each S_i is unary.

Most of the material in this talk appeared in *Eventual Periodicity and One-Dimensional Queries*, Notre Dame J. of Formal Logic 33:2 (1992), 273-290.

Decidability Results in Automata and Process Theory

FARON MOLLER

In this lecture we survey decidability results for various notions of equivalence over three classes of processes: regular (finite-state) processes, context-free processes, and basic parallel processes. In particular, we demonstrate the decidability of bisimulation equivalence over all three classes, using three different techniques: for regular processes the result follows by a brute force search for a bisimulation relation over the finite-state agents; for context-free processes we use a finite characterisation of infinite bisimulation relations; and for basic parallel processes we use a tableau-based decision procedure. We also demonstrate the undecidability of simulation and language equivalences

for basic parallel processes; the proof is a modification by Y.Hirshfeld of a proof by P.Jancar of the undecidability of bisimulation equivalence of labelled Petri nets.

Action Semantics

PETER D. MOSSES

Action Semantics is a framework that combines some of the best features of denotational semantics, operational semantics, and algebraic specifications. The combinator-based action notation provided by the framework appears to solve the modularity problems of previous semantic description techniques, and experiments have shown that action semantic descriptions scale up smoothly from simple pedagogical examples to practical programming languages.

In this talk, action semantics is introduced by showing how it can be used to describe a fragment of Standard ML. The intended interpretation of the required part of action notation is explained. The syntax of the described language is then extended with the main constructs of Concurrent ML, and it is shown how the semantic description can be extended to the new constructs – without changing the original semantic equations at all! The talk concludes by comparing action semantics with evolving algebras, and considering the possibility of hybrids involving features of both frameworks.

Universality and Powerdomains

KAY J. NÜSSLER

The class of all L–domains is not closed under the Plotkin–powerdomain–construction. Hence we are looking for an ordertheoretic property that gives a characterization of all L–domains $(D, \leq)$ whose Plotkin–Powerdomain $P[D]$ is again an L–domain. The demand that $(D, \leq)$ does not contain three special finite posets is equivalent to the condition that the powerdomain is an L–domain.

In the theory of denotational semantics of programming languages, various authors established the existance of special kinds of universal objects. It is a question of Plotkin whether there exists a Scott–domain $(D, \leq)$ whose Plotkin–powerdomain $P[D]$ is universal for the category of bifinite domains (SFP–domains), i.e. for every bifinite domain $(E, \leq)$ there exists an embedding projection pair of $(E, \leq)$ into $P[D]$. We showed that this question can be answered negatively and gave a finite counterexample. Moreover we find out that even the class of bifinite domains does not contain an object whose Plotkin–powerdomain is universal for the category of all finite domains with embedding projection pairs.

On Simple Lindström Extensions

MARTIN OTTO

In abstract model theory Lindström quantifiers provide a uniform formalism for the treatment of extensions of logics:
A new structural property – as given by an as yet not definable class of structures – is made available in the extension by allowing to assert this property of structures which are definably interpreted within the domain of discourse.

In finite model theory one looks for matches between logical expressibility (logical complexity) and computational complexity. While this approach has been very successful within the domain of linearly ordered structures, there are no definite results for even the most important complexity classes below NPTime. Most notably, fixed-point logic (FP) captures PTime on ordered structures, but is far too weak in the general case. No extension which exactly reaches up to PTime has been found, and in fact it has even been conjectured that no logic captures PTime in the general case (Gurevich, 88).

Hence the systematic investigation of extensions of FP is of interest. The adjunction of all PTime Lindström quantifiers corresponds to a trivial shift of the problem: the resulting 'logic' is trivially complete for PTime, but syntactically acceptable as a logic only if the class of all PTime structural properties is recursively enumerable. This latter condition is equivalent with the existence of a logic for PTime.

With a view to the idea of a closer or lower-level match between algorithmic analysis and logical, i.e. descriptive modelling, one might therefore focus on the adjunction of simple, low complexity or fundamental extra structural properties to fixed-point logic and try to leave the modelling of recursion to fixed-point generation. We here show that – for one conception of 'simplicity' expressed in terms of the existence of simple invariants – this attempt must fail. This conception of simplicity encorporates in particular the most liberal formalisation of cardinality properties, which also provide the key example for this analyis: even the adjunction of *all* cardinality based Lindström quantifiers to fixed-point logic does not reach up to the expressive power of the natural (and non-Lindström type) extension of fixed-point logic, which allows to assess and process cardinalities in a recursive way (see Grädel/Otto, CSL 92, for a detailed overview of this logic). These results extend to other properties governed by simple invariants, like monadic properties, or properties of equivalence relations.

In the present framework we naturally deal with infinite families of Lindström quantifiers across all arities. Known negative results in this context – based on different techniques and prerequisites – focus on finite Lindström extensions, cf. Hella, LICS 92, and Kolaitis/Väänänen LICS 92.

Evolving Algebras as a Foundation for Semantics–Based Programming Environments

ARND PÖTZSCH-HEFFTER

Semantics–based programming environments should enable the user to annotate programs, to specify program analyses, and to prove annotations and other program properties correct. Such environments heavily rely on an appropriate specification of the underlying programming language.

The talk introduced attributed occurrence structured — a formalism to specify the context-dependent syntax and static semantics of programming languages in terms of first–order structures — and proposed to use evolving algebras together with attributed occurrence structures (AOS for short) as a specification framework for semantics–based programming environments. Even though the use of AOS's is not necessary from a theoretical point of view, they are essential when it comes to the construction of realistic systems. AOS's are a generalization of attribute grammars; in particular, they allow to specify classes of graph structures.

We showed how the first–order language given by an EA+AOS–specification can be used to define program annotations and data flow analyses in terms of temporal logic formulas. The semantics of these formulas coincide with the runs of the EA semantics. We defined the weakest precondition transformer associated with an EA and gave the corresponding basic rule to prove invariance properties of programs.

Call-by-Value and Nondeterminism

KURT SIEBER

In order to study nondeterminism in a very 'pure' setting, we consider a simply typed purely functional call-by-value language with nondeterministic choice operators at all types. For this language we define a generic denotational semantics in terms of an arbitrary powerdomain construction. The three 'classical' instances of this generic semantics, obtained by the *Hoare-*, *Smyth-* and *Plotkin*-powerdomain, correspond to three different notions of observable behavior of nondeterministic programs, which we call *partial correctness* behavior, *total correctness* behavior and *overall* behavoir.

For all three powerdomain semantics we investigate the question of full abstraction. For the Hoare-powerdomain we obtain full abstraction by more or less traditional methods. For the Smyth-powerdomain full abstraction fails irreparably. Full abstraction also fails for the Plotkin-powerdomain, but here we can repair something: By adding an **exists**-operator, which can test all paths of a (nondeterministic) computation in parallel, we obtain full abstraction for the sublanguage, in which all

functions have only ground type results. But for the full language, even this highly parallel **exists**-operator is not sufficient to obtain full abstraction, and there is some evidence that this cannot be repaired by adding further operators to the language.

Dataflow Algebra: A Tale of Two Monads

EUGENE W. STARK

Dataflow networks constitute a paradigm for concurrent computation in which a collection of concurrently and asynchronously executing processes communicate by passing messages containing data values over FIFO communication channels. In previous work (reported in the 1992 LICS conference), I introduced a formal calculus, called "dataflow calculus," for dataflow networks, defined a bisimulation-based notion of network equivalence, and obtained a sound and complete axiomatization of the resulting equational theory. In this lecture, I consider the problem of defining a class of "dataflow algebras", having sufficient structure to provide an interpretation of the syntactic constructs (network terms and computation terms) of dataflow calculus. I provide motivation for the idea that dataflow algebras have simultaneously the structure of algebra for two related monads: one monad P that captures the notion of parallel composition of networks, and a second monad F, which captures the notion of forming feedback loops in terms of the "action" on networks of a a certain simple class of continuous functions. The two structures are related by a distributive law of feedback over parallel composition.

Extending First–Order Logic Using NP Operators

IAIN STEWART

We investigate why similar extensions of first-order logic using operators (that is, generalized quantifiers) corresponding to NP-complete decision problems apparently differ in expressibility: the logics capture either NP or L(NP). It had been conjectured that the complexity class captured is NP iff the operator is monotone. We show that this conjecture is false. However, we provide evidence supporting a revised conjecture involving finite variations of monotone problems.

On Types in Computer–Algebra–Systems Dynamic Recursive Types in Maple

BURKHARD WALD

From our point of view the computer–algebra–system Maple is just a programming language. Maple has an extensive type–system, which we call syntactical type-system, because Maple–datas are mathematical terms and hence syntactical things. The talk introduces a possibility to assign a type to a symbol, which gives the symbol a mathematical meaning. For this semantic propose we define new types.

EXAMPLE. There is a syntactic type integer. For type-assignment we define a type INT by recursion (integer are INT, sums of INT are INT and so on). In addition, if we make a type-assignment of INT to a symbol, this symbol gets type INT. Hence the set of terms which are of type INT changes dynamicly. At last we compare our approach with the new assume-functionality of release 2 of Maple V .

Dagstuhl-Seminar 9323: List of Participants 25.06.93

Uri **Abraham**
Ben Gurion University
Dept. of Mathematics and Computer Science
Beer Sheva
Israel
abraham@indigo.bgu.ac.il

Foto **Afrati**
National Technical University of Athens
Dept. of Elec. Eng. and Comp. Science
9 Heroon Politechnion
15773 Zographon
Greece
afrati@theseas.ntua.gr
tel.: +30-1-2232097 / 30-1-7757401

Egidio **Astesiano**
Dipartimento di Informatica e
Scienze dell'Informazione
DISI
Viale Benedetto XV
I-16132 Genova
Italy
astes@disi.unige.it
tel.: +39-10-353-80 31

Andreas **Baudisch**
Freie Universität Berlin
Fachbereich Mathematik
Arnimallee 3
D-14195 Berlin
Germany
baudisch@math.fu-berlin.de

Christoph **Beierle**
IBM Deutschland GmbH
Wissenschaftliches Zentrum Heidelberg
Institut für Wissensbasierte Systeme
Postfach 10 30 68
W-6900 Heidelberg
Germany
beierle@vnet.ibm.com
tel.: +49-6221-594393

Andreas **Blass**
University of Michigan
Dept. of Mathematics
Ann Arbor MI 48109-1003
USA
ablass@umich.edu
tel.: +1-313-763-1183

Egon **Börger**
Universita' di Pisa
Dip. di Informatica
Corso Italia 40
I-56100 Pisa
Italy
boerger@di.unipi.it
tel.: +39-50-51 02 30

Elias **Dahlhaus**
University of Sydney
Basser Dept. of Computer Science
New-South -Wales 2006
Australia
dahlhaus@cs.su.oz.au
tel.: +61-2-692-3756

Hans **Dobbertin**
Bundesamt für Sicherheit
Informationstechnik
Godesberger Allee 183
W-5300 Bonn 1
Germany
tel.: +49-228-768331 (privat)

Manfred **Droste**
FB Mathematik und Informatik
Postfach 10 37 64
W-4300 Essen 1
Germany
matb30@vm.hrz.uni-essen.de
tel.: +49-201-183 2505

H.-D. **Ebbinghaus**
Universität Freiburg
Abt. für Mathematische Logik und
Grundlagen der Mathematik
Albertstr. 23b
D-79104 Freiburg
Germany
hde@sun1.ruf.uni-freiburg.de
tel.: +49-761-203-23 84

Jörg **Flum**
Albert-Ludwig-Uni. Freiburg
Abt. für Mathematische Logik und
Grundlagen der Mathematik
Albertstr. 23b
D-79104 Freiburg
Germany
flum@sun1.ruf.uni-freiburg.de
tel.: +49-761-203-23 82

Erich **Grädel**
Lehrgebiet Math. Grundlagen
der Informatik
Mathematisches Institut
Ahornstr. 55
D-52074 Aachen
Germany
graedel@mjoli.informatik.rwth-aachen.de
tel.: +49-241-802-1080

Irene **Guessarian**
Université de Paris VI
Laboratoire Informatique
Théorique et Programmation
4 Place Jussieu
F-75252 Paris Cedex 5
France
ig@litp.ibp.fr
tel.: +33-1-4427-7029

Yuri **Gurevich**
University of Michigan
Dept. of Electrical Engineering and
Computer Science
Ann Arbor MI 48109-2122
USA
gurevich@eecs.umich.edu
tel.: +1-313-7 63 45 26

Reinhold **Heckmann**
Universität des Saarlandes
FB 14 - Informatik
Postfach 1150
D-66041 Saarbrücken
Germany
heckmann@cs.uni-sb.de
tel.: +49-681-302-24 54

Christian **Herrmann**
Technische Hochschule Darmstadt
Fachbereich Mathematik
Schlossgartenstr. 7
W-6100 Darmstadt
Germany
herrmann@mathematik.th-darmstadt.de

Wilfrid **Hodges**
Queen Mary and Westfield College
School of Mathematical Sciences
Mile End Road
London E1 4NS
Great Britain
w.hodges@qmw.ac.uk

Dietrich **Kuske**
Universität GHS Essen
Fachbereich Mathematik und Informatik
Postfach 10 37 64
W-4300 Essen 1
Germany
mem080@de0hrz1a.bitnet
tel.: +49-201-1833-659

Menachem **Magidor**
Hebrew University
Dept. of Mathematics
91904 Jerusalem
Israel
menachem@sunrise.huji.ac.il
tel.: +972-2-58-5355/6

Janos A. **Makowsky**
Technion - Haifa
Computer Science Dept.
Haifa 32 000
Israel
janos@cs.technion.ac.il
tel.: +972-4-332952

Greg L. **McColm**
University of South Florida (USF)
Dept. of Mathematics
4202 E. Fowler Ave. PHY 114
Tampa FL 33620-5700
USA
mccolm@math.usf.edu
tel.: +1 813-974-2643/3498 or 989-2590

Faron **Moller**
University of Edinburgh
Dept. of Computer Science
The King's Buildings
Mayfield Road
Edinburgh EH9-3JZ
Great Britain
fm@lfcs.ed.ac.uk
tel.: +44-31-650-51 50

Peter **Mosses**
Aarhus University
Computer Science Dept.
Ny Munkegade
DK-8000 Aarhus C
Denmark
pdmosses@daimi.aau.dk
tel.: +45-86-12 71 88

Kay J. **Nüßler**
Universität GHS Essen
Fachbereich Mathematik und Informatik
Postfach 10 37 64
W-4300 Essen 1
Germany
tel.: +49-201-737 569

Martin **Otto**
RWTH Aachen
Lehr- und Forschungsgebiet
Mathematische Grundlagen der Informatik
Ahornstr. 55
W-5100 Aachen
Germany
otto@joli.informatik.rwth-aachen.de

Leszek **Pacholski**
Insitute of Mathematics
Polish Academy of Science
Kopernika 18
PL-51 617 Wroclaw
Poland
tel.: +48-71-25 12 71

Arnd **Poetzsch-Heffter**
Cornell University
Department of Computer Science
4112 Upson Hall
Ithaca NY 14853-7510
USA
poetzsch@cs.cornell.edu
tel.: +1-607-255-9222

Dean **Rosenzweig**
University of Zagreb
FSB Katedra Matematiku
Dure Salaja 5
41000 Zagreb
Croatia
dean.rosenzweig@uni-zg.ac.mail.yu
tel.: +38-41-611944

Kurt **Sieber**
Universität des Saarlandes
Fachbereich 14 - Informatik
Postfach 1150
D-66041 Saarbrücken
Germany
sieber@cs.uni-sb.de
tel.: +49-681-302-32 35

Eugene W. **Stark**
State University of New York
at Stony Brook
Dept. of Computer Science
Stony Brook NY 11794-4400
USA
stark@cs.sunysb.edu
tel.: +1-516-632-8444

Iain A. **Stewart**
University College of Swansea
Dept. of Computer Science
Singleton Park
Swansea SA2 8PP
Great Britain
i.a.stewart@swansea.ac.uk
tel.: +44-792-20 56 78

Burkhard **Wald**
Universität GHS Essen
Hochschulrechenzentrum
Postfach 10 37 64
W-4300 Essen 1
Germany
wald@hrz.uni-essen.de
tel.: +49-201-183-2921

Zuletzt erschienene und geplante Titel:

A. Arnold, L. Priese, R. Vollmar (editors):
Automata Theory: Distributed Models, Dagstuhl-Seminar-Report; 54; 11.01.-15.01.93 (9302)

W. Cellary, K. Vidyasankar, G. Vossen (editors):
Versioning in Database Management Systems, Dagstuhl-Seminar-Report; 55; 01.02.-05.02.93 (9305)

B. Becker, R. Bryant, Ch. Meinel (editors):
Computer Aided Design and Test , Dagstuhl-Seminar-Report; 56; 15.02.-19.02.93 (9307)

M. Pinkal, R. Scha, L. Schubert (editors):
Semantic Formalisms in Natural Language Processing, Dagstuhl-Seminar-Report; 57; 23.02.-26.02.93 (9308)

W. Bibel, K. Furukawa, M. Stickel (editors):
Deduction , Dagstuhl-Seminar-Report; 58; 08.03.-12.03.93 (9310)

H. Alt, B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 59; 22.03.-26.03.93 (9312)

H. Kamp, J. Pustejovsky (editors):
Universals in the Lexicon: At the Intersection of Lexical Semantic Theories, Dagstuhl-Seminar-Report; 60; 29.03.-02.04.93 (9313)

W. Strasser, F. Wahl (editors):
Graphics & Robotics, Dagstuhl-Seminar-Report; 61; 19.04.-22.04.93 (9316)

C. Beeri, A. Heuer, G. Saake, S. Urban (editors):
Formal Aspects of Object Base Dynamics , Dagstuhl-Seminar-Report; 62; 26.04.-30.04.93 (9317)

R. V. Book, E. Pednault, D. Wotschke (editors):
Descriptional Complexity, Dagstuhl-Seminar-Report; 63; 03.05.-07.05.93 (9318)

H.-D. Ehrig, F. von Henke, J. Meseguer, M. Wirsing (editors):
Specification and Semantics, Dagstuhl-Seminar-Report; 64; 24.05.-28.05.93 (9321)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Algebra, Dagstuhl-Seminar-Report; 65; 07.06.-11.06.93 (9323)

Ch. Lengauer, P. Quinton, Y. Robert, L. Thiele (editors):
Parallelization Techniques for Uniform Algorithms, Dagstuhl-Seminar-Report; 66; 21.06.-25.06.93 (9325)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 67; 28.06.-02.07.93 (9326)

Ph. Flajolet, R. Kemp, H. Prodinger (editors):
"Average-Case"-Analysis of Algorithms, Dagstuhl-Seminar-Report; 68; 12.07.-16.07.93 (9328)

J.W. Gray, A.M. Pitts, K. Sieber (editors):
Interactions between Category Theory and Computer Science, Dagstuhl-Seminar-Report; 69; 19.07.-23.07.93 (9329)

D. Gabbay, H.-J. Ohlbach (editors):
Automated Practical Reasoning and Argumentation, Dagstuhl-Seminar-Report; 70; 23.08.-27.08.93 (9334)

A. Danthine , W. Effelsberg, O. Spaniol, (editors):
Architecture and Protocols for High-Speed Networks, Dagstuhl-Seminar-Report; 71; 30.08.-03.09.93 (9335)

R. Cole, E. W. Mayr, F. Meyer a.d.Heide (editors):
Parallel and Distributed Algorithms, Dagstuhl-Seminar-Report; 72; 13.09.-17.09.93 (9337)