Manfred Droste, Yuri Gurevich (editors):

Semantics of Programming Languages and Model Theory

Dagstuhl-Seminar-Report; 16
24.-28.6.1991 (9126)

ISSN 0940-1121

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

Das Internationales 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
Informatik, Bau 36
Universität des Saarlandes
W - 6600 Saarbrücken
Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: office@dag.uni-sb.de

DAGSTUHL SEMINAR

SEMANTICS OF PROGRAMMING LANGUAGES AND MODEL THEORY

Organized by:

Manfred Droste (Universität Dortmund and Universität Essen)
Yuri Gurevich (University of Michigan)

June 23-29, 1991

Topic of the conference was the interplay between semantics of programming areas and the mathematical areas of model theory and ordered structures. Participation of researchers from different but neighbouring fields proved very fruitful. Methods from algebra, logic and order theory were used for the solution of problems from denotational semantics of programming languages and in domain theory. A number of talks also dealt with the quickly developing area of evolving algebras and their applications for a complete description of the semantics of logic programming languages. Several further talks considered topics ranging, for instance, from functional programming languages and database models to linear logic, boolean algebras and model theory.

The lively interest in this field was documented by the number 37 participants from Germany and abroad, of which 27 presented talks. In spite of the considerable number of lectures, the atmosphere in Dagstuhl castle offered many opportunities for the discussion of open problems and research ideas in small groups, and stimulated new collaboration between several participants.

Abstracts

I. Evolving Algebras

Evolving Algebras

Yuri Gurevich

We explain evolving algebras (from the first principles) and survey the current state of the theory.

Evolving Algebras in Logic Programming

Egon Börger

This is a report on recent and ongoing work in semantics of logic programming languages which is based on Gurevich's new notion of evolving algebra (1988, see EATCS Bull. 43, Febr. 1991). I have defined *Prolog Algebras* which give a formal semantics for the full language Prolog, including the usual build - in predicates for control, dynamic code, manipulation of files, input, output, terms etc. (see SLNCS vol. 440 and 452, Springer MSRI Proc. of Berkeley 1989 workshop on Logic from CS). These Prolog algebras yield a formal model for the forthcoming ISO WG 17 draft proposal for a standard for Prolog. In joint work with D. Rosenzweig we have derived a formal definition of the *WAM* - a virtual machine model underlying most of the current Prolog Implementations - and were able to prove the *correctness of Prolog compilers* (satisfying certain natural conditions) with respect to the abstract Prolog specification by Prolog Algebras (see Proc. CSL'90, to appear in SLNCS). This WAM specification allowed us to produce an exhaustive analysis and formal definition of the *new ISO WG 17 view* on built - in predicates *for dynamic code* in Prolog and to suggest a formal specification for a uniform implementation of this view (see talk by D. Rosenzweig in this conference). In joint work with P. Schmitt we could show that Prolog Algebras naturally are extended to *Prolog 3 Algebras*, thus opening constraint logic programming to analysis and specification by evolving algebras (see Proc. CLS'90, SLNCS). Ongoing work with C. Beierle shows that a *correctness proof for the PAM* - Protus Abstract Maschine, the WAM extension to Protus-L, a logic programming language enriched by a polymorphic type discipline - can be given extending the WAM correctness proof by type term representing algebras and using constraint-functions for type conditions. Ongoing work with E. Riccobene shows that also *PARLOG* - a commercially available logic programming language with parallelism features which are transparent to the user - can be formally specified by evolving algebras defined putting together ideas from Prolog Algebras with ideas from Occam Algebras (developed by Y. Gurevich and L. Moss, see CSL'89, Proc. SLNCS) (see talk by E. Riccobene in this conference).

Analysis of Prolog Database Views and their Uniform Implementation

Dean Rosenzweig

The clarity and precision brought into the semantics of Prolog and its implementation by the evolving algebra approach of Y. Gurevich (cf. contributions by Y. Gurevich and E.Börger for this meeting) allow a precise analysis of some dark corners of the language, hitherto seemingly inaccessible to formal semantical methodologies.
Jointly with E. Börger, we have analyzed different views of "database operations" (asserta, assertz, retract) which the ISO WG 17 for standardization of Prolog has decided to allow on its meeting of November 1990. We have shown that all the views can be succinctly and uniformly expressed in a uniform way, parameterized only by a notion of a clause "being alive". We have also shown how they could be uniformly implemented by a modification of the "virtual copying" technique, introduced by Lindholm and O'Keefe 1987 for the logical view. Although our framework is rather abstract, free of details of implementation, all the "implementational tradeoffs" between different views become transparent through it. The analysis suggests making the view transparent to the user allowing him to control it and to link it to individual predicates, obtaining thus the right to strike his own compromises between logic (understanding of the program) and efficiency.

Fragments of Prolog and Evolving Algebras

Petr Štěpánek

It is shown that Evolving Algebras describing Algebraic Operational semantics of various fragments of Prolog can be constructed incrementally. The correctness of Evolving algebras and their extensions corresponding to larger fragments of the language is proved. In particular, the Evolving Algebra for pure Prolog is described and compared with its extension (due to Börger) for pure Prolog with the cut operator. It is shown that there is a homorphism that maps the latter algebra onto the former one. This homomorphism extends to a homomorphism of the evolution steps of both algebras during computations of pure Prolog programs. Moreover, the cut operator can be represented as an external dynamic function (oracle) of the algebra for pure Prolog.
(Joint work with Jan Hric.)

Some Connections between Evolving Algebras, Syntax, and Formal Language Theory

Lawrence S. Moss

The method of evolving algebras was proposed by Yuri Gurevich as a means of specifying the semantics of programming languages which is dynamic in a direct way, and which reflects the resource bounded aspects of computation. At the Dagstuhl meeting there were about five talks on applications of this framework. My talk began with a discussion of the EA approach to Occam, which I had worked out with Gurevich several years earlier. The key ideas are that

computation in a distributed language such as Occam is essentially determined by simple components which interact asynchronously, and that what is complicated is the definition of a run of the Occam machine.
The main point of the talk is to show that similar ideas are at work concerning the formalisms for natural language syntax. Specifically, several of the formalisms proposed for generating languages between the context-free and contex-sensitive can be given more simply as evolving algebras. Further, the EA specification is closer to the intuitions than the usual one. I illustrated this point with a discussion of Pereira's Extraposition Grammars.
Finally, I discussed how the EA approach was useful for me in some work with David Johnson (IBM T.J. Watson Res. Ctr.) on a formalization of Relational Grammar called Multistratal Axiomatic Grammar (MAG).

A Formal Specification of PARLOG Based on Evolving Algebras

Elvinia Riccobene

We provide a complete mathematical semantics for the parallel logic programming language PARLOG.
This semantics is abstract but nevertheless simple and supports the intuitive operational understanding of programs. It is based on Gurevich's notion of Evolving Algebras and it is obtained by adapting ideas from the description of (Standard) Prolog given by E. Börger and the specification of functional parallel computation phenomena of Occam devoloped by Y. Gurevich and L. Moss.
We develop a complete specification of the core of PARLOG which governs the computation of goals by user defined predicates. We give an explicit formalization of the two kinds of parallelism occurring in PARLOG: the AND- Parallelism and the (orthogonal) OR-Parallelism.
Both phenomena are described using an abstract notion of PARLOG terms and PARLOG substitution which is unburdened by representation details and implementation constraints.
(Joint work with Egon Börger.)

II. Denotational semantics and domain theory

Domains and the Denotational Semantics of Nondeterminism

Michael W. Mislove

For a high level programming language $\mathcal{L}$ and an operational semantics for $\mathcal{L}$, one can define a behaviour function $\mathcal{B} : \mathcal{L} \to \mathcal{OS}$. A denotional model $\mathcal{M} : \mathcal{L} \to \mathcal{M}$ is said to be *adequate* with respect to $\mathcal{B}$ if $(\forall p, q \in \mathcal{L})\ M(p) = M(q) \Rightarrow \mathcal{B}(\mathrm{p}) = \mathcal{B}(\mathrm{q})$. Conversely, M is said to be *fully abstract* with respect to $\mathcal{B}$ if $(\forall\ \mathrm{p,q} \in \mathcal{L})\ M(p) \neq M(q) \Longrightarrow (\exists C$ context $)$ $\mathcal{B}(\mathcal{C}(\mathrm{p})) \neq \mathcal{B}(\mathcal{C}(\mathrm{q}))$. (These terms are due to Plotkin.) After giving a brief history of the evolution of domains as denotational models for programming languages, we focus on the question of providing adequate and fully abstract models for "abstract" languages - i.e., ones whose syntax is given in terms of uninterpreted atomic actions. For a prototypical such language, we show how it is possible to craft operational models which precisely capture the three notions of nondeterminstic choice: angelic, demonic, and conventional nondeterminism. In the case of angelic nondeterminism, we show how to craft a related adequate and fully abstract denotational model using the family of nonempty Scott-closed subsets of a domain which provides a model for the deterministic sublanguage. Our construction uses the usual adjunction (i.e., the Hoare powerdomain), but augments it with an application of spectral theory. This added feature has two payoffs: first, it allows us to recapture the model of the deterministic sublanguage from within the nondeterministic model, and, second, it allows us to use mappings which are Lawson-continuous. This topology is a refinement of the Scott topology which is compact and Hausdorff; as a result, limit points in the model are unique when they exist. Finally, we show that none of the traditional powerdomains is adequate to model the conventional nondeterminism operational model. The work described comprises joint work with Frank J. Oles, IBM Research.

Decomposition of Domains

Achim Jung

We start from the observation that various negative results imply that no single class of domains suffices for modeling every computational paradigm. We therefore are looking for general results which give an overview over possible and necessary classes of domains, rather than searching for a single and universal category of domains. This goal is pursued by various researchers with various means, we just mention the two projects "Axiomatic Domain Theory" and "Classification of maximal ccc's".

The present work is concerned with the question which domains can be generated from flat domains by using hyperlimits as the single construction principle. We proceed by working backwards, i.e. by decomposing domains. We achieve a Decomposition Theorem which does not just give any representation but a non-redundant representation. This decomposition meshes well with the usual domain constructions, except of course the Plotkin powerdomain.

Coming back to the original question we find that flat domains generate a class $\mathcal{F}$ which is strictly contained in the class of distributive Scott-domains and which strictly contains concrete data structures.

Finite Axiomatizations for Universal Domains

Manfred Droste

In the theory of denotational semantics of programming languages, several authors established the existence of particular kinds of universal domains. Here we first describe a category-theoretic version of a general model-theoretic result, the Fraissé-Jónsson theorem, which gives, for a large class of categories, a necessary and sufficient condition for the existence of a universal homogeneous object. Moreover, such a universal homogeneous object is unique up to isomorphism and can, in many cases, be constructed effectively. We show for the categories of all ω-bifinite domains, all ω-bifinite L-domains, all ω-Scott- domains, and all ω-algebraic lattices, in each case with embedding-projection pairs as morphisms, that each of them contains a universal homogeneous object, (whereas the category of all coherent ω-Scott-domains does not). For each of these four categories $\mathcal{C}$ we introduce a finite set of axioms $S_{\mathcal{C}}$, formulated in a first order language of predicate calculus for posets, and show that an arbitary domain $(D, \leq) \in \mathcal{C}$ is the universal homogeneous object in $\mathcal{C}$ if and only if its subposet of compact elements satisfies all axioms in $S_{\mathcal{C}}$.
(Partly joint work with R.Göbel, Essen.)

Order-theoretic Properties of Powerdomains

Kay J. Nacken

The class of all Scott-domains is not closed under the Plotkin powerdomain construction. Hence, we are looking for an ordertheoretic property that gives a characterisation of all Scott-domains $(D, \leq)$ whose Plotkin powerdomains, $P[D]$, are again Scott-domains. The existence of an embedding of one of two special finite posets (W and M) into a Scott-domain $(D, \leq)$ is equivalent to the condition that the powerdomain $(P[D], \sqsubseteq_M)$ not be a Scott-domain.
It is an open problem of Plotkin whether there exists a Scott-domain $(D, \leq)$ whose Plotkin powerdomain $P[D]$ is universal for SFP-domains (i.e., for every SFP-domain $(E, \leq)$ there exists an embedding projection pair of $(E, \leq)$ into $P[D]$). We give a partial answer to this question: there exists a Scott-domain $(D, \leq)$ such that for any SFP-domain $(E, \leq)$ there exists a mub-embedding of $(E, \leq)$ into $P[D]$; here a mub-embedding is an order-embedding which preserves minimal upper bounds.

Powerdomain Constructions

Reinhold Heckmann

Given at least five different powerdomain constructions (lower L, Smyth's upper U, Plotkin's convex C, Buneman's sandwich S, Gunter's mixed M), one looks for a general theory of power constructions which answers the following questions: What are power constructions? How are different constructions related to each other? Are there more than the five constructions mentioned above? If so, how are these five constructions distinguished among all the others?
Powerdomain constructions are defined as Kleisli triples in the category of dcpo's producing commutative monoids. As a consequence of this definition, they are equipped with a "characteristic" semiring which is the powerdomain of the one-point-domain. Conversely, for every given semiring, there is an initial and a final power construction with just this characteristic semiring. New power constructions may be obtained from existing ones by product formation, core formation (removing junk), and restriction to sub-semirings.
The S constructions mentioned above may all be characterised in the framework of the general theory. They all are either initial or final or both for small "logical" semirings. Moreover, they are connected by a network of relations including products, core, and restriction. This network also suggests further yet unknown power constructions with interesting properties.

Denotational Semantics for Specification Languages

Wilfrid Hodges

A semantics is proposed for specification languages in general. The interpretation of a specification is to be a functor taking "given" systems to target systems. The class of specifiable functors is defined in three ways, the first an abstract domain-theoretic definition, the second in terms of initial models and the third in terms of the hereditarily finite universe of sets over the "given" system. It is shown that all three definitions describe the same class of functors up to natural equivalence. It is also shown that, although the specification language Z has no particular connection with this class of functors, real-life specifications in Z virtually always lie within a fragment of Z which closely matches the third definition above. In principle this yields a systematic translation from Z specifications into algebraic specification languages.

A Cartesian Closed Category of Parallel Algorithms between Scott Domains

Stephen Brookes

We present a category-theoretic framework for providing intensional semantics of programming languages and establishing connections between semantics given at different levels of intensional detail. We use a comonad to model an abstract notion of computation, and we obtain an intensional category from an extensional category by the co-Kleisli construction; thus, while an extensional morphism can be viewed as a function from values to values, an intensional morphism is akin to a function from computations to values. We explore the properties of

the particular example in which the underlying extensional category is the category of Scott-domains and continuous functions, and the notion of computation corresponds to an increasing sequence of data values. The resulting intensional category, whose morphisms we call algorithms, is cartesian closed. Application, currying and composition are continuous operations on algorithms, with respect to the pointwise order. Every algorithm has a continuous input-output function, and every continuous function is the input-output function of some algorithm. In fact, the algorithms for a given function form a complete lattice under the pointwise order. We define an intensional semantics for the λ-calculus and show that the extensional content of the meaning of each term is exactly the standard meaning of that term in the Scott model of the λ-calculus.
(This is joint work with Shai Geva.)

Adjunctions between Categories of Cpos

Frank J. Oles

In the hope of finding a way to turn the Plotkin powerdomain of a Scott domain back into a Scott domain, we investigate the possibilities for various adjunctions between categories of algebraic cpos and consistently complete algebraic cpos. As a negative result, we prove that Scott domains do not form a full reflective subcategory of SFP-objects. For an algebraic cpo D, possibly satisfying Property M, we consider four constructions:
(1) all Scott-closed subsets of D,
(2) all nonempty, Scott-closed subsets of D,
(3) all nonempty, Scott-compact, saturated subsets of D,
(4) all bounded, Scott-closed subsets of D
Each construction is described as a left adjoint. The first construction is left adjoint to the spectrum (i.e., $\vee$-primes) of a completely distributive complete algebraic lattice. The third construction does not always give a consistently complete cpo, but it does if D satisfies Property M. The last construction shows the most promise as a means of making the Plotkin powerdomain consistently complete because the application of this functor to a nondeterministic algebra gives a Scott semigroup (a consistently complete algebraic cpo with a Scott-continuous semigroup operation). This is joint work with Michael W. Mislove, Tulane Univ. and Oxford Univ.

Completion of Quasi-Uniform Spaces

Philipp Sünderhauf

According to Samson Abramsky, one can do domain theory just by considering the lattice of Scott-open sets. Now the Scott-topology of a continuous domain is induced by a quasi-uniformity in a natural fashion. We ask for a generalisation of the notions "complete" and "completion" from the theory of uniform spaces to the quasi-uniformities. It turns out that there is no completion of quasi-uniform spaces. In a slight modification of an idea of Michael Smyth we introduce topological quasi-uniform spaces; these are quasi-uniform spaces carrying an additional topology. For these structures we are able to present a notion of completeness and to construct a completion.

Bifinite Domains: Stable Case

Roberto M. Amadio

Let $Cpo_\wedge$ be the ccc of complete partial orders with "continuous glbs" of compatible pairs, and stable maps. We introduce the full subccc $Bif_\wedge$ of stable bifinites (or equivalently sfp) over $Cpo_\wedge$. Its objects are characterized as ω-algebraic $Cpos_\wedge$ satisfying a "combination" of property M, as in Smyth theorem, and property I, as introduced by Berry, that we call property (MI)*.
We "test" the category $Bif_\wedge$ via a series of classical constructions in domain theory. *Interalia* we show that:
(1) $Bif_\wedge$ is an ω-algebroidal category and it has a universal, homogeneous object. (2) The image of stable retraction over a stable bifinite is again a stable bifinite. (3) If $D \in Bif_\wedge$, and $Prj(D)$ denotes the collection of projections over D then $Prj(D) \in Bif_\wedge$.
Next we investigate which full, cartesian closed, sub-categories of ω- algebraic $Cpo_\wedge$ and stable maps are contained in $Bif_\wedge$. It is shown that property M and "2/3" of property I are necessary for preserving the ω-algebraicity of the functional space. The remaining "1/3" of property I is also necessary under rather mild hypothesis. As a fall out we show a stable, countably based, version of L-domains, introduced by Coquand, is contained in $Bif_\wedge$ and that such L-domains are the "largest ccc" under the assumption that principal ideals of domains are distributive.

Spaces of Retractions on Domains and Universal Retractions

Klaus Keimel

Let P be a domain, i.e. a directed complete partially ordered set with $\perp$. Let $[P \to P]$ denote the domain of all continuous (i.e. directed join preserving) maps $f : P \to P$ and $Ret(P)$ the subdomain of all retractions $r \in [P \to P]$. The question is whether there is a universal retraction, i.e. a retraction $\rho : [P \to P] \to Ret(P)$. The following results are due to M. Rothe (Diplomarbeit, Darmstadt 1991): Let P be a continuous domain such that $[P \to P]$ also is continuous. a) If $Ret(P)$ is a retract of $[P \to P]$, then $Ret(P)$ is continuous and every retraction $r \in Ret(P)$ is algebraic (i.e. the image $r(P)$ is an algebraic domain). b) Every retraction of P is algebraic iff P does not contain a chain order isomorphic to the rationals. These results tell us that one should consider the domain $AlgRet(P)$ of all algebraic retractions instead of $Ret(P)$. One then has: c) Is P bifinite (or a retract of a bifinite domain), then $AlgRet(P)$ also is bifinite (or a retract thereof) and there is a universal retraction $\rho : [P \to P] \to AlgRet(P)$. d) If L is an algebraic (or continuous) L-domain, then the same hold for the domain $AlgRet(P)$; but a universal retraction need not exist. These investigations are motivated by constructions of models for the polymorphic Lambda Calculus, where universal retractions are needed.

A Simple Method for Solving Domain Equations Effectively

Klaus Weihrauch

Let D'_s be the set of all fb-complete partial orders $\rho \subseteq \Sigma^* \times \Sigma^*$ with minimum, where ρ is fb-complete iff $\max_\rho E$ exists for all finite $E \subseteq A_\rho := \{x : (x,x) \in \rho\}$. Every Scott-domain is isomorphic to the completion $cpl(\rho)$ for some $\rho \in D'_s$. With an appropriate order $D_s := \{\emptyset\} \cup D'_s$ becomes a cpo $\overline{D}_s = (D_s, \leq)$ with algebraic basis $K(\overline{D}_s) = \{\rho \in D_s : \rho \text{ finite }\}$. It is now easy to define a function $sum : D_s \times D_s \to D_s$ such that f is continuous and $cpl(sum(\rho_1, \rho_2))$ can be called the sum of $cpl(\rho_1)$ and $cpl(\rho_2)$ (correspondingly for product, function space, etc.). For $f(\rho) := sum(\rho, \rho)$ let $\rho_f := \bigvee_i f^i(\emptyset)$ and $D := cpl(\rho_f)$ (which exists since $\rho_f \neq \emptyset$). Then $D = D+D$. More complicated domain equations can be solved accordingly, whenever $\rho_f \neq \emptyset$. If, however, $\rho_f = \emptyset$ then a simple trick leads to isomorphisms, eg. $D \cong D \times D$ or $D \cong D \to D$. The method is formulated with "constructive cpos" and is fully effective. It can applied correspondingly to bifinite domains.

III. Other topics

Deciding Boundedness for Uniformly Connected Datalog Programs

Irène Guessarian

A Datalog program is said to be bounded if the number of nested recursive calls needed to evaluate a given recursive query is bounded independently of the size of the input data base. We prove that boundedness is decidable for uniformly connected Datalog programs and some of their generalizations. Uniformly connected programs generalize chain programs, and, as in the case of chain programs, decidability of boundedness for uniformly connected programs can be reduced to finiteness of a context-free language.

Reasonable Extentions of the ML Type Discipline

A.J. Kfoury

The type discipline of ML does not allow various natural programs to be typed. These anomalies are mostly the result of two distinctive features of the discipline. The first is that ML forces all the occurrences of a λ-bound variable to have the same type and, even though let $x = N$ in M is considered syntactic sugar for $(\lambda x.M)N$, ML allows the occurrences of a let-bound variable to have different types. Put differently, ML treats λ-bound variables monomorphically while it treats let-bound variables polymorphically.
The second feature causing these anomalies is the monomorphic treatment of recursively defined functions. That is, ML restricts all the occurrences of a recursively defined function F

on both sides of its definition to have the same type.
We propose ways of extending the rules ABS, LET, and FIX in order to remedy these problems. In some cases the type-reconstruction remains decidable, in others it becomes undecidable, and in others still whether type-reconstruction is decidable is open.

Logical Systems are Pure Type Systems

Hans Tonino

For some time it is known that logics can be interpreted in type systems. This interpretation assigns types to formulae and lambda terms to proof-trees and is such that the derivability relation is preserved, the terms representing the proofs becoming inhabitants of the types representing the formulae. This phenomenon has become known as the Curry-Howard-De Bruyn isomorphism. Strictly speaking we are only entitled to speak of an isomorphism if the interpretation is bijective. In that case it would also be possible to reconstruct the formulae from the types and the proof-trees from the terms. This is however not always possible. It has for instance been shown that the calculus of constructions is not conservative over higher order, many sorted intuitionistic logic.
Fujita (1989) devised a type system which is isomorphic (in the Curry-Howard-De Bruyn sense) to the aforementioned higher order logic. This type system can easily be projected in the calculus of constructions. However, the proof of Fujita was not complete.
In my talk I will indicate in which way the isomorphism can effectively be defined, generalizing the result of Fujita, and completing his proof. My presentation of the type system uses the notion of a Pure Type System, introduced independently by Berardi (1988) and Testouw (1989). This makes the proofs mathematically easier to understand and more precise.

Computations as Proofs in Propositional Linear Logic

Andre Scedrov

Linear logic, introduced by Girard in 1986, is a refinement of classical logic, with a natural, intrinsic accounting of resources. It may be derived from a Gentzen-style sequent calculus for classical logic in three steps.
The first step is to eliminate two structural rules, contraction and weakening. We may view hypotheses as resources, and conclusions as requirements to be met by using the resources. Thus the formula "A implies A" means that the resource A can be used to meet the requirement A. Contraction rule allows any property that follows from two assumptions of a formula to be derived from a single assumption of that formula. Weakening allows deductions that do not use all of their hypotheses. Since contraction and weakening make it possible to use an assumption as little or as often as desired, these rules are responsible for what one may see in hindsight as a loss of control of resources in both classical and intuitionistic logic. Excluding these rules produces a linear system in which each assumption must be used exactly once. In the resulting linear logic, formulas indicate bounded or finite resources that cannot necessarily be discarded or duplicated.
The second step in deriving linear logic involves the propositional connectives. Briefly, the

change in structural rules leads naturally to two forms of conjunctions, one called "multiplicative" and the other "additive"; and similary two forms of disjunction. The multiplicative form requires partitioning of resources, while the additive form requires resource sharing.
Finally, in order to recover the full deductive power of classical logic, a storage or reuse operator, !, is added. Intuitively, the formula !A provides unlimited use of the resource A. Using a computational metaphor, the formula !A means that "the datum A is stored in the memory and may be referenced an unlimited number of times". There is also a dual modal operator, ?, definable from ! by using linear negation. The formula ?B allows the unlimited consumption of B.
Since the basic framework remains linear, unbounded reuse or consumption is only allowed "locally", at formules specifically marked with ! and ?. The resulting logic is natural from both proof-theoretic and computational standpoints.
In this joint work with P. Lincoln (Stanford and SRI), J. Mitchell (Stanfort), and N. Shanker (SRI) we establish an exact match between computations on generic machines and proofs in fragments of propositional linear logic. As a consequence, we obtain the results on complexity of provability for several fragments of propositional (quantifier-free) linear logic. This work has appeared in FOCS'90.
Perhaps our most notable result is that full propositional linear logic is undecidable. However, let us describe our results starting with the smallest fragment considered, the multiplicative fragment. We show that the decision problem for this fragment is NP. Moreover, if unrestricted weakening is allowed, then the multiplicative fragment becomes NP-complete. [The NP-completeness for the pure multiplicative fragment was obtained recently by M. Kanovich (Tver, USSR).]
There are two natural fragments extending pure multiplicative linear logic. We show that the first extension, with both multiplicative and additive connectives but not the modalities ! and ?, is PSPACE-complete. This fragment may be called core linear logic.
Let us note in passing that the second extension, with only multiplicative connectives and the modalities ! and ?, is at least as hard as the reachability problem for Petri nets (or, equivalently, commutative semi-Thue systems or vector addition systems). This follows from conservativity properties established in this work together with previous work of Asperti and Gunter-Gehlot relating linear logic and Petri nets. Although reachability is decidable (Kosarajn, STOCS'82), the best known lower bound is EXPSPACE (Lipton 1976, Mayr-Meyer 1982). A likely upper bound is primitive recursive in the Ackermann function (McAloon, Clote).
Finally we show that the provability in full propositional linear logic is undecidable (provability is trivially r.e. since the proof system is effective.) We also establish the undecidability of a noncommutative variant of linear logic (even without additive connectives).
The main open problems are the decidability of the multiplicatives with the modal operators ! , ?, and the decidability of core linear logic extended with second-order propositional quantifiers. (Adding first-order quantifiers only does not change the complexity of the fragments discussed here.)

Alternative Characterizations of Finitary Boolean Algebras

Lutz Heindorf

Roughly speaking a Boolean algebra is finitary iff it is almost countable and has a finite description up to isomorphism. This becomes a precise definition after a method of describing Boolean algebras is fixed.
In the talk we discuss three possibilities connected with the names of Hauf, Paljutin and Ketonen. The main result is that in all three cases the same class of finitary algebras is defined.

A Theory of Unary Pairfunctions

Burkhard Wald

We consider a class of functions which has pairs or nested pairs as their argument and pairs or nested pairs as their results. Examples for such Pairfunctions are the functions L, R, S, D, and B where $L(\langle a,b\rangle) = a, R(\langle a,b\rangle) = b$, $S(\langle a,b\rangle) = \langle b,a\rangle, D(a) = \langle a,a\rangle$, $B(\langle a,\langle b,c\rangle\rangle) = \langle\langle a,b\rangle,c\rangle$. Because we view these functions as unary functions, the set of all those functions becomes a semigroup with the usual composition of functions as the multiplication. The result is, that this semigroup is finitely presented. The presentation is given by 69 explicit equations over the generators L, S, D, and B.

How to Design Efficient Algorithms for Graphs Using Minors and Monadic Second Order Logic?

Detlef Seese

Many algorithmic problems in graph theory are NP-hard and (at least till now) have no solutions in polynomial time. A great number of them remain even NP-hard if they are restricted to smaller and simpler classes of graphs. But if the regarded class of graphs is very simple, as for instance the class of trees, or the class of series-parallel graphs, then some of the NP-hard problems have solutions in polynomial or even in linear time. This suggests the problem to find a structural reason for this "change in complexity".
In the talk it is tried to shed some light into this area.
Using tiling problems it is pointed out that the containment or definability of large grids is one of the reasons for high complexity.
This together with the fundamental results of N. Robertson and P.D. Seymour concerning the structure of graphs avoiding a given minor implies that one can concentrate the attention in the first step to graphs which are similar to trees, or more exactly have universally bounded tree width.
For the discription of the algorithmic problems it is used an extension of the very powerful monadic second order logic. Such problems are denoted as EMS problems. Using a variant of the machinery of interpretations it can be proved that each EMS problem over a class K of graphs of universally bounded tree width can be reduced to another EMS problem on a class

of binary trees. EMS problems for binary trees can be easily solved in polynomial time by automata theoretical methods.

Lattice Interpretation of Database Dependencies

Christian Herrmann

We report about a paper of the late Alan Day interpreting functional and multivalued dependencies into lattice word problems. By these means an undecidability proof for the implication problem of (functional dependencies and) embedded multivalued dependencies comes into reach.

Semantics in Database Models

Bernhard Thalheim

1. Semantic database models

Database design could be defined as the design of the logical and physical structure of a database in a given database management system to contain all the information required by the user in an organization **and required for an effective behavior** according to the complexity measures storage and read/write representation, computational simplicity and transparency.
Goals of database design support are: to support clear definition by ordinary designers, to support reasoning about system properties, to support redesign and refinement, to support effective implementation and to support orthogonality.
For that different important database concepts (complex objects and associations, integrity constraints, operations, behavior, views and distribution, accomodation of declarative, graphical and procedural declaration, persistence) should be used integrated and should be based on concepts of programming languages too (concurrency, abstraction (modularity, classes, types, clustering, grouping), hierarchies, inheritance). Therefore, fundamental components of semantic database models are: mechanisms for structuring data (objects, attributes, associations (aggregation, IsA-relationships, memberships, summarization)), logical constraints (local or global, static or dynamic), operations and behavior. Different semantic database models meet these requirements in a different manner. Most of them use graphical representation of parts. The common disadvantage of almost all semantical database models is the lack of operations.

2. An extended entity-relationship model

The entity-relationship model is one of the most important semantic database models. But it has several weaknesses (only first-order relationships, no theoretical basis for weak entities, unnatural IsA representation, shortcoming inherited from the network and the relational model). Based on the theoretical results of the relational theory (see for instance [4]) the entity-relationship model is extended to the **Higher-order Entity-Relationship Model** (HERM)

[5] by adding several constructs: nesting for attributes, relationships on relationships (higher-order relationships), clusters, integrity constraints, operations with pre- and postconditions, dynamic constraints for the description of the behavior.
It is based on the object-oriented approach to design. Objects are represented in the database by identifiers and values (according a predefined structure), have a global and a local semantics and use operations. Classes are collections of objects with the same structure, general semantics and operations. The type describes the structure, semantics and operations of a class. Since objects do not exist independently they should be considered by their dependence relation. Kernel objects are objects which exist independently from the others. Characteristic objects are objects characterizing or characterized by other objects. Furthermore there exist relationships among objects.
Integrity constraints introduced for the HERM are: classical constraints like functional, inclusion, exclusion, and multivalued dependencies and graphical constraints like path dependencies. The treatment of integrity contraints is based on the identifier and the generalized key concept.
The HERM algebra uses classical (generalized relational) operations as generic operations and encorporate user-definable operations with pre- and postconditions. Furthermore, query forms are definable in order to specify the behavior of the database. These are used for deriving decomposition restrictions.
This model allows the development of other design methods like, for instance, the design based on decomposition on units which is the pedant of the modular design method known in programming languages.
Another advantage of the HERM design is that schemes are simpler. For instance, the design example modelling the database used for the Mathematical Reviews of Teorey [3] uses 54 entity types and 58 relationship types. The HERM design of the same example is based on 11 entity types and 16 relationship types.
Furthermore, it can be shown that normalized HERM schems can be translated to normalized relational or normalized network schemes directly. The ERM-translation to relational schemes needs afterwards normalization and additional modeling of integrity constraints.

3. The application of constraints

The modeling capability is to be illustrated on three application areas.
First, the application of constraints to equivalence problems is discussed. Generally, the equivalence problem is undecidable. There is no finite axiomatization of scheme equivalence. But for several classes of schemes rules for equivalent restructuring can be developed.
Second, the translation of HERM schemes to network and relational schemes is based on four different approaches: the event-nonseparation, the event-separation, the union and the weak universal relation approach. Integrity constraints and operations are to be translated according to these approaches. The translations can be used for the recompilation of operations defined in the translated schemes to other translated schemes.
Third, it is shown how the integrity constraints can be used for derivation of correct implementations of the generic operations Insert, Delete, Atomar-Update and Update.
It should be noticed that these three capabilies are implementable. There was developed a prototype of a design system based on HERM.

References

[1] C. Beeri, Theoretical foundations for OODB's - a personal perspective. To appear in Database Engineering, 1991.

[2] K.-D. Schewe, B. Thalheim, I. Wetzel, J. W. Schmidt, Approaching object-oriented database design on the basis of SAMT. Submitted for publication, May 1991.

[3] T. Teorey, G. Wei, D.L. Bolton, and J.A. Koenig, ER model clustering as an aid for user communication in database design. Comm. ACM, Aug. 1989, 32, 8, 975-987.

[4] B. Thalheim, Dependencies in Relational Databases. Teubner, Leipzig, 1988.

[5] B. Thalheim, Theoretical fundamentals of entity-relationship modeling. Manuscript prepared for submitting, Rostock 1991 (see also LNCS 364, Springer 1989, 382-397; J. New Generation Comput. Syst. 1, 1988, 3, 211-228).

Traces and Pomsets: a Categorical View

S. Kasangian, G. Mauri, N. Sabadini

In this paper we consider two formalisms for describing concurrent processes, i.e. Mazurkiewicz's traces and pomsets. Pomsets are a generalization of traces, because the concurrency relation is not given a priori. Two actions may be concurrent or dependent according to different contexts, times and situations. In this paper we deal mainly with traces, discussing the relation between concurrency and causal independency among actions. Furthermore, we give a categorical treatment of traces and pomsets, introducing both traces or pomsets as enriched categories over 2-categories of trace observers (or pomsets observers). Abstractions and refinements are the effect of "change of base" of 2-categories.

Berichterstatter: K.-J. Nacken

Tagungsteilnehmer

Dr. R.M. Amadio
CRIN
B.P. 239
F-54506, Vandoeuvre-les Nancy Cedex
E: amadio@loria.cria.fr

Dr. C. Berline
Equipe de Logique
UFR de Mathematique
Universite Paris VII
2, Place Jussieu
F-75251 Paris Cedex 5

Prof. Dr. E. Börger
Dip. di Informatica
Universita' di Pisa
Corso Italia 40
I-56100 Pisa
E: boerger@dipisa.di.unipi.it

Prof. Dr. S. Brookes
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213
U S A
E: brookes@proof.ergo.cs.cmu.edu

Priv.-Doz. Dr. M. Droste
Fachbereich Mathematik und Informatik
Universität Essen
D-W4300 Essen 1
E: matb30@DE0HRZ1A.bitnet

Prof. Dr. E. Ebbinghaus
Abt. für Mathematische Logik und
Grundlagen der Mathematik
Universität Freiburg
Albertstr. 23b
D-W7800 Freiburg i. Br.

Prof. Dr. J. Flum
Abt. für Mathematische Logik und
Grundlagen der Mathematik
Universität Freiburg
Albertstr. 23b
7800 Freiburg

Prof. Dr. I. Guessarian
Dept. de Mathematique
Universite de Paris VII
2, Place Jussieu
F-75251 Paris Cedex 5
Frankreich

Prof. Dr. Y. Gurevich
Electrical Engineering and
Computer Science Dept.
University of Michigan
Ann Arbor, MI 48109-2122
USA
E: gurevich@dip.eecs.umich.edu

Dr. R. Heckmann
FB 10-Informatik
Universität des Saarlandes
D-W6600 Saarbrücken 11

Dr. L. Heindorf
Karl-Weierstraß-Institut
Mohrenstr. 39
D-O-1086 Berlin

Priv.- Doz. Dr. C. Herrmann
Fachbereich Mathematik
Technische Hochschule Darmstadt
Schloßgartenstr. 7
D-W6100 Darmstadt

Prof. Dr. W. Hodges
School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS, England
E: wilfrid@maths.qmw.ac.uk

Dr. A. Jung
Fachbereich Mathematik
Technische Hochschule Darmstadt
Schloßgartenstr.7
D-W6100 Darmstadt
E: xmatdb5r@ddathd21.earn

Prof. Dr. S. Kasangian
Dipartimento di Scienze dell' Informazione
Universita degli Studi di Milano
Via Moretto da Brecia, 9
I-20133 Milano
E: kasan@imiucca.csi.unimi.it

Prof. Dr. K. Keimel
Fachbereich Mathematik
Technische Hochschule Darmstadt
Schloßgartenstr.7
D-W6100 Darmstadt

Prof. Dr. A. Kfoury
Dept. of Computer Science
Boston University
Boston, Mass. 02215
U S A

Prof. Dr. M. Mislove
Dept. of Mathematics
Tulane University
New Orleans, Louisiana 70118
U S A
E: mt05amf@vm.tcs.tulane.edu

Prof. Dr. L. Moss
Universtity of Indiana
Dept. of Mathematics
Bloomington, Indiana
E: lmoss%prism.decnet@ucs.indiana.edu

Dipl.-Math. K.J. Nacken
Fachbereich Mathematik und Informatik
Universität Essen
D-W4300 Essen 1

Dr. F. J. Oles
Mathematical Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
U S A

Dr. D. Otth
Fachbereich Mathematik
ETH Zentrum
CH-8092 Zürich

Dr. M. Otto
Abt. für Mathematische Logik
und Grundlagen der Mathematik
Universität Freiburg
Albertstr. 23b
D-W7800 Freiburg

Dr. E. Riccobene
Dip. di Matematica
Universita di Catania
Viale Doria 6
I-95125 Catania
E: riccobene@mathct.cineca.it

Prof. Dr. D. Rosenzweig
FSB Kathedra Matematiku
Svencilista u Zagrebu
Dure Salaja 5
YU-41000 Zagreb
Jugoslawien

Prof. N. Sabadini
Dipartimento di Scienze dell' Informazione
Universita degli Studi di Milano
Via Moretto da Brecia, 9
I-20133 Milano

Prof. Dr. A. Scedrov
Dept. of Computer Science
University of Pennsylvania
Philadelphia, PA 19104-6395 U S A
E: andre@central.cis.upenn.edu

Prof. Dr. P. Schmitt
Fachbereich Informatik
Universität Karlsruhe
Englerstr. 2
D-W7500 Karlsruhe

Prof. Dr. D. Seese
Karl-Weierstrass-Institut für Mathematik
Mohrenstr. 39
D-O-1086 Berlin

Prof. Dr. P. Stepanek
Katedra Kybernetiky
University Karlovy
Malostranske nam. 25
11800 Praha 1, CSFR

Dipl.-Math. P. Sünderhauf
Fachbereich Mathematik
Technische Hochschule Darmstadt
Schloßgartenstr. 7
D-W6100 Darmstadt

Prof. Dr. B. Thalheim
Fachbereich Informatik
Universität Rostock
Albert-Einstein-Str. 21
D-O-2500 Rostock

Prof. Dr. W. Thomas
Institut für Informatik
Universität Kiel
Olshausenstr. 40
D-W2300 Kiel 1

Prof. Dr. J. Tiuryn
University of Warsaw
Palac Kultury i Nauki
00-901 Warszawa, Polen
E: warsaw@cc.uib.es

Prof. Dr. H. Tonino
Fac. of Technical Mathematics
and Informatics
Delft University of Technology
Julianalaan 132
NL-2628 BL Delft

Priv.-Doz. Dr. B. Wald
Hochschulrechenzentrum
Universität Essen
D-W4300 Essen 1

Prof. Dr. K. Weihrauch
Fachbereich Informatik
Fernuniversität Hagen
D-W5800 Hagen

Bisher erschienene und geplante Titel:

W. Gentzsch, W.J. Paul (editors):
Architecture and Performance, Dagstuhl-Seminar-Report; 1, 18.-20.6.1990; (9025)

K. Harbusch, W. Wahlster (editors):
Tree Adjoining Grammars, 1st. International Worshop on TAGs: Formal Theory and Application, Dagstuhl-Seminar-Report; 2, 15.-17.8.1990 (9033)

Ch. Hankin, R. Wilhelm (editors):
Functional Languages: Optimization for Parallelism, Dagstuhl-Seminar-Report; 3, 3.-7.9.1990 (9036)

H. Alt, E. Welzl (editors):
Algorithmic Geometry, Dagstuhl-Seminar-Report; 4, 8.-12.10.1990 (9041)

J. Berstel , J.E. Pin, W. Thomas (editors):
Automata Theory and Applications in Logic and Complexity, Dagstuhl-Seminar-Report; 5, 14.-18.1.1991 (9103)

B. Becker, Ch. Meinel (editors):
Entwerfen, Prüfen, Testen, Dagstuhl-Seminar-Report; 6, 18.-22.2.1991 (9108)

J. P. Finance, S. Jähnichen, J. Loeckx, M. Wirsing (editors):
Logical Theory for Program Construction, Dagstuhl-Seminar-Report; 7, 25.2.-1.3.1991 (9109)

E. W. Mayr, F. Meyer auf der Heide (editors):
Parallel and Distributed Algorithms, Dagstuhl-Seminar-Report; 8, 4.-8.3.1991 (9110)

M. Broy, P. Deussen, E.-R. Olderog, W.P. de Roever (editors):
Concurrent Systems: Semantics, Specification, and Synthesis, Dagstuhl-Seminar-Report; 9, 11.-15.3.1991 (9111)

K. Apt, K. Indermark, M. Rodriguez-Artalejo (editors):
Integration of Functional and Logic Programming, Dagstuhl-Seminar-Report; 10, 18.-22.3.1991 (9112)

E. Novak, J. Traub, H. Wozniakowski (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 11, 15-19.4.1991 (9116)

B. Nebel, C. Peltason, K. v. Luck (editors):
Terminological Logics, Dagstuhl-Seminar-Report; 12, 6.5.-18.5.1991 (9119)

R. Giegerich, S. Graham (editors):
Code Generation - Concepts, Tools, Techniques, Dagstuhl-Seminar-Report; 13, 20.-24.5.1991 (9121)

M. Karpinski, M. Luby, U. Vazirani (editors):
Randomized Algorithms, Dagstuhl-Seminar-Report; 14, 10.-14.6.1991 (9124)

J. Ch. Freytag, D. Maier, G. Vossen (editors):
Query Processing in Object-Oriented, Complex-Object and Nested Relation Databases, Dagstuhl-Seminar-Report; 15, 17.-21.6.1991 (9125)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Model Theory, Dagstuhl-Seminar-Report; 16, 24.-28.6.1991 (9126)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 17, 1.-5.7.1991 (9127)

A. Karshmer, J. Nehmer (editors):
Operating Systems of the 1990s, Dagstuhl-Seminar-Report; 18, 8.-12.7.1991 (9128)

H. Hagen, H. Müller, G.M. Nielson (editors):
Scientific Visualization, Dagstuhl-Seminar-Report; 19, 26.8.-30.8.91 (9135)

T. Lengauer, R. Möhring, B. Preas (editors):
Theory and Practice of Physical Design of VLSI Systems, Dagstuhl-Seminar-Report; 20, 2.9.-6.9.91 (9136)

F. Bancilhon, P. Lockemann, D. Tsichritzis (editors):
Directions of Future Database Research, Dagstuhl-Seminar-Report; 21, 9.9.-13.9.91 (9137)

H. Alt , B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 22, 07.10.-11.10.91 (9141)

F.J. Brandenburg , J. Berstel, D. Wotschke (editors):
Trends and Applications in Formal Language Theory, Dagstuhl-Seminar-Report; 23,14.10.-18.10.91 (9142)

H. Comon , H. Ganzinger, C. Kirchner, H. Kirchner, J.-L. Lassez , G. Smolka (editors):
Theorem Proving and Logic Programming with Constraints, Dagstuhl-Seminar-Report; 24, 21.10.-25.10.91 (9143)

H. Noltemeier, T. Ottmann, D. Wood (editors):
Data Structures, Dagstuhl-Seminar-Report; 25, 4.11.-8.11.91 (9145)

A. Borodin, A. Dress, M. Karpinski (editors):
Efficient Interpolation Algorithms, Dagstuhl-Seminar-Report; 26, 2.-6.12.91 (9149)

B. Buchberger, J. Davenport, F. Schwarz (editors):
Algorithms of Computeralgebra, Dagstuhl-Seminar-Report; 27, 16..-20.12.91 (9151)