Report on the Dagstuhl-Seminar 9403

Relational Methods in Computer Science

January 17 - 21, 1994

Organizers:
Chris Brink (University of Cape Town)
Gunther Schmidt (Universität der Bundeswehr München)

This Dagstuhl-Seminar has been attended by 35 computer scientists, logicians, and mathematicians from 14 countries and 5 continents.

Since the mid-1970's it had become clear that the calculus of relations is a fundamental conceptual and methodological tool in Computer Science just as much as in Mathematics. A number of seemingly distinct areas of research have in fact this much in common that their concepts and/or techniques come from the calculus of relations. However, it had also become clear that many opportunities for cross-pollination are being lost simply because there was no organised forum of discussion between researchers who, though they use the same concepts and methods, nonetheless perceive themselves as working in different fields.

The aim of this Dagstuhl Seminar was, therefore, to bring together researchers from various subdisciplines of Computer Science and Mathematics, all of whom use relational methods in their work, and to encourage the creation of an active network continuing after the Seminar to exchange ideas and results.

The talks focussed in particular on *Relational Models of Program Semantics*, *Kripke Semantics of Program Logic* (including relational approaches to e. g. dynamic logics, temporal logics, and modal logics), *Jónsson-Tarski Relation Algebras*, and *Relational Calculi and Methods in Application Fields* such as Databases, Computational Linguistics, Semantic Nets and Knowledge Representation.

It was felt that the meeting was a really necessary one to bring people together. A successor seminar is loosely planned to take place in Rio de Janeiro around July/August 1995. It is planned to have a collection of papers around the topic of the seminar published in a separate volume thereby trying to do some work in the direction of standardizing notation.

We want to thank all participants for their presentations and discussions. Our special thanks go to the Dagstuhl board for accepting this seminar to be held in SchloßDagstuhl. As always, Dagstuhl proved to be a perfect site with regard to lodging, leisuring, and lecturing. We appreciated the work of the Dagstuhl staff, who made us feel comfortable and enabled us to concentrate on our work. The technical staff in our home institutions greatly helped us in preparing the meeting and we owe them our sincere gratitude. Finally, thanks go to Claudia Hattensperger, who edited this report, based on the abstracts of the participants.

March 1994 Chris Brink and Gunther Schmidt

Induction and Recursion on Datatypes

Roland Backhouse[1]

Given a specification two fundamental, but separate, questions are whether there exists a solution to the specification and whether solutions are unique. In particular, in programming it is common to encounter specifications in the form of recursive equations. In this case the existence problem is usually resolved by appeals to the Knaster-Tarski theorem, and the unicity problem by inductive arguments.

Our concern is to develop a calculational theory of induction that enables one to readily identify relations admitting induction. To this end a notion of "admitting induction with respect to a datatype" has been identified and investigated.

In the talk a review was given of the highly compact formulations of "is well-founded" and "admits induction" in relation algebra. In each case three different (but equivalent) characterisations were given. Subsequently the use of one of the formulations of "admits induction" was illustrated by a proof of Newman's lemma (every locally confluent relation admitting induction is confluent).

Time did not permit discussion of how "admitting induction" and "is well-founded" are generalised to "F-reductive" (admitting induction with respect to a datatype) and "F-inductive" (being well-founded with respect to a datatype).

Fork Algebras

Gabriel Baum and Armando Haeberer

Fork algebras provide a useful basis for relational calculi for program derivation. In this presentation we examine some fundamental issues concerning fork algebras and their use in program derivation. Fork algebras arise as extensions of relational algebras with a new operator, called fork, which enables the introduction, by definition, of projections. The abstract calculus of fork algebras manipulates fork-relational terms without variables, free or bound, over individuals. This calculus provides our formalism for program derivation. Two basic issues concerning a formalism for program derivation concern its formal aspects (such as soundness and completeness) and its adequacy for reasoning and deriving programs from specifications. We, accordingly, address both issues. The (meta-)mathematical aspects of soundness and completeness of a calculus are connected to the limits, in principle, and there lies their importance. These are settled by two fundamental results, namely expressiveness (which shows that fork-relational terms encompass the expressive power of first-order formulas) and representability (which shows that any abstract model of our calculus is isomorphic to an intended one with relations of input-output pairs). The adequacy of our calculus for reasoning about and deriving programs has been illustrated by several examples elsewhere. We here provide some additional material indicating how various aspects of other approaches can be handled within ours and how we can cope with computational complexity and universal quantifiers.

[1] Joint work with H. Doornbos

Computing Kernels of a Graph – A Relation-Algebraic Approach Prototyping Relational Specifications Using Higher-Order Objects

Rudolf Berghammer

Given a directed graph $\mathcal{G} = (V, B)$ with set V of points and relation $B \in 2^{V \times V}$, a subset K of the point set is said to be a *kernel* if it is

a) *absorbant*, i.e., from every point x outside of K there is at least one point y in K such that B_{xy} holds, and

b) *stable*, i.e., for all points x and y contained in K the property B_{xy} does not hold.

In the talk we show how to compute kernels of a directed graph with relation-algebraic means. First, we consider kernels as vectors in the relational sense, i.e., as relations v fulfilling $v = v\,\mathsf{L}$. Then the kernel problem becomes a fixed point problem, where the function $f(v) = \overline{B\,v}$ under consideration is antitone. Using compositions of antitone functions and a relational description of direct sums, we are able to give easy proofs of generalizations of two well-known existence theorems for kernels, viz. of the theorems that every finite circuit-free and every finite bipartite directed graph has a kernel. The first theorem dates back to von Neumann, the later one is known as Richardson's theorem. Secondly, we consider kernels as elements of the powerset 2^V, i.e., as points (non-empty and injective vectors) in the relational sense. This leads to a prototyping-like enumeration algorithm for the set of all kernels of a directed graph. Also an execution of this algorithm using the RELVIEW system is demonstrated.

English as a Relational Language

Michael Böttner

A relational language in the sense of Suppes (see [1]) is a context-free language L with a relation-algebraic semantics. To construe natural languages as relational languages as proposed in various fragments by Suppes offers certain advantages like e.g. a variable-free semantics, a natural ontology of objects, properties and relations, representability of a sentence by a constituent tree etc.

In my talk, an illustration of this approach to the semantics of natural language is given by a fragment of English including fragments of syllogistic languages and extensions of nouns by adjectives. In addition, I propose solutions for Boolean combinations of nouns, collective verb phrases like e.g. *own a house together* and some anaphorical constructions arising from the reflexive, posessive, relative, and reciprocal pronouns.

References

[1] Suppes, P. (1991) *Languages for Humans and Robots*. Oxford: Blackwell.

Priestley Duality for Predicate Transformers

Chris Brink and Ingrid Rewitzky

We present a general framework for translating between relational (input-output) semantics and predicate transformer semantics of programs using a variation on an extension of *Priestley duality* for distributive lattices (see [3]) by Cignoli *et al* [1]. For predicate transformer semantics we consider an abstraction of Dijkstra's weakest precondition semantics [2], namely bounded distributive lattices with meet-homomorphisms. As for relational semantics we consider ordered topological spaces with certain binary relations, where the order is interpreted as an order of increasing information, the open sets as semi-observable properties (see, for example, [4]) and the relations are programs. Our predicate transformer semantics embeds into our relational semantics framework via the Cignoli/Priestley duality. We then give further justification to Smyth's claim that 'topological notions are basic to Computer Science' (see [4]) by giving computational interpretations of the topological notions surrounding Cignoli/Priestley duality.

References

[1] Cignoli, R., S. Lafalce and A. Petrovich. [1991]. Some remarks on Priestley duality for distributive lattices. *Order* 8. p 299-315.

[2] Dijkstra, E.W. [1976]. *A Discipline of Programming.* Prentice-Hall: Englewood Cliffs, New Jersey.

[3] Priestley, H.A. [1970]. Representation of distributive lattices by means of ordered Stone spaces. *Bulletin of London Mathematical Society* 2. p 186-190.

[4] Smyth, M.B. [1983]. Powerdomains and predicate transformers: a topological view. (J. Diaz (ed)). *Automata, Languages and Programming.* Lecture Notes in Computer Science 154. Berlin: Springer-Verlag. p 662-675.

Embedding a Demonic Semilattice in a Relation Algebra

Jules Desharnais[2]

We present a refinement ordering between binary relations, viewed as programs or specifications. This ordering induces a complete join semilattice that can be embedded in a relation algebra. This embedding then allows an easy proof of many properties of the refinement semilattice, by making use of the well-known corresponding properties of relation algebras. The operations of the refinement semilattice corresponding to join and composition in the embedding algebra are, respectively, demonic join and demonic composition. The weakest prespecification and postspecification operators of Hoare and He, defined over a relation algebra, also have corresponding operators in the semilattice. Finally, this work helps to understand better how demonic relational operators can be used in specifications.

[2] Joint work with N. Belkhiter, S. B. M. Sghaier, F. Tchier, A. Jaoua, Département d'informatique, Université Laval, Québec, QC, G1K 7P4 Canada and A. Mili, N. Zaguia, Department of Computer Science, University of Ottawa, Ottawa, ON, K1N 6N5 Canada

A Relation-Algebraic Model of Robust Correctness

Thomas F. Gritzner[3]

A new abstract relational approach is proposed to describe the semantics of a demonic nondeterministic language and to give a model of robust correctness, both based on Hoare's chaos semantics. We concentrate entirely on the semantic level and consider that relations are programs.

First, a relation-algebraic model of a simple demonic nondeterministic programming language without iteration and recursion is presented. Essentially, this model is given by the set $\mathcal{R}$ of strict, total, and upwards closed relations on a flat lattice, an idempotent functional $\mathcal{C}$ on relations which characterizes $\mathcal{R}$, and a basic set of semantical constants and operators... The set $\mathcal{R}$ of "allowed" interpretations of programs forms a complete lattice with respect to an ordering called $\sqsubseteq_{\mathcal{C}}$. Furthermore, the basic operators are at least monotonic. Using fixed point machinery, therefore, we can generalize the model to the full language including semantical operators for iteration and general recursion, too.

Next, the focus is put on the refinement of programs. In the case of demonic nondeterminism, refinement leads to robustly correct implementations. We show the correctness of the unfold/fold rule for robust correctness similarly to the proof of the erratic case, which has been achieved in 1990 by the second author.

We also investigate relationships to Dijkstra's wp-calculus and Morgan's specification statement. Analogous to the use of $\mathcal{R}$ and $\mathcal{C}$, these relational interpretations of $wp(R, Q)$ and $[P, Q]$ are essentially based on a characterization of the set $\mathcal{Q}$ of "allowed" interpretations of pre- and postconditions using an idempotent functional $\mathcal{B}$. We show that our ordering $\sqsubseteq_{\mathcal{C}}$ of robust correctness exactly corresponds to the ordering based on weakest precondition semantics and that Hoare triples of total correctness can be expressed by both wp and specification statements, in our approach.

Finally, first attempts are made to deal with weakest liberal conditions and to support the search for a model of strongest total postconditions.

RALF – A Relation-Algebraic Formula Manipulation System

Claudia Hattensperger

Relation algebra is based on a small set of axioms, hence, a proof-supporting (of course not an automatic) computer system can easily be implemented and so manipulations can be checked with computer assistance.

Interaction with RALF is via a graphical user interface. A RALF session is centered around one theorem with possibly an incomplete or complete proof attached to it. Theorems are built up over the usual language of relations enriched by some additional (predefined or user-defined) functions and predicates using the propositional connectives. There are two major modes, one for proving and one for inspecting finished proofs. The theorem is presented as a tree to make even complex expressions easy to grasp.

[3] Joint work with R. Berghammer

While proving, the user will mark a subtree for transformation, and the system will show all the mathematically correct transformations it could apply to this expression according to its rule base. Upon selection by the user the system applies the chosen rule. As the proof strategy consists of reducing the formula to a primitively true one, RALF examines after each transformation whether the proof is finished, i.e., whether RALF recognizes the expression as true or false.

Metarules like

$$A = B \longleftarrow (A < B) \text{ META\_AND } (A > B)$$

have been implemented to split the proof into subproofs administered by the system.

The theorem and its proof can be saved, regardless of whether the proof is finished or not. Thus, the user is able to load a theorem with unfinished proof and continue proving it. Proven theorems can be reused as transformation rules in proofs.

RALF is a multi-user system implemented in C within OpenWindows 3.0.

Handling Intervals with Relation Algebra

Robin Hirsch

Given a representation of a relation algebra we construct relation algebras of *pairs* and of *intervals*. If the representation happens to be *complete, homogeneous* and *fully universal* then the pair and interval algebras can be constructed direct from the relation algebra. If, further, the original RA is ω-categorical we show that the interval and pair algebras are too. The complexity of relation algebras is studied and it is shown that every pair algebra with infinite representation is intractable. Applications include constructing an interval algebra with metric and interval expressivity.

Membership of Datatypes

Paul Hoogendijk[4]

A goal of a theory of datatypes is to identify concepts that are common to *all* (or a large number of) datatypes and that are fundamental to the specification and/or solution of a variety of program problems. An example of such a concept is the notion of a membership relation. Datatypes record the presence of elements, so one would expect datatype F to come equipped with a membership relation ϵ such that b(ϵ)x holds precisely when b is an element of data structure x. Indeed, this notion of membership is so common that its definition is usually taken for granted. During the talk, we give a formal definition of a membership relation and give some of its properties.

Another concept is the notion of a *strong* relator. A relator is said to be strong if it has a so-called (tensorial) *strength*. We will show that, under some mild conditions, the existence of membership implies that the relator is strong, and that its strength is unique. Related to strong functors, we have the notion of a *strong* natural transformation: apart from being natural it must satisfy the appropriate coherence

[4] Joint work with O. de Moor and P. Freyd

condition with respect to the strengths of both relators. We will show that every natural transformation between two relators with membership is strong. A corrollary of this result is that for a relator F with membership, every monad (F, η, μ) is a *strong* monad i.e. η and μ are strong.

Rectangular Decomposition of n-ary Relations: Application to Database Decomposition

Ali Jaoua[5]

System decomposition is a central problem in several areas of computer science, such as software engineering, database or human-computer interaction. The behaviour of many systems may be described by an n-ary relation. Hence, it becomes important to find optimal methods for n-ary relation decomposition as a generalisation of binary relation decomposition. Two well known methods for binary relation decompositions are: the Galois lattice of maximal rectangles, and the minimal set of optimal rectangles. The second method is more attractive because it is less expensive than Galois lattice in terms of space. We call rectangle RE any cartesian product of two sets D and C (i.e. RE = D $\times$ C). Starting from an n-ary relation, we replace any tuple by all possible pairs of its elements. Each element v is replaced by a pair (v,i) where i represents its order. As a result of this transformation we obtain an equivalent binary relation that does not depend on the order of its elements. When we apply rectangular decomposition on obtained binary relation, we can deduce some methods to organise the initial n-ary relation. The proposed method can be used to minimise redundancy, for database and system structuring, data analysis and classification, and for automatic entity generation. Experimentation of this approach on real databases has given surprisingly good results.

Relation-Algebraic Treatment of Term Graphs

Wolfram Kahl

The main concern of our research is to arrive at an algebraic characterisation of graph reduction and, more generally, term graph rewriting with a power comparable to that of combinatory rewriting systems. Relation-algebraic notation and reasoning are the tools we employ for realising our efforts.

The talk first gave a short account of the heterogeneous way of dealing with products and also sketched treatment of other datatypes, such as sequences.

This was then put to use for defining a notion of *simple term graph* capturing only the interaction between node labelling and the successor function. We introduced subgraph and quotient graph constructions. The key parts of the well-definedness proofs of these constructions were presented in two fashions. One fashion is purely relation-algebraic reasoning employing product types, that is, working with constructions equivalent to fork. The other fashion is to introduce variables for points (univalent total

[5] Joint work with H. Ammari, K. Arour, H. Ounelli, N. Belkhiter, Département d'informatique, Université Laval, Québec, QC, G1K 7P4 Canada

vectors always present in concrete relation algebras) and thus softening the component-free approach. As in this latter setting the power of predicate logic is available at the metalevel, proofs trend to be shorter than in the former, where that power has to be simulated within relation calculus.

We continued to give a notion of *term graphs with variables*, including variable binding and variable identity as primitive concepts in the definition of the graph, instead of accepting them as derived concepts as in usual settings. Since variable binding has to be postulated as dominating in the graph theoretic sense, the proofs of the subgraph and quotient graph constructions have to include proofs for that. The former works out nicely in both fashions; for the latter, however, the "point proof" already is so complicated that trying to construct a "product proof" becomes extremely hard and unrealistic in an application oriented context.

The whole exercise was presented here as an example of application of relation algebra to a different field. We conclude that the very compact notation of relation algebra and the concise reasoning it supports can be put to great use in applications — in our case, the more complicated proofs for combinatory term graph rewriting are hardly imaginable without having this toolbox available.

We pleaded that for relation algebra to find more followers in application fields, more tools have to be provided to support calculations, including tools for more confident treatment of "point proofs" and easier handling of "product proofs".

Connections between Predicate Transformer Semantics, Relational Semantics, and Demonic Semantics

Roger D. Maddux

Consider a language $\mathcal{L}$ with *predicates*: $B, \ldots,$ *basic commands*: havoc, abort, skip, ..., and *compound commands*: $S;T$, $S \,\mathsf{or}\, T$, $\mathsf{while}\, B \,\mathsf{do}\, S$, ...

A **predicate transformer interpretation** of $\mathcal{L}$ consists of (1) a complete Boolean algebra $\mathcal{B} = \langle |\mathcal{B}|, +, \cdot, ^-, 0, 1\rangle$, (2) for every predicate B, an element d_B of $\mathcal{B}$, (3) for every command S, two predicate transformers $\mathrm{wp}_S\,(\text{-})\,, \mathrm{wlp}_S\,(\text{-}) : \mathcal{B} \to \mathcal{B}$, such that

(a) $\mathrm{wp}_S\,(x) = \mathrm{wlp}_S\,(x) \cdot \mathrm{wp}_S\,(1)$,
(b) $\mathrm{wlp}_S\,(\text{-})$ is universally conjunctive and $\mathrm{wp}_S\,(\text{-})$ is positively conjunctive,
(c) $\mathrm{wlp}_{\mathsf{havoc}}\,(x) = 0\dagger x$ and $\mathrm{wp}_{\mathsf{havoc}}\,(x) = 0\dagger x$,
(d) $\mathrm{wlp}_{\mathsf{abort}}\,(x) = 1$ and $\mathrm{wp}_{\mathsf{abort}}\,(x) = 0$,
(e) $\mathrm{wlp}_{\mathsf{skip}}\,(x) = x$ and $\mathrm{wp}_{\mathsf{skip}}\,(x) = x$,
(f) $\mathrm{wlp}_{S_0;S_1}\,(x) = \mathrm{wlp}_{S_0}\Big(\mathrm{wlp}_{S_1}\,(x)\Big)$ and $\mathrm{wp}_{S_0;S_1}\,(x) = \mathrm{wp}_{S_0}\Big(\mathrm{wp}_{S_1}\,(x)\Big)$,
(g) $\mathrm{wlp}_{S\,\mathsf{or}\,T}\,(x) = \mathrm{wlp}_S\,(x) \cdot \mathrm{wlp}_T\,(x)$ and $\mathrm{wp}_{S\,\mathsf{or}\,T}\,(x) = \mathrm{wp}_S\,(x) \cdot \mathrm{wp}_T\,(x)$,
(h) $\mathrm{wlp}_{\mathsf{while}\,B\,\mathsf{do}\,S}\,(x)$ is the greatest fixed point of $\overline{\mathrm{d}_B}\cdot x + \mathrm{d}_B \cdot \mathrm{wlp}_S\,(\text{-})$ and $\mathrm{wp}_{\mathsf{while}\,B\,\mathsf{do}\,S}\,(x)$ is the least fixed point of $\overline{\mathrm{d}_B} \cdot x + \mathrm{d}_B \cdot \mathrm{wp}_S\,(\text{-})$.

A **relational interpretation** of $\mathcal{L}$ consists of (1) a complete relation algebra $\mathcal{A} = \langle |\mathcal{A}|, +, \cdot, ^-, 0, 1, ;, ^\smile, 1'\rangle$, (2) for every predicate B, an element d_B of $\mathcal{A}$ satisfying $\mathrm{d}_B;1 = \mathrm{d}_B$, (3) for every command S, an element r_S of $\mathcal{A}$ and an element e_S of $\mathcal{A}$ satisfying $\mathrm{e}_B;1 = \mathrm{e}_B$, such that

(c′) $\mathrm{r}_{\mathsf{havoc}} = 1$ and $\mathrm{e}_{\mathsf{havoc}} = 0$,

(d′) $r_{\mathsf{abort}} = 0$ and $e_{\mathsf{abort}} = 1$,
(e′) $r_{\mathsf{skip}} = 1'$ and $e_{\mathsf{skip}} = 0$,
(f′) $r_{S;T} = r_S ; r_T$ and $e_{S;T} = e_S + r_S ; e_T$,
(g′) $r_{S \,\mathsf{or}\, T} = r_S + r_T$ and $e_{S \,\mathsf{or}\, T} = e_S + e_T$,
(h′) $r_{\mathsf{while}\, B\, \mathsf{do}\, S}$ is the least fixed point of $\overline{d_B} \cdot 1' + d_B \cdot r_S ; (\text{-})$ and $e_{\mathsf{while}\, B\, \mathsf{do}\, S}$ is the greatest fixed point of $d_B \cdot (e_S + r_S ; (\text{-}))$.

The associated transformers of a relational interpretation are defined by $\mathrm{wlp}_S(x) = \overline{r_S ; \overline{x}}$ and $\mathrm{wp}_S(x) = \overline{r_S ; \overline{x}} \cdot \overline{e_S}$. This produces a predicate transformer intepretation.

A **demonic interpretation** of $\mathcal{L}$ consists of (1) a complete relation algebra $\mathcal{A}$, (2) for every predicate B, an element d_B of $\mathcal{A}$ such that $d_B ; 1 = d_B$, (3) for every command S, an element r_S of $\mathcal{A}$ such that
(c″) $r_{\mathsf{havoc}} = 1$
(d″) $r_{\mathsf{abort}} = 0$
(e″) $r_{\mathsf{skip}} = 1'$
(f″) $r_{S;T} = r_S ; r_T \cdot \overline{r_S ; \overline{\overline{r_T ; 1}}}$,
(g″) $r_{S \,\mathsf{or}\, T} = (r_S + r_T) \cdot r_S ; 1 \cdot r_T ; 1$,
(h″) $r_{\mathsf{while}\, B\, \mathsf{do}\, S}$ is the least fixed point of $\overline{d_B} \cdot 1' + d_B \cdot r_S ; (\text{-}) \cdot \overline{r_S ; \overline{\overline{(\text{-}) ; 1}}}$.

In a demonic interpretation, the role of $\overline{e_S}$ is played by $r_S ; 1$, so its associated transformers are defined by $\mathrm{wlp}_S(x) = \overline{r_S ; \overline{x}}$ and $\mathrm{wp}_S(x) = \overline{r_S ; \overline{x}} \cdot r_S ; 1$. This produces the $\mathrm{wp}_{\text{-}}(\text{-})$ half of a predicate transformer interpretation.

Relational and demonic interpretation look different, since the demonic one uses "demonic composition" and "demonic union" in place of the usual ones. Any map from predicates to domain elements of $\mathcal{A}$ that is well-behaved on the basic commands can be extended to a unique relational interpretation, and also to a unique demonic interpretation. But there a connection between the two interpretations so obtained. For every relational intepretation define $\hat{r}_S = r_S \cdot \overline{e_S}$. If $e_S + r_S ; 1 = 1$ for every command S, then $\hat{r}_S$ is a demonic interpretation. Furthermore, it doesn't matter whether r_S or $\hat{r}_S$ is used in the definition of $\mathrm{wp}_{\text{-}}(\text{-})$, because they both produce the same function. The fact that demonic intepretations give half of a predicate transformer interpretation now follows from the fact that relational interpretation produces predicate transformer intepretations. Finally, the demonic interpretation obtained from a well-behaved map on the basic statements can be obtained by first extending the map to a relational iterpretation, and then using $\hat{r}_S$ in place of r_S.

Does it Make a Difference? Extending Weak Associative Relation Algebras with the Difference Operator

Maarten Marx[6]

The variety RRA of Representable Relation Algebras has, due to its great expressive power, very bad meta–properties. Among other things it is not finitely Hilbert style axiomatizable and its equational theory is undecidable. Especially its complexity makes RRA sometimes not very suitable for applications in Computer Science.

[6] Joint work with S. Mikulás, I. Németi and A. Simon

The variety RWA of Weak Associative Representable Relation Algebras (defined below) on the contrary is finitely axiomatizable (a result of R. Maddux [3]) and its equational theory is decidable (proved by I. Németi [7]). This is a rather general phenomenon in algebraic logic, captured by the following slogan.

Slogan 1 Relativization is a way of turning negative results into positive ones.

We loose however a lot of expressive power by relativization; operators like the universal modality $\Diamond$, the difference operator $\mathcal{L}_D$, and the counting modalities $\Diamond_1$ and $\Diamond_2$ (all defined below), are term–definable in RRA, but aren't anymore in RWA.

We show that we can add the difference operator to RWA without loosing it's nice properties. Similar results can be obtained if one adds (all) counting modalities to RWA. The theorems reported here form an example of a research project the authors are currently working on and which is captured by slogan 1 and the following slogan.

Slogan 2 "Relativize to turn things positive, and then inject as much extra power as you can without loosing "positiveness"."

As argued in Sain [9], [11], [10], the difference operator and the counting modalities can help us in proving more program properties (they add proof-theoretic power to logics of programs and actions).

Definition 1 An algebra $A = \langle \mathcal{P}(W), \cup, -, \circ^W, ^{-1}, Id \rangle$ is called a full *RWA* if W is a reflexive and symmetric binary relation and $x \circ^W y \stackrel{\text{def}}{=} (x \circ y) \cap W$. RWA denotes the variety generated by all full *RWA*'s.

Definition 2 Let W be a set and define the following unary operations on $\mathcal{P}(W)$:

$$\begin{array}{lcl} \Diamond x & \stackrel{\text{def}}{=} & \{w \in W : (\exists v) : v \in x\} \\ \mathcal{L}_D x & \stackrel{\text{def}}{=} & \{w \in W : (\exists v) : v \neq w \ \&\ v \in x\} \\ \Diamond_n x & \stackrel{\text{def}}{=} & W \text{ if } \mid x \mid \geq n \text{ else } \Diamond_n x = \emptyset \end{array}$$

It is easy to see that having the $\mathcal{L}_D$ and the booleans we can define the universal modality as $\Diamond x \stackrel{\text{def}}{=} \mathcal{L}_D x \cup x$, which means the same as $\Diamond_1$ and the counting modality $\Diamond_2 x \stackrel{\text{def}}{=} \mathcal{L}_D(x \cap \mathcal{L}_D x)$.

Let RWA^D be RWA enriched with the difference operator.

Theorem 3 *The equational theory of* RWA^D *is decidable.*

Theorem 4 RWA^D *is a discriminator variety axiomatizable by finitely many axioms.*

The above results also have applications in the field of (Weak) Peirce Algebras.

For information on the difference operator we refer to Sain [10] and the PhD theses of de Rijke [8] and Venema [12], an extended discussion about the counting operators can be found in van der Hoek [2]. Peirce Algebras are discussed in Brink et al. [1] and in de Rijke [8], Weak Peirce Algebras in Marx [4]. An extended discussion about *arrow logic* and more on relativization of relation algebras can be found in Marx et al. [5].

References

[1] Brink, C., Britz, K., and Schmidt, R. *Peirce Algebras.* To appear in: Formal Aspects of Computing (1993).

[2] van der Hoek, W. *Modalities for Reasoning About Knowledge and Quantities.* PhD thesis, Amsterdam 1992.

[3] Maddux, R. *Some varieties containing relation algebras.* Trans. Amer. Math. Soc. 272 (1982), 501-526.

[4] Marx, M. *Dynamic Arrow Logic with Pairs.* CCSOM Preprint, to appear in [6], Amsterdam 1992.

[5] Marx, M., Mikulás, S., Németi, I., and Sain, I. *Investigations in Arrow Logic.* CCSOM Preprint to appear in [6], Amsterdam 1992.

[6] Marx, M., Masuch, M., and Pólós, L. (eds) *Arrow Logic and Multi–Modal Logics (Logic At Work).* Preproceedings of the Applied Logic Conference, 1992, Amsterdam.

[7] Németi, I. *Decidability of Relation Algebras with Weakened Associativity.* Proc. Amer. Math. Soc. volume 100, Number 2, 1987.

[8] de Rijke, M. *Extending Modal Logic.* ILLC Dissertation Series, 1993-4, University of Amsterdam.

[9] Sain, I. *Successor axioms for time increase the program verifying power of full computational induction.* Preprint Math. Inst. Hungar. Acad. Sci. Budapest, No. 23/1983 (1983).

[10] Sain, I. *Is "Some–Other–Time" sometimes better than "Sometime" in proving partial correctness of programs?* Studia Logica, Vol. XLVII, No. 3, 1988, 279–301.

[11] Sain, I. *Nonstandard Logics of Programs.* Dissertation, Hung. Acad. Sci. Budapest (in Hungarian) 1986.

[12] Venema, Y. *Many-Dimensional Modal Logic.* PhD thesis, Amsterdam 1992.

Rectangular Density Implies Representability

Szabolcs Mikulás[7]

It is known that every rectangularly dense *atomic* cylindric (CA) and quasi-polyadic equality (QPEA) algebra is representable as an algebra of relations, cf. Henkin et al. [2]. In this abstract we claim that the atomicity condition is not necessary, i.e., that every rectangularly dense CA and QPEA is representable.

For notation and basic definitions we refer the reader to [2].

Definition 1 Let α be any ordinal and V_α be one of CA_α or QPEA_α. Let $\mathcal{A} \in \mathsf{V}_\alpha$ and $a \in A$. We say that a is **rectangular** iff

$$\mathsf{c}_{(\Gamma)}a \cdot \mathsf{c}_{(\Delta)}a = \mathsf{c}_{(\Gamma\cap\Delta)}a$$

for all finite subsets Γ and Δ of α.

We say that $\mathcal{A}$ is **rectangularly dense** iff

$$(\forall 0 \neq a \in A)(\exists 0 \neq b \in A)(b \leq a \;\&\; b \text{ is rectangular.})$$

[7]Joint work with I. Németi

Let us formulate our main theorem.

Theorem 2 Let α be any ordinal and $\mathsf{V}_\alpha \in \{\mathsf{CA}_\alpha, \mathsf{QPEA}_\alpha\}$. Let RV_α denote the class of representable elements of V_α and VR_α denote the class of rectangularly dense elements of V_α. Then

$$\mathsf{RV}_\alpha = \mathbf{SP}\mathsf{VR}_\alpha.$$

Theorem 2 above is a consequence of the following two theorems.

Theorem 3 Let $\alpha \in \omega$ and V_α be as in Theorem 2 above. Let $\mathcal{A}$ be a simple, rectangularly dense element of V_α. Then $\mathcal{A}$ is representable.

The following theorem is a consequence of a more general one in Andréka et al. [1].

Theorem 4 Let $\alpha \in \omega$ and V_α be as in Theorem 2 above. Let $\mathcal{A}$ be a rectangularly dense element of V_α, and assume that the universe of $\mathcal{A}$ is countable, $|A| \leq \omega$. Then there are simple elements $\mathcal{A}_a$ $(0 < a \in A)$ of V_α such that every $\mathcal{A}_a$ is rectangularly dense and $\mathcal{A} \in \mathbf{SP}\{\mathcal{A}_a : 0 < a \in A\}$.

References

[1] H. Andréka, S. Givant, I. Németi, *Perfect Extensions of Boolean Algebras with Operators and Derived Algebras*, 1992.

[2] L. Henkin, J. D. Monk, A. Tarski, *Cylindric Algebras, Parts I., II.*, North-Holland, 1971, 1985.

Ideal Streams

Bernhard Möller

We introduce operators and laws of an algebra of formal languages, a subalgebra of which corresponds to the algebra of (multiary) relations. An essential operation is the join which models gluing of traces. The closure with respect to this operations gives rise to a Kleene algebra so that all laws known from regular algebra apply in this setting.

This algebra is then used to give a simplified semantics for (sets of) finite and infinite streams, based on the ideal completion. In this way we also give a simple treatment of non-determinacy. We show how some essential operations on streams and notions concerning correctness can be expressed algebraically and show the use of the equational laws in reasoning about streams. The approach is illustrated with the formal description and correctness proof for the alternating bit protocol.

This study is part of an attempt to single out a framework for program development at a very high level of discourse, close to informal reasoning but still with full formal precision.

The Connection between Predicate Logic and Demonic Relation Calculus

Thanh Tung Nguyen

We show that the demonic relation calculus—an algebraic apparatus for defining the denotational semantics of Dijkstra's guarded-command language—is isomorphic to the predicate transformer calculus. Here is the main result:

Theorem. Let $U \equiv \prod_{1 \leq k \leq n} D_k$ be a state space. Let
- x$\mathcal{R}$ be the space of (binary) relations on U,
- $\mathbf{I} \in \mathcal{R}$ the identity relation,
- "$\cdot$" the left-restriction,
- "$;\,;$" the demonic composition,
- "$\oplus$" the demonic union,
- "$\sqsubseteq$" the restriction-of ordering,
- $\mathcal{T}$ the space of predicate transformers—strict and positively conjunctive functions from state predicates to state predicates,
- $\epsilon \in \mathcal{T}$ the identity predicate transformer,
- $\perp \in \mathcal{T}$ the constantly-false predicate transformer,
- "$*$" the logical restriction defined by $(P * \gamma)(Q) = P \wedge \gamma(Q)$ for any Q,
- "$\circ$" the usual function composition,
- "$\sqcap$" the conjunction defined by $(\gamma \sqcap \delta)(Q) = \gamma(Q) \wedge \delta(Q)$ for any Q,
- "$\rhd$" the less-defined-than ordering defined by $\gamma \rhd \delta \Leftrightarrow \gamma = \gamma(\mathsf{true}) * \delta$,
- $\mathcal{R}_{\sqsubseteq}$ the algebraic cpo $\langle \mathcal{R}, \mathbf{I}, \emptyset, \cdot, ;\,;, \oplus, \sqsubseteq \rangle$,
- $\mathcal{T}_{\rhd}$ the algebraic cpo $\langle \mathcal{T}, \epsilon, \perp, *, \circ, \sqcap, \rhd \rangle$,
- $\mathcal{R}^*_{\sqsubseteq}$ the space of relationals—continuous functions from relations to relations,
- $\mathcal{T}^*_{\rhd}$ the space of transformators—functions from predicates transformers to predicate transformers,
- $\mathbf{fix}_{\sqsubseteq}$ the function from relationals to theirs least fixed points, and
- $\mathbf{fix}_{\rhd}$ the function from transformators to their least fixed points.

Then we have

$$\begin{array}{llcll} (\mathsf{iso1}) & \mathcal{R}_{\sqsubseteq} & = & \mathcal{T}_{\rhd} & \text{up to } \mathbf{wp}\text{-isomorphism} \\[2ex] (\mathsf{iso2}) & \mathcal{R}^*_{\sqsubseteq} & = & \mathcal{T}^*_{\rhd} & \text{up to } \nabla\text{-isomorphism} \\[2ex] (\mathsf{iso3}) & \mathbf{wp} \circ \mathbf{fix}_{\sqsubseteq} & = & \mathbf{fix}_{\rhd} \circ \nabla. & \end{array}$$

where
- $\mathbf{wp}$ is the function from relations to predicate transformers obtained by Currying the $\mathbf{wp}$ operator, that is, for any $\mathbf{R}$, $\mathbf{wp}(\mathbf{R})(Q) = \mathbf{wp}(\mathbf{R}, Q)$ for any Q,
- ∇ from relationals to transformators defined by $\nabla(\Phi) = \mathbf{wp} \circ \Phi \circ \mathbf{wp}^{-1}$ for any Φ.

Towards Automating Dualities

Hans Jürgen Ohlbach

Dualities between different theories occur frequently in mathematics and logic — between syntax and semantics of a logic, between structures and power structures, between relations and relational algebras, to name just a few. A structure is a set with some relations on it. The power structure is the powerset of the set and for each relation on the set there is a corresponding function on the power structure. The duality problem is the problem to find for particular properties of the relation corresponding properties of the function and vice versa. The functions in power structures correspond to operators in nonclassical logics. Therefore this duality problem corresponds to the correspondence problem in nonclassical logics. In the talk I show for the case of structures and power structures how corresponding properties of the two related structures can be computed fully automatically by means of quantifier elimination algorithms and predicate logic theorem provers. The method is illustrated with a number of examples solved with help of the theorem prover OTTER.

Relational Semantics and Relational Proof Systems for Nonclassical Logics

Ewa Orlowska

Modelling incomplete information in the relational framework is discussed. We consider information systems such that the explicit data given in the system have the form of a list of objects and properties. From explicit data we derive implicit information that is modelled by means of classes of information relations. The relations generate algebras of relations that serve as basis of Kripke frames for the underlying information logics. Relational semantics for information logics is defined such that formulas are interpreted as relations. Relational proof systems for information logics are given.

Defining relational logics and presenting relational proofs with the graphic logical editor of ATINF (R. Caferra, M. Herment, E. Orlowska). Specification of relational logics in the graphic editor of ATINF is done in terms of a definition language that is based on the calculus of construction. Communication with interference tools is realized at the level of a presentation language. After syntactic verification, proofs are displayed in boxes. Various options for manipulation of boxes are available.

Tarski's Vision Revisited: Mathematics Founded on a Calculus of Binary Relations

Vaughan Pratt

Tarski's vision was to found mathematics on the Peirce-Schroeder calculus of binary relations abstracted to the variety RA. We organize the class of all binary relations between any pair of sets into a category whose morphisms transform rows covariantly and columns contravariantly, called the category Chu(2) of Chu spaces over 2. Chu(K) generalizes this to K-valued matrices, i.e. whose elements are drawn from a fixed set K.

Interpreting the operations of linear logic constructively in Chu(K) yields a calculus CLL that closely parallels that of RA. CLL differs from RA in the following essential ways.

1. Constructive. Entailments $R \vdash S$ are truth-valued in RA (0 or 1), set-valued in CLL (the set of proofs or moves from R to S).

2. Contravariant. All relations go from a covariant set A to a contravariant set X, in the sense that a morphism from R to R' consists of two maps $f : A \longrightarrow A', g : X' \longrightarrow X$, satisfying $f(a)R'x = aRg(x)$. The effect is to make converse (transpose) play the role that complement-of-converse plays for RA.

3. Concurrent. RA's sequential (noncommutative) composition is replaced by parallel (commutative) interaction or orthocurrence, as with the structure of the six events when a sequence of three trains passes through a sequence of two stations.

4. Concrete. In CLL the row index set of R is treated as its underlying set, obtained as $!R$. RA relations have no corresponding uniform notion of underlying set.

This organization works for relations whose truth values come from any fixed set K, not just 2. When $K = 2^n$, this category realizes (fully and concretely embeds) the category Str_n of n-ary relational structures and their homomorphisms. In turn all major categories of mathematics embed in Str_n for some (typically small) n, e.g. n=3 for groups and semigroups, 4 for monoids, etc., making Chu(2^n) a universal self-dual category for everyday mathematics.

Zooming In. Zooming Out.

Maarten de Rijke[8]

In the talk I draw attention to a phenomenon that seems to be appearing in many research areas nowadays: the phenomenon of *combined ontologies*. This term is used to refer to ontologies that consist of multiple component structures together with links between them. Examples and applications of combined ontologies can be found in the semantics of object oriented programming, verification of real-time systems, temporal databases, generative linguistics, the semantics of natural languages, and in many other fields.

The talk presents examples of combined ontologies, it mentions some of the logical issues they give rise to, and it concludes with some problems.

[8]Joint work with P. Blackburn

Unifying State-Based Formalisms for Proving Data Refinement

Willem–P. de Roever[9]

A number of state-based formalisms exist for proving data refinement: e.g., the method of representation invariants and auxiliary variables [2], VDM [1], Z [3, 5], Hoare [6], Back [7], Gardiner & Morgan [8]. Are these methods essentially different or do they amount to the same? We prove that Reynolds' method, VDM, Z, Hoare's method and Back's method all amount to *forward refinement* (modulo certain minor restrictions), and are therefore, by a result of [4] incomplete. The formalism we use to prove these results is a mixture of Hoare logic and a relational calculus. The key technique used is as follows: give a specification $\{pre_A\}A\{post_A\}$ of an abstract operation A, and a representation invariant α, we deduce a specification $\{pre_c\}C\{post_c\}$ at the concrete level such that whenever C satisfies the latter, C forward simulates the maximal solution $pre_A \rightsquigarrow post_A$ satisfying $\{pre_A\}A\{post_A\}$ with respect to α. The same question is answered for backwards simulation, U and U^{-1} simulation. Of the above mentioned formalisms which are proved equivalent two deal with total correctness: VDM and Back's. These are proved equivalent to forward simulation with respect to a formalism for proving total correctness in which the Smyth order is used as refinement relation.

References

[1] Cliff B. Jones. *Systematic Software Development using VDM.* Prentice-Hall, 1986.

[2] J.C. Reynolds. *The Craft of Programming.* Prentice-Hall, 1981.

[3] Anthony Diller. *Z – An Introduction to Formal Methods.* Wiley, 1990.

[4] C.A.R. Hoare, Jifeng He, and J.W. Sanders. *Prespecification in data refinement.* Information Processing Letters, 25:71-76, 1987.

[5] J.M. Spivey. *The Z Notation: A Reference Manual.* Prentice-Hall, second edition, 1992.

[6] C.A.R. Hoare. *Proofs of correctness of data representation.* Acta Informatica, 1:271-281, 1972.

[7] R.J.R. Back. *Data Refinement in the Refinement Calculus.* Reports on computer science & mathematics 68, Åbo Akademi, 1988.

[8] P. Gardiner and C. Morgan. *A single complete rule for data refinement.* ACM Transactions on Programming Languages and Systems, 1993.

An Algebraic Treatment of Graph Algorithms

Martin Russling

In books on algorithmic graph theory, algorithms are usually presented without formal specification and formal development. Here, in contrast, an algebra of formal languages and relations is used to *derive* graph algorithms. The use of this algebra is illustrated by derivations of a shortest path, a hamiltonian circuits, and a sorting algorithm.

All derivations are formal, understandable and concise.

[9] Joint work with J. Coenen and K. Engelhardt

The algebra proved to be suitable not only for dealing with graph algorithms but also for describing streams and deriving tree, pointer and sorting algorithms.

Fork Algebras in Usual as Well as in Non–Well–Founded Set Theories

Ildikó Sain[10]

At the Dagstuhl conference the question was raised whether Proper Fork Algebras would be finitely axiomatizable in the non–well founded set theories. It was proposed that this question is important. Most of the theorems below are true in usual set theory as well as in the set theories without the axiom of Foundation proposed so far in the literature. The theorems that are true in *both* kinds of set theory are marked as "(without Foundation)" after the number of the theorem.

The literature of Fork Algebras has been alive and productive for at least four years by now, see e.g. Veloso–Haeberer [13], [5], [14], [4].

On binary relations, say R and S, there are the well known set theoretic operations, e.g. $R \cup S$, $R \cap S$, $\ldots$, $R \circ S$, R^{-1}, Id (the identity relation). Recall that Proper Relation Algebras (PRA's) have these as operations (together with complementation of course).

¿From the above mentioned four–year old literature we recall a new binary set theoretic operation $\bigtriangledown$ called *"fork"* between binary relations. So, $R \bigtriangledown S$ is a new binary relation derived from R and S.

Notation: $\langle x, y\rangle$ is the usual set theoretic ordered pair of x and y. I.e., $\langle x, y\rangle = \{\{x\}, \{x, y\}\}$.

Definition 1 Let R and S be binary relations. Then $R \bigtriangledown S$ is a new relation defined as follows. $R \bigtriangledown S \stackrel{\text{def}}{=} \{\langle x, \langle y, z\rangle\rangle : xRy \ \&\ xSz\}$. $\mathcal{A}$ is called a Proper Fork Algebra (a PFA) iff $\mathcal{A} = \langle \mathcal{B}, \bigtriangledown\rangle$, where $\mathcal{B}$ is a PRA (i.e. a set relation algebra) closed under $\bigtriangledown$, and the operation $\bigtriangledown$ of $\mathcal{A}$ is as defined above.

The two books Aczél [1] and Barwise–Etchemendy [3] together review the literature of set theories without the axiom of Foundation and they seem to mention all Foundation–free set theories which were seriously proposed one time or another. What comes below is true in all the set theories without Foundation collected in [1], [3]. In our proofs we use only a small part of their axioms.

Theorem 1 (without Foundation) *(i) The class* IPFA *of isomorphic copies of Proper Forking Algebras is not axiomatizable by* any *set of first order sentences.*
(ii) The quasi–variety generated by the class PFA *is not finitely axiomatizable.*
(iii) The variety generated by the class PFA *is not finitely axiomatizable either.*
(iv) Statements (i)–(iii) above remain true in our usual set theory (ZF).

Theorem 2 ([7, Thm.1]; in ZF)
The equational theory $\mathrm{Eq}(\mathsf{PFA})$ *of* PFA *is* Π^1_1*–complete.*

[10] Joint work with I. Németi

Corollary 1 ([9, Thm.25.1]) Eq(PFA) *is not recursively enumerable.*

Definition 2 ([6], [7]) $\mathcal{A}$ is called a True Pairing Algebra (TPA for short) iff $\mathcal{A} = \langle \mathcal{B}, p, q \rangle$ where $\mathcal{B}$ is a PRA with greatest element $U \times U$, and $p \stackrel{\text{def}}{=} \{\langle\langle x, y\rangle, x\rangle : x, y \in U\}$ and $q \stackrel{\text{def}}{=} \{\langle\langle x, y\rangle, y\rangle : x, y \in U\}$ are distinguished constants of $\mathcal{A}$.

Theorem 3 (without Foundation) PFA *and* TPA are term definitionally equivalent.

Corollary 2 *Theorems 1–2 and Corollary 1 above are true for* TPA's in place of PFA's.

The above imply that *there are no representation theorems in terms of* PFA*'s or* TPA's; not even in set theories without the Axiom of Foundation.

Definition 3 $\mathcal{A}$ is a Nonstandard Fork Algebra (NFA) iff there are a set U and an injective function $f : U \times U \longrightarrow U$ such that $\mathcal{A} = \langle \mathcal{B}, \bigtriangledown^f \rangle$, where $\mathcal{B}$ is a PRA with greatest element $U \times U$, and for any $R, S \in B$, $R \bigtriangledown^f S = \{\langle x, f(y, z)\rangle : xRy \ \& \ xSz\}$ is also in B.

The fact that NFA behaves well was essentially proved by Tarski more than 40 years ago:

Theorem 4 (Tarski 1953) INFA *is a finitely axiomatizable variety (where* INFA *is the class of isomorphic copies of members of* NFA*).*

See Theorem 5 above and Tarski–Givant [12, items (i)–(iii) of section 8.4 on p.242] and Tarski [11, p.604, Theorem (VII)]. Maddux gave a purely algebraic proof for Tarski's theorem and this proof is disc Maddux gave a purely algebraic proof of Tarski's theorem and this proof is discussed substantially (with careful references etc.) in [12]. NFA is thoroughly investigated in [12] (in a term definitionally equivalent form, see Theorem 3 above).

What the Fork Algebra papers call "expressibility of first–order logic in the equational theorey Eq*(*NFA*) of* NFA*" was also proved by Tarski more than 40 years ago, see e.g. the footnote on p.242 in [12] and [11].*

In our opinion, because of these and of Theorem 4 above, papers investigating NFA should quote [12]. We note that the theory developed and carefully presented in [12] was continued e.g. in Andréka–Jónsson–Németi [2], Maddux [6], Németi [8].

POSITIVE RESULTS: Instead of only binary relations, consider all possible finitary relations over some set U.

Notation:

Ref(U) is the set of all finitary relations over U,

i.e. $\text{Ref}(U) \stackrel{\text{def}}{=} \{R :$ for some finite n, R is an n–ary relation over $U\}$.

Theorem 5 ([9], [10]) *It is possible to define set theoretic operations $f_1, \ldots, f_9$ on finitary relations such that (i), (ii) below hold.*

(i) The class REL *defined below is a finitely axiomatizable variety up to isomorphisms;*

REL $\stackrel{\text{def}}{=}$ The variety generated by $\{\langle \text{Ref}(U), f_1, \ldots, f_9\rangle : U$ is a set$\}$.

(ii) First order logic is expressible in (a natural way in) the equational language of REL.

References

[1] Aczél, P., Non–well–founded Sets. CSLI Lecture Notes No. 14, 1988.

[2] Andréka, H., Jónsson, B., and Németi, I., Free Algebras in Discriminator Varieties, Algebra Universalis, 28, 1991, 401–447.

[3] Barwise, J. and Etchemendy, J., The Liar, Oxford University Press, 1989.

[4] Berghammer, R., Haeberer, A. M., Schmidt, G., and Veloso, P. A. S., Comparing Two Different Approaches to Products in Abstract Relation Algebras. In Proceedings of the Third International Conference on Algebraic Methods and Software Technology, AMAST, Springer Verlag, 1993.

[5] Haeberer, A. M. and Veloso, P. A. S., Partial Relations for Program Derivation: Adequacy, Inevitability and Expressiveness. In Constructing Programs From Specifications - Proceedings of the IFIP TC2 Working Conference on Constructing Programs From Specifications, N.H., IFIP WG 2.1, 1993, 319–371.

[6] Maddux, R. D., Zeitschrift für math. Logik und Grundlagen d. Math., 1989, Part I in Bd. 35, Part II in Bd. 39 (1993).

[7] Mikulás, Sz., Sain, I., and Simon, A., Complexity of Equational Theory of Relational Algebras with Projection Elements. Bull. Section of Logic, 21, 3, 1992, 103–111, Full paper: Preprint of Math. Inst. Hungar. Acad. Sci. 1991.

[8] Németi, I., Dissertation for D.Sc. with Hung. Academy of Sciences, Budapest, 1986.

[9] Sain, I., On the Search for a Finitizable Algebraization of First Order Logic (Shortened Version A). Math. Inst. Hungar. Acad. Sci., 1988, 78 pp.

[10] Sain, I., On Finitizing First Order Logic and its Algebras. To appear, 1994.

[11] Tarski, A., Collected Papers, Vol. 4, Birkhäuser, 1986, 603–604.

[12] Tarski, A. and Givant, S., A Formalization of Set Theory Without Variables. 41, Providence, Rhode Island, 1987.

[13] Veloso, P. A. S. and Haeberer, A. M., A Finitary Relational Algebra for Classical First–order Logic. Bull. Section of Logic (Warsaw–Lódz), Vol. 20, No. 2, 1991, June, 52–62.

[14] Veloso, P. A. S., Haeberer, A. M., and Baum, G., Submitted to Automatic Programming in the Journal of Symbolic Computation.

Modal Rule Correspondences

Holger Schlingloff[11]

In this work we consider both the syntactical and semantical effects of adding certain derivation rules to the basic (multi–) modal system **K**. For a large class of modal axiom schemes we give simple rules corresponding to the same semantical condition.
E.g. the relation-algebraic condition

$$R^+ \subseteq S$$

can be characterized by the axiom

$$[S](p \to [R]p) \to (p \to [S]p)$$

[11] Joint work with W. Heinle, Ludwig–Maximilians–Universität München, Germany

as well as by the equivalent rule

$$p \to [R]p \vdash p \to [S]p.$$

In all cases these equivalences are derivable, i.e. proven without reference to the completeness of the logic.

Since our global consequence approach invalidates the deduction theorem, the use of rules increases definability in modal logics. We show that our method can be used to yield correspondences for rules extending the modal language.
E.g. the *disjunction rule*

$$[R]p \vdash p \qquad \text{corresponds to} \qquad \langle R^{\smile} \rangle \mathit{true},$$

and the *start–rule*

$$\mathit{start} \to [R]p \vdash p \qquad \text{corresponds to} \qquad \langle R^{\smile} \rangle \mathit{start}.$$

Special attention is given to the *irreflexivity–rule*

$$p \wedge \Box\neg p \to q \vdash q,$$

which can be used to complete certain incomplete logics, since it corresponds to irreflexivity and atomicity (i.e. the *point axiom*) of the underlying general frame. However, unlike in relation algebra, in modal logic the point axiom is *not* sufficient to determine the Kripke–structures among all modal algebras. This is proved by constructing an incomplete modal logic in which the irreflexivity rule is derivable.

A Relational Investigation on the Laws of Information Transmission

Gunther Schmidt

The talk started with some seemingly independent observations. Firstly, there exists a theory of functions in the presence of an ordering "is less defined than" but none for relations. Secondly, papers on wp-calculus usually deal with composition, if, while, but not with parallel composition of processes; at least not in the sense that they deduce the wp-rules for parallel composition also from the general fixedpoint principles. Thirdly, relational semantics has been given to sequential programs, but so far there is none for parallel ones.

It seems that all three situations can be handled constructing a new class of embedded relational algebras. As a first result it has been proved that there exists a multiplicative embedding of the algebra of relations between the sets A, B into the algebra of relations between the corresponding powersets. This embedding assigns the function $f_R := \mathsf{syq}(R^{\mathrm{T}}\varepsilon, \varepsilon)$ to the relation R, where syq denotes the symmetric quotient and ε the "is-element-of"-relation between a set and its powerset. The function f_R is continuous in the sense that $f_R^{\mathrm{T}}\,\mathrm{lub}(X) = \mathrm{lub}(f_R^{\mathrm{T}}X)$ for all relations X (including $X = 0$) where the least upper bound is taken with regard to the powerset ordering.

This embedding leads to defining new boolean operations on the subset of continuous functions f_R according to those for R.

Having achieved this, we consider a new type of relations. Instead of working with boolean matrices, we work with matrices the coefficients of which are the boolean functions mentioned before. So between an element a and an element b there may, or may not, just exist the relationship. Rather, there may be a difficult transfer of partialities of information established by the continuous function.

The new approach serves then as a means of giving new semantics to the Haeberer-Veloso fork-algebras. Furthermore, it is the basis for the construction of a nonstandard model of a relation algebra satisfying the unsharpness conjecture saying that there exist models of relations algebras where the relational product does not distribute over fork.

Modelling the transfer of the degrees of partiality of information by such functions f_R, is inherently a nonstrict approach. The question then arises as to how to manage the transition to the strict case. Here another relation algebra has been constructed studying the extremal case of the above one considering $|A| = 0$ and $|B| = 1$. The powersets then have 1 or 2 elements, respectively.

Peirce Algebras and Their Applications in Artificial Intelligence and Computational Linguistics

Renate Schmidt

In [1] we present a two-sorted algebra, called a *Peirce algebra*, of relations and sets interacting with each other. In a Peirce algebra, sets can combine with each other as in a Boolean algebra, relations can combine with each other as in a relation algebra, and in addition we have both a set-forming operator on relations (the Peirce product of Boolean modules) and a relation-forming operator on sets (a cylindrification operation). Peirce algebras provide useful formalisations for various fields in Computer Science. In this talk I focus on the application of Peirce algebras in artificial intelligence and computational intelligence. In particular, I show that the so-called *terminological logics* arising in knowledge representation (originating with a system called KL-ONE) have evolved a semantics best described as a calculus of relations interacting with sets [1,2,3]. In computational linguistics P. Suppes (1976,1979,1981) and M. Böttner (1992) use concrete Peirce algebras as a relational formalisation of the semantics of the English language. In [4] I link both these applications and show that Peirce algebra provides a useful bridge for utilising the linguistic investigations for the problem of finding adequate terminological representations for given information formulated in ordinary English.

References

[1] Brink, C., Britz, K. and Schmidt, R. A. (1994), Peirce Algebras. *Formal Aspects of Computing* **6**, 1–20.

[2] Brink, C. and Schmidt, R. A. (1992), Subsumption Computed Algebraically. *Computers and Mathematics with Applications* **23**(2–5), 329–342.

[3] Schmidt, R. A. (1991), Algebraic Terminological Representation. Master's Thesis, University of Cape Town.

[4] Schmidt, R. A. (1993), Terminological Representation, Natural Language & Relation Algebra. In the *Proceedings GWAI-92*, Vol. 671 of *LNAI*, 357–371.

Axiomatizing $\mathsf{Crs}_\alpha^\mathcal{G}$

András Simon[12]

Consider the following classes of algebras:

Definition 1 *Let $\alpha > 1$ be an ordinal and $\mathcal{G} \subseteq \mathcal{P}(\alpha)$. Then*

$$\mathsf{Crs}_\alpha^\mathcal{G} \stackrel{\text{def}}{=} \mathbf{S}\,\mathbf{P}\{\langle \mathcal{P}(V), \cup, -, V, \mathrm{C}_\Gamma^{[V]}, \mathrm{D}_{ij}^{[V]}\rangle_{\Gamma\in\mathcal{G}, i,j\in\alpha} : V \subseteq {}^\alpha U \quad \textit{for some set } U\}.$$

Here $\mathrm{D}_{ij}^{[V]} = \{f \in V : f(i) = f(j)\}$ *and* $\mathrm{C}_\Gamma^{[V]}\, x = \{f \in V : (\exists g \in x) f\lceil(\alpha\backslash\Gamma) = g\lceil(\alpha\backslash\Gamma)\}$ *if* $x \subseteq V$.

In this paper we we present a finite schema $\Sigma_\alpha^\mathcal{G}$ of equations in the language of $\mathsf{Crs}_\alpha^\mathcal{G}$. ($\Sigma_\alpha^\mathcal{G}$ is not finite schema in Monk's sense but it is finite schema in the sense in which Resek's axioms for Crs_α are.)

Theorem 1 *Assume that $\mathcal{G}$ does not contain infinite subsets of α except possibly α itself. Then $\mathbf{I}\,\mathsf{Crs}_\alpha^\mathcal{G}$ is axiomatized by $\Sigma_\alpha^\mathcal{G}$.*

Theorem 2 *If $\mathcal{G}$ does not satisfy the assumption of Theorem 1 then $\mathbf{I}\,\mathsf{Crs}_\alpha^\mathcal{G}$ is not axiomatizable by any set of first order formulas. In fact, it is not even a pseudo-elementary class.*

If $\mathcal{G}$ is the set $\{\{i\} : i \in \alpha\}$ of all singletons, then $\mathsf{Crs}_\alpha^\mathcal{G}$ coincides with the class Crs_α of cylindric relativized set algebras. Resek [2] proved that Andréka's strong non-finitizability result for cylindric set algebras Cs_α does not extend to Crs_α. Our Theorem 1 shows that the same holds for $\mathsf{Crs}_\alpha^\mathcal{G}$ (in place of Crs_a). By contrast, Németi showed (see [1]) that Monk's non-finitizability of Cs_α does extend to Crs_α. See [2] for more information. Inclusion of the generalized cylindrifications C_Γ for finite Γ's is a natural step first suggested by S. Comer, who noticed that these operations are not term-definable in Crs_α.

References

[1] J. D. Monk, *Lectures on Cylindric Relativized Set Algebras*, in: C. Rauszer ed. Algebraic Methods in Logic and in Computer Science, Banach Center Publications no. 28, Warsaw, 1993.

[2] I. Németi, *Algebraization of quantifier logics, an introductory overview,* 11th version, Preprint, Math. Inst. Hung. Acad. Sci., abstracted in Studia Logica L, 3-4(1991) 485-569.

[3] D. Resek, *Some results on Relativized Cylindric Algebras,* Doctoral dissertation, University of California, Berkeley, 1975.

[12]This is part of a joint work with M. van Lambalgen. Research supported by the Hungarian National Foundation for Scientific Research grants no. 1911 and 2258

Z-like Formal Development in Classical Set Theory

John Staples[13]

We begin by comparing the exploitation of abstraction in formal development methods with the use of abstraction in mathematics. It is suggested that it would be beneficial to exploit in the formal methods area the close working relationship between abstract and concrete theories which is a feature of much mathematics. Expected benefits include simpler, more modular abstract theories which are complementary rather than competitive. It might also be an apt way to reconcile the algebraic and model-based approaches to formal development.

A key issue is to identify and use at the concrete level the structures which the abstract theories characterise; it is not satisfactory to lose sight of the desired structures during translation from the abstract to the concrete. The paper illustrates this issue by taking the Z language as indicative of several abstractions relevant to formal specification. Classical set-theoretic counterparts of a range of Z concepts are developed, with the emphasis on mathematical aptitude rather than strict compatibility with the Z draft standard. This work is intended as a step towards a formal development methodology more comprehensive than Z, in which Z-like structures are characterised abstractly, and are also visible and useable in the underlying set theory. Examples of Z structures addressed include: Z variables, predicates, schemas, schema operations and quantification over Z variables. As an example of the potential of this approach to integrate abstractions not found in Z, we also demonstrate the compatibility of the approach with Hehner's informal characterisation of procedural statements as predicates relative to a frame of declared variables

Flownomials: Regular Expressions for Distributed Computation

Gheorghe Ştefanescu[14]

First, we survey an extension of Kleene's calculus of regular expressions to cope with atoms having many-input/many-output connecting ports, called the *calculus of flownomials*.

The syntax of the calculus is given by

$$E := E \oplus E | E \cdot E | E \uparrow^c \mid \wedge_m^a \mid {}^aX^b \mid \vee_a^n \mid x (\in X).$$

Here x is an atom, $\oplus$ is sum (parallel composition), $\cdot$ is (sequential) composition and $\uparrow$ is feedback and the constants are $\wedge_m^a$ (block ramification), ${}^aX^b$ (block transposition) and $\vee_a^n$ (block identification).

This new setting allows to single out some critical axioms

$$(1) f \cdot \wedge_m^b = \wedge_m^a \cdot (mf) \qquad\qquad (2) \vee_a^n \cdot f = (nf) \cdot \vee_b^n$$

[13]Joint work with Peter Kearney and Owen Traynor, all at Software Verification Research Centre, Department of Computer Science, The University of Queensland, Australia

[14]Partially joint work with J.A. Bergstra, S.L. Bloom, V.E. Căzănescu and Z. Esik

which are used in two ways: a *strong* version (arbitrary f) and a *weak* one (f relation).

If both strong axioms (1) and (2) are used, then this setting is equivalent to that of the classical regular expressions, hence it models the input-output behaviour of sequential nondeterministic programs. The case with (1) weak and (2) strong models bisimilar process graphs. The case with $m \leq 1$, (1) weak and (2) strong models the input behaviour of deterministic flowchart programs. Etc. The general case with both (1) and (2) weak gives an algebra for flowgraph programs (*not* for their behaviours) and may be applied to various graph-models used in distributed computation, e.g. dataflow nets, process graphs, synchronization nets, systolic automata, etc.

Finally we present a result showing that the equational theory of relations specified by the extension of regular expressions with conversion is decidable and display an equational axiomatisation.

Completeness through Flatness in Two-Dimensional Temporal Logic

Yde Venema

In various logic-related disciplines like artificial intelligence, computer science or natural language semantics, there is an approach towards a formal representation of the notion of time which is inspired by modal logic. In the last two decades, all these research areas have seen an interest in the development of two-dimensional modal systems, i.e. formalisms in which the truth of formulas of the language is evaluated at pairs of time points instead of at the points themselves.

In our talk we introduce a temporal logic TAL and prove that it has several nice features: many known formalisms with a two-dimensional flavor can be expressed in TAL. We first pin down the expressive power of TAL to the three-variable fragment of first-order logic; we prove that this induces an expressive completeness result of 'flat' TAL with respect to monadic first order logic (over the class of linear flows of time). Then we treat axiomatic aspects: our main result is a completeness proof for the set of formulas that are 'flatly' valid in well-ordered flows of time and in the flow of time of the natural numbers.

Relational Datatypes with Laws

Jaap van der Woude

A brief impression is given of the Spec calculus, an Eindhoven form of relational algebra. The introduction of a minimal typing for specs via domain kinds was examplified by the derivation of the monotype domain operator and the induced Galois connection between monotypes and vectors. A composition with the Galois connection consisting of spec composition and residuals led to the notion of monotype factor, which turns out to have all properties of the weakest liberal precondition predicate transformer. Guided by this simple and elegant wlp-expression programs are considered to be pairs (R,A), where R is the I/O relation and A is the domain of guaranteed termination.

The catenation of programs is derived from the composition of the (immediate) erratic predicate transformers. The comparison of three different partial orders on the programs was exploited to introduce two choice operators (the usual nondeterministic choice and the "unusual" fair choice (parallel)). Finally (following the work of Henk Doornbos) a program transformer of the form F.(R,A) = (F.R,R@A) was defined to generate program kF= (μ F, μ (μ F@)). It was shown that kF is the usual Egli-Milner least fixpoint provided the program transformer F is Egli-Milner monotonic, F and R@ are spec-monotonic and @A is antitonic. Since all transformers, Egli-Milner monotonically constructed using constants catenation and the choices, satisfy these conditions, this generalises the recursively defined programs with erratic nondeterminism.

Dagstuhl-Seminar 9403

Participants

Roland C. **Backhouse**
Eindhoven University of Technology
Dept. of Mathem. and Computing Science
P.O. Box 513
NL-5600 MB Eindhoven, The Netherlands
rolandb@win.tue.nl
tel.: +31-40-472744

Gabriel **Baum**
Univ. Nacional de la Plata
Dept. of Computer Science
Buenos Aires, Argentina
gbaum@unlp.edu.ar
tel.: +54-21-4 27 38

Rudolf **Berghammer**
Christian-Albrecht-Universität Kiel
Inst. für Informatik und Prakt. Mathem.
Preusserstrasse 1-9
D-24105 Kiel, Germany
rub@informatik.uni-kiel.d400.de
tel.: +49-431-56 04-86

Michael **Böttner**
MPI - Nijmegen
Max Planck Institute for Psycholinguistics
PB 310
NL-6500 AH Nijmegen, The Netherlands
boettner@mpi.nl
tel.: +31 (80) 521 911

Chris **Brink**
University of Cape Town
Department of Mathematics
Rondebosch 7700, South Africa
cbrink@maths.uct.ac.za

Jules **Desharnais**
Université Laval
Département d'Informatique
Québec QC G1K 7P4, Canada
desharn@ift.ulaval.ca
tel.: +1-418-656-3760

Thomas **Gritzner**
TU München
Institut für Informatik
D-80290 München, Germany
gritzner@informatik.tu-muenchen.de
tel.: +49-89-2105-8174

Armando **Haeberer**
Pontifícia Universidade Católica do
Rio de Janeiro
Departamento de Informática
Rua Marquês de São Vicente 225
Gávea Rio de Janeiro RJ 22453, Brazil
armando@inf.puc-rio.br
tel.: +55-21-524-9530

Claudia **Hattensperger**
Universität der Bundeswehr München
Fakultät für Informatik
D-85577 Neubiberg, Germany
claudia@informatik.unibw-muenchen.de
tel.: +49-89-6004-3194

Robin **Hirsch**
Imperial College of Science
Department of Computing
180 Queen's Gate
London SW7 2BZ, Great Britain
rdh@doc.ic.ac.uk
tel.: +44-71-589-5111 ext. 4987

Paul **Hoogendijk**
Eindhoven University of Technology
Vakgroep Informatica
P.O. Box 513
NL-5600 MB Eindhoven, The Netherlands
paulh@win.tue.nl
tel.: +31-40-47-4452

Ali **Jaoua**
University of Tunis, Faculté des Sciences
Campus Universitaire, Le Belvédère 1060
Tunis, Tunisia
ALIJAOUA%TNEARN.bitnet
@vm.urz.Uni.Heidelberg.de
tel.: +216-1-662-886

Wolfram **Kahl**
Universität der Bundeswehr München
Fakultät für Informatik
D-85577 Neubiberg, Germany
kahl@informatik.unibw-muenchen.de
tel.: +49-89-6004-3198

Roger D. **Maddux**
Iowa State University
Dept. of Mathematics
400 Carver Hall
Ames Iowa 50011-2066, USA
maddux@iastate.edu
tel.: +1-515-294-8134

Maarten **Marx**
University of Amsterdam
Oude Turfmarkt 151
NL 1012GC Amsterdam, The Netherlands
marx@ccsom.nl
tel.: +31-20-5 25 25 38

Szabolcs **Mikulás**
Hungarian Academy of Sciences
Mathematical Institute
Pf. 127
H-1364 Budapest, Hungary
h3762mik@ella.hu
tel.: +36-1-1 17 80 50

Bernhard **Möller**
Universität Augsburg
Institut für Mathematik
D-86135 Augsburg, Germany
moeller@uni-augsburg.de
tel.: +49-821-598-2164

Thanh Tung **Nguyen**
IMACS
Avenue des Glycines 62
B-1950 Kraainem, Belgium

Hans J. **Ohlbach**
Max-Planck-Institut für Informatik
Im Stadtwald
D-66123 Saarbrücken, Germany
ohlbach@mpi-sb.mpg.de
tel.: +49-681-302 5366

Ewa **Orlowska**
Polish Academy of Science
Institute of Theoretical and
Applied Computer Science
Azaliowa 29
PL-04-539 Warsaw, Poland
orlowska@plearn.bitnet
tel.: +48-2-6 13 23 19

Vaughan **Pratt**
Stanford University
Department of Computer Science
Margaret Jacks Hall
Stanford CA 94305, USA
pratt@cs.stanford.edu
tel.: +1-415-494-2545

Ingrid M. **Rewitzky**
University of Cape Town
Department of Mathematics
Rondebosch 7700, South Africa
ingrid@oryx.mth.uct.ac.za
tel.: +227-221-650-3213

Maarten de **Rijke**
CWI
P.O. Box 40 79
NL-1009 AB Amsterdam, The Netherlands
mdr@cwi.nl
tel.: +31-20-592-4080

Willem P. de **Roever**
Christian-Albrecht-Universität Kiel
Inst. für Informatik und Prakt. Mathem.
Preusserstrasse 1-9
D-24105 Kiel, Germany
wpr@informatik.uni-kiel.de
tel.: +49-431 56 04 71 / 74

Martin **Russling**
Universität Augsburg
Institut für Mathematik
D-86135 Augsburg, Germany
russling@uni-augsburg.de
tel.: +49-821-598-2118

Ildikó **Sain**
Hungarian Academy of Sciences
Mathematical Institute
P.O.Box 127
H-1364 Budapest, Hungary
H1468SAI@ella.hu

Holger **Schlingloff**
TU München
Institut für Informatik
D-80290 München, Germany
schlingl@informatik.tu-muenchen.de
tel.: +49-89 48095 140

Gunther **Schmidt**
Universität der Bundeswehr München
Fakultät für Informatik
D-85577 Neubiberg, Germany
schmidt@informatik.unibw-muenchen.de
tel.: +49-89-6004 2449 / (Secr. 2263)

Renate **Schmidt**
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken, Germany
schmidt@mpi-sb.mpg.de
tel.: +49-681-302-5430

András **Simon**
Hungarian Academy of Sciences
Mathematical Institute
P.O.Box 127
H-1364 Budapest, Hungary
H1468SAI@ella.hu

John **Staples**
University of Queensland
Software Verification Research Centre
Queensland 4072, Australia
staples@cs.uq.oz.au
tel.: +61-7-365-2048

Gheorghe **Ştefanescu**
Romanian Academy
Institute of Mathematics
Calea Grivitei 21
RO-70700 Bucharest, Romania
ghstef@imar.ro

Thomas **Ströhlein**
TU München
Institut für Informatik
D-80290 München, Germany
stroehle@informatik.tu-muenchen.de
tel.: +49-89-2105-8181

Yde **Venema**
Universiteit Utrecht
Department of Philosophy
Heidelberglaan 8
NL-3584 CS Utrecht, The Netherlands
yde@phil.ruu.nl

Michael **Winter**
Universität der Bundeswehr München
Fakultät für Informatik
D-85577 Neubiberg, Germany
trash@hermes.unibw-muenchen.de
tel.: +49-89-6004-2158

Jaap van der **Woude**
TU Eindhoven
Fac. Wiskunde en Informatica
Den Dolech 2
Eindhoven, The Netherlands
japie@win.tue.nl
tel.: +31-40-47 27 47