01081 Applications of Kleene Algebra

18.02.2001-23.02.2001

organized by

Roland Backhouse
Dexter Kozen
Bernhard Möller

Kleene algebra (KA) [23, 16, 24] is an algebraic system for calculating with sequential composition, choice and finite iteration. It was first introduced by Kleene in 1956 and further developed by Conway in 1971. It has reappeared in many contexts in mathematics and computer science. Its classical application has been within the theory of formal languages, where it is one of many equivalent approaches to the description of regular languages.
Within the field of efficient algorithms it has been applied to path problems on graphs (being closely related to the algebra of closed semirings [2]), to convex hull algorithms and formal treatment of pointer algorithms [5, 6, 10, 14, 12].
In compiler construction, Kleene algebra can be used to prove the correctness of optimization techniques for loop constructs [25].
More recently, Kleene algebra has been successfully applied to the semantic description of imperative programs with non-deterministic choice [4]. It covers both angelic and demonic composition and choice [17, 18].
Moreover, it allows a simple algebraic incorporation of assertions [21] as well as modal and dynamic logic [31, 26]. Also, there are close relations with interval and temporal logic [29, 22], the duration calculus [33, 34] and timed automata [3]. Further applications concern switching theory [30].
Finally, a particular form of Kleene algebra corresponds to quantales with unit elements [32, 1, 11]. Hence there is a correspondence with linear logic.
Extensions of Kleene algebra deal with infinite iteration. They have been used in the description and verification of protocols and in proofs about concurrent systems in general [15]. Related systems are iteration theories [9], the computation calculus [19] and the theory of ω-languages. Dropping one of the distributivity requirements for Kleene algebras leads to a system that is close to process algebras such as ACP or μCRL [7, 20]. Finally, there are close connections to network algebra [8, 13].
These tracks of research have so far been undertaken in a rather isolated manner. The aim of this seminar was to bring the researchers from these tracks together for fruitful interaction and for helping the subject to more public visibility. To

our knowledge, this was the first international sysmposion dedicated to the *applications* of Kleene algebra.
Compared with other algebraic approaches to semantics, such as relation algebra or sequential calculus, Kleene algebra and its relatives enjoy a particularly simple axiomatisation, since they do not use a notion of (pseudo-)converse and the corresponding axioms.
Since its basic features and rules are known even to beginner students of computer science, Kleene algebra will be more easily accepted than other formal systems and thus may serve as an effective vehicle for formal treatment of various subjects in computer science.

Contents

Part I
Basic Theory

1 Factor Theory Revisited

Roland Backhouse, University of Nottingham

Conway's book "Regular algebra and finite machines" has been mentioned frequently at this seminar. However, one important contribution in his book that has not been mentioned is his study of so-called "factors" of a regular language. The goal of this talk is to bring this to everyone's attention. Conway's "factors" are called "residuals" in relation algebra and "weakest prespecifications" in the programming literature; the fact that the same concept is known by various names attests to its importance. Conway's contribution was to show that the factors of a regular language can be organised in a matrix, which he called the factor matrix. This matrix has a number of special properties – for example, the matrix is reflexive and transitive. Conway's account of the factor matrix is however very wordy, making it difficult to read and check. In one case, the unfortunate omission of the word "not" in a sentence caused me a great deal of confusion when first reading his text!! In this talk, I show how factors are formulated using the now standard Galois connection defining the residuals of a relation. This allows one to give precise, calculational formulations of the factor matrix. I also mention the relation between the factor "graph" and the Knuth, Morris, Pratt string matching algorithm (see Backhouse and Lutz, ICALP 1977). Prompted by an earlier talk I show how factors are used to formulate the well-foundedness of a relation. This formulation is used to present a calculational proof of Newman's lemma (see Doornbos, Backhouse and Van der Woude, TCS, 1997).

2 Omega Algebra: the good, the bad, and the ugly

Ernie Cohen, Telcordia Technologies

We describe omega algebra, an extension of Kleene algebra to omega-regular expressions. The omega algebra axioms are complete for the standard equational theory of omega-regular expressions, and have proved to be an effective tool for reasoning about total correctness of asynchronous programs. Nevertheless, in contrast to the Kleene case, it's not clear that we have the "right" axioms:

- There appear to be useful theorems from relational algebra that are not provable in omega algebra. Examples include theorems giving conditions for well-foundedness of the union of well-founded relations, and rules for backward data refinement.
- A suitable theory for probabilistic programs seems to require a binary (rather than unary) omega operator.

3 Automata on Guarded Strings

Dexter Kozen, Cornell University

Guarded strings were introduced by Kozen and Smith [1996] as a model for Kleene algebra with tests (KAT). Guarded strings are like ordinary strings, except that between each letter there is an atom of a free Boolean algebra. The regular sets of guarded strings play the same role in KAT as the regular sets of ordinary strings do in Kleene algebra.
Recently we have found guarded strings useful in other contexts. In a recent paper, we developed a complete Gentzen-style sequent calculus for partial correctness and conjectured that the decision problem for this system was PSPACE-complete. We were able to verify this conjecture using guarded strings. Our proof required the development the elementary theory of finite automata on guarded strings, a generalization of the theory of finite automata on ordinary strings.
In this talk I will introduce automata on guarded strings and give several basic constructions, including determinization, state minimization, and an analog of Kleene's theorem, which are generalizations of the analogous results in classical automata theory. I will show that a central result of the theory of BDDs (Boolean decision diagrams), namely that reduced ordered BDDs are unique, is a special case of the Myhill-Nerode theorem for a class of automata on guarded strings.

4 A complete set of rational identities

Daniel Krob, Université Paris 7 Denis Diderot

A rational identity is a pair (E, F) of two rational expressions whose interpretations give two identical languages. A set of rational identities is said to be complete iff one can deduce from it every possible rational identity using only deductions involving usual rational manipulations and substitutions.
We show how to construct a complete set of rational identities consisting of the two identities

(M) $(ab)^* = 1 + a(ba)^*b$

(S) $(a + b)^* = a^*(ba^*)^*$

and of a family of identities indexed by all the symmetric groups. This system was conjectured to be complete by Conway.

5 Inductive *-Semirings

Zoltan Ésik, University of Szeged
(Joint work with Werner Kuich)

One of the most well-known induction principles in computer science is the fixed point induction rule, or least pre-fixed point rule. Inductive *-semirings are partially ordered semirings equipped with a star operation satisfying the fixed point equation and the fixed point induction rule for linear terms. Inductive *-semirings are extensions of continuous semirings and the Kleene algebras of Conway and Kozen.
We develop, in a systematic way, the rudiments of the theory of inductive *-semirings in relation to automata, languages and power series. In particular, we prove that if S is an inductive *-semiring, then so is the semiring of matrices $S^{n\times n}$, for any integer $n \geq 0$, and that if S is an inductive *-semiring, then so is any semiring of power series $S\langle\langle A^*\rangle\rangle$. As shown by Kozen, the dual of an inductive *-semiring may not be inductive. In contrast, we show that the dual of an iteration semiring is an iteration semiring. Kuich proved a general Kleene theorem for continuous semirings, and Bloom and Ésik proved a Kleene theorem for all Conway semirings. Since any inductive *-semiring is a Conway semiring and an iteration semiring, as we show, there results a Kleene theorem applicable to all inductive *-semirings. We also describe the structure of the

initial inductive *-semiring and conjecture that any free inductive *-semiring may be given as a semiring of rational power series with coefficients in the initial inductive *-semiring. We relate this conjecture to recent axiomatization results on the equational theory of the regular sets.

6 On some particular Conway semirings

Werner Kuich, Technische Universität Wien
(Joint work with Zoltan Ésik)

We consider locally closed semirings and rationally additive semirings. A semiring S is called locally closed if for all $a \in S$ there is some integer k such that $1 + a + \ldots + a^k = 1 + a + \ldots + a^{k+1}$. In any locally closed semiring we may define a star operation $a \mapsto a^*$, where a^* is the above finite sum. We prove that when S is locally closed and commutative, then S is a Conway semiring and, moreover, an iteration semiring.
Rationally additive semirings are a generalization of complete and continuous semirings. We prove that every rationally additive semiring is a Conway semiring and, moreover, an iteration semiring. We then characterize the semirings of rational power series with coefficients in N_∞, the semiring of natural numbers equipped with a top element, as the free rationally additive semirings.

7 The Greibach-Normal-Form Theorem in Continuous Kleenean Algebras

Hans Leiss, CIS - Universität München

The well-known Greibach-Normal-Form theorem for context-free grammars can be understood as a theorem about systems of polynomial inequations

$$x_i \geq p_i(x_1, ..., x_n), \quad (i = 1, ..., n),$$

where the unknowns x_i correspond to the syntactical categories of the grammar and the polynomials p_i describe the syntactic constructions of expressions of categories x_i.

We first showed the following
Theorem: Let KA $= (K, +, 0, ;, 1, *)$ be a continuous Kleenean algebra and let A be a subset of 'atoms' of K in the sense that no sum of products of elements of A is greater or equal to 1. Then for each system

$$x_1 \geq p_1(x_1, ..., x_m), \; ..., \; x_m \geq p_m(x_1, ..., x_m) \tag{1}$$

of polynomial inequations, in which each monomial of a polynomial p_i is a product of variables $x_1, ..., x_m$ and atoms a in A, but neither equal to 1 nor to some x_j, there is another system

$$x_1 \geq q_1(x_1, ..., x_m, ..., x_k), \; ..., \; x_k \geq q_k(x_1, ..., x_m, ..., x_k) \tag{2}$$

of polynomial inequations, such that

- the least solutions of (1) and (2) in KA agree in $x_1, ..., x_m$, and k is at most $m(m+1)$,
- each monomial of the polynomials $q_1, ..., q_k$ is a product of variables and atoms whose leftmost factor is an atom,

Proof sketch: The continuity assumption on KA is used to ensure the existence of least pre-fixed points. (1) can be written as a matrix inequation

$$X \geq X; R(X) + T(X), \tag{3}$$

where $X; R(X)$ collects the monomials whose leading factor is a variable and $T(X)$ collects those whose leading factor is an atom. In the Kleenean algebra of matrices over KA, the least solutions of (3) and of

$$X \geq T(X); R(X)^* \tag{4}$$

agree, where $R(X)^*$ is the iteration matrix of $R(X)$. But since $R(X)^*$ is the least solution of $Y \geq R(X); Y + 1$, the least solution of (1) coincides with the X-components of the least solution of the system

$$X \geq T(X); Y, \quad Y \geq R(X); Y + 1. \tag{5}$$

Essentially, (5) amounts to (2): first, in the polynomial inequations corresponding to $X \geq T(X); Y$, the monomials have atomic leading factors. Second, since no monomial of (1) was a variable, the matrix $R(X)$ has no entries 1, so leading factors of monomials in the polynomial inequations corresponding to $Y \geq R(X); Y + 1$ are atoms or variables among X; the latter can be replaced by their associated polynomial of $X \geq T(X); Y$ whose monomials have atoms as leading factors.
Remark: The unit matrix 1 can be avoided by considering instead the system

$$X \geq T(X); Y + T(X), \quad Y \geq R(X); Y + R(X), \tag{6}$$

where Y corresponds to $R(X); R(X)^*$. For context-free grammars, this means that epsilon-rules do not occur in the grammar in Greibach-Normal-Form.
Note that the left-recursion of (3) is replaced by a right-recursion in (5). The resulting algorithm is quadratic in the grammar size and similar to the one of Rosenkrantz 1967, but instead of using power series of matrices, we use properties of least pre-fixed points.
The second part of the talk showed that the continuity assumption can be weakened to assumptions about existence of least pre-fixed points. In the language of semirings extended by a least-fixed-point operator μ, the essential transformation

$$(\mu X.(X; R(X) + T(X)), ...) \;=\; \mu(X, Y).(T(X); Y, R(X); Y + 1) \tag{7}$$

can be proven in a formal theory of 'Kleene algebra with recursion', axiomatized by

- axioms for idempotent semirings in 0,1,+,; ,
- Park's axiom of pre-fixed point induction,
- axioms for a basic connection between left- and right-recursion:

$$\begin{aligned} \forall a, b: \quad \mu X.(X; a + b) &= b; (\mu X.(a; X + 1)) \\ \mu X.(a; X + b) &= (\mu X.(X; a + 1)); b \end{aligned}$$

Using $a^* := \mu X.(a; X + 1)$, this theory embeds D.Kozen's theory of Kleenean algebra.

8 Kleene-ing Up Semantics

Bernhard Möller, Univerität Augsburg
(Partially joint work with Jules Desharnais)

Kleene algebras provide a convenient and powerful algebraic axiomatisation of a complete lattice that is endowed with a sequential composition operation. The particular kind of Kleene algebras we are considering is equivalent to Boolean quantales. Models include formal languages under concatenation, relations under

standard composition, sets of graph paths under path concatenation and sets of streams under concatenation.
The least and greatest fixpoint operators of a complete lattice allow definitions of the finite and infinite iteration operators * and $^\omega$, resp.
Elements of Kleene algebras can be used, among others, as abstractions of the input-output semantics of nondeterministic programs or as models for the association of pointers with their target objects. In the first case, one seeks to distinguish the subclass of elements that correspond to deterministic programs. In the second case one is only interested in functional correspondences, since it does not make sense for a pointer to point to two different objects.
We discuss several candidate notions of determinacy and clarify their relationship. Some characterizations that are equivalent in the case where the underlying Kleene algebra is an (abstract) relation algebra are not equivalent for general Kleene algebras.
In relational semantics, the input-output semantics of a program is a relation on its set of states. We generalize this in considering elements of Kleene algebras as semantical values. In a nondeterministic context, the demonic semantics is calculated by considering the worst behavior of the program. In this paper, we concentrate on while loops. While calculating the semantics of a loop is difficult, showing the correctness of any candidate abstraction is much easier. For deterministic programs, Mills has described a checking method known as the while statement verification rule. A corresponding programming theorem for nondeterministic iterative constructs is proposed, proved and applied to an example. This theorem can be considered as a generalization of the while statement verification rule to nondeterministic loops.
In standard Kleene algebra it is assumed that the composition operation is universally disjunctive in both arguments. This entails monotonicity and strictness w.r.t. the least element 0 that plays the role of $\perp$ in denotational semantics. However, full strictness does not make sense when one wants to give an algebraic account of systems with lazy evaluation. Therefore we study a "one-sided" variant of KAs in which composition is strict in one argument only. This treatment fits well with systems such as the calculus of finite and infinite streams which is also used in or R. Dijkstra's computation calculus.
There is some choice in what to postulate for the other argument. Whereas Dijkstra stipulates positive disjunctivity, we investigate how far one gets if only monotonicity is required. The reason is that we want to enable a connection to process algebra. There only one of the distributivity laws for composition over choice is postulated to preserve the temporal succession of choices.

9 A Compositional and Complete Axiom System for Interval Temporal Logic

Ben C. Moszkowski, De Montford University Leicester

Interval Temporal Logic (ITL) is a temporal logic which includes a basic construct for the sequential composition of two formulas as well as an analog of Kleene star. Within ITL one can express both finite-state automata and regular expressions. Its notation makes it suitable for logic-based modular reasoning involving periods of time, refinement and assumptions and commitments. Various conventional imperative programming language constructs can be directly expressed in ITL and executable subsets of ITL are available. In addition, operators for projecting between different levels of time granularity exist (although not considered here). Zhou Chaochen, Hoare and Ravn have developed a real-time ITL extension called the Duration Calculus for hybrid systems.
After introducing ITL, our presentation considers two aspects of ITL's theory. The first is a compositional methodology for specification and proof using ITL. We show how assumptions and commitments based on fixpoints of various ITL operators provide a flexible way to modularly reason about safety and liveness. In addition, some techniques are described for compositionally transforming and refining ITL specifications.
The remainder of our presentation deals with a complete axiom system for ITL. Prior to our work, no one had proved the completeness of a relatively simple ITL deductive system supporting infinite time and permitting infinite sequential iteration comparable to omega-regular expressions. We have developed a complete axiomatization for such a version of quantified ITL over finite domains and can show completeness by representing finite-state automata in ITL and then translating ITL formulas into them. The proof of completeness illustrates some nice links between automata, regular expressions and temporal logics.

10 Iteration in Process Algebra

Alban Ponse, University of Amsterdam

In this talk I provide an overview of the extension of ACP-style process algebra with various operations that model some form of iteration.
Process algebra is an algebraic framework that aims to support specification and analysis of concurrent processes. It comprises primitives for parallelism and syn-

chronization. A typical point of departure in process algebra is the distinction between internal and external behavior, which allows for verification by characterization of external behavior (by applying abstraction", i.e., renaming internal actions to the silent step TAU). In this way, the parallel composition of a number of communicating components may give rise to a simple external behavior that indicates that the parallel "implementation" equals the specification of its external behavior.
In process algebra, a (potentially) infinite behavior is usually represented as the solution of a system of guarded recursive equations. An alternative approach is to extend process algebra with operations that model some form of recursion. In 1984, Milner was the first to do so: he considered the Kleene star operation in the context of process algebra, and raised a few questions concerning expressiveness and axiomatizability in bisimulation semantics. In 1993, various forms of iteration were added to ACP-style process algebra, notably (the original, binary version of) the Kleene star which does not presuppose a special constant SKIP (or 1, or epsilon) and is therefore easier to combine with parallel operators.
Further contents of the talk:

- An overview of some basic questions that emerged, mainly about axiomatizability and expressivity in bisimulation semantics, and of some of the answers that were found;
- The role of some typical special constants (DELTA for deadlock, SKIP, and TAU) in the context of iteration;
- Explanation of a general fairness principle in the setting with iteration (i.e., TAU(TAU * x) = TAU x);
- An overview of variations of the Kleene star that were proposed in the process algebra literature (and of associated basic results), among which some non-regular operations (the latter leading to much stronger expressiveness results).

Conclusion: Iterative operations in process algebra provide a relatively simple introduction to the specification of (potentially) infinite behavior; and - perhaps most relevant at this particular Seminar - should be related to (more) standard approaches in the setting of "Kleene Algebra".

11 Kleene algebra of two dimensional words - a model for interactive systems

Gheorghe Stefanescu, University of Bucharest

The talk focuses on an extension of Kleene algebra to cope with concurrent object-oriented systems. It is part of the MixNA (Mixed Network Algebra) project aiming to get an algebraic formalism for such systems; see Part IV of the recent book: G. Stefanescu, "Network Algebra", Springer-Verlag, London, 2000 for some more informations on this model.
Two-dimensional (or planar) words are proposed as an abstraction for the interaction running patterns of concurrent object-oriented systems. They come with a natural algebraic structure which extend the usual Kleene algebra with a new set of (horizontal) identity/composition/star operations. These new operations are used to model objects' interaction. We present 2-dimensional regular expressions, 2-dimensional grammars and 2-dimensional automata as devices for representing languages of planar words and give some hints on applying them to concurrent object-oriented systems.

Part II
Applications

12 Application of Dynamic Logic to Complexity of Path Constraints

Natasha Alechina, University of Nottingham
(Joint work with Stephane Demri and Maarten de Rijke)

Path constraints were introduced by Abiteboul and Vianu [PODS'97] as a means to optimise queries over data represented as an edge-labelled graph. A path inclusion constraint says that all vertices which are reachable from the root of the graph by a p-path are also reachable by a q-path, where p and q are regular expressions built from edge labels, wild card #, ;,+,*. Abiteboul and Vianu showed that the implication problem for path inclusion constraints (whether a set of path inclusion constraints implies another path inclusion constraint) is decidable in EXPSPACE.

In this talk, we introduce a new variant of propositional dynamic logic, PDL^{path}, which includes a wild card, converse operator and a nominal "root". We show that the satisfiability problem for PDL^{path} is EXPTIME-complete, by an easy adaptation of known results [de Giacomo]. Path inclusion constraints can be expressed in PDL^{path} as $[p]\langle q\hat{}\rangle$ root. Backward constraints ("from any vertex reachable by a p-path from the root, we can come back by a q-path") are expressible by $[p]\langle q\hat{}\rangle$root. This gives us a new tighter EXPTIME upper bound on the complexity of the implication problem for inclusion and backward constraints.
A PSPACE lower bound follows from the known results for regular expressions.

13 Applications of regular algebra to language processing problems

Roland Backhouse, University of Nottingham

Many functions on context-free languages can be expressed in the form of the least fixed point of a function whose definition mimics the grammar of the given language. Examples include the function returning the length of the shortest word in a language, and the function returning the smallest number of edit operations required to transform a given word into a word in a language. This paper presents the basic theory that explains when a function on a context-free language can be defined in this way. It is shown how the theory can be applied in a methodology for programming the evaluation of such functions.

14 Reading a Little Kleene Algebra Calculation

Kieran Clenaghan, University of York

Kleene algebra provides for (among other things) the manipulation of expressions involving transitive closure. It therefore has applications in the calculation of algorithms from specifications based on transitive closure.
In this talk we give a reading of a simple example calculation, that of Dijkstra's shortest paths algorithm. This is based on work by R. Backhouse and others. The point of the talk is to illustrate how the calculation can be carried out, and hopefully followed, in full detail on a couple of blackboards. The intention is to

show the compactness and effectiveness of the algebra, whilst at the same time paying attention to the readability of the calculation.
The talk prompts the obvious question: can the algebra be applied to a variety of interesting algorithm calculations? B. Möller and others have shown that it can, but there is scope for much more investigation. A particular question presents itself: can the calculation be generalised "nicely" to give a derivation of Knuth's generalisation of Dijkstra's algorithm? The generalistion is to find best cost trees in a hypergraph (paths become trees when a graph is generalised to a hypergraph).

15 System Support for Kleene Algebras

Thorsten Ehm, Universität Augsburg

This talk gives a short overview of two systems we are developing to assist our daily work with Kleene algebras. The first one is a library consisting of a set of Hugs functions used to generate atomic Kleene algebras. These are applied to test if formulas and equations using Kleene algebraic operations hold. This is no proof but may – if the equation passes the test – confirm the researcher of the correctness of his assumption. On the otherhand – if the test fails – one can get information what went wrong and possibly correct the equation. Additionally there are also functions for parsing equations and testing them on manually typed-in algebras. The talk gives a short overview which problems arise due to the immense amount of combination possibilities and proposes improvements. The second part closes the gap between these tests and a formally correct proof by introducing a proof system for several different axiomatizations of Kleene algebras. This tool is based on the KIV system, a formal proof system mainly used for software specification and verification. Using this, proofs in the built-in algebras can be done in full detail and rigor.

16 Kleene Algebra with Tests and Compiler Optimization: Verification of Cache Blocking in LU Decomposition with Pivoting

Dexter Kozen, Cornell University

Kleene algebra with tests (KAT), introduced by the author [1997], combines programs and assertions in a purely equational system. A Kleene algebra with tests is a Kleene algebra with an embedded Boolean subalgebra. KAT strictly subsumes propositional Hoare logic, is of no greater complexity, and is deductively complete over relational models. Moreover, KAT requires nothing beyond classical equational logic, in contrast to Hoare logic, which requires a specialized syntax involving partial correctness assertions.
KAT has been applied successfully in various low-level verification tasks involving communication protocols, basic safety analysis, concurrency control, and local compiler optimizations. A useful feature of KAT in this regard is its ability to accommodate certain basic equational assumptions regarding the interaction of atomic instructions. This feature makes KAT ideal for reasoning about the correctness of low-level code transformations.
In this talk I will report on the use of KAT in a substantial compiler verification task. Mateev, Menon, and Pingali [2000] have described a series of source-level compiler transformations for automatic cache blocking in LU decomposition with partial pivoting. These transformations are used primarily in large applications to enhance locality of reference. In attempting to verify the correctness of these transformations, Mateev, Menon, and Pingali observed that the standard approach involving symbolic dependence analysis is inadequate. One major complication is that, although the transformations are semantically correct, they do not preserve definition-use dependencies. This led them to consider other approaches that exploited knowledge of the semantics of the basic operations. They proposed a new system called "fractal symbolic analysis," in which programs are repeatedly simplified until symbolic analysis becomes feasible. The semantics is not preserved in the simplification process, but the equality of the simplified programs implies the equality of the original programs.
In this talk I will demonstrate that the verification task studied by Mateev, Menon, and Pingali can be adequately handled by KAT in a purely equational way with no extraneous constructs. I will introduce four new basic rules governing the interaction of atomic programs and tests. These rules are schematic in nature and play roughly the same role as the assignment rule in Hoare logic, but are more versatile.

17 Calculating With Pointer Structures

Bernhard Möller, Universität Augsburg

In calculational program design one derives implementations from specifications using semantics-preserving deduction rules. The aim of modern algebraic approaches is to make both specification and calculation more concise and perspicuous by compacting logic into algebra as much as possible.
We present such an algebraic approach to the calculation of programs with pointer structures. It is based on the particular Kleene algebra of relations and partial maps. However, the basic definitions for the selective updating operation and reachability questions can even be given for general Kleene algebras.
We investigate sufficient criteria for preservation of substructures under selective updating. The approach is illustrated with some simple examples such as list concatenation and reversal, tree rotation and search tree insertion and deletion. The approach covers also cyclic structures like cyclic lists or threaded trees.

18 Church-Rosser Proofs in Kleene Algebra

Georg Struth, Universität Freiburg

We apply Kozen's Kleene algebra to proving Church-Rosser theorems for non-symmetric transitive relations, quasiorderings and equations. The approach nicely balances algebraic conciseness, expressive confinement and computational power. Specifications and proofs are simple, elegant and readable. Proofs are based on simple natural properties of the regular operations and algebraically reconstruct precisely the standard diagrammatic arguments. By definition of the star operation, induction is replaced by fixed point computation and thereby deduction by mere calculation. Large parts of proofs are even amenable to automation, since Kleene algebra is complete for the algebra of regular events. This makes the approach well-suited for mechanization. Our claim of simplicity is formally supported by short and straightforward specifications and highly automatic proofs with the Isabelle proof checker.
The step from Church-Rosser theorems to Church-Rosser theorems modulo a congruence or Newman's lemma leads beyond Kleene algebra. Using an omega-operation and allegorial tabulation techniques, we can express and derive the obligate well-foundedness assumptions and avoid the usual invention of complex induction orderings and measures in proofs. Proving Newman's lemma in other variants of Kleene algebra should be of general interest for analyzing systems

with infinite behaviour.

19 Algebraic Reasoning about Guarded Loops

Joakim von Wright, Turku Center for Computer Science
(Based on joint work with Ralph-Johan Back)

It is well known that universally conjunctive predicate transformers are isomorphic to relations over the underlying state space and thus form a model for classical Kleene algebra. In a total correctness framework, universally conjunctive predicate transformers model programs that always terminate, with + as demonic choice (meet) and 0 as miracle (top).
In order to model possibly nonterminating programs, the larger class of conjunctive predicate transformers is considered (those that distribute over nonempty meets of predicates). By introducing a strong iteration operator which is a least fixpoint (where the Kleene star is a greatest fixpoint) we can model nontermination, but we have to give up the axiom x0=0.
By introducing special guard elements (also known as tests or predicates) we can model two notions of correctness for nonterminating programs and we can also model while-loops and more general guarded loops, and verify numerous transformation rules for such loops, preserving total correctness.
We can go one step further and model interaction. For this, we extend our analysis to a framework of monotonic predicate transformers, with both angelic and demonic nondeterminism. We weaken the distributivity axiom x(y+z) = xy+xz to an inequality, and the monotonic predicate transformers become a model. Thus, we can verify transformation rules that hold for interactive systems.

References

[1] S. Abramsky, S. Vickers: *Quantales, observational logic and process semantics.* Math. Struct. Comp. Science **3**, 161–227 (1993)

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman: *The design and analysis of computer algorithms.* Reading, Mass.: Addison Wesley 1974

[3] E. Asarin, O. Maler, P. Caspi: *A Kleene theorem for timed automata.* In: G. Winskel (Ed.): Proc. LICS'97, 160-171, 1997.

[4] R.-J. Back, J. von Wright: *Reasoning algebraically about loops.* Acta Informatica **36** 295–334 (1999)

[5] R.C. Backhouse, B.A. Carré: *Regular algebra applied to path-finding problems.* J. Institute of Mathematics and its applications **15**, 161–186 (1975)

[6] R.C. Backhouse, A.J.M. van Gasteren: *Calculating a path algorithm.* In: R.S. Bird, C.C. Morgan, J.C.P. Woodcock (eds.): Mathematics of program construction. Lecture Notes in Computer Science **669**. Berlin: Springer 1993,32–44

[7] J.A. Bergstra, I. Bethke, and A. Ponse: *Process algebra with iteration and nesting.* The Computer Journal, **37**, 243-258 (1994)

[8] J.A. Bergstra, G. Ştefănescu: *Network algebra with demonic relation operators.* Report P9509, PRG, University of Amsterdam, 1995. Revised Version: Revue Roumaine de Mathematique Pure et Appliquée (to appear)

[9] S.L. Bloom, Z. Esik: *Iteration Theories: The Equational Logic of Iterative Processes.* Berlin: Springer 1993

[10] Francis Bossut, Max Dauchet, and Bruno Warin: *A Kleene theorem for a class of planar acyclic graphs.* Information and Computation, **117**, 251–265 (1995).

[11] C. Brown, D. Gurr: *A representation theorem for quantales.* Journal of Pure and Applied Algebra, 85:27-42 (1993)

[12] T. Brunn, B. Möller, M. Russling: *Layered graph traversals and hamiltonian path problems – An algebraic approach.* In: J. Jeuring (ed.): Mathematics of Program Construction. Lecture Notes in Computer Science **1422**. Berlin: Springer 1998, 96–121

[13] V.E. Cazanescu, G. Ştefănescu: *Feedback, iteration and repetition.* INCREST Preprint 42/1988. Published in: G. Paun (ed.): Mathematical aspects of natural and formal languages. World Scientific 1994, 43-62.

[14] K. Clenaghan: *Calculational graph algorithmics: reconciling two approaches with dynamic algebra.* CWI Amsterdam, Report CS-R9518, 1995

[15] E. Cohen: *Separation and reduction.* In: R. Backhouse, J. Oliveira (eds.): *Mathematics of program construction.* Lecture Notes in Computer Science **1837**. Berlin: Springer 2000, 45–59

[16] J.H. Conway: *Regular algebra and finite machines.* London: Chapman and Hall 1971

[17] J. Desharnais, B. Möller: *Characterizing determinacy in Kleene algebra.* Special Issue on Relational Methods in Computer Science, Information Sciences — An International Journal (to appear)

[18] J. Desharnais, B. Möller, F. Tchier: *Kleene under a demonic star.* In: T. Rus (ed.): *Algebraic Methodology and Software Technology.* Lecture Notes in Computer Science **1816**. Berlin: Springer 2000, 355–370

[19] R.M. Dijkstra: *Computation calculus bridging a formalization gap.* Science of Computer Programming **37**, 3–36 (2000)

[20] W.J. Fokkink: *Axiomatisations for the perpetual loop in process algebra.* In: P. Degano, R. Gorrieri, A. Marchetti-Spaccamela (eds.): Proc. 24th ICALP. Lecture Notes in Computer Science **1256**. Berlin: Springer 1997, 571–581

[21] M. Hollenberg: *Equational axioms of test algebra.* In: M. Nielsen, W. Thomas (Eds.): Computer Science Logic, 11th International Workshop, CSL '97, Annual Conference of the EACSL, Aarhus, Denmark, August 23-29, 1997, Selected Papers. Lecture Notes in Computer Science **1414**. Berlin: Springer 1998, 295-310

[22] B. von Karger: *Temporal algebra.* Universität Kiel, Habilitationsschrift 1997

[23] S.C. Kleene: *Representation of events in nerve nets and finite automata.* In: C. Shannon, J. McCarthy (eds.): Automata Studies. Princeton University Press 1956, 3–41

[24] D. Kozen: *A completeness theorem for Kleene algebras and the algebra of regular events.* Information and Computation **110**, 366-390 (1994)

[25] M.-C. Patron, D. Kozen: *Certification of compiler optimizations using Kleene algebra with tests*, Report 99-1779, Computer Science Department, Cornell University, Dec. 1999.

[26] D. Kozen, J. Tiuryn: *On the completeness of propositional Hoare logic.* In: J. Desharnais (ed.): *RelMiCS 2000, 5th International Seminar on Relational Methods in Computer Science.* Université Laval, Québec, Jan. 2000, 195-202.

[27] B. Möller: Towards pointer algebra. Science of Computer Programming **21**, 57–90 (1993)

[28] B. Möller: Calculating with acyclic and cyclic lists. Special Issue on Relational Methods in Computer Science, Information Sciences — An International Journal **119/3–4**, 135–154 (1999)

[29] B. Moszkowski: Some very compositional temporal properties. In: E.-R. Olderog (ed.): Programming concepts, methods and calculi. IFIP Transactions A-56. Amsterdam: North-Holland 1994, 307–326

[30] T. Ninomiya, M. Mukaidono: *Clarifying the axioms of Kleene Algebra based on the method of indeterminate coefficients.* Proc. 29th IEEE International Symposium on Multiple-Valued Logic (ISMVL 99), Albert-Ludwigs-University, Freiburg im Breisgau , Germany, on May 20-22, 1999

[31] V. Pratt: *Dynamic algebras as a well-behaved fragment of relation algebras.* In: C.H. Bergman, R.D. Maddux, D.L. Pigozzi (eds.): Algebraic Logic and Universal Algebra in Computer Science. Lecture Notes in Computer Science **425**. Berlin: Springer 1988, 77–110

[32] K.I. Rosenthal: *Quantales and their applications.* Pitman Research Notes in Mathematics Series, Vol. 234. Longman Scientific & Technical 1990

[33] Zhou, C.: *Duration calculi: an overview.* In: D. Bjørner, M. Broy, I.V. Pottosin (eds.): Formal methods in Programming and their Applications. Lecture Notes in Computer Science **735**. Berlin: Springer 1993, 256–266

[34] Zhou, C., A.P. Ravn, H. Rischel, J.U. Skakkebæk: *Specification of embedded, real-time systems.* Proc. 1992 Euromicro Workshop on Real-Time Systems. IEEE Computer Society Press 1992, 116-121