Proof Automation in Kleene Algebra

Peter Höfner

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany
`hoefner@informatik.uni-augsburg.de`

Abstract. It has often been claimed that model checking, special purpose automated deduction or interactive theorem proving are needed for formal program development. Recently, it has been demonstrated that off-the-shelf automated proof and counterexample search is an interesting alternative if combined with the right domain model. Furthermore it has been shown that variants of Kleene algebra might provide light-weight formal methods with heavy-weight automation.
In this paper we give a brief overview of a number of program analysis and computer mathematics tasks where (variants of) Kleene algebra combined with automated theorem provers are already applied.

1 Introduction

Formal systems verification often requires the integration of domain-specific knowledge. This is usually achieved through higher-order theorem proving at the expense of computational power or through model checking at the expense of expressive power. For automated deduction, this task is still a challenge.

In [8, 10] an alternative approach to automated deduction is proposed: domain-specific algebras for standard provers instead of domain-specific provers for standard algebras. More concretely, automated reasoning in Kleene algebra with the paramodulation-based theorem prover Prover9 and the counterexample generator Mace4 is investigated. Kleene algebras are particularly suitable for the task:

– They have widespread applications ranging from program analysis and semantics to combinatorial optimisation and concurrency control.
– They offer a concise syntax for modelling actions, programs or state transitions under non-deterministic choice, sequential composition and iteration.
– They provide a uniform semantics for various program analysis tasks that support cross-theory reasoning with various approaches.
– They come with a simple first-order equational calculus that yields particularly short and abstract proofs.

The last item paves the way for first-order automated reasoning. In this paper we give an overview of the potential of Kleene algebra for proof automation. In particular, we present a number of program analysis and computer mathematics tasks like proof automation of separation and reduction theorems in concurrency control and program verification in Hoare logic. We chose Prover9 and Mace4 [13] because it integrates automated deduction with counterexample search. Any other paramodulation-based theorem prover should yield similar results.

To further underpin the applicability for non-experts, we consistently use a rather naïve black-box approach to theorem proving and avoid sophisticated encodings, refined proof orderings, hints or proof planning, and excessive running times. Much stronger results could be obtained with reasonable additional effort.

2 Idempotent Semirings and Iteration Algebras

Idempotent semirings form the algebraic basis for the proof experiments. They provide operators for non-deterministic choice and sequential composition in a first-order equational calculus.

A *semiring* is a structure $(S, +, ;, 0, 1)$ such that $(S, +, 0)$ is a commutative monoid, $(S, ;, 1)$ is a monoid, multiplication distributes over addition and 0 is an annihilator, i.e., $a0 = 0a = 0$.[1] An *i-semiring* is a semiring where $+$ is idempotent. The *natural order* $\leq$ on S is given by $a \leq b \Leftrightarrow a + b = b$; it has 0 as its least element; addition and multiplication are isotone with respect to it.

Tests model assertions and restrictions in i-semirings. Formally, a *test* in an i-semiring is an element $p \leq 1$ that has a complement q relative to 1, i.e., $p+q = 1$ and $p \cdot q = 0 = q \cdot p$. The set of all tests of S is denoted by $\mathsf{test}(S)$. It is not hard to show that $\mathsf{test}(S)$ forms a Boolean algebra.

More interesting behaviours of programs and transition systems arise from finite and infinite iteration.

A *Kleene algebra* is an i-semiring S extended by an operation $^* : S \to S$ that satisfies the *star unfold* and the *star induction* axioms $1+xx^* = x^*$, $1+x^*x \leq x^*$, $y + xz \leq z \Rightarrow x^*y \leq z$ and $y + zx \leq z \Rightarrow yx^* \leq z$.

An *omega algebra* is a Kleene algebra S extended by an operation $^\omega : S \to S$ that satisfies the *omega unfold* and the *omega coinduction* axiom $x^\omega \leq xx^\omega$ and $z \leq y + xz \Rightarrow z \leq x^\omega + x^*y$.

I-semiring, semiring with tests, Kleene algebra and omega algebra are encoded in Prover9/Mace4 by implementing the axioms in a simple, straightforward way. The encodings can be found at [1].

3 The Standard Calculi

In this and the following sections we only aim at illustrating the main ideas, achievements and difficulties of the approach. The technical details of all proofs in this paper, including the Prover9 input and output files that provide complete information about the the proof search and the time and memory used, can be found at [1]. Up to now we have built up a database with more than 300 theorems.

The standard calculi of i-semiring, tests, Kleene and omega algebra can automatically be verified with Prover9. An important example for reasoning in Hoare logic is the equivalence of $px\neg q = 0$, $px \leq xq$, $x\neg q \leq \neg px$, and $px = pxq$. Overall we have proved over 200 theorems for the basic calculi, most of them from scratch. We can also detect some non-theorems, e.g. $x^{\omega *} = x^\omega$, with the model generation tool Mace4. Sumarising, we point out important facts and remarks for automated deduction:

[1] Multiplication ; is left implicit

– While equations like $x(yx)^* = (xy)^*x$ could as well be decided by automata, automated deduction can also verify implications such as $x \le y \Rightarrow x^* \le y^*$.
– The induction axioms act as elimination rules that simplify expressions.
– These axioms formalise (co)induction without external measures (e.g. length of a sequence) in first-order equational logic.
– Proving complex statements sometimes needs simple intermediate lemmas.

4 Automating Reduction and Separation

The expressions $(x+y)^*$ or $(x+y)^\omega$ can be interpreted as the repeated concurrent execution of two processes x and y. In this context, *reduction laws* such as $(x+y)^* = x^*(yx^*)^*$ connect concurrency with interleaving while *separation laws* infer global system properties from those of the particular processes. We present two examples that show how such derivations can be automated.

Our first example abstracts the relational encoding of the Church-Rosser theorem for abstract reduction systems:

$$y^*x^* \le x^*y^* \Rightarrow (x+y)^* \le x^*y^* \ .$$

It says that repeated concurrent executions of x and y can be reduced to an x-sequence followed by a y-sequence if all x-sequences have priority over y-sequences.

The Church-Rosser theorem is usually proved by induction over the number of y^*x^*-peaks that arise from $(x+y)^*$, i.e., with an external induction measure (cf. [16]); first-order proofs with the internal induction provided by Kleene algebra can also be given [15]. Prover9 automatically shows the law in about $3\,s$.

The second example is a separation theorem [2]. It states that, in the presence of a suitable commutation condition, concurrent processes terminate iff individual processes do. The theorem can be specified in omega algebra [15] if termination of a process x is expressed as $x^\omega = 0$ (absence of infinite iteration):

$$yx \le x(x+y)^* \Rightarrow (x^\omega + y^\omega = 0 \Leftrightarrow (x+y)^\omega = 0) \ .$$

The claim is proved in about $270\,s$. This shows that it is possible to reason automatically about program termination, although finiteness cannot be expressed within first-order logic. Moreover explicit (bi)simulation is not needed.

These two proof experiments show that in many cases, despite the associativity and commutativity laws involved, Prover9 can prove some impressive facts.

5 Modal Semirings

The scope of Kleene algebra can be considerably extended by adding modalities. The resulting formalism is related to propositional dynamic and temporal logics.

An i-semiring S is called *modal* [14] if it can be endowed with a total (forward) diamond operation $|x\rangle : \mathsf{test}(S) \to \mathsf{test}(S)$, for each $x \in S$, that satisfies

$$|x\rangle p \le q \Leftrightarrow xp \le qx \quad \text{and} \quad |xy\rangle p = |x\rangle |y\rangle p \ .$$

Intuitively, $|x\rangle p$ characterises the set of states with at least one x-successor in p, i.e., the preimage of set p under the action x. According to the aforementioned property $xp \le qx \Leftrightarrow xp = qxp$ and the fact that xp models the right-restriction by a test p, $|x\rangle p$ is the least set from which every element of p can be reached via action x. Dually, backwards operators $\langle x|p$ can be defined and related with x-predecessor. Modal boxes can be defined, as usual, via de Morgan duality: $|x]p = \neg|x\rangle\neg p$ and $[x|p = \neg\langle x|\neg p$. Modal semirings can be extended to modal Kleene/omega algebra without any further modal axioms.

Modalities enjoy a rich calculus and symmetries that are expressed by Galois connections and conjugations [14], for instance, $|x\rangle p \leq q \Leftrightarrow p \leq [x|q$. These and further standard laws can be proved automatically from the axioms.

Our experiments confirm that the number of axioms introduced through the different layers considerably inhibits proof search. They further show that the following laws for modalities are very useful in practice.

$$|x\rangle(p+q) = |x\rangle p + |x\rangle q, \qquad |x+y\rangle p = |x\rangle p + |y\rangle p, \qquad |x\rangle 0 = 0,$$
$$p + |x\rangle|x^*\rangle p = |x^*\rangle p \qquad |x\rangle p \leq p \Rightarrow |x^*\rangle p \leq p. \qquad (1)$$

6 Automating Hoare Logic

It is well-known that the programming constructs of Dijkstra's guarded command language can be encoded in Kleene algebra. Validity of the while-rule

$$\frac{\{p\wedge q\}\ x\ \{q\}}{\{q\}\ \text{while}\ p\ \text{do}\ x\ \{\neg p\wedge q\}}$$

for instance, is expressed as $pqx\neg q = 0 \Rightarrow q(px)^*\neg(p+q) = 0$. Using such an encoding of Hoare triples, validity of all rules of Hoare logic (except assignment) can then verified in Kleene algebra [11]. Prover9 failed to prove the above implication from scratch. This negative result shows that choosing the right algebra and the appropriate level of abstraction is important for a successful automation.

Hence, we use modal Kleene algebra to show that the rules of Hoare logic (except the assignment rule) are theorems that can be automated. Further, we show how partial correctness proofs of concrete programs can be automated up to domain specific calculations. Encodings of validity of the Hoare rules in modal Kleene algebra can be found in [14]; e.g., validity of the while-rule is encoded as

$$\langle x|pq \leq q \Rightarrow \langle (px)^*\neg p|q \leq \neg pq\ .$$

The implication follows immediately and automatically from (1). Validity of the remaining rules (except assignment) follows immediately and automatically, too.

The considerations about Hoare logic has abstracted from the assignment rule $\{p[e/x]\}\ x := e\ \{p\}$ which can be encoded as $p[e/x] \leq |\{x := e\}]p$. Using the rule, we now verify an algorithm for division of integers in modal Kleene algebra. We use abstraction for the Kleene algebra part and the assignment rule at the leaves of the proof tree as an interface to the specific calculations with integers.

```
funct Div $(n,m) \equiv k := 0;\ l := n;$
                  while $m \leq l$ do $k := k+1;\ l := l-m;$
```

We overload arithmetic notation by consistently writing arithmetic expressions in brackets. Setting $x_1 \hat{=} \{k := 0\}$, $x_2 \hat{=} \{l := n\}$, $y_1 \hat{=} \{k := k+1\}$, $y_2 \hat{=} \{l := l-m\}$, $r \hat{=} \{m \leq l\}$ and using the precondition and the postconditions $p \hat{=} \{0 \leq n\}$, $q_1 \hat{=} \{n = km + l\}$, $q_2 \hat{=} \{0 \leq l\}$, $q_3 \hat{=} \{l < m\} = \neg r$ yields the theorem

$$\langle x_1x_2(ry_1y_2)^*\neg r|p \leq q_1q_2\neg\ r,$$

which can be automatically proved from $p \leq |x_1x_2](q_1q_2)$ and $q_1q_2r \leq |y_1y_2](q_1q_2)$. The assumptions have precisely the form of the assignment rule; they cannot be analysed by Prover9. In [8] a proof by hand is given, but full automation could be achieved by integrating a solver for a suitable fragment of arithmetic.

This shows that partial correctness proofs can be fully automated in modal Kleene algebra in the presence of domain-specific solvers. This particular case would require a solver for simple arithmetic; other proofs might e.g. require solvers for data structures like lists, arrays or stacks.

7 More Automation

Here we have presented automation within standard calculi, concurrency control and Hoare logic. We have successfully applied the combination of Prover9 and variants of Kleene algebra for automation of more program analysis and computer mathematics tasks. More details can be found in [8–10].

Dynamic Logics. Modal Kleene algebra is strongly related to variants of dynamic algebra developed by Kozen, Parikh and Pratt (cf. [6]). It has been shown (automatically and by hand) that Segerberg's induction law is a theorem of modal Kleene algebra, which means that propositional dynamic logic is subsumed by Kleene algebra. Therefore automated deduction with propositional dynamic logic is now available via modal Kleene algebra. This treatment of modal logic is completely axiomatic whereas previous approaches usually translate the Kripke semantics for modal logics more indirectly into first-order logic. An extension to first-order dynamic logics seems feasible.

Linear Temporal Logics. It is well known that the operators of linear temporal logics can be expressed in propositional dynamic logics. The operators of next-step and until can be defined by $\mathsf{X}p = |x\rangle p$ and $p\mathsf{U}q = |(px)^*\rangle q$; the operators for finally and globally are $\mathsf{F}p = |x^*\rangle p$ and $\mathsf{G}p = |x^*]p$, where x stands for an arbitrary action. Adding some axioms ([12]) to those of modal Kleene algebra yield automated proofs in linear temporal logics in this setting with Prover9.

Modal Correspondence Theory. We have automated a modal correspondence proof of Löb's formula, which in modal logic expresses well-foundedness of a transitive relation. A discussion of these notions and a proof by hand can be found in [5]. We replayed the proof step-wise. Beyond this example, further modal correspondence results can easily be automated.

Boolean and Relation Algebra. We have also shown that relation algebra is surprisingly appropriate for automated deduction. We have proved a basic calculus with more than 100 theorems, some of them of considerable complexity. They provide a basic library of well-known facts about Boolean algebra, Boolean algebra with operators and relation algebra, on which more complex investigations and applications in program verification can build.

Demonic Refinement Calculus. We have encoded a predicate for inequation in Prover9 to allow inequational reasoning. Using this alternative approach, we automatically verify some key refinement laws for concurrent systems, which are far more sophisticated than the above examples. For example we have proved a classical data refinement law [4, 17] and Back's atomicity refinement theorem [3].

8 Conclusion

Compared to our expectations, the number and difficulty of the theorems we could prove automatically came as a surprise. We chose the experiments according to their relevance for formal methods as well as according to complexity. They show that domain-specific algebras can successfully be combined with general purpose theorem provers. The experiments suggest that modal Kleene algebras provide the appropriate level of abstraction to formalise and reason about programs and systems in a simple, concise and efficient way. Furthermore they suggest that off-the-shelf theorem proving technology can be very successful if combined with the appropriate algebra. The approach therefore seems very promising as a light-weight formal method with heavy-weight automation.

Further applications for the approach might be more sophisticated applications of relation algebra as well as the combination with hybrid systems [7].

Acknowledgements. I thank Georg Struth for the joint work. Further I thank Bernhard Möller for fruitful discussions and remarks.

References

1. `http://www.dcs.shef.ac.uk/~georg/ka`.
2. L. Bachmair and N. Dershowitz. Commutation, transformation, and termination. In J. Siekmann, editor, *CADE 8*, LNCS 230, pages 5–20. Springer, 1986.
3. R.-J. Back. A method for refining atomicity in parallel algorithms. In E. Odijk, M. Rem, J.-C. Syr, editors, *Parallel Architectures and Languages Europe*, LNCS 366, pages 199–216. Springer, 1989.
4. R.-J. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction.* Graduate Texts in Computer Science. Springer, 1998.
5. J. Desharnais, B. Möller, G. Struth. Kleene algebra with domain. *ACM Trans. Comp. Logic*, 7(4):798–833, 2006.
6. D. Harel, D. Kozen, J. Tiuryn. *Dynamic Logic.* MIT Press, 2000.
7. P. Höfner and B. Möller. Towards an algebra of hybrid systems. In W. MacCaull, M. Winter, I. Duentsch, editors, *RelMiCS*, LNCS 3929, pages 121–133, 2006.
8. P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfennig, editor, *CADE 2007*, LNCS 4603, pages 279–294, 2007.
9. P. Höfner and G. Struth. Automated reasoning in relation algebras and Boolean algebras with operators. Technical report, Department of Computer Science, University of Sheffield, 2007. (to appear).
10. P. Höfner and G. Struth. Can refinement be automated? In E. Boiten, J. Derrick, G. Smith, editors, *Refine 2007*, EATCS, pages 53–73, 2007. (to appear).
11. D. Kozen. On Hoare logic and Kleene algebra with tests. ACM Trans. Comp. Logic, 1(1):60–76, 2000.
12. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems—Specification.* Springer, 1991.
13. W. McCune. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/mace4`.
14. B. Möller and G. Struth. Algebras of modal operators and partial correctness. Theo. Comp. Sc., 351(2):221–239, 2006.
15. G. Struth. Abstract abstract reduction. J. Log. Algebr. Prog., 66(2):239–270, 2006.
16. Terese, editor. *Term Rewriting Systems.* Cambridge University Press, 2003.
17. J. von Wright. From Kleene algebra to refinement algebra. In E. A. Boiten and B. Möller, editors, *MPC*, LNCS 2386, pages 233–262. Springer, 2002.