Automated Reasoning in Kleene Algebra

Peter Höfner and Georg Struth

Department of Computer Science, University of Sheffield, United Kingdom
{p.hoefner,g.struth}@dcs.shef.ac.uk

Abstract. It has often been claimed that model checking, special purpose automated deduction or interactive theorem proving are needed for formal program development. We demonstrate that off-the-shelf automated proof and counterexample search is an interesting alternative if combined with the right domain model. We implement variants of Kleene algebras axiomatically in Prover9/Mace4 and perform proof experiments about Hoare, dynamic, temporal logics, concurrency control and termination analysis. They confirm that a simple automated analysis of some important program properties is possible. Particular benefits of this approach include "soft" model checking in a first-order setting, cross-theory reasoning between standard formalisms and full automation of some (co)inductive arguments. Kleene algebras might therefore provide light-weight formal methods with heavy-weight automation.

1 Introduction

Formal systems verification and computer mathematics 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, however, this task is still a challenge. Over the last decades, considerable effort has been put into the development of special purpose calculi for automated deduction with algebraic theories, but the practical impact of this approach on formal methods has been rather limited. Nevertheless, the specific balance of expressive and computational power and the user-friendliness of automated deduction could considerably increase the practical applicability of formal software verification.

This paper proposes an alternative approach to automated deduction for systems verification: domain-specific algebras for standard provers instead of domain-specific provers for standard algebras. More concretely, we investigate the potential of automated reasoning in Kleene algebra with the resolution- and paramodulation-based Prover9 and the counterexample generator Mace4 [2].

Over the last few years, variants of Kleene algebras emerged as fundamental structures in computing. They found widespread applications ranging from program analysis and semantics to combinatorial optimisation and concurrency control. Kleene algebras seem particularly suitable for our task: They offer a concise syntax for modelling actions, programs or state transitions under nondeterministic choice, sequential composition and iteration. They provide a uniform semantics for various program analysis tasks that supports cross-theory

reasoning with modal, relational, trace-based, language-based and event-based approaches. They come with a simple first-order equational calculus that yields particularly short and abstract proofs, and they are supported by powerful automata-based decision procedures. Kleene algebras have already been integrated into higher-order theorem provers [22, 15, 3] and their applicability as a formal method has successfully been demonstrated in that setting. But their potential for automated deduction has not yet been explored.

At first sight, feeding an automated prover with the Kleene algebra axioms and some meaningful conjecture might seem hopeless: Kleene algebras contain an commutative idempotent additive monoid and a multiplicative monoid that interact via distributivity. So one would rather expect the prover to get lost in term rearrangements and complex unifications. But our proof experiments on program verification, logics of programs and modal correspondence theory support the opposite, and perhaps surprising conclusion: The combination of Kleene algebra and state-of-the-art theorem proving technology makes it often possible to prove theorems of considerable complexity and practical relevance.

Our main contributions are as follows: First, we specify Kleene algebras [16], omega algebras [6] and their modal extensions [9, 19] in Prover9 and Mace4. We chose this particular tool primarily because it integrates automated deduction with counterexample search. Any other paramodulation-based theorem prover should lead us to similar conclusions. We can automatically verify the standard calculus of these structures. We can prove more than 100 theorems, most of them from scratch, and fail on a very small number of statements. Second, we apply our approach to a number of program analysis and computer mathematics tasks: Proof automation of separation and reduction theorems in concurrency control; automated program verification in Hoare logic; automated verification of the axioms of propositional dynamic logic and linear temporal logic; an automated modal correspondence proof of Löb's formula.

These experiments confirm the feasibility of our approach. Many proofs were fully automatic, interaction (introduction of lemmas) was only needed for some more complex statements. The example tasks, which are rather advanced, have been chosen for particular reasons: The concurrency control examples show some (co)inductive arguments and termination analysis within first-order logic. The examples from Hoare, dynamic and temporal logic demonstrate the versatility and practical relevance of the approach. The correspondence proof shows that some non-trivial mathematics (viz. a second-order frame property) can be automated and that abstraction is often a key to success. We believe that the approach can be extended to a light-weight formal method with a particularly high degree of automation.

Our experiments also pose some interesting research questions for automated deduction and formal methods. A further discussion can be found in the conclusion and the respective sections.

The emphasis of this paper is rather on the universality of Kleene algebra than on a detailed particular application. We will therefore only survey the

specifications and concrete proofs with Prover9 and Mace4. The complete input and output files for each proof discussed can be found at a web-site [1].

Also, to further underpin the applicability for non-experts in automated deduction, 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 therefore be obtained with reasonable additional effort.

2 Idempotent Semirings

Idempotent semirings form the algebraic basis for the proof experiments of this paper. They provide the appropriate level of abstraction for modelling actions, programs or state transitions under non-deterministic choice and sequential composition in a first-order equational calculus. This makes them very suitable for resolution-based and paramodulation-based theorem proving.

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 from the left and right and 0 is a left and right zero of multiplication. A semiring S is *idempotent* (an *i-semiring*) if $(S, +)$ is a semilattice with $x + y = \sup(x, y)$. We usually omit the multiplication symbol. The semilattice-order $\leq$ on S has 0 as its least element; addition and multiplication are isotone with respect to it.

The specification for Prover9/Mace4 is

```
x+y = y+x.                  % additive commutative monoid
x+0 = x.
x+(y+z) = (x+y)+z.
x;1 = x & 1;x = x.          % multiplicative monoid
x;(y;z) = (x;y);z.
x+x = x.                    % additive idempotence
0;x = 0 & x;0 = 0.          % multiplicative zeroes
x;(y+z) = x;z+x;y.         % distributivity laws
(x+y);z = x;z+y;z.
```

The definition of $\leq$ can be added, `x<=y <-> x+y=y`, but we usually work with equations to profit from the rewrite-based simplification techniques of Prover9. In contrast, human reasoning with i-semiring is largely order-based.

Every semiring comes with an *opposite* semiring in which the order of multiplication is swapped. The associated duality gives theorems for free.

Tests of a program or sets of states of a transitions system can also be modelled in this setting. Such objects are needed, e.g., for expressing conditions in if-then-else statements or loops, or the propositions of modal logics. It is natural to assume that these objects form a Boolean algebra. They can be integrated into i-semirings as follows: A *test* in an i-semiring S is an element of a Boolean subalgebra $\mathsf{test}(S) \subseteq S$ (the *test algebra* of S) such that $\mathsf{test}(S)$ is bounded by 0 and 1 and multiplication coincides with lattice meet. We will write $x, y \ldots$ for arbitrary semiring elements and $p, q, \ldots$ for tests. Idempotent semirings admit at least the test algebra $\{0, 1\}$ and can have different test algebras. We use predicates for embedding tests; `c(p)` represents the complement $\neg p$ in Prover9.

```
test(p) -> p;c(p) = 0 & p+c(p) = 1.
test(0) & test(1).
test(p) -> c(c(p)) = p.
test(p) & test(q) -> c(p+q) = c(p);c(q).
test(p) & test(q) -> c(p;q) = c(p)+c(q).
test(p) -> test(c(p)).
```

The first line expresses existence and uniqueness of complements. The remaining lines induce the Boolean algebra of tests from a given set of tests. This can be verified with Prover9.

Idempotent semirings with tests are expressive enough for (indirectly) encoding Hoare logic without the assignment and the loop-rule [17]. Validity of a Hoare triple $\{p\}x\{q\}$ is captured by $px\neg q = 0$: no action x transforms a precondition p into a postcondition $\neg q$. We will discuss an automation of Hoare logic and the associated weakest liberal precondition semantics in Section 5 and 7.

The standard calculus of i-semirings and tests can automatically be verified with Prover9. A non-trivial example is the equivalence of

$$px\neg q = 0, \qquad px \leq xq, \qquad x\neg q \leq \neg px, \qquad px = pxq.$$

This equivalence is important, e.g., for reasoning in Hoare logic and with modal Kleene algebras.

3 Iteration Algebras

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

A *Kleene algebra* [16] is an i-semiring S extended by an operation $^* : S \to S$ that satisfies the *star unfold* and the *star induction* axiom

$$1 + xx^* = x^*, \qquad\qquad y + xz \leq z \Rightarrow x^*y \leq z$$

and their duals with respect to opposition. The induction axioms are encoded as equations, e.g, `(y+x;z)+z = z -> x*;y+z = z.` The expression x^* abstractly represents the reflexive transitive closure of x. The transitive closure of x is defined as $x^+ = xx^*$.

An *omega algebra* [6] is a Kleene algebra S extended by an operation $^\omega : S \to S$ that satisfies the *omega unfold* and the *omega coinduction* axiom

$$x^\omega \leq xx^\omega, \qquad\qquad z \leq y + xz \Rightarrow z \leq x^\omega + x^*y.$$

By these definitions, x^*y and $x^\omega + x^*y$ are the least and greatest fixed points of $\lambda z.y + xz$. The elements x^* and x^ω arise as special cases.

The following facts are interesting for automated deduction: First, the induction axioms act as star and omega elimination rules that simplify expressions. Second, these axioms formalise (co)induction without external measures (e.g. length of a sequence) in first-order equational logic. Third, there are strong

connections with standard automata-based decision procedures: While the equational theory of Kleene algebra is that of regular expressions [16], the uniform word problem is undecidable. Similar results hold for omega algebras and ω-regular expressions [6].

The following identities, e.g., can be proved automatically: $0^* = 1 = 1^*$, $1 \leq x^*$, $xx^* \leq x^*$, $x^*x^* = x^*$, $x \leq x^*$, $x^*x = xx^*$, $x^{**} = x^*$, $1 + xx^* = x^* = 1 + x^*x$, $x(yx)^* = (xy)^*x$ and $(x + y)^* = x^*(yx^*)^*$. $0^\omega = 0$, $x \leq 1^\omega$, $x^\omega = x^\omega 1^\omega$, $x^\omega = xx^\omega$, $x^\omega y \leq x^\omega$, $x^*x^\omega = x^\omega$, $x^{+\omega} = x^\omega$ and $(x + y)^\omega = (x^*y)^\omega + (x^*y)^*x^\omega$.

While these identities could as well be decided by automata, automated deduction can also verify implications such as

$$x \leq y \Rightarrow x^* \leq y^*, \qquad x \leq y \Rightarrow x^\omega \leq y^\omega, \qquad xz \leq zy \Rightarrow x^*z \leq zy^*.$$

We sometimes need some simple intermediate lemmas, obtained from proofs by hand, for proving more complex statements, but nothing beyond.

We can also detect some non-theorems, e.g. $x^{\omega *} = x^\omega$, with the model generation tool Mace4, but only one statement considered, $x^\omega x^\omega = x^\omega$, could neither be proved nor refuted (automatically or by hand); Mace4 can generate all idempotent semirings, Kleene algebras and omega algebras with < 20 elements. This conjecture and refutation game with Prover9 and Mace4 is very helpful in general. Table 1 shows that the number of Kleene algebras grows very fast with the number of elements.

#elements	#KAs	#KAs (up to iso.)	#KAs with test (up to iso.)
1	1	1	1
2	1	1	1
3	3	3	3
4	39	20	21
5	753	149	149
6	23357	1488	1491
7	1052475	18554	
8	69199211		

Table 1. Enumeration of Kleene algebras

Mace4 can check all Kleene algebras with less than 15 elements in a few minutes on a desktop PC[1]. It takes, for example, $\sim$ 20s to check that $px\neg q = 0$ and $px + xq = xq$ are equivalent in all Kleene algebras with 15 elements.

Generation of Kleene algebras with Mace4 requires isomorphism checking and therefore storing models (7 elements need $>$ 2GB RAM). Interestingly, Conway's classical book on regular algebras [7] lists 21 Kleene algebras with four elements. We found that his examples (5.) and (7.) are flawed and another one is missing. According to the Mace4 manual, the integrated isomorphism checking should be taken with a grain of salt. But our numbers for < 7 elements are confirmed by Jipsen's computations with the GAP system [14].

[1] We used a Pentium 4 CPU, 1.6GHz, 384MB RAM.

4 Automating Concurrency Control

In this and the following sections we only aim at illustrating the main ideas, achievements and difficulties of the approach. All technical details of all proofs in this paper, including the Prover9 and Mace4 input and output files, that provide complete information about the the proof search and the time and memory used, can be found at a web-site [1]. All proofs have been done from scratch, i.e., with the full sets of axioms plus isotonicity of addition, multiplication, star and omega, but without any further assumptions, unless otherwise stated.

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. Kleene algebras are very useful for deriving such laws [6, 23]. We present two examples that show how such derivations can be automated.

Our first example is the reduction law

$$y^*x^* \leq x^*y^* \Rightarrow (x+y)^* \leq x^*y^*$$

which says that repeated concurrent executions of x and y can be reduced to an x-sequence followed by a y-sequence (both possibly void) if all x-sequences have priority over y-sequences. This statement abstracts the relational encoding of the Church-Rosser theorem for abstract reduction systems.

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. [24]). However, equational proofs with the internal induction provided by Kleene algebra can also be given [23]. We can automatically prove the reduction law in about 3s; we can also automate an abstraction of the proof by induction on the number of peaks.

This result is a first step towards further proof automation that seems now feasible, viz. an automated proof of (the abstract part of) the Church-Rosser theorem of the λ-calculus. Equational proofs in Kleene algebra have already been given [23]. An essential feature of the proof method is abstraction. Properties about λ-terms are proved separately (e.g. in a higher-order prover [20]) and represented abstractly as *bridge lemmas* within Kleene algebra. These are then used as hypotheses at the algebraic level that is suitable for automation.

Reasoning about abstract reduction systems is traditionally diagrammatic. Kleene algebra provides a semantics for a considerable part of diagrammatic reasoning [10] which can therefore be verified by using a theorem prover in the background.

Our second example is a separation theorem due to Bachmair and Dershowitz [4]. It states that, in the presence of a suitable commutation condition, concurrent processes terminate iff individual processes do. The theorem can be specified and proved by hand in omega algebra [23]. In this setting, termination of a process x can be expressed as $x^\omega = 0$ (absence of infinite iteration). The separation theorem can therefore be stated as

$$yx \leq x(x+y)^* \Rightarrow (x^\omega + y^\omega = 0 \Leftrightarrow (x+y)^\omega = 0). \tag{1}$$

The implication $(x+y)^\omega = 0 \Rightarrow x^\omega + y^\omega = 0$ does not depend on the hypothesis. It can be proved in less than one second.

The converse direction requires a series of lemmas, at least with our naïve approach. Our search for automation lead us to a simpler proof than that in [23]. First, we can prove automatically that the hypothesis is equivalent to $y^+x \leq x(x+y)^*$ and to $y^*x \leq x(x+y)^*$. We then attempted to prove that

$$x^\omega = 0 \wedge yx \leq x(x+y)^* \Rightarrow y^*x \leq x^+y^*, \tag{2}$$

but failed. The essential part is proving $x(x+y)^* \leq x^+y^* (= x^\omega + x^+y^*)$ from the hypotheses. By omega coinduction, it suffices to show that $x(x+y)^* \leq xy^* + xx(x+y)^*$, which can be done automatically, but using the identity $(x+y)^* = y^* + y^*x(x+y)^*$, which itself can be done automatically. The coinduction step, however, let the search explode.

This proof is essentially a step-wise replay of a proof by hand. The main problem of proving is that applications of isotonicity, which are trivial in an inequational context, require intricate unifications in the equational case. A combination of equational and inequational reasoning would be very beneficial here. Equation (2) allows us to replace the hypothesis $yx \leq x(x+y)^*$ by the computationally simpler $y^*x \leq x^+y^*$ whenever x terminates.

The remaining lemmas for (1), $x^*(x^*y)^\omega = (x^*y)^\omega$, and that $x^\omega = 0$ and $yx \leq x(x+y)^*$ imply $(y^*x)^\omega = 0$, are again automatic. Our separation theorem is then immediate from the lemmas.

This second example shows that it is possible to reason automatically about program termination in a first-oder setting, although finiteness cannot be expressed within first-order logic. The proofs are essentially coalgebraic and use the coinduction axiom of omega algebra. 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. Some proofs, however, require an amount of interaction that is similar to higher-order proof checkers: proving individual lemmas automatically first and then using them as hypotheses for the main goal in a second round.

5 Automating Hoare Logic: A First Attempt

It is well-known that the programming constructs of Dijkstra's guarded command language can be encoded in Kleene algebra [17]. In particular,

$$\mathsf{if}\ p\ \mathsf{then}\ x\ \mathsf{else}\ y = px + \neg py \qquad \text{and} \qquad \mathsf{while}\ p\ \mathsf{do}\ x = (px)^*\neg p.$$

Using the above encoding of Hoare triples, validity of the rules of Hoare logic (except assignment) can then be expressed—although quite indirectly—and verified in Kleene algebra [17]. Validity of the while-rule

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

for instance, is expressed as $pqx\neg q = 0 \Rightarrow q(px)^*\neg(p+q) = 0$. We could not prove from scratch that this implication is a theorem of Kleene algebra. However, we can immediately prove that $qyq = qy \Rightarrow qy^*q = qy^*$, from which the above implication follows by isotonicity and substitution. Again, an encoding based on inequalities and in particular a theorem prover that can handle chaining rules for transitive relations might resolve this problem.

This negative result illustrates the fact that choosing the right algebra and the appropriate level of abstraction is important for a successful automation. First, the rules of Hoare logic are superfluous for verifying programs with Kleene algebra, but they are sound with respect to the algebraic semantics. Therefore, we don't even need to bother about verifying them. Second, the difficulty with proving validity of the while-rule reflects the general problems with verifying programs in this setting. In the following sections, we will show that a modal extension of Kleene algebra considerable simplifies this purpose.

6 Modal Semirings

The scope of Kleene algebras can be considerably extended by adding modalities. As we will see, the resulting formalism is similar to propositional dynamic logic but also strongly related to temporal logics.

An i-semiring S is called *modal* [19] 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

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

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 $xp \leq qx \Leftrightarrow xp = qxp$ and the fact that xp models the right-restriction of an action (e.g. a relation) by a set p, $|x\rangle p$ is the least set from which every element of p can be reached via action x. Therefore, the above definition of diamonds captures the usual Kripke semantics with the modal syntax at the left-hand side and the relational semantics at the right-hand side of the equivalence.

A *domain* operation $\mathsf{dom} : S \to \mathsf{test}(S)$ is obtained from the diamond operator as $\mathsf{dom}(x) = |x\rangle 1$. Alternatively, domain can be axiomatised on i-semirings, even equationally, from which diamonds are defined as $|x\rangle p = \mathsf{dom}(xp)$. By this axiomatisation, $\mathsf{dom}(x)$ is the least set that does not restrict action x from the left, which is indeed a natural condition for a domain operation.

Dually, backward diamond operators can be defined via semiring opposition, $\langle x|p \leq q \Leftrightarrow px \leq xq$ and $\langle xy|p = \langle y|\langle x|p$, and related with a notion of codomain. 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 algebras without any further modal axioms.

The equational axioms of (co)domain can easily be implemented in Prover9, boxes and diamonds can be defined relative to them. Modal operators must be totalised to functions of type $S \times S \to \mathsf{test}(S)$ by setting, e.g., $|x\rangle y = 0$ if $y \notin \mathsf{test}(S)$.

Modalities enjoy a rich calculus and symmetries that are expressed by Galois connections and conjugations [19], for instance, $|x\rangle p \leq q \Leftrightarrow p \leq [x|q$ and $p|x\rangle q = 0 \Leftrightarrow q\langle x|p = 0$. These and further standard laws can be proved automatically from the axioms. While dualities transform theorems, Galois connections and conjugations generate them. We therefore need not prove statements that follow generically from Galois connections or that are duals of other statements. This particular advantage of the algebraic approach saves a lot of work in practice.

Our experiments confirm that the number of axioms introduced through the different layers considerably inhibits proof search. Particular sources of complexity are the complementation axioms of the test algebra and the domain axioms that are computationally not sufficiently meaningful. In contrast, the following laws for modalities are very useful in practice.

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

They can be automatically verified; only the last implication requires a simple intermediate lemma. The relevance of this alternative approach to modalities over Kleene algebras has further been explored in [11]. Essentially, the above laws define a *Kleene module*, a two-sorted structure over a Kleene algebra and a Boolean algebra in which the diamond operator acts as a scalar product. By using Kleene modules, we can completely dispense with domain (and even with Boolean complements, if necessary) and thereby considerably guide the proof search.

7 Automating Hoare Logic

We will now use modal Kleene algebra—instead of the previous non-modal approach—to show that the rules of Hoare logic (except the assignment rule) are theorems of modal Kleene algebra that can easily be automated. We also argue that the rules of the weakest liberal precondition calculus come for free by dualising the calculus of modal diamonds, which has to a large extent been automated. Finally, we show how partial correctness proofs of concrete programs can be automated in Kleene algebra up to domain specific calculations.

Encodings of validity of the Hoare rules in modal Kleene algebras can be found in [19]. In particular, validity of the while-rule is encoded as

$$\langle x|pq \leq q \Rightarrow \langle(px)^*\neg p|q \leq \neg pq. \tag{3}$$

Dualisation yields $|xp\rangle q \leq q \Rightarrow \neg p|(xp)^*\rangle q \leq \neg pq$ and we can now apply the rules of our forward diamond calculus for Prover9. Obviously, there is almost nothing to prove. (3) follows immediately and automatically from the diamond induction law of Kleene modules and isotonicity of multiplication in an inequational encoding.

Validity of the remaining rules (except assignment) follows immediately from the Kleene module laws, too. The weakening rule, for instance, reduces to an

isotonicity property. Up to a trivial induction over proofs in Hoare logic, this yields an automation of the soundness proof of Hoare logic with respect to the Kleene algebra semantics given in [19].

The standard completeness proof uses Hoare's weakest liberal precondition semantics. For each postcondition p and terminating action x it computes the weakest precondition (or the greatest set) for which each x-transition leads to p. This is precisely captured by $|x]p$. The calculus of weakest liberal preconditions therefore is just the calculus of forward box operators. It can be obtained without proof by dualising the diamond calculus. Based on these results, a simple calculational completeness proof of Hoare logic has been given, but it uses structural induction with respect to the programming constructs [19]. A partial automation is certainly possible. While the induction is schematic, the base case and the induction step are entirely equational and can be automated.

The previous considerations about Hoare logic abstracted from the assignment rule $\{p[e/x]\}\ x := e\ \{p\}$ which can be encoded as $\langle\{x := e\}|\ p[e/x] \leq p$ or, by the Galois connection, as

$$p[e/x] \leq |\{x := e\}]p.$$

Using this rule, we will now completely verify an algorithm for division of a nonnegative integer n by a positive integer m in modal Kleene algebra. We will 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.

$$\begin{array}{l}\mathsf{funct}\ \ \mathrm{Div} \equiv k := 0;\ l := n;\\ \qquad\qquad\ \ \mathsf{while}\ \ m \leq l\ \mathsf{do}\ k := k + 1;\ l := l - m;\end{array}$$

We will consistently write arithmetic expressions in brackets and therefore overload arithmetic notation. Setting

$$x_1 \hat{=} \{k := 0\},\quad x_2 \hat{=} \{l := n\},\quad y_1 \hat{=} \{k := k + 1\},\quad y_2 \hat{=} \{l := l - m\},\quad r \hat{=} \{m \leq l\}$$

and using the precondition and the postconditions

$$p \hat{=} \{0 \leq n\},\quad q_1 \hat{=} \{n = km + l\},\quad q_2 \hat{=} \{0 \leq l\},\quad q_3 \hat{=} \{l < m\} = \neg r$$

yields the Hoare triple $\{p\}\ x_1x_2(ry_1y_2)^*\neg r\ \{q_1q_2\neg r\}$. Its translation to modal Kleene algebra obliges us to prove

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

This can easily be done automatically from the hypotheses

$$p \leq |x_1]|x_2](q_1q_2) \qquad \text{and} \qquad q_1q_2r \leq |y_1]|y_2](q_1q_2).$$

The assumptions themselves have precisely the form of the assignment rule; they cannot be further analysed by Prover9. We give a proof by hand, but a full

automation could be achieved by integrating a solver for a suitable fragment of arithmetics. For the first hypothesis we calculate

$$\begin{aligned}|x_1]|x_2](q_1q_2) &= |\{k := 0\}]\ |\{l := n\}](q_1q_2) \geq (\{n = km + l\}\{0 \leq l\})[k/0][l/n]\\ &= \{n = 0m + n\}\{0 \leq n\} = \{0 \leq n\} = p.\end{aligned}$$

For the second hypothesis we calculate

$$\begin{aligned}|y_1]|y_2](q_1q_2) &\geq (\{n = km + l\}\{0 \leq l\})[l/(l - m)][k/(k + 1)]\\ &= \{n = (k + 1)m + (l - m)\}\{0 \leq (l - m)\}\\ &\geq \{n = km + l\}\{0 \leq l\}\{m \leq r\}.\end{aligned}$$

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 arithmetics. Other proofs might require solvers, e.g., for data structures like lists, arrays or stacks or for more complex numeric domains. Integrating such solvers into state of the art theorem provers would therefore have immediate practical relevance for program analysis and verification. The special syntax and the specific inference rules of the Hoare calculus are not at all needed.

8 Automating Dynamic Logics

Modal Kleene algebras are very similar to propositional dynamic logics. More precisely, they are strongly related to variants of dynamic algebras developed by Kozen, Parikh and Pratt (cf. [12]). The axioms of dynamics algebras look like those of Kleene modules, but the induction axiom of Kleene modules is replaced by Segerberg's induction axiom

$$|x^*\rangle p - p \leq |x^*\rangle(|x\rangle p - p), \tag{4}$$

with $p - q$ defined as $p\neg q$. However, while dynamic algebras use a Boolean algebra in the second argument of diamonds, there is no Kleene algebra in the first argument, only a term algebra of Kleenean signature (cf. [11, 19]).

It has been shown that (4) is a theorem of modal Kleene algebra, which means that propositional dynamic logic is subsumed by modal Kleene algebra. We can give a step-wise automated proof of (4). We can prove that

$$p \leq |x^*\rangle q + p, \qquad |x\rangle|x^*\rangle q \leq |x^*\rangle q + p, \qquad q \leq |x^*\rangle q$$

where q replaces $|x\rangle p - p$. With these hypotheses we can show that (4) follows from distributivity, the induction law of Kleene modules and the Galois connection $p - q \leq r \Leftrightarrow p \leq q + r$, which holds in Boolean algebra.

Another variant of dynamic algebra uses the additional axiom $|p?\rangle q = pq$ where $? : B \to K$ embeds tests into actions. In Kleene algebra, the embedding of tests is left implicit and this axiom reduces to $|p\rangle q = pq$. The proof can be automated from scratch as well.

Automated deduction with propositional dynamic logic is now available via modal Kleene algebras. 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 (cf. [21, 8]). These translational approaches therefore reason in one particular model whereas ours, beyond relations, also covers models based on traces, paths and languages. Finally, an extension to first-order dynamic logics seems feasible.

9 Automating 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. We can also define the initial state by $\mathsf{init}_x = [x|0$; it is the set of states with no x-predecessors. A set of axioms has been proposed by Manna and Pnueli [18] and further been adapted and explained by von Karger [25] to a setting of second-order quantales. However, von Karger's axioms can easily be translated to the first-order setting of modal Kleene algebras.

$$
\begin{aligned}
|(px)^*\rangle q &= q + p|x\rangle|(px)^*\rangle q, & \langle(xp)^*|q &= q + p\langle(xp)^*|\langle x|q,\\
|(px)^*\rangle 0 &\leq 0, & \langle x|0 &= 1,\\
|x^*](p \to q) &\leq |x^*]p \to |x^*]q, & [x^*|(p \to q) &\leq [x^*|p \to [x^*|q,\\
|x^*]p &\leq p|x]|x^*]p, & |x^*](p \to |x]p) &\leq |x^*](p \to |x^*]p),\\
p &\leq [x||x\rangle p, & p &\leq |x]\langle x|p,\\
\mathsf{init}_x \leq |x^*](p \to [x|q) \to |x^*](p \to [x^*|q), & & \mathsf{init}_x &\leq |x^*]p \to |x^*][x|p,\\
|x](p \to q) &= |x]p \to |x]q, & [x|(p \to q) &= [x|p \to [x|q,\\
\langle x|p &\leq [x|p, & |x\rangle p &= |x]p.
\end{aligned}
$$

These axioms split into two groups. Those in the first five lines are theorems of modal Kleene algebra; the remaining ones express the particular properties of the underlying model and therefore need not be proved. But also for the first five lines there is nothing to prove: a closer inspection shows that they are instances of general theorems of Kleene algebras that have already been automated, e.g, $|x\rangle p - |x\rangle q \leq |x\rangle(p-q)$. The axioms in the fifth line, in particular, are instances of generic cancellation laws of Galois connections. The second axiom in the fourth line is a dual variant of Segerberg's induction axiom (4).

Adding the axioms in the three last lines to those of modal Kleene algebras allows one to perform automated proofs in linear temporal logics in this setting with Prover9. These axioms encode relational properties in the sense of modal correspondence theory. Those in the sixth line encode confluence of x, the remaining ones encode linearity of x and the fact that there is no upper endpoint. We did not attempt to further automate this analysis. Instead we will provide a more significant correspondence result in the next section.

Since variants of Dijkstra's guarded command language can also be specified in Kleene algebra, our approach could provide a kind of "soft model checking" in a first-order setting. System specifications can be written in the guarded command language, system properties can be specified in linear temporal logic. Proofs or refutations can then be attempted in Prover9 and Mace4. Again, our approach seems to allow an extension to first-order linear temporal logic.

10 Automating Modal Correspondence Theory

We will now consider an example from modal logics to further demonstrate the balance between expressive and computational power of modal Kleene algebras.

We will automate a modal correspondence proof of Löb's formula (cf. [5]), which in modal logic expresses well-foundedness of a transitive relation. In its usual form Löb's formula is written as $\Box(\Box p \rightarrow p) \rightarrow p$. To represent it algebraically, we first replace $\Box$ by $|x]$ and then dualise the result to forward diamonds. This yields

$$|x\rangle p \leq |x\rangle(p - |x\rangle p). \tag{5}$$

We must express transitivity and well-foundedness. An element x is *transitive* if $xx \leq x$, which implies that $|x\rangle|x\rangle p \leq |x\rangle p$. Furthermore, x is *well-founded* if

$$p - |x\rangle p = 0 \Rightarrow p = 0. \tag{6}$$

This notion coincides with the usual set-theoretic notion. The expression $p - |x\rangle p$ abstractly represents the x-maximal elements of a set p, i.e., the set of elements of p from which no further x-transition is possible. Formula (6) therefore says that only the empty set has no x-maximal element, whence x is well-founded (with respect to increasing chains).

A deeper discussion of these notions and a proof by hand can be found in [9]. The proof has two steps. The first one shows that a transitive element is equal to its transitive closure. We can automatically prove $|x\rangle|x\rangle p \leq |x\rangle p \Rightarrow |x^+\rangle p \leq |x\rangle p$. The second one shows that well-foundedness of x is equivalent to

$$|x\rangle p \leq |x^+\rangle(p - |x\rangle p). \tag{7}$$

Equivalence of (6) and (5) for transitive elements is then obvious.

While the proof that (7) implies (6) is immediate with Prover9, the converse implication is more complex. First, note that the antecedent of (6) is equivalent to $p \leq |x\rangle p$. So if we can show that

$$|x\rangle p - |x^+\rangle(p - |x\rangle p) \leq |x\rangle(|x\rangle p - |x^+\rangle(p - |x\rangle p)) \tag{8}$$

then (7) follows from (6). We can do a step-wise proof from the Kleene module axioms in which $p - q$ is consistently replaced by $p\neg q$. The arising difficulties show that we do not work at the right level of abstraction.

Since a modal operator $|a\rangle$ operates on the Boolean algebra of tests, we can lift our considerations to the function space. As usual, this is done by stipulating,

for all $f, g : \mathsf{test}(S) \to \mathsf{test}(S)$, that $f \leq g$ iff $\forall p \in \mathsf{test}(S).f(p) \leq g(p)$. The operations of Kleene algebras can be lifted as well, e.g., $(f+g)(p) = f(p)+g(p)$, $(fg)(p) = f(g(p))$ and $1(p) = p$. It can be shown that the structure induced on the function space is again a Kleene algebra (except for one induction axiom) [19]. The structure induced is even richer. In particular, we obtain $(f - g)(p) = f(p) - g(p)$. Lifting (8) and setting $f = |a\rangle$ yields

$$f - f^+(1 - f) \leq f(f - f^+(1 - f)).$$

We can now prove automatically that $f-f^+(1-f) \leq f((1-(1-f))-f^+(1-f))$. The remaining step requires an application of the inequality $1 - (1 - f) \leq f$, which we can prove automatically in Boolean algebra. However, this isotonicity step requires an intricate matching in our equational encoding, which could not be done by the prover in a reasonable amount of time.

This experiment illustrates the benefits of the abstraction and lifting techniques that come with the algebraic approach. It also illustrates the limitations of our naïve equational encoding that cannot sufficiently cope with isotonicity.

The standard correspondence result for Löb's formula is model-theoretic; it strongly uses implicit set theory and infinite chains and its frame property is second-order. In contrast, our approach is entirely calculational and therefore more suitable for automation. In particular, modal Kleene algebra allows us to express syntax and semantics in one and the same formalism. Beyond this example, further modal correspondence results can easily be automated.

11 Conclusion

We implemented variants of Kleene algebras in the automated deduction system Prover9 and the associated counterexample generator Mace4. We automatically verified the standard calculus of these algebras and proved some non-trivial statements that are relevant to systems verification and modal correspondence theory. We used the theorem-proving technology in a rather naïve way and did not put much effort into tuning syntactic orderings or using the selection and hint mechanisms provided. We usually stopped the prover after searching for a few minutes and introduced step-wise proofs when proofs from scratch were not successful with this approach. The immediate benefit of the black-box approach is that it yields a very conservative estimation of the possibilities and limitations of the approach, which is very valuable with respect to industrial applicability.

Compared to our initial expectations, the number and difficulty of the theorems we could prove came as a surprise. The Church-Rosser proof from scratch in a few seconds, for instance, seems quite impressive. We chose our experiments due to their practical relevance for computer mathematics and formal methods as well as due to their complexity. They support our claim that domain-specific algebras can successfully be combined with general purpose theorem provers; a direction that certainly deserves further investigation.

For mathematicians, our experiments underpin that automated deduction with complex algebraic structures is feasible, sometimes surprisingly simple and

fast. Routine proofs can often be fully automated even by non-experts in automated deduction. In the context of formal methods, automated proof support, e.g. for B or Z, is still a challenge. Our approach has the potential to improve this situation. Our 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. While special purpose theorem provers have often been deemed necessary this task, our experiments 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. In particular, the interplay of conjectures and refutations—a kind of "soft model checking"—seems very useful in practice.

Our experiments also pose some interesting research questions for automated deduction. First, equational reasoning should be complemented by reasoning with inequalities (viz. chaining calculi), an issue that has so far rather been neglected in implementations. During the submission phase of this paper, we have encoded inequalities as predicate in Prover9 together with the obvious axioms. Using this alternative approach, we could automatically verify some key refinement laws for concurrent systems, which are far more sophisticated than the examples treated in this paper [13]. The equational coding failed on most of these examples. Second, an integration of domain-specific solvers and decision procedures promises a full automation of partial correctness analysis of programs and beyond. Third, we cannot sufficiently exploit the symmetries and dualities of Kleene algebra within Prover9, and, although some forms of (co)induction are supported by Kleene algebra, structural induction is not possible. A combination of other tools that support these tasks would be very helpful.

In this paper we could only outline the first steps of our new approach to automated program analysis. In the future, we plan to build up a library of automatically verified theorems of Kleene algebra. The development of a tool that combines diagrammatic reasoning about transition systems with formal verification through automated deduction seems very interesting. We will also further pursue the specification and automated verification of programs and protocols via the guarded command language and the modal apparatus provided by Kleene algebra. And, last but not least, we are planning to continue transforming our approach into an applicable and strongly automated formal method.

Acknowledgements. We are grateful to the anonymous referee of a previous paper [23] for challenging us to automate Kleene algebras. We would also like to thank Peter Jipsen, Wolfram Kahl, Dexter Kozen and Renate Schmidt for inspiring discussions on deduction with these structures.

References

1. `http://www.dcs.shef.ac.uk/~georg/ka`.
2. Prover9 and Mace4. `http://www.cs.unm.edu/~mccune/mace4`.

3. K. Aboul-Hosn and D. Kozen. KAT-ML: An interactive theorem prover for Kleene algebra with tests. *Journal of Applied Non-Classical Logics*, 16(1–2):9–33, 2006.
4. L. Bachmair and N. Dershowitz. Commutation, transformation, and termination. In J. H. Siekmann, editor, *8th International Conference on Automated Deduction*, volume 230 of *LNCS*, pages 5–20. Springer, 1986.
5. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.
6. E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction (MPC 2000)*, volume 1837 of *LNCS*, pages 45–59. Springer, 2000.
7. J. H. Conway. *Regular Algebra and Finite Machines*. Chapman & Hall, 1971.
8. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic Journal of the IGPL*, 8(3):265–292, 2000.
9. J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Trans. Computational Logic*, 7(4):798–833, 2006.
10. M. Ebert and G. Struth. Diagram chase in relational system development. In M. Minas, editor, *3rd IEEE workshop on Visual Languages and Formal Methods (VLFM'04)*, volume 127 of *ENTCS*, pages 87–105. Elsevier, 2005.
11. T. Ehm, B. Möller, and G. Struth. Kleene modules. In R. Berghammer, B. Möller, and G. Struth, editors, *Relational and Kleene-Algebraic Methods in Computer Science*, volume 3051 of *LNCS*, pages 112–123. Springer, 2004.
12. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
13. P. Höfner and G. Struth. Can refinement be automated? Technical Report CS-07-08, Department of Computer Science, University of Sheffield, 2007.
14. P. Jipsen. Personal communication.
15. W. Kahl. Calculational relation-algebraic proofs in Isabelle/Isar. In R. Berghammer, B. Möller, and G. Struth, editors, *Relational and Kleene-Algebraic Methods in Computer Science*, volume 3051 of *LNCS*, pages 179–190. Springer, 2004.
16. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
17. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Trans. Computational Logic*, 1(1):60–76, 2000.
18. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems—Specification*. Springer, 1991.
19. B. Möller and G. Struth. Algebras of modal operators and partial correctness. *Theoretical Computer Science*, 351(2):221–239, 2006.
20. T. Nipkow. More Church-Rosser proofs (in Isabelle/HOL). *J. Automated Reasoning*, 26(1):51–66, 2001.
21. H. J. Ohlbach, A. Nonnengart, M. de Rijke, and D. Gabbay. Encoding Two-Valued Nonclassical Logics in Classic Logic. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 21, pages 1403 – 1485. Elsevier, 2001.
22. G. Struth. Calculating Church-Rosser proofs in Kleene algebra. In H. de Swart, editor, *Relational Methods in Computer Science, 6th International Conference*, volume 2561 of *LNCS*, pages 276–290. Springer, 2002.
23. G. Struth. Abstract abstract reduction. *Journal of Logic and Algebraic Programming*, 66(2):239–270, 2006.
24. Terese, editor. *Term Rewriting Systems*. Cambridge University Press, 2003.
25. B. von Karger. Temporal algebra. *Mathematical Structures in Computer Science*, 8(3):277–320, 1998.