Automated Engineering of Relational and Algebraic Methods in Isabelle/HOL

Simon Foster[1], Georg Struth[1], and Tjark Weber[2]

[1] Department of Computer Science, The University of Sheffield
{s.foster,g.struth}@dcs.shef.ac.uk
[2] Computer Laboratory, University of Cambridge
tw333@cam.ac.uk

Abstract. We present a new integration of relational and algebraic methods in the Isabelle/HOL theorem proving environment. It consists of a fine grained hierarchy of algebraic structures based on Isabelle's type classes and locales, and a repository of more than 800 facts obtained by automated theorem proving. We demonstrate further benefits of Isabelle for hypothesis learning, duality reasoning, theorem instantiation, and reasoning across models and theories. Our work forms the basis for a reference repository and a program development environment based on algebraic methods. It can also be used by mathematicians for exploring and integrating new variants.

1 Introduction

Kleene and relation algebras provide semantics for programs and a basis for integrating logics for computing systems. Relational and algebraic reasoning is at the core of popular program development methods for imperative, concurrent and functional programs. Algebraic approaches seem particularly useful for modelling and analysing the information and control flow in computing systems. Specifications are often very compact, concise and generic in this setting; proofs are based on first-order equational reasoning, hence calculational, and often much shorter than set-theoretic ones. This makes algebraic methods very suitable for automation. Applications of these methods in program development, however, may require additional mechanisms for reasoning about data structures or data types and higher-order features for fixed points or (co)recursion.

Interactive theorem proving (ITP) systems such as Isabelle/HOL [29] and Coq [7] have been used to implement Kleene and relation algebras [32,21,30], as have special-purpose systems [2]. It has also been shown that automated theorem proving (ATP) systems are quite successful at proving algebraic theorems at textbook level [17,18] and able to support simple program analysis tasks [5].

The strenghts and weaknesses of ITP and ATP are almost orthogonal: Apart from supporting theory hierarchies and proof management, ITP systems offer the expressiveness of higher-order logic for modelling and reasoning. But they require considerable mathematical expertise and sophisticated tactics for feeding manual proofs into the tool. ATP systems, in contrast, have traditionally been

obtimised towards proof search performance; even non-expert users can discharge many calculational proof obligations fully automatically. But these tools are typically limited to first-order logic, and proof management, hypothesis learning and modular reasoning are usually not supported.

Currently, however, there are strong trends to make these two worlds converge. Isabelle, in particular, is being transformed into a theorem proving environment that combines higher-order modelling and reasoning with ATP systems, satisfiability modulo theories (SMT) solvers, decision procedures and counterexample generators. So does this make the best of both worlds available for the automated engineering of relational and algebraic methods?

This paper comes to an overall very positive answer. It proposes Isabelle/HOL as a way forward in engineering relational and algebraic methods. Algebraic theorem proving is now perhaps just as easy with Isabelle as with ATP systems. Important additional benefits arise from Isabelle's higher-order features and SMT integration. Our main contributions are as follows.

We implement a theory hierarchy for relation and Kleene algebras using Isabelle's local specification facilities. This hierarchy includes many popular variants such as basic process algebras, probabilistic Kleene algebras, demonic refinement algebras, omega algebras, Kleene algebras with (anti)domain and modal Kleene algebras. Theorems are inherited across the hierarchy by subclass and sublocale proofs, and the difference between reasoning in Kleene algebras and relation algebras often vanishes.

We prove more than 800 facts in this setting, all of them by ATP using Isabelle's Sledgehammer tool [9]. It turns out that Sledgehammer is very good at learning hypotheses for proofs. Since Isabelle reconstructs all proofs produced by Sledgehammer's external ATP systems with an internally verified tool, the degree of automation is slightly lower than by standalone ATP, and a few difficult theorems do not succeed by ATP alone in one full sweep. Nevertheless, most of the basic facts attempted could be proved automatically in one single step.

We show that Isabelle's higher-order features are very valuable for automated theory engineering. Dualities can be formalised and used to obtain theorems for free. Higher-order variables can be used for instantiating theorems, for instance, from abstract Galois connections or conjugations. Set-theoretic specifications of algebras with explicit carrier sets support mechanised reasoning in universal algebra. By formally linking abstract algebras with concrete models, we obtain a seamless transition between pointwise and pointfree reasoning.

These results suggest that our Isabelle/HOL formalisation has considerable potential for turning relation and Kleene algebras into program development and verification tools that are relatively lightweight yet offer a high degree of automation. We propose it as the basis for a standardised repository for algebraic methods from which further variants, a more extensive library of theorems and proofs, and more concrete program analysis environments can be engineered.

Beyond these technical contributions, another main purpose of this paper is to serve as a tutorial introduction to the relational and algebraic methods community.

2 Isabelle/HOL

Isabelle/HOL [29] is a popular ITP system based on higher-order logic. It has recently mutated from a metalogical framework for specifying logics [31] into a theorem proving environment with integrated decision procedures, ATP systems, SMT solvers and counterexample generators. Intricate mathematical proofs and industrial verification tasks have been carried out with this tool (e.g. [25,22]). For our purposes, the following features of Isabelle are particularly interesting.

First, Isabelle offers a user interface with notation support [29, §4]. Standard notations for Kleene and relation algebras can therefore be defined and used. LaTeX code can automatically be generated from proofs and specifications. In fact, this paper was generated from our Isabelle theory files, with the added benefit that its entire technical content has been formally verified.

Second, the proof language Isar [36] supports a proof style that is natural and easily readable for humans. Equational reasoning, in particular, can be translated almost unchanged into Isabelle [4], and Isar proofs could be published directly in mathematical texts.

Third, Isabelle offers facilities for theory hierarchies and modules through local specifications [16]. A local specification consist of parameters and assumptions: for instance, the operations of Boolean algebra, and the axioms satisfied by these operations. Hierarchies of local specifications can be established through extension or proof. Relation algebra, for instance, can be specified as an extension (*subclass* in Isabelle parlance) of Boolean algebra, and all theorems proved in the latter context become automatically available in the former.

Fourth, Isabelle's Sledgehammer tool [9] integrates various external ATP systems. To obtain trustworthy facts, all external ATP proofs are reconstructed either by proof search using the internal Isabelle-verified theorem prover Metis [19], or more directly by Isar. Sledgehammer selects hypotheses among local verified lemmas and calls the external ATP systems. Since Metis is less efficient, the external provers can be used iteratively to minimise hypothesis sets before Metis is called. Alternatively, an Isar proof can be generated that attempts a stepwise reconstruction of external proofs. SMT solvers, notably Z3, have recently been integrated [10] as an alternative to ATPs. Finally, also counterexample generators such as Nitpick [8] are now part of Isabelle. They allow a game of proof and refutation when developing and prototyping theories.

In contrast to previous rather monolithic ATP proofs in Kleene and relation algebra, where theorems were often attempted directly from the axioms, the user now owns the means of production: proofs can be performed at any level of granularity, from fully automated proofs to textbook-style manual Isar proof scripts, in which the individual proof steps can be automated.

Examples of these features and the different styles of proof are given in the remainder of this paper. Isabelle is well documented; and more information can be found at the tool web site [1]. Our complete repository, including all facts used in this paper, can be found online [33]. All hypotheses used in the proofs presented can easily be found by searching their names in the repository.

3 Implementing a Kleene Algebra Hierarchy

We informally use the term "Kleene algebra" for a family of algebras based on variants of idempotent semirings—or dioids—which are extended with operations for finite or infinite iteration. Different variants correspond to different system semantics and different intended applications, inlcuding processes, probabilistic systems, program refinement, sequential or concurrent program analysis. Program semantics for partial or total correctness and formalisms such as dynamic, temporal or Hoare logics can be obtained by adding further axioms.

Isabelle's local specification facilities allow us to build a modular theory hierarchy for Kleene algebras and to inherit theorems across theories and models. We outline the approach in this section. To simplify the presentation, the code in this paper sometimes differs slightly from that in the repository.

Our hierarchy starts with the axiomatic class of join semilattices.

Class *join-semilattice* = *plus-ord* +
  assumes *add-assoc*: $(x+y)+z = x+(y+z)$
  and *add-comm*: $x+y = y+x$
  and *add-idem*: $x+x = x$

It expands a predefined class *plus-ord*, which provides notation for addition and order and connects them via $x \leq y \equiv x + y = y$ by the semilattice axioms.

Linking the equational view on semilattices with the order-based one requires showing that join semilattices are partial orders.

Sublass (**in** *join-semilattice*) *order*
Proof
  fix $x\ y\ z$
  show $x \leq x$ **by** (*metis add-idem leq-def*)
  show $x \leq y \Longrightarrow y \leq x \Longrightarrow x = y$ **by** (*metis add-comm leq-def*)
  show $x \leq y \Longrightarrow y \leq z \Longrightarrow x \leq z$ **by** (*metis add-assoc leq-def*)
  show $x < y \longleftrightarrow x \leq y \wedge \neg\ (y \leq x)$ **by** (*metis strict-leq-def*)
qed

The individual proof goals are prescribed by Isabelle, in particular that for $<$ which is usually a definition. Isabelle's built-in ATP system Metis is called on each goal with the hypotheses indicated. More information about such proofs can be found in the following sections. All Isabelle theorems for partial orders are now available for join semilattices.

The next level of our hierarchy implements variants of semirings and dioids, whose multiplication symbol is provided by a predefined class *mult*.

Class *near-semiring* = *plus* + *mult* +
  assumes *mult-assoc*: $(x\cdot y)\cdot z = x\cdot(y\cdot z)$
  and *add-assoc'*: $(x+y)+z = x+(y+z)$
  and *add-comm'*: $x+y = y+x$
  and *distr*: $(x+y)\cdot z = x\cdot z+y\cdot z$

Class *near-dioid* = *near-semiring* + *plus-ord* +
  assumes *idem*: $x+x = x$

Near-dioids form the basis of process algebras like CCS or ACP [6]. By definition, near-diods are near-semirings, and all near-semiring theorems are automatically inherited. But the link with semilattices requires proof.

Sublass (**in** *near-dioid*) *join-semilattice* — automatic proof omitted

Other variants of semirings and dioids can be obtained by extension, e.g.,

Class *pre-dioid* = *near-dioid* +
  assumes *subdistl*: $z{\cdot}x \leq z{\cdot}(x{+}y)$

Class *semiring* = *near-semiring* +
  assumes *distl*: $x{\cdot}(y{+}z) = x{\cdot}y{+}x{\cdot}z$

Class *dioid* = *semiring* + *near-dioid*

Predioids form the basis for game algebras [15] and probabilistic Kleene algebras [27]. Semirings and dioids have various applications in different fields of mathematics and computing. We have extended these structures with additive and multiplicative units in the usual ways. Different variants are needed again for different applications.

At the next level of the hierarchy we add a Kleene star to our variants of semirings with 1. We only show the extremal points of this level.

Class *left-near-kleene-algebra* = *near-dioid-one* + *star* +
  assumes *star-unfoldl*: $1{+}x{\cdot}x^\star \leq x^\star$
  and *star-inductl*: $z{+}x{\cdot}y \leq y \longrightarrow x^\star{\cdot}z \leq y$

Class *kleene-algebra* = *left-kleene-algebra-zero* +
  assumes *star-inductr*: $z{+}y{\cdot}x \leq y \longrightarrow z{\cdot}x^\star \leq y$

Isolation of the left star unfold and left induction axiom is important for variants such as probabilistic Kleene algebras. A right unfold law can be proved from the three axioms for all variants except near Kleene algebras.

At the final level of the hierarchy we extend different algebraic variants with an omega operation for infinite iteration. Since omega algebras [11] will not appear any further in this paper, we refer to our repository for details.

Our repository also contains semigroups, semirings and Kleene algebras with domain and antidomain [12,14], including modal Kleene algebras [28] and demonic refinement algebras [37]. Domain semirings essentially subsume Kleene algebras with tests [23]. Most theories have been developed to the present state of knowledge. A proof environment for dynamic logics, temporal logics or Hoare logic can be obtained with minor effort. Structures such as concurrent Kleene algebras, action algebras and action lattices, and Kleene algebras with converse are under development.

4 Integrating Relation Algebras

Our implementation of relation algebras follows Roger Maddux's book [26], which itself is based on Alfred Tarski's original paper [35]. It uses Hunting-

ton's axioms for Boolean algebras. These are rather minimalist, and we have added definitions for the partial order, the maximal and minimal element, and meet.

Class *boolean-algebra* = *plus-ord* + *uminus* + *one* + *zero* + *mult* +
  assumes *join-assoc*: $(x+y)+z = x+(y+z)$
  and *join-comm*: $x+y = y+x$
  and *compl*: $x = -((-x)+(-y))+(-((-x)+y))$
  and *one-def*: $x+(-x) = 1$
  and *zero-def*: $(-\ 1) = 0$
  and *meet-def*: $x \cdot y = -((-x)+(-y))$

For applications in Section 6, we introduce the concept of *conjugate* functions over a Boolean algebra [20], which gives rise to Boolean algebras with operators and Galois connections. In particular, they yield theorems for free.

Definition (**in** *boolean-algebra*)
  conjugation-p f g $\equiv \forall\, x\ y.\ (f(x) \cdot y = 0 \longleftrightarrow x \cdot g(y) = 0)$

Next we obtain relation algebras from Boolean algebras and show that every relation algebra is a dioid.

Class *relation-algebra* = *boolean-algebra* + *composition* + *unit* + *converse* +
  assumes *comp-assoc*: $(x;y);z = x;(y;z)$
  and *comp-unitr*: $x;e = x$
  and *comp-distr*: $(x+y);z = x;z + y;z$
  and *conv-invol*: $(x^\smile)^\smile = x$
  and *conv-add*: $(x+y)^\smile = x^\smile + y^\smile$
  and *conv-contrav*: $(x;y)^\smile = y^\smile;x^\smile$
  and *comp-res*: $x^\smile;(-(x;y)) \leq -y$

Sublocale *relation-algebra* $\subseteq$ *dioid-one-zero* $(op\ +)\ (op\ ;)\ (op \leq)\ (op <)\ 0\ e$
  — automatic proof omitted

In this case, we establish a sublocale (instead of a subclass) relationship because different signatures need to be matched.

To link relation algebras with Kleene algebras, we add a reflexive-transitive closure operation.

Class *relation-algebra-rtc* = *relation-algebra* + *star* +
  assumes *rtc-unfoldl*: $e+x;x^\star \leq x^\star$
  and *rtc-inductl*: $z+x;y \leq y \longrightarrow x^\star;z \leq y$
  and *rtc-inductr*: $z+y;x \leq y \longrightarrow z;x^\star \leq y$

Sublocale *relation-algebra-rtc* $\subseteq$
  kleene-algebra $(op\ +)\ (op\ ;)\ (op \leq)\ (op <)\ 0\ e\ (op\ ^\star)$
  — automatic proof omitted

All facts for Kleene algebra are now available for relation algebras; the difference between reasoning in Kleene algebra and relational reasoning becomes often invisible for users.

Our implementation of relation algebra includes the most important textbook concepts and theorems about functions, subidentities or tests, domain and range elements, vectors and points.

Relation algebras have previously been implemented in Isabelle/HOL by Gritzner and von Oheimb [30]. At that time, however, type classes, locales and ATP integration were not available, and proving relational theorems required strong user interaction. Their system therefore uses a translation to relational atom structures and complex algebras to increase the degree of automation, which was certainly impressive in 1997. A comparison nicely illustrates the technological advances in automated and interactive theorem proving over the last decade.

5 Automated Algebraic Proofs with Isabelle

Our development of a repository of verified theorems for relational and algebraic methods in Isabelle is largely based on ATP via Sledgehammer. To our knowledge, it is the first larger case study on automated algebraic reasoning within Isabelle.

In this paper, we can only present a few example proofs out of the more than 800 proofs in our repository. Our main intention is to illustrate the range of possibilities available and to compare this new approach to our previous work with standalone ATP systems.

We show that, in Kleene algebras, the Kleene star indeed axiomatises the reflexive transitive closure. (In our repository, these proofs have been obtained in slightly weaker algebras.) The first proof goal, namely that $x^\star$ is reflexive, transitive, and extends x, can be stated as follows.

Lemma (**in** *kleene-algebra*) *star-rtc1*: $1+x+x^\star\cdot x^\star \leq x^\star$
— proof below

To prove this lemma, we called Sledgehammer, which itself called a number of external ATP systems. In this case the external ATPs returned with proofs within seconds. In a second round, the ATP systems were rerun to minimise the hypothesis set needed. Finally, Isabelle's built-in ATP system Metis performed proof search on a selected hypothesis set to reconstruct the external proof within Isabelle. In our example, this yields

by (*metis add-lub star-ext star-ref star-trans*)

This list shows that Sledgehammer has "learned" appropriate hypotheses for proving this fact. Instead of the star axioms, for instance, two lemmas are used that have previously been verified. This sort of proof management is not supported by standalone ATP systems.

Alternatively, the SMT solver Z3 can be used, whose proof is again reconstructed internally:

by (*smt add-lub star-ext star-ref star-trans*)

On our repository, the overall performance of Z3 is comparable to Metis, but both are less successful than the external ATP systems. This explains why hypothesis minimisation is an important feature of Sledgehammer.

A third variant is asking Sledgehammer for an Isar proof. In our example, the following Isar proof script is generated for lemma *star-rtc1*.

Proof −
have $1 \leq x^\star$ **by** (*metis star-ref*)
hence $1 \leq x^\star \wedge x \leq x^\star$ **by** (*metis star-ext*)
hence $1 + x \leq x^\star$ **by** (*metis add-lub*)
hence $1 + x \leq x^\star \wedge x^\star \cdot x^\star \leq x^\star$ **by** (*metis star-trans*)
thus $1 + x + x^\star \cdot x^\star \leq x^\star$ **by** (*metis add-lub*)
qed

Individual proof steps are again verified by Metis, but, due to the change of granularity, Isar proofs can often be checked more quickly. The Isar proof feature of Sledgehammer is, however, still experimental and often rather unstable.

It now remains to show that $x^\star$ is the *least* element above x that is reflexive and transitive. The proof, which can be found in our repository, can again be fully automated, but Metis takes several minutes.

A particular benefit of the ITP approach is that users own the means of production. Instead of fully automated proofs, textbook-style proofs can be given as Isar scripts whenever Sledgehammer fails or the user chooses so. The following example shows such a proof in which, of course, the user's expertise is needed.

Lemma (**in** *kleene-algebra*) *star-slide*: $(x{\cdot}y)^\star{\cdot}x = x{\cdot}(y{\cdot}x)^\star$
Proof −
have $x{\cdot}(y{\cdot}x)^\star \leq x{\cdot}(y{\cdot}(x{\cdot}y)^\star{\cdot}x{+}1)$
by (*metis add-comm mult-assoc star-unfoldr-eq star-slide1*
mult-isor add-iso mult-isol)
hence $x{\cdot}(y{\cdot}x)^\star \leq (x{\cdot}y{\cdot}(x{\cdot}y)^\star{+}1){\cdot}x$
by (*metis distl mult-assoc mult-oner distr mult-onel*)
hence $x{\cdot}(y{\cdot}x)^\star \leq (x{\cdot}y)^\star{\cdot}x$
by (*metis add-comm star-unfoldl-eq*)
thus *?thesis* **by** (*metis antisym-conv star-slide1*)
qed

In the first step, the well-known star unfold law $1 + x^\star \cdot x = x^\star$, which has previously been verified, is used. The second and third step use essentially distributivity and star unfold. The slide law $(x \cdot y)^\star \cdot x \leq x \cdot (y \cdot x)^\star$, which again has been verified before, is used in the final step.

Our experiments suggest that handwritten proofs in relation algebra and Kleene algebra can usually be translated directly into readable Isar scripts.

The theory hierarchy, combined with ATP systems and counterexample generators, helps finding the weakest structure in which theorems hold. We were able, for instance, to prove all identities that were known to hold in Kleene algebras and all formulas in omega algebras—including the above slide rule—without right star induction. These empirical observations suggest that weaker variants of Kleene (and omega) algebras may already be complete for the algebra of (omega-)regular events.

6 Higher-Order Features

Apart from managing ATP proofs, Isabelle's higher-order features are very useful for theory engineering. Here we give only two examples: the exploitation of duality for domain and range semirings, and the instantiation of theorems of Boolean algebras with operators in relation algebras.

The domain of a relation is the set of all states on which a relation is enabled. In the semiring of relations, $domain(x) \equiv \{(p,p) \mid \exists q.\,(p,q) \in x\}$. More abstractly, we use a domain operation d and the following axioms [14].

Class *domain-semiring* = *semiring-one-zero* + *plus-ord* + *domain-op* +
  assumes *d1*: $x+(d(x)\cdot x) = d(x)\cdot x$
  and *d2*: $d(x\cdot y) = d(x\cdot d(y))$
  and *d3*: $d(x)+1 = 1$
  and *d4*: $d(0) = 0$
  and *d5*: $d(x+y) = d(x)+d(y)$

We can prove that domain elements are precisely the fixpoints of the domain operation. This domain characterisation is not merely an equivalence between domain semiring identities; it involves an existential quantifier.

Lemma (**in** *domain-semiring*) *d-fixpoint*: $(\exists\, y.\ x = d(y)) \longleftrightarrow d(x) = x$
  — automatic proof omitted

Semiring duality is duality with respect to opposition, that is, the order of multiplication is swapped. The notion extends to semirings with one and zero, and preservation of theorems can be expressed by the following lemma, which states that the opposite contravariant multiplication induces again a semiring.[3]

Definition (**in** *mult*) $x \odot y \equiv y \cdot x$

Lemma (**in** *semiring-one-zero*) *dual-semiring-one-zero*:
  class.semiring-one-zero 0 1 (*op* +) (*op* $\odot$) — automatic proof omitted

In the context of domain semirings, the dual of domain is range:

Class *range-semiring* = *semiring-one-zero* + *plus-ord* + *range-op* +
  assumes *r1*: $x+(x\cdot r(x)) = x\cdot r(x)$
  and *r2*: $r(x\cdot y) = r(r(x)\cdot y)$
  and *r3*: $r(x)+1 = 1$
  and *r4*: $r(0) = 0$
  and *r5*: $r(x+y) = r(x)+r(y)$

Sublocale *range-semiring* $\subseteq$ *domain-semiring r* (*op* +) (*op* $\leq$) (*op* $<$) *0 1* (*op* $\odot$)
  — automatic proof omitted

This sublocale expression states that range semirings are duals of domain semirings with respect to opposition. It allows us to obtain statements about range directly by dualising domain statements. In fact, all statements about range in

[3] The logically equivalent **Sublocale** *semiring-one-zero* $\subseteq$ *semiring-one-zero* 0 1 (*op* +) (*op* $\odot$) is not accepted by Isabelle for technical reasons.

our repository have been obtained this way. The following range export law, for instance, is automatically derived from its dual domain export law, which has been proved by other means.

Lemma (**in** *range-semiring*) *range-export*: $r(x{\cdot}r(y)) = r(x){\cdot}r(y)$
by (*metis dual.domain-export opp-mult-def*)

In the proof, the domain export law $d(d(x){\cdot}y) = d(x){\cdot}d(y)$, which has previously been proved, is dualised by the sublocale statement above. The above definition of $\odot$ is also needed in the proof. The hypotheses have been found automatically by Sledgehammer.

Next we show how abstract theorems about conjugate functions in Boolean algebras with operators can automatically be instantiated to more concrete theorems about relation algebras. In our hierarchy, relation algebras form a subclass of Boolean algebras, hence conjugation is available for relation algebras, too.

The following lemma shows that the functions $\lambda y.\, x; y$ and $\lambda y.\, x^{\smile}; y$ are conjugate in relation algebras. This is, in fact, one of the famous Schröder rules.

Lemma (**in** *relation-algebra*) *schroeder-1*: $(x;y){\cdot}z = 0 \longleftrightarrow y{\cdot}(x^{\smile};z) = 0$
Proof −
have $(x;y){\cdot}z = 0 \longleftrightarrow (z^{\smile};x){\cdot}y^{\smile} = 0$ **by** (*metis conv-invol peirce*)
thus *?thesis* **by** (*metis conv-invol conv-zero conv-contrav conv-times meet-comm*)
qed

Lemma (**in** *relation-algebra*) *schroeder-1-var*:
conjugation-p $(\lambda\ y\ .\ x\ ;\ y)$ $(\lambda\ y\ .\ x^{\smile}\ ;\ y)$
by (*metis conjugation-p-def schroeder-1*)

Lemma *schroeder-1* proves the Schröder law explicitly from Peirce's formula $(x; y) \cdot z^{\smile} = 0 \longleftrightarrow (y; z) \cdot x^{\smile} = 0$, which can be found in our repository. It is used in the proof of Lemma *schroeder-1-var* to express the conjugation property. The following modular law of relation algebra, for instance, can then be obtained automatically by instantiating an abstract modular law of Boolean algebras with operators that holds for conjugate functions.

Lemma (**in** *boolean-algebra*) *modular-1*:
assumes *conjugation-p f g* **shows** $f(x){\cdot}y \leq f(x{\cdot}g(y)){\cdot}y$
— proof omitted

Corollary (**in** *relation-algebra*) *modular-1′*: $(x;y){\cdot}z \leq (x;(y{\cdot}(x^{\smile};z))){\cdot}z$
by (*metis schroeder-1-var modular-1*)

Such higher-order features are particularly useful for modal semirings and modal Kleene algebras, where dualities, conjugations and Galois connections relate the forward and backward box and diamond operators. Dualities can then be used as theorem transformers and conjugations as theorem generators. Theorems for free can thus be effectively realised by theory engineering in Isabelle.

7 Abstract versus Set-Theoretic Classes

This section discusses an alternative set-theoretic specification of algebraic structures that uses explicit carrier sets. The abstract type-based approach described in the previous sections is usually sufficient to reason *in* algebraic structures, e.g., to prove identities in Kleene algebra. However, limitations show up when reasoning *about* these structures, for instance about subalgebras. Explicit carrier sets overcome these limitations in Isabelle/HOL. They allow specifications that are more appropriate for mechanising model theory or universal algebra.

To keep our example simple, we show a carrier-based specification for domain semigroups [12] instead of domain semirings.

Class *carrier-semigroup* = *mult* +
  fixes *carrier* :: *'a set*
  assumes *m-closed*: $[\![x{\in}carrier;\ y{\in}carrier]\!] \Longrightarrow x{\cdot}y \in carrier$
  and *m-assoc*: $[\![x{\in}carrier;\ y{\in}carrier;\ z{\in}carrier]\!] \Longrightarrow (x{\cdot}y){\cdot}z = x{\cdot}(y{\cdot}z)$

Class *carrier-domain-semigroup* = *carrier-semigroup* + *domain-op* +
  assumes *d-closed*: $x{\in}carrier \Longrightarrow d(x) \in carrier$
  and *d1*: $x{\in}carrier \Longrightarrow d(x){\cdot}x = x$
  and *d2*: $[\![x{\in}carrier;\ y{\in}carrier]\!] \Longrightarrow d(x{\cdot}d(y)) = d(x{\cdot}y)$
  and *d3*: $[\![x{\in}carrier;\ y{\in}carrier]\!] \Longrightarrow d(d(x){\cdot}y) = d(x){\cdot}d(y)$
  and *d4*: $[\![x{\in}carrier;\ y{\in}carrier]\!] \Longrightarrow d(x){\cdot}d(y) = d(y){\cdot}d(x)$

In contrast to our previous abstract specifications, each assumption is now relativised to the carrier set and closure conditions for the operations have been added, as usual in algebra.

By using carrier sets, we can now prove automatically that the set of domain elements in a domain semigroup forms a domain subsemigroup. Because Isabelle/HOL does not offer dependent types, this would be difficult, if not impossible, to state without explicit carrier sets.

Lemma (**in** *carrier-domain-semigroup*) *domain-subsemigroup*:
  class.carrier-domain-semigroup (*op* ·) $\{x{\in}carrier.\ d(x){=}x\}$ *d*
  — automatic proof omitted

The lemma states that the set of all elements x in the carrier that satisfy $d(x) = x$ endowed with the operations $\cdot$ and d forms a domain semigroup with carrier.

This example suggests that metalogical statements could still be proved by ATP when using classes with explicit carrier sets, although set theory is now involved. Additional experiments suggest that carrier sets may cause substantial overhead and more fragile proof automation, but further evidence is needed.

The abstract and the set-theoretic level can be linked, and theorems can be transferred between them. Given an abstract algebraic structure, the universal set over its type, i.e., $\{x.\ True\}$, constitutes a suitable carrier set for the corresponding set-theoretic structure. Conversely, given a structure with explicit carrier set C, the subtype of all elements in C constitutes a suitable base type for the corresponding abstract structure. However, due to Isabelle/HOL's lack of dependent types, this subtype can be defined only when C does not depend on local parameters.

8 Integrated Point-Wise and Point-Free Reasoning

We have discussed how a hierarchy of algebraic structures can be defined in Isabelle, and we explained how this hierarchy is useful for organising theory engineering. However, in intended models, the theorems thus obtained are conditional: e.g., $0^* = 1$, *provided* 0, 1 and $\cdot^*$ denote the respective operations of, e.g., a Kleene algebra. We cannot apply these theorems in concrete models unless we know that the operations in these models satisfy the axioms of (in this case) Kleene algebra. Three important models of Kleene algebras are sets of traces, formal languages, and binary relations [13]. In this section we sketch how abstract "point-free" reasoning in relation and Kleene algebra can be formally linked with "point-wise" reasoning in concrete models. A full account can again be found in our repository.

A *trace* is given by a list of odd length whose first element is a state, and in which states and actions alternate. Isabelle provides pair types $'\alpha \times '\beta$ and a polymorphic list type $'\alpha$ *list*, but has limited support for predicative subtypes. Therefore, the following (equivalent) characterisation of traces is easier to work with formally: a trace is a pair consisting of an initial state and a list of transitions, where each transition is a pair of action and successor state.

Types $('\sigma, '\alpha)$ *trace* $= '\sigma \times ('\alpha \times '\sigma)$ *list*

We define functions *first* and *last* that extract the first and last state of a trace, respectively. Multiplication of traces t and u is a partial operation that is defined when the last state of t is equal to the first state of u.

Definition $t \cdot u \equiv$ *if* $last(t) = first(u)$ *then* $(\pi_1(t), \pi_2(t) \mathbin{@} \pi_2(u))$ *else undefined*

Here π_1, π_2 are the projections for pairs, and @ denotes list concatenation.

Multiplication can be lifted to a complex product on sets of traces in the usual way. HOL is a logic of total functions; *undefined* above—contrary to common mathematical usage—is a constant of the logic that merely denotes some completely unspecified value. In the complex product, we only consider pairs of traces whose product is defined.

Definition $T \cdot U \equiv \bigcup t{\in}T.\ \bigcup u{\in}\{u \in U.\ last(t) = first(u)\}.\ \{t \cdot u\}$

The empty set is the multiplicative zero, and the set $\bigcup p\ \{(p, [])\}$ of all single-state traces is the multiplicative unit. In fact, sets of traces form a Kleene algebra (where addition is given by set union, the order coincides with the subset relation, and the star operation is given by arbitrary iterations of multiplication, as in language theory). Isabelle provides the **Interpretation** command to formally establish this relationship.

Interpretation *kleene-algebra* $(op \cup)$ $(op \cdot)$ $(op \subseteq)$ $(op \subset)$ $\emptyset$ $(\bigcup p.\ \{(p, [])\})$ $(\cdot^\star)$
— proof omitted

Isabelle sets up a proof obligation that requires the user to show that the operations listed indeed satisfy the axioms of Kleene algebras. Simple axioms can be verified automatically, while harder ones—in particular verification of the

star axioms, which now require induction—need user interaction. The hierarchical approach leads to well-structured interpretation proofs: the fact that sets of traces form left Kleene algebras (with 0) and Kleene algebras (with 0) in which also the right star induction axiom holds can be verified incrementally. This is useful, for instance, when attempting completeness proofs for the weakest possible axiomatisation.

In addition to the trace model of Kleene algebras, we have formalised the models of formal languages (i.e., sets of words, where words are implemented as lists, and word multiplication is given by list concatenation) and binary relations (i.e., sets of ordered pairs, with multiplication given by relative product).

Having established, for instance, that binary relations form a Kleene algebra, Isabelle immediately makes all abstract theorems of Kleene algebra proved in the system available for binary relations as well. For instance,

Lemma $y^\star \circ x^\star \leq x^\star \circ y^\star \Longrightarrow (x + y)^\star \leq x^\star \circ y^\star$
— automatic proof omitted

is the instance of the abstract Church-Rosser theorem of Kleene algebra for binary relations (with $\circ$ denoting relative product and $\cdot^\star$ the reflexive-transitive closure of a relation). We can therefore seamlessly switch back and forth between point-free abstract reasoning (at the algebraic level) and point-wise concrete reasoning (in the model of binary relations). Krauss and Nipkow [24] artfully explore this connection to decide (in)equations of binary relations using an equivalence checker for regular expressions. Making their integrated decision procedure for regular expressions available for deciding identities in Kleene algebra could increase the proof automation for this class significantly.

Models such as traces, languages and binary relations carry, of course, a richer structure than what has been captured by Kleene or relation algebras. This can to a certain extent be captured abstractly by defining Kleene algebras over quantales or relation algebras over complete Boolean algebras. While a specification of these structures in Isabelle is straightforward, the suitability of Sledgehammer in this higher-order context remains another interesting open question.

9 Future Directions

The work in this paper documents only the initial steps towards an Isabelle repository for algebraic methods. The directions for future work that arise from this work are perhaps more important than the results obtained so far. We envisage three main directions:

(a) The creation of a standardised repository for relational and algebraic methods that includes the most important variants, models and theorems, and reflects the state of the art in the field.
(b) The integration of this repository into a development and verification environment for programs and computing systems that combines lightweight relational specification languages with heavyweight automation.

(c) The exploration of more advanced mathematics, e.g., the model theory and universal algebra of Kleene algebras and relation algebras within Isabelle.

These directions can best be addressed through a joint effort within the RAMiCS community, and the repository, its notation, conceptualisation, structure and design is therefore open to additions and debate. A possible way forward is the creation of a Wiki to which any researcher in the area will be able to contribute through a moderated process. A minimal requirement would be that all documents checked in must compile with Isabelle.

Using standalone ATP systems for reasoning automatically about Kleene and relation algebras showed that proofs of calculational statements at textbook level can usually be automated. But there are several limitations, in particular the fact that reasoning about data structures and data types such as numbers, arrays, lists is not sufficiently supported, and that proofs by induction are not possible. The presence of both SMT solvers and higher-order features in Isabelle/HOL seems of great benefit here and certainly deserves further exploration. Automating large parts of program analyses in this new setting seems possible, although we cannot provide any empirical evidence yet. The development and integration of decision procedures for fragments of Kleene algebras and relation algebras seems also very beneficial in this respect.

Finally, we are not aware that a systematic formalisation of model theory or universal algebra in Isabelle/HOL has so far been attempted. Our proof experiments show that some simple metamathematical concepts and proofs can efficiently be handled by ATP, which might facilitate this endeavour. Completeness proofs for variants of Kleene algebras are particularly important for integrating decision procedures and further enhancing automation.

10 Conclusion

We have shown that Isabelle/HOL is a highly useful environment for automated theorem proving in relation algebras and variants of Kleene algebras that overcomes previous limitations of standalone ATP proofs with these structures. Main advantages include theory hierarchies, proof management, hypothesis learning, cross-theory reasoning, automatisation of duality and abstraction/instantiation, integration of abstract and model-based reasoning and other higher-order features that enable metatheory reasoning. These suitably complement the sheer proof power of ATP systems on calculational proofs. The integration of SMT solvers into Isabelle promises additional benefits for reasoning about data structures and data types.

These results provide further evidence that algebraic and relational methods are very suitable as lightweight formal methods with heavyweight automation; and that our Isabelle repository is a significant step in that direction.

But there is still scope for improvement from the tool side, too. The existing automation gap between the external ATP systems and the internal proof reconstruction needs to be closed, and a wider range of ATP systems should

be integrated. For this purpose, ATP output should be further standardised (cf. TSTP [34]). Other valuable features would be better ATP support for order-based reasoning (ordered chaining calculi [3]), better sort or type support, and enhanced techniques for hypothesis learning and, more generally, large hypothesis sets.

Acknowledgments The authors would like to thank Andy Gordon, Tony Hoare, Peter Jipsen and Makarius Wenzel for fruitful discussions. We acknowledge funding from EPSRC grants EP/F067909/1 and EP/G031711/1.

References

1. Isabelle website, http://isabelle.in.tum.de/. Accessed February 20, 2011.
2. Aboul-Hosn, K., Kozen, D.: KAT-ML: an interactive theorem prover for Kleene algebra with tests. J. Applied Non-Classical Logics 16(1-2), 9–34 (2006)
3. Bachmair, L., Ganzinger, H.: Ordered chaining calculi for first-order theories of transitive relations. J. ACM 45(6), 1007–1049 (1998)
4. Bauer, G., Wenzel, M.: Calculational reasoning revisited (an Isabelle/Isar experience). In: Boulton, R.J., Jackson, P.B. (eds.) TPHOLs 2001. LNCS, vol. 2152, pp. 75–90. Springer (2001)
5. Berghammer, R., Struth, G.: On automated program construction and verification. In: Bolduc, C., Desharnais, J., Ktari, B. (eds.) MPC 2010. LNCS, vol. 6120, pp. 22–41. Springer (2010)
6. Bergstra, J.A., Fokkink, W.J., Ponse, A.: Process algebra with recursive operations. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, pp. 333–389. Elsevier (2001)
7. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development, Coq'Art: the Calculus of Inductive Constructions. Springer (2004)
8. Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 131–146. Springer (2010)
9. Böhme, S., Nipkow, T.: Sledgehammer: Judgement day. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 107–121. Springer (2010)
10. Böhme, S., Weber, T.: Fast LCF-style proof reconstruction for Z3. In: Kaufmann, M., Paulson, L.C. (eds.) ITP 2010. LNCS, vol. 6172, pp. 179–194. Springer (2010)
11. Cohen, E.: Separation and reduction. In: Backhouse, R.C., Nuno Oliveira, J. (eds.) MPC 2000. LNCS, vol. 1837, pp. 45–59. Springer (2000)
12. Desharnais, J., Jipsen, P., Struth, G.: Domain and antidomain semigroups. In: Berghammer, R., Jaoua, A., Möller, B. (eds.) RelMiCS 2009. LNCS, vol. 5827, pp. 73–87. Springer (2009)
13. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM TOCL 7(4), 798–833 (2006)

14. Desharnais, J., Struth, G.: Internal axioms for domain semirings. Science of Computer Programming 76(3), 181–203 (2011)
15. Goranko, V.: The basic algebra of game equivalence. Studia Logica 75, 221–238 (2003)
16. Haftmann, F., Wenzel, M.: Local theory specifications in Isabelle/Isar. In: TYPES 2008. pp. 153–168 (2008)
17. Höfner, P., Struth, G.: Automated reasoning in Kleene algebra. In: Pfenning, P. (ed.) CADE 2007. LNCS, vol. 4603, pp. 279–294. Springer (2007)
18. Höfner, P., Struth, G.: On automating the calculus of relations. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS, vol. 5195, pp. 50–66. Springer (2008)
19. Hurd, J.: System description: The Metis proof tactic. In: Benzmueller, C., Harrison, J., Schuermann, C. (eds.) ESHOL 2005. pp. 103–104. arXiv.org (2005)
20. Jónsson, B., Tarski, A.: Boolean algebras with operators, Part I. American Journal of Mathematics 73, 891–939 (1951)
21. Kahl, W.: Calculational relation-algebraic proofs in Isabelle/Isar. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS 2003. LNCS, vol. 3051, pp. 178–190. Springer (2003)
22. Klein, G., et al.: seL4: Formal verification of an OS kernel. Comm. ACM 53(6), 107–115 (2010)
23. Kozen, D.: Kleene algebra with tests. ACM TOPLAS 19(3), 427–443 (1997)
24. Krauss, A., Nipkow, T.: Proof pearl: Regular expression equivalence and relation algebra. Journal of Automated Reasoning (2011), to appear
25. Mackenzie, D.: What in the name of Euclid is going on here? Science 307(5714), 1402–1403 (Mar 2005)
26. Maddux, R.D.: Relation Algebras. Elsevier (2006)
27. McIver, A., Weber, T.: Towards automated proof support for probabilistic distributed systems. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS, vol. 3835, pp. 534–548. Springer (2005)
28. Möller, B., Struth, G.: Algebras of modal operators and partial correctness. Theoretical Computer Science 351(2), 221–239 (2006)
29. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL – A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
30. von Oheimb. D., Gritzner, T.F.: RALL: Machine-supported proofs for relation algebra. In: CADE 1997. pp. 380–394 (1997)
31. Paulson, L.C.: Isabelle: The next seven hundred theorem provers. In: Lusk, E.L., Overbeek, R.A. (eds.) CADE 1988. LNCS, vol. 310, pp. 772–773. Springer (1988)
32. Struth, G.: Abstract abstract reduction. J. Logic and Algebraic Programming 66(2), 239–270 (2006)
33. Struth, G., et al.: Isabelle algebraic methods repository (2011), http://www.dcs.shef.ac.uk/~georg/isa. Accessed February 20, 2011.
34. Sutcliffe, G., Suttner, C.: The TPTP problem library for automated theorem proving. http://www.tptp.org. Accessed February 20, 2011.
35. Tarski, A.: On the calculus of relations. J. Symbolic Logic 6(3), 73–89 (1941)
36. Wenzel, M.: Isabelle/Isar— a versatile environment for human-readable formal proof documents. Ph.D. thesis, Institut für Informatik, Technische Universität München, Germany (2002)
37. von Wright, J.: Towards a refinement algebra. Science of Computer Programming 51(1-2), 23–45 (2004)