Calculational Relation-Algebraic Proofs in Isabelle/Isar

Wolfram Kahl

Department of Computing and Software
McMaster University

Abstract. We propose a collection of theories in the proof assistant Isabelle/Isar that support calculational reasoning in and about heterogeneous relational algebras and Kleene algebras.

1 Introduction and Related Work

Abstract relational algebra is a useful tool for high-level reasoning that, through appropriate models, provides theorems in fields such as data mining, fuzzy databases, graph transformation, and game theory. Frequently, once an application structure is identified as a model of a particular relation-algebraic theory, that theory becomes the preferred reasoning environment in this application area. Since relation-algebraic reasoning typically follows a very calculational style, and, due to the expressive power of its constructs and rules, also proceeds in relatively formal steps, one would expect that computer support for this kind of reasoning should be relatively easy to implement. Since the number of rules that can be applied in any given situation tends to be quite large, and expressions can become quite complex, computer support also appears to be very desirable.

Some applications, such as fuzzy relations [Fur98], or graph transformation [Kah01, Kah02], involve structures where complements in particular may not be available. These structures therefore require weaker formalisations, such as Dedekind categories, or other kinds of allegories [FS90]. Besides the category structure encompassing composition and identities, allegories are equipped with meet and converse, and are closely related with data-flow graphs. The dual view of control-flow graphs corresponds to Kleene algebras which besides composition and identities feature join and iteration (via the Kleene star). Recent years have seen a rapid growth of interest in computer science applications of Kleene algebras and related structures, studied from a relational perspective, see e.g. [DM01]. Since all of these structures still share a considerable body of common theory, it appears desirable to structure the theory support for relation-algebraic reasoning in such a way that the organisation of results reflects the necessary premises in an intuitive way. On the "data-flow side", the different kinds of allegories proposed in [FS90] offer themselves naturally for this structuring; on the "control-flow side", we will use Kleene algebras and several extensions like Kleene algebras with tests [Koz97] and Kleene algebras with domain [DMS03].

R. Berghammer et al. (Eds.): RelMiCS/Kleene-Algebra Ws 2003, LNCS 3051, pp. 178–190, 2004.

While allegories, being extensions of categories, correspond to the *heterogeneous* approach to relation algebras of [SS93, SHW97], Kleene algebras are mostly studied in a *homogenous* setting. Since we believe that the heterogeneous approach with its strongly-typed flavour has significant advantages in particular when it comes to computer science applications, we will adopt it throughout. Therefore, in the remainder of this paper, "Kleene algebra" always means "typed Kleene algebra" in the sense of [Koz98], and then, both allegories and typed Kleene algebras are considered as kinds of categories.

Furthermore, since complex applications will require the possibility to reason about relational algebras built from other relational algebras via certain construction principles — for example product algebras or matrix algebras — it becomes a necessity to allow reasoning not only *within* a single relational algebra, but also *about* several structures and the connections between them, using operations from both *in a single formula*. Therefore, relational algebras must become addressable as objects, as in the relational programming framework RATH [KS00].

Quite a few projects in the recent past have strived to provide computer-aided proof assistance for abstract relation-algebraic reasoning, each with its particular motivation and priorities, all quite different from the approach we are proposing in this paper.

The relation-algebraic formula manipulation system and proof checker RALF [BH94, HBS94, KH98] was designed as a special-purpose proof assistant for abstract (heterogeneous) relation algebras with the goal of supporting proofs in the calculational style typical for relation-algebraic reasoning. RALF has a graphical user interface presenting goal formulae in their tree structure, a feature that allows easy interactive selection of the subexpressions to be transformed by proof steps. During interaction, only the current sub-goal is visible; document output is generated in the calculational style of proof presentation. RALF is based on a fixed axiomatisation, and only supports reasoning *within* a single relation algebra.

Math∫pad is a flexible quasi-WYSIWYG syntax-directed editing environment for mathematical documents that has been designed to support calculational proof presentation. It has been connected with the theorem prover PVS to enable checking of relation-algebraic proofs contained in Math∫pad documents [VB99]. Although the infrastructure is general, the system has been used only *within* concrete relation algebra; there appears to be no provision so far for working in weaker theories, nor for reasoning *about* multiple relation-algebraic structures in the same context.

An interesting related experiment is "PCP: Point and Click Proofs", a proof assistant based on a small JavaScript rewriting engine that allows users to interactively construct proofs of properties in a wide range of mathematical structures, characterised by equational and quasi-equational theories [Jip03, Jip01]. Currently, (homogeneous) relation algebra and Kleene algebra are already supported by the system, which is extensible and still under development. It appears

not to be geared to the addition of a type system as required for heterogeneous relation algebras, and is also limited to reasoning *within* structures.

A previous formalisation of heterogeneous relation algebras in Isabelle, RALL [vOG97], uses the Isabelle/HOL type system to support reasoning in abstract *heterogeneous* relation algebras with minimal effort, but at the cost of limiting itself to reasoning *within* a *single* relation algebra, as well. Besides interactive tactical proving, with special tactics allowing to approach the calculational style, RALL also explores automatic proving using the isomorphism of *atomisation* from relation-algebraic formulae into predicate logic, which allows Isabelle's classical reasoner to tackle the atomised versions of relation-algebraic formulae. This approach is not available for weaker structures like allegories or Dedekind categories, which were also not considered in RALL.

Struth realised a formalisation in Isabelle-1999 of untyped Kleene algebras via a hierarchy of axiomatic type classes, and used this to fully formalise Church-Rosser proofs in Kleene algebras [Str02]. Although Struth did all his reasoning within a single structure, axiomatic type classes [Wen97] do support reasoning "between" several structures; but they impose severe limitations to reasoning *about* structures.

It turns out that our objectives, namely *calculational reasoning* both *within* and *about* models of *different theories* with a relation-algebraic flavour are quite nicely supported by recent developments in the theorem prover Isabelle [NPW02]:

For reasoning *within* abstract algebraic structures in essentially the same way as it is done in pencil-and-paper mathematics, the concept of *locales* has been introduced into Isabelle [KWP99]. If such locales are based on records, this also allows reasoning *about* algebraic structures in the sense that several instances of the same structure can be used simultaneously in a single context.

While internally Isabelle still is a tactical theorem prover, the addition of the interpreted "Isar" language for "Intelligible semi-automated reasoning" allows proofs to be structured in the same way as in traditional mathematical proof presentation [Nip03, Wen02].

One additional aspect of Isar is its support for calculational reasoning [BW01]; this has been designed to also support user-defined transitive relations, such as the inclusion ordering of relations.

These features together allow us to provide a collection of theories extending from categories via different kinds of allegories and Kleene algebras up to heterogeneous relation algebras, geared towards calculational reasoning *in and about* relational algebras in a way that allows easy connection with concrete application theories.

Filling in these theories to a degree that they will become a useful starting point for applications is still ongoing work. From the examples we show in the next section, the reader may obtain a first flavour of working with the current state of our theory collection, which is not yet optimally tuned towards the automatic proving support provided by Isabelle. Once this is achieved, we will turn to extending the capabilities of Isabelle/Isar for the purpose of further

streamlining of calculational proof support for relational reasoning, for example to eliminate certain repetitive patterns, but also to include special-purpose decision procedures. The main purpose of this paper is to make a case that user-friendly mechanised support for fully formal calculational reasoning in relational algebras appears to become realistically achievable in the near future.

After the examples in the next section, we proceed to give an overview over the organisation of our theories in Sect. 3, and in Sect. 4 we discuss some technical details underlying the possibility of reasoning about and between structures.

2 Example Proofs

Essentially following [FS90], we define allegories as a special kind of categories. However, while Freyd and Scedrov treat categories and allegories as one-sorted algebraic structures with morphisms only, we follow the more usual approach of providing a separate sort for objects which serve as *source* and *target* of morphisms.

The inclusion ordering $\sqsubseteq$ in allegories is defined from the meet (or intersection) operation $\sqcap$:

$$\textit{incl-def}\,[\textit{iff}]\colon [\![\ R : a \leftrightarrow b;\ S : a \leftrightarrow b\ ]\!] \Longrightarrow (R \sqsubseteq S) = (S \sqcap R = R)$$

In the higher-order logic HOL, terms of type *bool* are used *en lieu* of propositions, so equality also takes on the role of propositional equivalence. The presence of the assumptions $[\![\ R : a \leftrightarrow b;\ S : a \leftrightarrow b\ ]\!]$ implies that the equivalence $(R \sqsubseteq S) = (R \sqcap S = R)$ only needs to hold if R and S belong to the same *homset* $a \leftrightarrow b$, i.e., if R and S are well-defined morphisms between two well-defined objects a and b.

As a first example proof, we show $R \sqcap S \sqsubseteq R$ via the above definition of inclusion from the algebraic properties of $\sqcap$:

$$\begin{aligned} R \sqcap (R \sqcap S) &= (R \sqcap R) \sqcap S && \text{associativity} \\ &= R \sqcap S && \text{idempotence} \end{aligned}$$

The Isar proof consists of the same steps. Since in our design, the well-typedness for relation-algebraic expressions is not dealt with by HOL's type system, but rather by homset membership, we also need to aid the system to explicitly discharge all well-typedness conditions — the automatic reasoning tactic *auto* is sufficient in most cases, aided by introducing the typings as introduction rules with [*intro*].

```
lemma (in Allegory) meet-decr1:
  assumes R-t[intro]: R : a ↔ b
  assumes S-t[intro]: S : a ↔ b
  shows R ⊓ S ⊑ R
proof −
 have R ⊓ (R ⊓ S) = (R  ⊓ R) ⊓ S      by (rule meet-assoc [THEN sym], auto)
 also have ... = R ⊓ S                by (subst meet-idem, auto)
 finally show ?thesis                 by (rule-tac incl-contract, auto)
qed
```

Obviously, we were able to transfer our proof into Isar without significant loss of readability or conciseness.

As a slightly more complicated example, we show that for univalent relations F, the equality $R \sqcap S;F = (R;F^\smile \sqcap S);F$ holds — this can be shown by a cyclic inclusion chain:

$$\begin{array}{lll} R \sqcap S;F & \sqsubseteq (R;F^\smile \sqcap S);F & \text{modal rule} \\ & \sqsubseteq R;F^\smile;F \sqcap S;F & \text{meet-subdistributivity} \\ & \sqsubseteq R \sqcap S;F & F \text{ univalent} \end{array}$$

Here, quite a few implicit steps were hidden, and some of these technical details need to be made explicit in the Isar proof. Some remain hidden; for example, modal rules come in four shapes (equivalent by conversion and commutativity of meet), so the Isabelle theory for allegories binds the theorem reference *modal* to the set containing all four shapes of modal rules. Also, since longer inclusion chains do not uniquely determine to which inclusions the antisymmetry rule should be applied, we need to split the calculation into two separate inclusion chains:

```
lemma (in Allegory) unival-meet-escape-1:
  assumes F-u: univalent F
  assumes F-t[intro]: F : b ↔ c
  assumes R-t[intro]: R : a ↔ c
  assumes S-t[intro]: S : a ↔ b
  shows     R ⊓ S ; F = (R ; F⌣ ⊓ S) ; F
proof −
 have R ⊓ S ; F ⊑ (R ; F⌣ ⊓ S) ; F                    by (rule modal, auto)
 moreover have (R ; F⌣ ⊓ S) ; F ⊑ R ⊓ S ; F
  proof −
  have (R ; F⌣ ⊓ S) ; F ⊑ (R ; F⌣) ; F ⊓ S ; F       by (rule meet-cmp, auto)
  also have (R ; F⌣) ; F = R ; (F⌣ ; F)             by (rule cmp-assoc, auto)
  also from F-u have F⌣ ; F ⊑ Id c                   by (rule univalent, auto)
  also (incl-mon-trans) have R ; Id c = R           by (rule right-id, auto)
  also show ?thesis                                 by (rule calculation, best+)
  qed
 ultimately show ?thesis                            by (rule incl-antisym, best+)
qed
```

For this proof, a few additional points need explanation. While in most cases, *auto* can discharge all well-typedness conditions, here there are two cases where *auto* does not succeed, and we used the best-fit-first reasoning tactic *best* instead. Since several conditions had to be discharged, application of *best* had to be iterated — *auto* attempts to discharge all open subgoals, so never needs iteration. One of the "**also**" occurrences has an explicitly specified transitivity rule *incl-mon-trans* — this is necessary for cases where the transitivity rule cannot be uniquely determined by Isabelle.

Furthermore, the conclusion of the second inclusion chain looks strange: instead of "**finally show** *?thesis*", which is an abbreviation for "**also from** *calculation* **show** *?thesis*" (where *calculation* is the name of the local register that

Isar uses to accumulate the result of calculational proofs), we now have "**also show** *?thesis* **by** ... *calculation*" — the reason for this is that *calculation* as produced by the previous chain is not a simple fact, but a meta-logic quantification carrying many (well-typedness) assumptions, so that it is not easily accessible to the automatic proof tools and instead needs to be explicitly applied to *?thesis*, the statement to be shown, as a rule.

The typing assumptions have, as always, again been flagged with [*intro*] as introduction rules, so *auto* will use them to discharge the type correctness conditions that all the other rule applications introduce. The assumption about univalence has not been flagged in this way — if it had been, we would not have needed to refer to it via "from *F-u*" for making use of univalence of F. The decision not to make this assumption available to Isabelle's automated proof tools therefore contributes to traceability.

Finally, experienced Isar users will note that instead of showing the theorem in the last line of the proof by "*rule incl-antisym*", we might have declared this as the rule organising the whole proof by inserting it after the opening "proof" in the place where "−" explicitly excludes any proof structuring rule. However, *incl-antisym* has not only the two inclusions as premises, but also two well-typedness conditions, which then would have to be dealt with explicitly, while with our approach they are implicitly discharged by the last "*best+*".

Although this second proof employed a larger number of the features of Isabelle/Isar, calculational proofs typically need neither the full complexities of the Isar proof language, nor all the capabilities of the Isabelle tactics language used in the arguments to "by".

The features that are used are limited to a set that can be learned relatively easily, and the way in which these features are used mostly follows rather systematic patterns. Even for some unexpected failures there are recipes, like "if *auto* fails to discharge type correctness conditions, then try *best+* and *(simp_all (no_asm_simp))*".

Even though more understanding of Isabelle and Isar is definitely helpful, and for many basic and auxiliary pre-proven lemmas it is necessary to remember their names for being able to invoke them explicitly, we expect the learning curve for doing calculational Isar proofs based on our theory library to be much smoother than for Isar proofs in predicate logic.

3 Theory Organisation

The kernel of our theory collection (which is still work in progress) is a hierarchy of theories defining the structures of discourse and providing useful facts and derived concepts:

– **Categories:** built on homsets, identities, and composition. Provides concepts like mono-, epi-, and isomorphisms, initial and terminal objects, direct sums and products. Functors will also be included here.

- **Ordered categories:** categories with an ordering $\sqsubseteq$ on every homset. Provides monotonicity and transitivity rules for reasoning with inclusions. Predicates for bounds and residuals, where existing, are also already defined here.
- **Allegories:** extend ordered categories by meets and converse. Provides concepts like univalent, total, ..., symmetric, transitive, ... Relators come in at this level.
- **Distributive allegories:** extend allegories by joins $\sqcup$ and least relations $\perp\!\!\!\perp$.
- **Division allegories:** extend distributive allegories by left and right residuals, and therefore also by symmetric quotients.
- **Dedekind categories**, or **locally complete distributive allegories** allow arbitrary joins, and therefore also greatest relations $\top\!\!\!\top$.
- **Relation algebras** have homsets that are Boolean algebras and therefore also provide complements.
- **Atomic relation algebras satisfying the Tarski rule** are the setting used in a large part of the literature, for example in [SS93].
- **Concrete relation algebras with subobjects, quotients, finite sums and finite products** are the setting of the relational programs of RelView [BBS97, BHL99]. This theory will allow verification of RelView programs in a recognised theorem prover.

Each of these classes of structures is defined in Isabelle/Isar/HOL as a *locale*, which corresponds to the mathematical habit of "assuming an arbitrary, but fixed category/allegory/... throughout".

Besides this "relational hierarchy", we also add support for Kleene algebras and related structures, since reasoning in these structures shares many aspects with relational reasoning. Although Kleene algebra is usually presented as a homogeneous algebra, we axiomatise heterogeneous versions on top of ordered categories since this way their relations with our hierarchy of categories and allegories are more natural.

- Heterogeneous **Kleene algebras**, according to Kozen's axiomatisation [Koz91], share the join structure with distributive allegories, and add the Kleene star, which corresponds to the reflexive transitive closure present in the relational hierarchy in Dedekind categories.
- **Residuated Kleene algebras** add residuals to Kleene algebras, these are shared with division allegories.
- **Action lattices** [Koz94] essentially are residuated Kleene algebras where homsets are lattices.
- **Kleene algebras with domain** (KAD) [DMS03] add to Kleene algebras a domain operator corresponding to that definable in allegories by $\mathsf{dom} R := \mathbb{I} \sqcap R ; R^{\smile}$.

A simplified overview of our theory dependency graph is shown in Fig. 1.

In order to be able to profit from the shared structures, we provide them separately in small theories. For example, the shared join properties are collected in our "heterogeneous idempotent semirings" (HISR), defined on top of locally ordered categories with join.

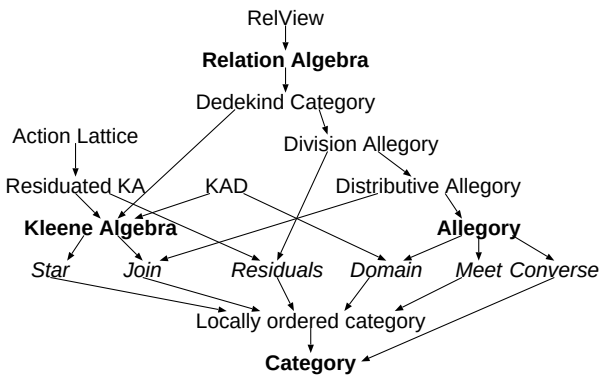

Fig. 1. Theory dependency summary

For the domain elements of Kleene algebras with domains, we do not follow the approach of embedding a Boolean algebra of tests into the subidenties. Instead we observe that the key properties of multiplicatively-idempotent subidentities presented in [DMS03] do not depend on the semiring structure, but already hold in ordered categories. We included a separete theory to collect those properties and use them to axiomatise the domain operator; all the resulting material is then made available to both allegories and Kleene algebras with domain. (See [DG00] for argumentation why the domain operator should be primitive in allegories.)

4 Structure Representation Aspects

For treating functors and relators, we obviously need to be able to deal with at least two categories/allegories in the same context. For structures encapsulated in Isabelle's locales, this implies that each of these locales has to be based on a record type. Such a record is essentially an explicit data structure aggregating all the information defining an algebra; for allegories and relation algebras, a similar organisation can be found in the Haskell library RATH [KS00]. In comparison with Haskell, the extensible records allowed by Isabelle simplify the setup; only the linear hierarchy that is enforced on extensions appears slightly unintuitive in some contexts, but does not lead to theoretical or practical problems. For example, within Kleene algebras, there is always a meet component accessible, but nothing can be proved about it. By organising record extensions and axiomatic theory definitions (i.e., locales) into separate files we achieved a setup which guarantees that reasoning in, for example, Kleene algebras is not burdened in any way by the presence of meets in the underlying records.

For being able to access derived concepts simultaneously in two structures (as, for example, in the statement "relators preserve univalence"), a particular way of defining them with explicit reference to the underlying structure is

required; this structure is of type $('o, 'm, 'r)$ *Allegory-scheme*, the type of all extensions of allegory structure records — these are currently the smallest records containing converse and inclusion. In addition, such derived concepts should, for the sake of demonstrable consistency, be introduced via meta-level equations — for technical reasons, these currently cannot employ user-defined syntax. Therefore, definitions of derived concepts cannot be stated in a simple intuitive shape. For example, ideally we would be able to define univalence by stating something like the following:

$$R : a \leftrightarrow b \Longrightarrow \mathit{univalent}\ R == (R^{\smile} ; R \sqsubseteq \mathit{Id}\ b)$$

Instead, we have to define an internal identifier *isUnivalent* and declare *univalent* with index "1" for reference to possibly different structures as user-level syntax for this. For the definition, we have to refer to all allegory primitives via the internal identifiers, such as *incl* for $\sqsubseteq$, and have to explicitly supply the structure argument that corresponds to the index position. We also have to impose the well-typedness conditions via a conditional, with the other branch containing the pseudo-value *arbitrary* about which nothing can be proven. For user-friendly access to such a definition, an additional lemma is necessary that contains the well-typedness condition as an assumption:

```
constdefs
  isUnivalent :: ('o, 'm, 'r) Allegory-scheme ⇒ 'm ⇒ bool (univalent1 - [1000] 999)
  isUnivalent s R == if isMor s R
                     then incl s (cmp s (conv s R) R) (CId s (Ctrg s R))
                     else arbitrary

lemma (in ConvOrdCat) univalent-def:
  R : a ↔ b ⟹ univalent R = (R˘ ; R ⊑ Id b)
by (unfold isUnivalent-def, auto)
```

All subsequent reasoning will then use that lemma — its name, *univalent-def*, has been chosen to essentially hide the real definition. For derived predicates, such as univalence, this "definition-lemma" might be almost sufficient; typically, more user-friendly lemmas are also provided, like the one used in the second example proof:

```
lemma (in ConvOrdCat) univalent[elim?]:
  assumes univalent R
  assumes [simp]: R : a ↔ b
  shows  R˘ ; R ⊑ Id b
by (rule univalent-def [THEN iffD1])
```

For derived operations, one also needs to state and prove the derived rules that help to automatically discharge well-typedness conditions.

This way of defining derived concepts applies in the same way to user-defined extensions to our theory as to our theory library itself. Although this is of course somewhat inconvenient, there is a systematic procedure to produce the necessary material for user-defined extensions. Adhering to that procedure guarantees that

new derived concepts can be used with the same flexibility as those contained in the library, and with the same safety with respect to consistency.

5 Conclusion and Outlook

We have shown that with new features of Isabelle, calculational reasoning in relation algebras can be supported in a way that makes proofs both readable and writable — indeed, in many cases producing a correct proof with the support of the XEmacs interface ProofGeneral [Asp00] is probably slightly easier than producing a good-looking proof directly in LaTeX.

In particular, we have been able to add the flexibility of dealing with several relation algebras at the same time, and of instantiating the abstract arguments in concrete structures, without incurring a prohibitive cost in terms of dealing with well-typedness conditions — the careful arrangement of homset rules allows us to discharge those systematically and automatically.

For the future, we plan to tackle correctness proofs of RelView programs as realistic case studies, and hope to be able to enlist more automated support from Isabelle for "trivial" calculational proof steps.

Another natural application of our framework and its many small component theories will be a study of the recently proposed "Kleene algebras with relations" [Des03].

While the proofs we have shown are by themselves not interesting at all, they demonstrate the style of prover-supported calculational reasoning that is possible already now. As an important step towards more user-friendly support for calculational reasoning we plan to incorporate decision procedures, where feasible, into the corresponding theories, for example for the equational theory of allegories [DG00], or for Kleene algebras. Once such decision procedures are available they can eliminate many tedious steps that are currently still necessary.

Acknowledgements

I am grateful to Millie Rhoss de Guzman and Hitoshi Furusawa for their comments on previous versions of this paper and many related discussions, and also to the anonymous reviewers for their valuable comments and to Kevin Everets for careful proofreading.

References

[Asp00] David Aspinall. Proof General: A generic tool for proof development. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1785 of *LNCS*, pages 38-42. Springer, 2000. See also `http://www.proofgeneral.org`. 187

[BBS97] Ralf Behnke, Rudolf Berghammer, and Peter Schneider. Machine support of relational computations: The Kiel RELVIEW system. Technical Report 9711, Institut für Informatik und Praktische Mathematik, Christian-Albrechts- Universität Kiel, June 1997. 184

[BH94] Rudolf Berghammer and Claudia Hattensperger. Computer-aided manipulation of relational expressions and formulae using RALF. In Bettina Buth and Rudolf Berghammer, editors, *Systems for Computer-Aided Specification, Development and Verification*, Bericht Nr. 9416, pages 62-78. Universität Kiel, 1994. 179

[BHL99] Rudolf Berghammer, Thorsten Hoffmann, and Barbara Leoniuk. Rechnergestützte Erstellung von Prototypen für Programme auf relationalen Strukturen. Technical Report 9905, Institut für Informatik und praktische Mathematik, Christian-Albrechts-Universität Kiel, July 1999. 184

[BW01] Gertrud Bauer and Markus Wenzel. Calculational reasoning revisited, an Isabelle/Isar experience. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher-Order Logics: TPHOLs 2001*, volume 2152 of *LNCS*, pages 75-90. Springer, 2001. 180

[Des03] Jules Desharnais. Kleene algebras with relations. In Rudolf Berghammer and Bernhard Möller, editors, *Proc. RelMiCS 7, International Seminar on Relational Methods in Computer Science, in combination with the 2nd Intl. Workshop on Applications of Kleene Algebra*, LNCS. Springer, 2003. (Invited Talk). 187

[DG00] Dan Dougherty and Claudio Gutiérrez. Normal forms and reduction for theories of binary relations. In Leo Bachmair, editor, *Rewriting Techniques and Applications, Proc. RTA 2000*, volume 1833 of LNCS, pages 95-109. Springer, 2000. 185, 187

[DM01] Jules Desharnais and Bernhard Möller. Characterizing determinacy in Kleene algebras. *Information Sciences*, 139:253-273, 2001. 178

[DMS03] Jules Desharnais, Bernhard Möller, and Georg Struth. Kleene algebra with domain. Technical Report 2003-7, Universität Augsburg, Institut für Informatik, 2003. 178, 184, 185

[dS02] H.C.M. de Swart, editor. *Proc. RelMiCS 6, International Workshop on Relational Methods in Computer Science, Oisterwijk near Tilburg, Netherlands, 16-21 October 2001*, volume 2561 of LNCS. Springer, 2002.

[FS90] Peter J. Freyd and Andre Scedrov. *Categories, Allegories*, volume 39 of *North-Holland Mathematical Library*. North-Holland, Amsterdam, 1990. 178, 181

[Fur98] Hitoshi Furusawa. *Algebraic Formalisations of Fuzzy Relations and Their Representation Theorems*. PhD thesis, Department of Informatics, Kyushu University, March 1998. 178

[HBS94] Claudia Hattensperger, Rudolf Berghammer, and Gunther Schmidt. RALF – A relation-algebraic formula manipulation system and proof checker. Notes to a system demonstration. InMaurice Nivat, Charles Rattray, Teodore Rus, and Giuseppe Scollo, editors, AMAST '93, Workshops in Computing, pages 405-406. Springer, 1994. 179

[Jip01] Peter Jipsen. Implementing quasi-equational logic on the web. Talk given at the AMS Sectional Meeting, University of South Carolina, March 16-18 2001. `http://www.chapman.edu/~jipsen/PCP/usctalk.html`. 179

[Jip03] Peter Jipsen. PCP: Point and click proofs. Web-based system at URL: `http://www.chapman.edu/~jipsen/PCP/PCPhome.html, 2003`. 179

[Kah01] Wolfram Kahl. A relation-algebraic approach to graph structure transformation, 2001. Habil. Thesis, Fakultät für Informatik, Univ. der Bundeswehr München, Techn. Bericht 2002-03. 178

[Kah02] Wolfram Kahl. A relation-algebraic approach to graph structure transformation. In de Swart [dS02], pages 1-14. (Invited Talk). 178

[KH98] Wolfram Kahl and Claudia Hattensperger. Second-order syntax in HOPS and in RALF. In Bettina Buth, Rudolf Berghammer, and Jan Peleska, editors, *Tools for System Development and Verification*, volume 1 of *BISS Monographs*, pages 140-164, Aachen, 1998. Shaker Verlag. ISBN: 3-8265-3806-4. 179

[Koz91] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inform. and Comput.*, 110(2):366-390, 1991. 184

[Koz94] Dexter Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78-88. MIT Press, 1994. 184

[Koz97] Dexter Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, pages 427-443, May 1997. 178

[Koz98] Dexter Kozen. Typed Kleene algebra. Technical Report 98-1669, Computer Science Department, Cornell University, March 1998. 179

[KS00] Wolfram Kahl and Gunther Schmidt. Exploring (finite) Relation Algebras using Tools written in Haskell. Technical Report 2000-02, Fakultät für Informatik, Universität der Bundeswehr München, October 2000. see also the RATH page `http://ist.unibw-muenchen.de/relmics/tools/RATH/`. 179, 185

[KWP99] Florian Kammüller, Markus Wenzel, and Lawrence C. Paulson. Locales - a sectioning concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher-Order Logics, 12th International Conference, TPHOLs'99*, volume 1690 of *LNCS*, pages 149-166. Springer, 1999. 180

[Nip03] Tobias Nipkow. Structured proofs in Isar/HOL. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs, International Workshop TYPES 2002*, LNCS. Springer, 2003. 180

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of LNCS. Springer, 2002. 180

[SHW97] Gunther Schmidt, Claudia Hattensperger, and Michael Winter. Heterogeneous relation algebra. In Chris Brink, Wolfram Kahl, and Gunther Schmidt, editors, *Relational Methods in Computer Science*, Advances in Computing Science, chapter 3, pages 39-53. Springer, Wien, New York, 1997. 179

[SS93] Gunther Schmidt and Thomas Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists*. EATCS-Monographs on Theoretical Computer Science. Springer, 1993. 179, 184

[Str02] Georg Struth. Calculating Church-Rosser proofs in Kleene algebra. In de Swart [dS02], pages 276-290. 180

[VB99] Richard Verhoeven and Roland Backhouse. Towards tool support for program verification *and* construction. In Jeanette Wing, Jim Woodcock, and Jim Davies, editors, *FM '99 - Formal Methods*, volume 1709 of LNCS, pages 1128-1146. Springer, September 1999. 179

[vOG97] David von Oheimb and Thomas F. Gritzner. RALL: Machine-supported proofs for relation algebra. In William McCune, editor, *Conference on Automated Deduction - CADE-14*, volume 1249 of LNCS, pages 380-394. Springer, 1997. 180

[Wen97] Markus Wenzel. Type classes and overloading in higher-order logic. In Elsa L. Gunter and Amy Felty, editors, *Theorem Proving in Higher-Order Logics, TPHOLs '97*, volume 1275 of *LNCS*, pages 307-322. Springer, 1997. 180

[Wen02] Markus M. Wenzel. *Isabelle/Isar - A Versatile Environment for Human-Readable Formal Proof Documents*. PhD thesis, Technische Universität München, Fakultät für Informatik, February 2002. 180