Towards Automation for Pointer Algebra*

Han-Hing Dang

Institut für Informatik, Universität Augsburg, D-86159 Augsburg, Germany
{h.dang}@informatik.uni-augsburg.de

Abstract. First-order automated verification within algebraic structures has proven to be widely applicable. In this work we deal with automation for pointer algebra. We evaluate the adequacy of its original axioms for theorem proving tools. Moreover we give a different algebraic approach for projections which turns out to be more suitable for automation.

1 Introduction

Automated theorem provers are powerful and used nowadays for a variety of verification tasks. Especially algebraic approaches have successfully been applied to automatic reasoning for different applications [5, 6, **?**,8]. This work concentrates on an algebraic calculus for the derivation of abstract properties for pointer structures, namely pointer algebra [3]. It has proven to be an appropriate abstraction for reasoning about pointer structures and deduction of pointer algorithms. Moreover the algebra provides a suitable calculus to derive reachability constraints and to describe allocation or projection. Projection is used to contrain reachability observations to pointer links. We focus on that operation and evaluate the efficiency of automated reasoning tools to derive properties of it.

2 Basics

This work discusses a basic model of the calculus presented in [2]. It is based on matrices and called *fuzzy model*. To give a concrete definition, first assume a finite set L of edge labels with standard operations $\cup, \cap$ and a finite set V of nodes. This set is now lifted for the model to $\mathcal{P}(L)^{V \times V}$, i.e., matrices over a set V of nodes with with sets of L-labels as entries. In particular the structure $(\mathcal{P}(L)^{V \times V}, \cup, ;, 0, 1, \top)$ forms a quantale with greatest element $\top$ where

$$\begin{aligned}
(M \cup N)(x,y) &=_{df} M(x,y) \cup N(x,y) \ , \\
(M;N)(x,y) &=_{df} \textstyle\bigcup_{z \in V} (M(x,z) \cap N(z,y)) \ , \\
\top(x,y) &=_{df} L \ , \\
0(x,y) &=_{df} \emptyset \ , \\
1(x,y) &=_{df} \begin{cases} L \text{ if } x = y \\ \emptyset \text{ otherwise} \end{cases} \ .
\end{aligned}$$

* Research was partially sponsored by DFG Project ALGSEP — Algebraic Calculi for Separation Logic

for arbitrary matrices $M, N \in \mathcal{P}(L)^{V \times V}$ and $x, y \in V$. As a concrete example consider a tree structure where $L =_{df} \{\mathsf{left}, \mathsf{right}, \mathsf{val}\}$. As usual the labels left and right represent links to the left and right son of a node resp. The destination of the label val can be interpreted as the value of a node. We will use "label" and "link" as synonym in the following. Figure 1 now gives a sample matrix and its corresponding labelled directed graph, i.e., a tree for $V =_{df} \{v_i, c_i : 1 \leq i \leq 3\}$.

	v_1	v_2	v_3	c_1	c_2	c_3
v_1	$\emptyset$	$\{\mathsf{left}\}$	$\{\mathsf{right}\}$	$\{\mathsf{val}\}$	$\emptyset$	$\emptyset$
v_2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathsf{val}\}$	$\emptyset$
v_3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{\mathsf{val}\}$
c_1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
c_2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
c_3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

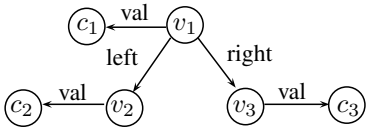

Fig. 1. Example matrix and the corresponding labelled graph

Next we give some further abstract definitions to present the main algebraic structure. We will calculate on quantales since the used matrices also form such a structure as mentioned above. Abstractly, we denote a quantale by $(S, +, \cdot, 0, 1, \top)$ with greatest element $\top$ and natural order $\leq$, defined by $x \leq y \Leftrightarrow_{df} x + y = y$. Furthermore, $(S, \leq)$ forms a complete lattice while $+$ coincides with binary suprema and binary infima are denoted by $\sqcap$. Clearly, $\sqcap$ in the matrix model is the pointwise $\cap$. Moreover we define a *test* as an element $p \leq 1$ that has a complement relative to 1, i.e., there exists an element $\neg p$ that satisfies $p + \neg p = 1$ and $p \cdot \neg p = 0 = \neg p \cdot p$. We will use the letters p, q for tests. Concretely a test in our model is a matrix that has non-empty entries at most on its diagonal.

To abstractly define further particular operations in our model we need special test elements that restrict the entries of a matrix to links of a set $L' \subseteq L$. Adequate elements are called *scalars* in the algebra and will be denoted by the letters $\alpha, \beta, \ldots$. A scalar α in the algebra is a test that additionally commutes with $\top$, i.e., $\alpha \cdot \top = \top \cdot \alpha$. We will use the notion $L(\alpha)$ to denote to the set L' that a scalar α comes with. A scalar α in the model is a matrix that has exactly the set $L(\alpha)$ on its diagonal for each element $x \in V$.

Finally we define left residuals by the Galois connection $x \leq y \backslash z \Leftrightarrow x \cdot y = z$ and symmetrically right residuals. By our completeness assumption residuals always exist. In particular for an arbitrary element x residuals w.r.t. a test p can be explicitly characterised by

$$p \backslash x = x + \neg p \cdot \top \quad \text{and symmetrically} \quad p / x = x + \top \cdot \neg p \ . \tag{1}$$

A proof can be found in [2]. Based on the defined operations and particular elements of the algebra we now give a definition for projections in the algebra. Projections are used to constrain the set of considered labels for reachability

oberservations. In the matrix model, projection w.r.t. to a scalar α is used to restrict each entry of a matrix exactly to $L(\alpha)$. This operation is defined in pointer algebra by residuals and cut operations, also known from Goguen categories [9].

The cut operators $\cdot^\uparrow$ and $\cdot^\downarrow$ are axiomatised for arbitrary x, y as follows

$$\begin{array}{ll} x^\uparrow \leq y \Leftrightarrow x \leq y^\downarrow \ , & (x \cdot y^\downarrow)^\uparrow = x^\uparrow \cdot y^\downarrow \ , \\ \alpha \text{ is a scalar and } \alpha \neq 0 \text{ implies } \alpha^\uparrow = 1 \ , & (x^\downarrow \cdot y)^\uparrow = x^\downarrow \cdot y^\uparrow \ . \end{array} \tag{2}$$

By this projections $P_\alpha(x)$ for arbitrary scalars α and elements x are defined by

$$P_\alpha(x) =_{df} \alpha \cdot (\alpha \backslash x)^\downarrow \ . \tag{3}$$

To explain this definition consider first the term $\alpha \backslash x$ in our concrete model. This can be rewritten using Equation (1) into $x + \neg\alpha \cdot \top$, i.e., residuals add to each edge of the matrix x all labels not in $L(\alpha)$. The cut operation is then used to keep only the edges of $\alpha \backslash x$ that are labelled with L while all other edges not completely labelled will be erased, i.e., are mapped to $\emptyset$. Finally $\alpha \cdot (\alpha \backslash x)^\downarrow$ reduces all L entries to $L(\alpha)$.

3 A First Attempt: Automating some Properties

Since projections are basically used to reduce the set of graph edges to relevant links between pointer structures, they immediately play a crucial role for reachability properties [2]. Reasoning about reachability can get a tedious task in larger graph structures without suitable machine support. Hence, our main interest was first to test to which extent the algebraic approach with its given axiomatisation is suitable for automation. We started our attempt by evaluating several properties of projections using the automated theorem prover[1] *Prover9* [7]. This ATP system was used since it performed best for automated reasoning tasks within basic algebraic structures [1].

For this we encoded about 30 basic theorems about projections and tried to prove them automatically without adding auxiliary theorems to the set of assumptions. We set a time limit of 300 seconds for each theorem which is in general enough for ATP systems to find a proof. Unfortunately the ATP system was in general only able to automatically derive simple theorems. Input-files will be made available online at [4]. In detail the system proved 12 basic properties in time; as e.g. $P_0(x) = 0$ and $P_1(x) \leq x$. Particularly, for the latter the system inferred $P_1(x) = x^\downarrow$ and consequently derived by the definition of cut operations $P_1(x)^\uparrow = P_1(x) \cdot 1^\uparrow = P_1(x)$. By this the claim was easily implied from $(x^\downarrow)^\uparrow \leq x$ which follows from instantiating the Galois connection in (2).

However, theorems as $P_1(x) \cdot P_1(y) \leq P_1(x \cdot y)$ and $P_\alpha(0) = 0$ could not be derived within 300 seconds. Although a hand-written derivation for the first law would only require similar results as in the automated proof for $P_1(x) \leq x$. Moreover guiding the proof search by adding auxiliary theorems as $x^\downarrow \leq x$, $(x^\downarrow)^\uparrow \leq x$ and isotony laws was not successful either.

[1] We abbreviate the term automated theorem prover to ATP in the following.

Therefore projections with the above axiomatisation seems to be difficult to handle for Prover9. This could be due to the axiomatisation of $\cdot^{\downarrow}$ through $\cdot^{\uparrow}$ by a Galois connection since the latter operation is not directly used by projections. Moreover using residuals without their explicit characterisation (1) seem to additionally exacerbate the proof search. However including that characterisation does not improve the results significantly. Therefore we suppose that also automating reachability properties that include Kleene star axioms and projections will perform likewise bad with ATP systems.

Hence, instead of tweaking the proof search, we follow a different idea to overcome difficulties for automation. The main idea is to find another characterisation of projections that has to be more suitable for automation tasks than the original one. We discuss such an approach in the next section.

4 A Different Approach for Projections

To improve the results for automatically deriving properties of projections we propose a different characterisation for projections. The central idea is to define a new operator so that projections can be expressed with fewer operations and axioms than in Equation (3). Notice that the original definition of projections uses residuation, the cut operation $\cdot^{\downarrow}$ and scalars.

To understand the new operator consider first the concrete matrix $\alpha\,;M$ for an arbitrary matrix M and scalar α in our model. The product $\alpha\,;M$ reduces the labels of each edge in M to at most $L(\alpha)$. In contrast, projection of a matrix M w.r.t. to a scalar α eliminates all edges in M that are not at least labelled by $L(\alpha)$ and reduces entries that contain $L(\alpha)$ to $L(\alpha)$. Hence arbitrary projections always satisfies $P_\alpha(M) \subseteq \alpha\,;M$.

Starting from $\alpha\,;M$, it is therefore only additionally needed to eliminate all edges with links $L' \subset L(\alpha)$. To get that elimination under control we propose a special binary operation $\odot$. Its general purpose is maintaining only labelsets which are present in both matrices it is applied to.

Concretely the operator $\odot$ is defined for arbitrary matrices M, N as follows

$$(M \odot N)(x,y) \;=_{df}\; \begin{cases} M(x,y) \text{ if } M(x,y) = N(x,y) \\ \emptyset \qquad\qquad \text{otherwise .} \end{cases}$$

This operation is now used in combination with the matrix $\alpha\,;\top$ to eliminate all entries of a matrix other than $L(\alpha)$. Notice that $(\alpha\,;\top)(x,y) = \alpha$ for all $x, y \in V$. By this we turn to a different abstract definition for projections

$$P_\alpha(M) \;=_{df}\; (\alpha \cdot M) \odot (\alpha \cdot \top)\ . \tag{4}$$

We continue to use the symbol $\odot$ also in the abstract setting. Finally to include this operation in our abstract treatment we need to characterise its behavior and interplay with existing semiring operations. We will use all following laws as axioms for Prover9 in the next section. It can be seen first that $\odot$ is associative, idempotent and commutative, i.e., for arbitrary x, y, z we have

$$x \odot x = x, \quad x \odot y = y \odot x, \quad x \odot (y \odot z) = (x \odot y) \odot z \tag{5}$$

In particular $\odot$ additionally satisfies

$$\begin{array}{cc} x \odot y \leq x \sqcap y \ , & (x \sqcap 1) \odot \top = x \odot 1 \ , \\ (x \odot \top) \cdot (y \odot \top) \leq (x \cdot y) \odot \top \ , & (x \odot \top) + (y \odot \top) \leq (x + y) \odot \top \ . \end{array} \tag{6}$$

Concretely $x \sqcap y$ is an upper bound for $x \odot y$. In particular if a matrix has only entries in its diagonal it suffices to consider a $\odot$-product with 1 instead of $\top$. Moreover, in general we only have semi-distributivity laws for $\odot$. These can be strenghtened to equations when considering the $\odot$-product with $\alpha \ ; \top$ instead of $\top$ for scalars α where $L(\alpha)$ is a singleton set. Additionally to achieve certain properties we need to state that $\alpha \neq 1 \ \Rightarrow \ \alpha \odot 1 = 0$ for scalars α. This law is dual to the implication for scalars in (2). To connect the abstract version of $\odot$ with the new definition of projections (4) we further assume for arbitrary tests p and matrices x

$$p \cdot ((x + \neg p \cdot \top) \odot \top) = (p \cdot x) \odot (p \cdot \top) \ . \tag{7}$$

This equation replays a behavior for projections as described for its original definition in Equation (3). In the next section we again evaluate the given characterisation for projections with Prover9.

5 Towards Positive Results

It can be seen that we still need to use scalars as in the original characterisation for projections. Moreover we have now more axioms for our new operations than for the cut operations and residuals. However, the results by this characterisation seem to be promising. Most of the theorems can now be proved immediately without adding any further auxiliary theorems. In detail, we were able to fully automatically derive 27 of 31 given theorems ranging from simpler to more difficult ones.

Consider e.g. problematic theorems of Section (3) as $P_1(x) \cdot P_1(y) \leq P_1(x \cdot y)$ and $P_\alpha(0) = 0$ which were difficult to derive using the original axioms. Both can now easily be shown by the new characterisation. The former simplifies to $(x \odot \top) \cdot (y \odot \top) \leq (x \cdot y) \odot \top$ which was assumed while the latter can be got from calculating $P_\alpha(0) = (\alpha \cdot 0) \odot (\alpha \cdot \top) = 0 \odot (\alpha \cdot \top) \leq 0 \sqcap (\alpha \cdot \top) = 0$. Since distributivity laws was problematic to automatically derive from the original axioms it was reasonable to include the laws in the new characterisation for Prover9.

Moreover the distributivity law $(x \odot \top) \cdot (y \odot \top) \leq (x \cdot y) \odot \top$ can be shown to be equivalent to $((\alpha \cdot x) \odot (\alpha \cdot \top)) \cdot ((\alpha \cdot y) \odot (\alpha \cdot \top)) \leq (\alpha \cdot x \cdot y) \odot (\alpha \cdot \top)$. Although the latter form also includes scalars it has improved our results slightly.

In general we conjecture that proofs become simpler using this characterisation since less operations are involved. For this we included a behavior of residuals (cf. (1)) with a new defined operation in our approach without any interplay with a second operator as in (2).

6 Conclusion and Outlook

Our presented characterisation for projections in pointer algebra seems to be more suitable for automated verification tasks. Although more axioms are given to the ATP system, proofs become easier and more manageable for automation. The evaluated results on a small subset of basic theorems seem to be very promising. In particular we were able to double the number of goals which could be automatically derived.

The presented approach still needs to be checked for completeness. Although most of the theorems that could be shown using the original axioms could be derived with the new approach it might still need to be adjusted.

Using this approach for projections we hope to be also able to improve automation for abstract reachability properties. Including various algorithms as case examples for the abstract approach to automatically derive correctness would be also interesting. This has to be investigated as future work.

Acknowledgements: I am very grateful to Peter Höfner and Bernhard Möller for valuable comments and remarks.

References

1. H.-H. Dang and P. Höfner. First-order theorem prover evaluation w.r.t. relation- and Kleene algebra. In R. Berghammer, B. Möller, and G. Struth, editors, *Relations and Kleene Algebra in Computer Science — PhD Programme at RelMiCS 10/AKA 05*, number 2008-04 in Technical Report, pages 48–52. Institut für Informatik, Universität Augsburg, 2008.
2. T. Ehm. *The Kleene Algebra of Nested Pointer Structures: Theory and Applications*. PhD thesis, Fakultät für Angewandte Informatik, Universität Augsburg, 2003.
3. T. Ehm. Pointer Kleene algebra. In R. Berghammer, B. Möller, and G. Struth, editors, *RelMiCS*, volume 3051 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 2004.
4. P. Höfner. ATPPortal. `http://ursaminor.informatik.uni-augsburg.de:8080/atpportal/`.
5. P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfennig, editor, *Automated Deduction — CADE-21*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 279–294. Springer, 2007.
6. P. Höfner and G. Struth. On automating the calculus of relations. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning (IJCAR 2008)*, volume 5159 of *Lecture Notes in Computer Science*, pages 50–66. Springer, 2008.
7. W. McCune. Prover9 and Mace4. <`http://www.cs.unm.edu/~mccune/prover9`>. (accessed May 24, 2011).
8. G. Struth. Reasoning automatically about termination and refinement. In S. Ranise, editor, *6th International Workshop on First-Order Theorem Proving*, volume Technical Report ULCS-07-018, Department of Computer Science, pages 36–51. University of Liverpool, 2007.
9. M. Winter. A new algebraic approach to L-fuzzy relations convenient to study crispness. *Information Sciences*, 139(3-4):233–252, 2001.