On the Algebraic Derivation of Garbage Collectors

Han-Hing Dang

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

Abstract. We give an algebraic characterisation of reachability and non-reachability in modal Kleene algebras. By this we derive a general algorithm for garbage collectors and present some further algebraic optimisations. The given approach is fully expressible in first-order logic and hence provides an abstract and general framework for automated and machine-verified derivations of garbage collecting algorithms.

1 Introduction

Many popular programming frameworks as JAVA or .NET nowadays come with an automated garbage collecting system to support memory management and facilitate programming. It is enormously relieving not to take care of each allocated address, since every resource has to be freed if it has no further usage.

The intention of the present work is to apply formal reasoning, in particular algebraic techniques, to derive garbage collecting systems. We propose an algebraic foundation that allows general and abstract derivations of correct garbage collecting algorithms. The used algebra provides due to its simple (in)equations an elegant way for proving recursive equations that specify such algorithms. The derivation process can also be supported and machine-guided by the use of first-order theorem proving systems.

2 Basics and Definitions

Generally, we start from an *idempotent semiring*, i.e., a structure $(S, +, \cdot, 0, 1)$, where $(S, +, 0)$ forms an idempotent commutative monoid and $(S, \cdot, 1)$ a plain monoid. The natural order $\leq$ on an idempotent semiring is defined by $x \leq y \Leftrightarrow_{df} x + y = y$. Additionally, multiplication is defined to distribute over $+$ while 0 is an annihilator for $\cdot$, i.e., for arbitrary x we have $x \cdot 0 = 0$ and $0 \cdot x = 0$.

Obviously, relations form such a structure and can be used to represent graph structures used in garbage collecting algorithms. There, the natural order coincides with the inclusion order $\subseteq$ while $+$ abstracts $\cup$ and $\cdot$ abstracts relational composition ; . The 0 element represents the empty set while 1 denotes the identity relation.

Reasoning about graphs often requires referring to sets of nodes. For this, we assume for semirings special subidentities $p \leq 1$, called *tests*. These elements are defined to have a complement relative to 1 which is denoted by $\neg p$. It satisfies $p + \neg p = 1$ and $p \cdot \neg p = 0 = \neg p \cdot p$. Thus tests have to form a Boolean subalgebra. By this, $+$ coincides with the binary supremum $\sqcup$ and $\cdot$ with the binary infimum $\sqcap$ on tests. The latter one translates to $\cap$ in relations. The greatest test is 1 while 0 is the smallest one.

Next, we define an *image* operator and denote it by $\_^\urcorner$. It that returns a subidentity that represents e.g. relationally the set of all direct successor nodes. Dually, one can define domain elements; but these are not of interest in this paper. For arbitrary element x and test p the *codomain* $\_^\urcorner$ is axiomatised as follows

$$x \leq x \cdot x^\urcorner \ , \qquad (x \cdot p)^\urcorner \leq p \ .$$

By this we define modal operators called *diamond* and *box* [1]

$$\langle x | p \ =_{df} \ (p \cdot x)^\urcorner, \qquad [x|p \ =_{df} \ \neg \langle x | \neg p = \neg(\neg p \cdot x)^\urcorner \ . \tag{1}$$

The diamond $\langle x|p$ calculates all existing direct successor nodes under an element x starting from p. Dually, $[x|p$ represents the set of all nodes that can be only directly reached by all nodes in p under x.

The diamond operation is isotone in both arguments while the box operation is only isotone in its second and antitone in its first argument. Since we are interested in reachability observation in graph structures we extend the algebraic structure to a *Kleene algebra* [4] by an iteration operator * axiomatised by

$$\begin{array}{ll} 1 + x \cdot x^* \leq x^* \ , & x \cdot y + z \leq y \ \Rightarrow \ x^* \cdot z \leq y \ , \\ 1 + x^* \cdot x \leq x^* \ , & y \cdot x + z \leq y \ \Rightarrow \ z \cdot x^* \leq y \ . \end{array}$$

This implies that a^* is the least fixed-point μ_f of the equation $f(x) = 1 + a \cdot x$.

3 Characterising Reachability and Non-Reachability

It has been shown in [2, 3] that the algebra defined in the previous section can be used to state and to derive general properties of reachability calculations.

The fundament of that is an algebraic mapping *reach* that calculates all nodes/objects somehow reachable from a given set of starting nodes/objects. For its definition assume a test p and a semiring element a. Then

$$reach(p, a) \ =_{df} \ \langle a^* | p \ . \tag{2}$$

By this *reach* is the smallest fixpoint μ_f of the equation $f(q) = p + \langle a|q$. Some more useful consequences are e.g.,

$$p \leq reach(p,a) \ , \qquad reach(p+q, a) = reach(p,a) + reach(q,a) \ .$$

Distributivity of *reach* in its first argument states that its calculation can be split, i.e., *reach* can be independently calculated on disjoint subsets of starting nodes. The overall result equals the union of all intermediate results. Moreover, *reach* is isotone in both arguments which will facilitate inequational reasoning.

It is not difficult to derive from the definition of *reach* the trivial results $reach(0,a) = 0$ and $reach(p,0) = p$. These equations can be used as termination conditions for a recursive calculation of *reach*. For the recursion steps, the following equation can be shown

$$reach(p,a) = p + reach(\langle a|p, a) \ .$$

From this, a recursive algorithm for *reach* is directly derivable:

$$\begin{aligned} reach(p,a) = \ & \mathsf{if}\ p = 0\ \mathsf{then}\ 0 \\ & \mathsf{else}\ p + reach(\langle a|p, a) \ . \end{aligned}$$

Unfortunately this recursion has a major problem. It will not terminate in general. For example with relations, assume a test $p = \{(1,1)\}$ and $a = \{(1,2),(2,1)\}$. It can be seen that a contains a reachable cycle. Hence, p will never become 0, i.e., never gets empty.

To overcome this deficiency, *reach* has to be modified so that there exists a decreasing value with each recursion step. We give a variant that decreases a in each step. Concretely, one can derive the recursion $reach(p,a) = p + reach(\langle a|p, \neg p \cdot a)$ using a so-called induction rule for *reach*

$$p \leq q \ \wedge \ \langle a|q \leq q \ \Rightarrow \ reach(p,a) \leq q \ . \tag{3}$$

The corrected solution overcomes the above problem since it deletes all currently visited nodes in $\neg p \cdot a$ with each recursion. Hence, if $p \neq 0$ then the calculation is continued on a reduced graph with less nodes and edges.

The presented results can now be further used to dually derive an algorithm that calculates all unreachable nodes w.r.t. a given set of starting nodes. We define

$$noreach(p,a) \ =_{df} \ \neg reach(p,a) \ . \tag{4}$$

One can immediately conclude by the definition of box in (1) that

$$noreach(p,a) = \neg reach(p,a) = \neg\langle a^*|p = [a^*|\neg p \ .$$

By the fixpoint theory of De Morgan duals, we immediately get another characterisation of *noreach* as $noreach(p,a) = \nu q.\ \neg p \cdot [a^*|q$. Obviously, we can derive dual properties for *noreach* like

$$noreach(p+q,a) = noreach(p,a) \cdot noreach(q,a) \tag{5}$$

or antitonicity of *noreach* in both arguments.

Note that $\cdot$ on tests coincides with binary infima. Hence, the distributivity property for *noreach* resolves, dually to *reach*, to the intersection of all intermediate calculations of *noreach* instead of their union. A similar induction rule for *noreach* reads

$$p \leq q \cdot [a|p \ \Rightarrow \ p \leq noreach(a,q) \ . \tag{6}$$

Moreover, one gets a similar recursive definition for *noreach* with

$$\begin{aligned} noreach(p,a) = \ & \mathsf{if}\ p = 0\ \mathsf{then}\ 1 \\ & \mathsf{else}\ \neg p \cdot noreach(\langle a|p, \neg p \cdot a) \ . \end{aligned}$$

4 Further Optimisations

We continue to refine and optimise the derived algorithm for *noreach* of the previous section. The presented modifications will end up in a simple imperative form of an algorithm for *noreach*.

First, by simple algebraic transformations one can show

$$noreach(\langle a|p, \neg p \cdot a) = noreach(\neg p \cdot \langle a|p, \neg p \cdot a) \ .$$

Hence, one can immediately derive a slightly modified version from the previous one that additionally deletes all redundant starting nodes for the next recursion step; they are also ruled out in its second argument $\neg p \cdot a$. Such nodes have already been visited and need not to be considered anew. They lie e.g. on cycles with at most two edges.

Next, using a so-called accumulator r as an additional argument, we can easily transform the algorithm into a tail recursion by

$$\begin{aligned} noreach(p,a) = {} & tnoreach(1,p,a) \ , \\ tnoreach(r,p,a) = {} & \mathsf{if}\ p = 0\ \mathsf{then}\ r \\ & \mathsf{else}\ tnoreach(\neg p \cdot r, \neg p \cdot \langle a|p, \neg p \cdot a) \ . \end{aligned} \tag{7}$$

Initially r contains all nodes and p is initialised with all root nodes. For the subsequent recursion all root nodes will be deleted from r while the second argument will be replaced with all direct successors of the roots apart from the roots themselves. The algorithm terminates if r contains all nodes unreachable from the root nodes.

Using the accumulator r, we have also introduced an argument that decreases with each recursion step. Hence, the idea arises to use r in a test for termination. For this, we need to assume an invariant for r that holds within each recursion

$$\neg p \cdot r \le [a|r \ . \tag{8}$$

This assumption is required since no interplay between p, a and r was characterised by the derivation in (7). To better understand this property we rewrite it into $\langle a|\neg r \le p + \neg r$. Since r denotes in each recursion step all currently unreachable nodes, $\neg r$ contains the already visited ones. Now, the inequation says that all a-successors of the visited nodes are either contained in p, the current set of starting nodes, or have been reached before.

This allows an algebraic derivation of the following recursion

$$\begin{aligned} tnoreach(r,p,a) = {} & \mathsf{if}\ p = 0\ \mathsf{then}\ r \\ & \mathsf{else}\ tnoreach(\neg p \cdot r, \neg p \cdot r \cdot \langle a|p, a) \ . \end{aligned} \tag{9}$$

A proof can be found in the appendix. One advantage of this recursion schema is that it can be immediately rewritten into imperative form (cf. Fig. 1). For better readability, we used $r - p =_{df} \neg p \cdot r$ and replaced $\neg p \cdot r$ in line 4 with the updated r of line 3 in Fig. 1.

Another argument for preferring version (9) is that e.g., relationally, the calculation of $\neg p \cdot a$ means deletion of all nodes of p in a and their incident edges. First, deleting reachable nodes in a is not desired and thus would require e.g., for algorithm (7) a copy of the whole graph a, which is very space inefficient. Moreover, the calculation of all ingoing edges to nodes in p can be very inefficient as one has to keep track of all such links or might have to traverse the whole graph. Therefore it is more intuitive to give a solution that does not modify a and still guarantees termination as in (9).

```
noreach
p := roots; r := 1;
while (p != 0) {
   r := r − p;
   p := r · ⟨a|p;
}
return  r;
```

Fig. 1. An algebraic and imperative version of *noreach*

Another interpretation of $\neg p \cdot a$ can be a marking of p-nodes as visited like in common *Mark and Sweep* collectors. But this would require an adequate adaptation of the multiplication operation $\cdot$ so that in $\langle a|p$ only all unmarked successors of unmarked nodes are considered.

5 Conclusion and Future Work

We have presented algebraic definitions of reachable and unreachable elements within the setting of a modal Kleene algebra. From this we derived recursive specifications to define an abstract algorithm for garbage collection. Furthermore, we algebraically proved some optimisations of the algorithm and presented a corresponding imperative version of it. The algebra-based approach is fully first-order and comes with simple (in-)equational laws. The main intention of this work is to provide a simple general framework that additionally allows the inclusion of automated first-order theorem proving systems to help in the guidance of deriving garbage collecting systems.

As future work we are planning to abstractly specify concurrent behaviour within our setting. Algorithm (9) can e.g., be seen as an algebraic variant of the optimized fixpoint iteration algorithm given in [5]. Now, an idea would be to consider executions of a system by sequences of elements, i.e., $a_1 a_2 \ldots a_n$. By this, it is possible to characterise e.g., with the presented approach that garbage only grows or dually that the set of reachable nodes is monotonically decreasing over time. This reflects the behaviour that whenever references to reachable nodes have been lost, only a run of the garbage collector can regain such nodes back. Thus, no program running in parallel is able to recover garbage by itself. Assuming r denotes the set of roots, that behaviour is simply given by

$$r \le p \;\Rightarrow\; \langle a_{i+1}|reach(p, a_i) \le reach(p, a_i)\ .$$

Acknowledgements: I thank Bernhard Möller for his valuable comments and for drawing my attention to this particular research topic. This work was partially funded by the DFG project *MO 690/9-1 AlgSep — Algebraic Calculi for Separation Logic.*

References

1. Desharnais, J., Möller, B., Struth, G.: Modal Kleene algebra and applications — A survey. Journal of Relational Methods in Computer Science 1, 93–131 (2004)
2. Ehm, T.: The Kleene algebra of nested pointer structures: Theory and applications, PhD Thesis (2003). `http://www.opus-bayern.de/uni-augsburg/frontdoor.php?source_opus=89`
3. Ehm, T.: Pointer Kleene algebra. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS. LNCS, vol. 3051, pp. 99–111. Springer (2004)
4. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Information and Computation 110(2), 366–390 (1994)
5. Pavlovic, D., Pepper, P., Smith, D.R.: Formal derivation of concurrent garbage collectors. In: 10th International Conference on Mathematics of Program Construction, MPC 2010, Québec City, Canada. pp. 353–376. No. 6120 in LNCS, Springer (2010)

6 Appendix

Correctness proof of algorithm (9) *:*
First, we state a result used in the following: $noreach(p, a) \leq \neg p$. We show that algorithm (7) and (9) are equal. The trivial case $p = 0$ is obvious. Moreover, using results for *reach* and antitonicity of *noreach*, it remains to show

$$\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq \neg p \cdot r \cdot noreach(\langle a|p, a) \ .$$

Using $1 = r + \neg r$ and property (5), the right hand-side resolves to $\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \cdot noreach(\neg r \cdot \langle a|p, a)$. Hence, the above inequation can be reduced to

$$\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq noreach(\neg r \cdot \langle a|p, a) \ .$$

Using the backwards box induction rule (6) the above inequation is implied by

$$\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq \neg(\neg r \cdot \langle a|p) \cdot [a|(\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a)) \ .$$

Next, using $noreach(p, a) \leq \neg p$, we conclude $\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq \neg p \cdot r \cdot \neg(r \cdot \langle a|p) \leq r \cdot \neg(r \cdot \langle a|p) \leq r \cdot \neg\langle a|p \leq \neg(\neg r \cdot \langle a|p)$. Thus, the inequation reduces further to $\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq [a|(\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a))$. Now, by multiplicativity of box in its second argument, i.e., $[a|(p \cdot q) = [a|p \cdot [a|q$ and $noreach(r \cdot \langle a|p, a) \leq [a|(noreach(r \cdot \langle a|p, a))$, it remains to show

$$\neg p \cdot r \cdot noreach(r \cdot \langle a|p, a) \leq [a|\neg p \cdot [a|r \ .$$

This is implied by $\neg p \cdot r \cdot \neg(r \cdot \langle a|p) \leq [a|\neg p \cdot [a|r$. Next, we calculate for the left-hand side $\neg p \cdot r \cdot \neg(r \cdot \langle a|p) = \neg p \cdot r \cdot \neg\langle a|p = \neg p \cdot r \cdot [a|\neg p$. The resulting inequation $\neg p \cdot r \cdot [a|\neg p \leq [a|\neg p \cdot [a|r$ is implied by the invariant (8), i.e., $\neg p \cdot r \leq [a|r$. $\square$