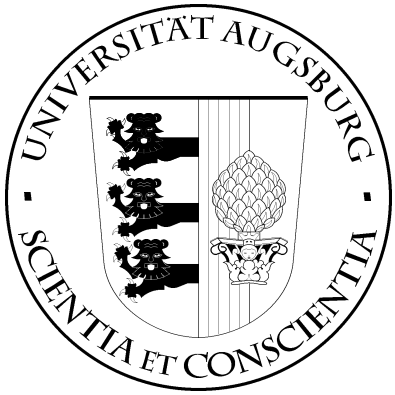
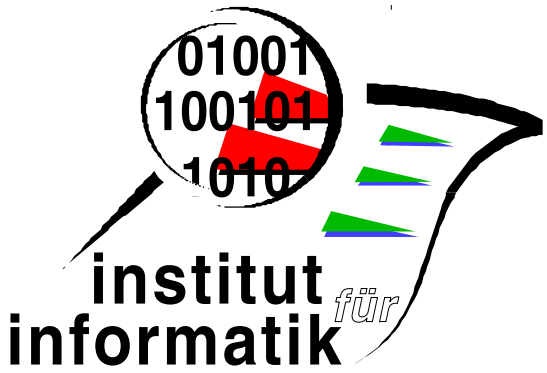
Universität Augsburg



Kleene Algebras and Pointer Structures

Thorsten Ehm

Report 2003-13 July 2003



Institut für Informatik
D-86135 Augsburg

Kleene Algebras and Pointer Structures

Thorsten Ehm

Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
Ehm@informatik.uni-augsburg.de

Abstract Kleene algebras (KA) have turned out to be an appropriate tool to formally describe algebraic systems in various areas. Despite this universal applicability there often proofs are easy and half as long as in concrete KAs. In this paper we describe how to use KAs to model edge-labeled directed graphs. As an application we show how the relational pointer algebra developed by B. Möller can be treated with this technique.
Keywords: Kleene algebra, pointer algebra, pointer structures

1 Introduction

Many areas that have to be treated formally demand a powerful but also concise calculus. As these two desires affect each other, we are forced to find a compromise between them. Kleene algebra has turned out to be an algebraic system that is simple in its treatment on the one hand and of high expressive power on the other.

There are some applications, for example in automata or graph theory, where one has to cope with several equally structured KAs. A standard method in this case is to switch to matrix algebra with Kleene algebraic entries. The problem is that the more abstract treatment avoids access to the matrix elements. Moreover such an algebra is not always closed under Kleene algebra operations. Kozen proposed as solution the definition of action lattices [25], which are action algebras [32] enhanced with an additional meet operator. These are closed under the formation of matrices, but in this case the difference to relational algebra is so small that the abstraction to KAs by only omitting the converse operation does not make much sense anymore.

The main goal of this paper is to give a technique how to handle several (equally shaped) Kleene algebras in one. To make the paper self-contained we give full proofs for all lemmas. The reader may wish to skip some of these. As an application we show how this framework can be used to model pointer structures and prove properties about them. More precisely, we have in mind a set of records that can be pointer-linked by various selectors. This is an abstract view of a labeled graph where the nodes represent records and selectors are modeled by labeled links. Möller [27] has shown how a relational version of such a calculus can be used to derive correct pointer algorithms from an abstract functional specification. Such a pointer algebra also can be used as a formal basis for the semantics to the mostly Hoare-logic or wp-calculus based methods for the verification of pointer algorithms [6,31,3,7,4,34].

This paper is structured as follows: Section 2 defines the notions of Kleene algebra, operators, enhancements and states some properties. In Section 3 it is shown how the toolkit can be used to handle several Kleene algebras simultaneously without loosing the ability to have access to the distinct elements inside. In the theoretical Sections 2 and 3 we mostly will use labeled graphs as a running example to motivate our calculations. As an application the abstract treatment of pointer structures and related operations in such a Kleene algebra is shown in Section 4.

2 Kleene Algebra

This section gives the definition of Kleene algebra, several operations and their properties as well as some extensions. An important part is the relation between scalars and ideals and the derivation of an extension that later is used to handle injection and projection of elements. As the basis for developing such an extended Kleene algebra we use the axiomatization of KA as given by Kozen [26]:

Definition 1 (Kleene algebra). *A* Kleene algebra $(\mathcal{K}, +, \cdot, 0, 1, ^*)$ *is an idempotent semiring with star:*

$$a + (b + c) = (a + b) + c \quad (1) \qquad a \cdot (b + c) = a \cdot b + a \cdot c \quad (8)$$
$$a + b = b + a \quad (2) \qquad (a + b) \cdot c = a \cdot c + b \cdot c \quad (9)$$
$$a + 0 = a \quad (3) \qquad 0 \cdot a = 0 \quad (10)$$
$$a + a = a \quad (4) \qquad a \cdot 0 = 0 \quad (11)$$
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \quad (5) \qquad 1 + a \cdot a^* = a^* \quad (12)$$
$$1 \cdot a = a \quad (6) \qquad 1 + a^* \cdot a = a^* \quad (13)$$
$$a \cdot 1 = a \quad (7) \qquad b + a \cdot c \leq c \to a^* \cdot b \leq c \quad (14)$$
$$b + c \cdot a \leq c \to b \cdot a^* \leq c \quad (15)$$

As usual, inequations are defined using the join operator:

$$a \leq b \overset{\text{def}}{\Leftrightarrow} a + b = b$$

A useful proof tool to work in a poset are the rules of indirect equality:

$$a = b \Leftrightarrow (\forall c.\ c \leq a \Leftrightarrow c \leq b) \Leftrightarrow (\forall c.\ a \leq c \Leftrightarrow b \leq c)$$

As an abbreviation we define $a^+ = a \cdot a^*$. We also mention some standard laws that hold for * and $^+$:

Lemma 1.

1. $1 \leq a^*$
2. $a \leq a^*$
3. $a^* \cdot a^* = a^*$
4. $(a^*)^* = a^*$
5. $(a + b)^* = a^* \cdot (b \cdot a^*)^*$
6. $a \cdot (b \cdot a)^* = (a \cdot b)^* \cdot a$
7. $a^+ \cdot a^* = a^+ = a^* \cdot a^+$

The proofs are trivial or can be found in [24].

Sometimes Lemma 1.5 does not give the required simplification of a starred join. Instead we need a sort of recursive rewrite rule:

Lemma 2 (Star decomposition).

$$(a + b)^* = (1 + (a + b)^* \cdot b) \cdot a^* = a^* \cdot (1 + b \cdot (a + b)^*)$$

Proof. We only prove the first equality. The second is shown symmetrically.

$$(a + b)^* = (a^* \cdot b)^* \cdot a^* = (1 + (a^* \cdot b)^* \cdot a^* \cdot b) \cdot a^* = (1 + (a + b)^* \cdot b) \cdot a^*$$

□

2.1 Predicates

Of special interest are elements that are less than or equal to the neutral element 1. They can be used to describe domain and range of elements in a straightforward way, serve as tests or model sets of states. In other disciplines these elements are also called partial identities, monotypes or coreflexives. We nevertheless will call them predicates, as this models quite well the semantics of such elements in the context of program verification and construction.

Definition 2 (predicate). *A* predicate *of a Kleene algebra is an element s with $s \leq 1$.*

In the sequel the set of all predicates is denoted by $\mathcal{P} = \{s : s \leq 1\}$ and elements of $\mathcal{P}$ by s and t. We will use a similar approach as Kozen in [26] but identify the Boolean sort with the set of *all* predicates.

Definition 3 (KAT). *A* Kleene algebra with tests (KAT) *is a KA where the set of predicates $\mathcal{P}$ forms a Boolean lattice $(\mathcal{P}, +, \cdot, \neg, 0, 1)$ with $\neg$ denoting the complement in $\mathcal{P}$.*

Predicates in KAT are idempotent with regard to composition, since $\cdot$ coincides with the meet operation. So some properties directly derived from lattice theory are only valid for elements of $\mathcal{P}$.

Corollary 1. *Consider a KAT and $s, t \in \mathcal{P}$.*

1. $s \cdot s = s$
2. *a)* $s + \neg s = 1$
   b) $s \cdot \neg s = 0$
3. *a)* $\neg(s + t) = \neg s \cdot \neg t$
   b) $\neg(s \cdot t) = \neg s + \neg t$

2.2 Residuals and top

Residuation goes back to de Morgan [9] who called the respective rule "Theorem K". In the meantime residuals are well understood. They also play an essential rôle in relation algebra [35] and form the basis of division allegories [16]. In earlier approaches [14] we used residuated KAs which are defined as KAs with existing left and right residuals:

$$b \leq a\backslash c \overset{\text{def}}{\Leftrightarrow} a \cdot b \leq c \overset{\text{def}}{\Leftrightarrow} a \leq c/b$$

The structure of a residuated Kleene algebra is equivalent to action algebras introduced by Pratt [32]. An advantage of residuals is that we get a top element for free, which is $\top = 0\backslash 0$. Note that $0\backslash 0$ is not the only representation of $\top$. In fact, all expressions of the form $0\backslash a$ are equal to $\top$. The distributivity laws (8) and (9) as well as the composition laws for 0 in Definition 1 follow directly from the existence of residuals. But since in the sequel we just need residuals with respect to predicates which can be defined using a top element and the existence of residuals is such a strong demand, we just enhance the algebra with a top element.

Definition 4 (KA with top). *A Kleene algebra $(\mathcal{K}, +, \cdot, 0, 1, ^*)$ with top is a KA with an element $\top$ defined by: $\forall a \in \mathcal{K}.\ a \leq \top$*

In residuated KAs the residuals of predicate s can be expressed by

$$s\backslash a = a + \neg s \cdot \top \tag{16}$$

$$a/s = a + \top \cdot \neg s \tag{17}$$

Proof. We only show the first equality. Assume a residuated KA, then

"$\geq$": $a + \neg s \cdot \top \leq s\backslash a \Leftrightarrow s \cdot (a + \neg s \cdot \top) \leq a \Leftrightarrow s \cdot a \leq a$

"$\leq$": $s\backslash a = s \cdot (s\backslash a) + \neg s \cdot (s\backslash a) \leq a + \neg s \cdot \top$ □

So we use $\backslash$ as well as $/$ in KAs with top as abbreviation defined by terms (16) and (17). We summarize the most important properties of residuals with respect to predicates:

Lemma 3. *1.* $0\backslash a = \top$
2. $1\backslash a = a$
3. $s\backslash \top = \top$
4. $1 \leq s\backslash s$
5. $s \cdot (s\backslash a) = s \cdot a$
6. $s\backslash(s \cdot a) = s\backslash a$
7. $(s\backslash a)/t = s\backslash(a/t)$

Proof. 1. $0\backslash a = a + \neg 0 \cdot \top = a + \top = \top$
2. $1\backslash a = a + \neg 1 \cdot \top = a$
3. $s\backslash \top = \top + \neg s \cdot \top = \top$
4. $s\backslash s = s + \neg s \cdot \top \geq s + \neg s = 1$
5. $s \cdot (s\backslash a) = s \cdot (a + \neg s \cdot \top) = s \cdot a$
6. $s\backslash(s \cdot a) = s \cdot a + \neg s \cdot \top = a + \neg s \cdot \top = s\backslash a$
7. $(s\backslash a)/t = (a + \neg s \cdot \top)/t = a + \neg s \cdot \top + \top \cdot \neg t = (a/t) + \neg s \cdot \top = s\backslash(a/t)$ □

The laws given for $\backslash$ hold symmetrically for $/$. Further the Galois connection holds restricted to predicates in a KA with top and residuation defined by (16) and (17):

Lemma 4.

$$s \cdot a \leq b \Leftrightarrow a \leq s\backslash b$$
$$a \cdot s \leq b \Leftrightarrow a \leq b/s$$

Proof. We only how the first claim. the second follows symmetrically. Assume $s \cdot a \leq b$, then $a = s \cdot a + \neg s \cdot a \leq b + \neg s \cdot \top = s\backslash b$. Now assume $a \leq s\backslash b$, then $s \cdot a \leq s \cdot (s\backslash b) \overset{3.5}{=} s \cdot b \leq b$ □

A nice but not very valuable thing is that with these definitions of restricted residuals the pure induction rule (see [32]) can be proved for predicates:

Lemma 5 (pure induction). $(s\backslash s)^* = (s\backslash s)$

Proof. $(s\backslash s)^* \leq (s\backslash s)$ follows from star induction (Definition 1.15) and

$$\begin{aligned}
&1 + (s\backslash s) \cdot (s\backslash s) \\
&= s + \neg s + (s + \neg s \cdot \top) \cdot (s + \neg s \cdot \top) \\
&= s + \neg s + s + \neg s \cdot \top \cdot s + \neg s \cdot \top \cdot \neg s \cdot \top \\
&\leq s + \neg s \cdot \top \\
&= (s\backslash s)
\end{aligned}$$

whereas the other inequality is trivial. □

2.3 Domain and Codomain

For an abstract definition of domain and codomain we use an equational axiomatization based on the one given in [11]. As we will see in Section 2.6 it suffices to manifest only the propositions of a predomain operator, since locality will follow from a later added axiom.

Definition 5 (domain). *The domain operation* $\ulcorner$ *is defined by:*

$$a \leq \ulcorner a \cdot a \tag{18}$$
$$\ulcorner(s \cdot a) \leq s \tag{19}$$

The codomain operator $\urcorner$ can be defined symmetrically. From these laws it follows that domain and codomain distribute over joins and therefore are monotonic. The here used axiomatization first was introduced in [11]. In [30] properties of domain and codomain operators are presented in standard Kleene algebra (see Appendix A for an axiomatization). Most of the rules also hold in our environment except the ones mentioning the meet operator.

Lemma 6.

1. $\ulcorner s \leq 1$
2. $\ulcorner s = s$
3. $\ulcorner s \cdot s = s$
4. $\ulcorner(s \cdot \top) = s$

Proof. 1. $\ulcorner s = \ulcorner(s \cdot 1) \overset{(19)}{\leq} s \leq 1$
2. $s \overset{(18)}{\leq} \ulcorner s \cdot s \leq \ulcorner s \cdot 1 = \ulcorner s$ and $\ulcorner s \leq s$ follows from the proof of 1.
3. Immediate by 2. and Boolean algebra
4. $\ulcorner(s \cdot \top) \overset{(19)}{\leq} s \overset{2.}{=} \ulcorner s \leq \ulcorner(s \cdot \top)$ □

In KAs with top we can also define domain by the Galois connection

$$\ulcorner a \leq s \Leftrightarrow a \leq s \cdot \top$$

which is the standard way in SKAs. Equality of these definitions is an easy proof shown in [11].

2.4 Ideals and Scalars

We will introduce the notions of ideals [20,21] and scalars [22]. These later are used to single out certain regions or parts of Kleene algebra elements. To be able to handle the elements of equally shaped Kleene algebras simultaneously the elements of each algebra are tagged by a scalar. Both, the set of ideals and the set of scalars are closed under application of the Kleene operations join and composition. The access to differently tagged algebras is based on a one-to-one correspondence between ideals and scalars, which does not hold in all KAs. We will first give the definitions of ideals and scalars and afterwards derive conditions under which a bijection can be established.

Definition 6 (ideal). *A right ideal is an element $j \in \mathcal{K}$ that satisfies $j = j \cdot \top$. Symmetrically we define the notion of left ideals. An ideal then is an element that is a left and a right ideal (that fulfills $\top \cdot j \cdot \top = j$).*

Intuitively an ideal corresponds to a completely connected graph where all arrows are identically labeled with the respective selector. As we will see later each of these graphs plays the rôle of a top element in the subalgebra for a fixed selector set. It is evident that every non-trivial Kleene algebra has at least two ideals, namely 0 and $\top$ the not at all and the completely connected graph. As an example, in abstract relation algebra by the Tarski rule ($a \neq 0 \Rightarrow \top \cdot a \cdot \top = \top$) these are the only ones. An algebra in which the Tarski rule holds is called *simple* in [36]. So in every simple Kleene algebra there are only these two ideals. From every Kleene element a we can get an ideal by composing it with the top element from both sides ($\top \cdot a \cdot \top$) and these are the only ideals. As mentioned above,

Lemma 7. *The set $\mathcal{J} = \{j \mid \top \cdot j \cdot \top = j\}$ of ideals is closed under join and composition.*

Proof. Let $j, k \in \mathcal{J}$, then $\top \cdot (j + k) \cdot \top = \top \cdot j \cdot \top + \top \cdot k \cdot \top = j + k$
$\top \cdot (j \cdot k) \cdot \top = (\top \cdot j) \cdot (k \cdot \top) = j \cdot k$

□

In the sequel we will denote elements of $\mathcal{J}$ by j and k. A rôle comparable to that of ideals in the whole algebra play scalars in the set of predicates. A scalar is a predicate that commutes with the top element.

Definition 7 (scalar). *An element $\alpha \in \mathcal{P}$ is called a* scalar *iff $\alpha \cdot \top = \top \cdot \alpha$.*

Scalars are similar to ideals except that they are not completely connected. There are only pointers from each node to itself via all selectors described by the scalar. In the sequel we will use the terminology selector interchangeably for scalars as this mimics the purpose of scalars as handles for selecting parts of a graph or pointer structure. The notion of a scalar comes from fuzzy relation theory. There it is used as discrimination level for an α-cut. The α-cut of a fuzzy set A is a crisp set A_α that contains all elements that have a membership grade greater or equal to α in A. In this context the notion of crispness describes that there is no uncertain information. This means the membership grades are either 0 or 1. We will see later how such an α-cut can be used to project out an α-labeled subgraph. There is no need to be familiar with fuzzy theory. The interested reader is referred to [23]. As above we can see, that in a non-trivial algebra there are at least the two scalars 0 and 1. The set $\mathcal{S}$ of scalars is closed under join, composition and complement:

Lemma 8. *The set $\mathcal{S} = \{\alpha \mid \alpha \leq 1 \wedge \alpha \cdot \top = \top \cdot \alpha\}$ of scalars forms a Boolean lattice*

Proof. Clearly all resulting elements are predicates and therefore it suffices to show commutativity with the top element:

1. $(\alpha + \beta) \cdot \top = \alpha \cdot \top + \beta \cdot \top = \top \cdot \alpha + \top \cdot \beta = \top \cdot (\alpha + \beta)$
2. $\alpha \cdot \beta \cdot \top = \alpha \cdot \top \cdot \beta = \top \cdot \alpha \cdot \beta$
3. We show $\leq$ ($\geq$ symmetrically): $\neg\alpha \cdot \top = \neg\alpha \cdot \top \cdot (\alpha + \neg\alpha)$
$= \neg\alpha \cdot \top \cdot \alpha + \neg\alpha \cdot \top \cdot \neg\alpha$
$= \neg\alpha \cdot \alpha \cdot \top + \neg\alpha \cdot \top \cdot \neg\alpha$
$\leq \top \cdot \neg\alpha$

□

In the sequel we will use Greek letters α, β, γ for scalars. Scalars not only commute with top but also show some other nice commutativity properties:

Lemma 9. *Let $\alpha \in \mathcal{S}$ be a scalar and $a \in \mathcal{K}$, then*

1. $\alpha \backslash a = a / \alpha$
2. $\alpha \cdot a = a \cdot \alpha$

Proof. 1. $\alpha \backslash a = a + \neg\alpha \cdot \top = a + \top \cdot \neg\alpha = a / \alpha$
2. By indirect equality and Lemma 4: $\alpha \cdot a \leq b \Leftrightarrow a \leq \alpha \backslash b \Leftrightarrow a \leq b / \alpha \Leftrightarrow a \cdot \alpha \leq b$

□

2.5 Establishing the bijection

As mentioned in the previous section, access to parts of the structure is based on a bijective correspondence between ideals and scalars. In all KATs with an additional domain operator it holds that there exists an injective mapping from scalars to ideals. The way back is a little more complicated.

Lemma 10. *Define* $i_{\mathcal{SJ}}(\alpha) \stackrel{\text{def}}{=} \alpha \cdot \top$. *Then for a scalar* α *the element* $j = i_{\mathcal{SJ}}(\alpha)$ *is an ideal and* $i_{\mathcal{SJ}} : \mathcal{S} \to \mathcal{J}$ *is injective.*

Proof. $\top \cdot j \cdot \top = \top \cdot (\alpha \cdot \top) \cdot \top = \alpha \cdot \top \cdot \top \cdot \top = \alpha \cdot \top = j$, so j is an ideal.
Let now be $\alpha, \beta \in \mathcal{S}$ and $i_{\mathcal{SJ}}(\alpha) = i_{\mathcal{SJ}}(\beta)$, then $\alpha = \ulcorner(\alpha \cdot \top) = \ulcorner(\beta \cdot \top) = \beta$. □

At this point we give two examples of ideals and scalars in standard models of Kleene algebra. First consider $LAN = (\mathcal{P}(A^*), \cup, \cdot, \bullet, \emptyset, \epsilon)$ the algebra of regular languages over an alphabet A. Here the structure of predicates is minimal. We only have $\emptyset$ and ϵ which correspond to 0 and 1. So these also are the only scalars and by Lemma 10 the corresponding ideals are $\emptyset$ and A^*. A more interesting structure of predicates shows the path algebra $PAT = (\mathcal{P}(A^*), \cup, \bowtie, \emptyset, A \cup \{\epsilon\})$ with $\bowtie$ denoting the join of two paths by concatenation of the paths and removing one of the common last or first element. So we have the four scalars $\emptyset, \{\epsilon\}, A, A \cup \{\epsilon\}$ and the corresponding ideals $\emptyset, \{\epsilon\}, \top, \top \setminus \{\epsilon\}$.

To be able to map also ideals injectively to scalars to get a one-to-one correspondence we have to do some more work. First we change our focus from Kleene algebra to standard Kleene algebra (SKA) as defined in [8] (For an axiomatization see Appendix A). This is more restrictive and based on a complete lattice structure. So there is an additional meet operation. We now show how to port a result from Dedekind categories to Kleene algebras by using SKAs.

In [22] it is shown that $f(a) = a \sqcap 1$ is an injective mapping $f : \mathcal{J} \to \mathcal{S}$ from ideals to scalars in a Dedekind category. To prove this the modular laws

$$Q \cdot R \sqcap S \leq Q \cdot (R \sqcap Q^\smile \cdot S)$$
$$Q \cdot R \sqcap S \leq (Q \sqcap S \cdot R^\smile) \cdot R$$

are used. To avoid unnecessary parenthesis we assume that composition binds more tightly than meet. Although arbitrary KAs need not have a converse operation, the proof only needs weaker versions of these laws by using $\top$ as the conversed element. So we can demand that the modular laws only hold for the top element:

Definition 8 (weakly modular KA). *We say a SKA is* weakly modular *if the modular laws hold for* $\top$*:*

$$\top \cdot a \sqcap b \leq \top \cdot (a \sqcap \top \cdot b)$$
$$a \cdot \top \sqcap b \leq (a \sqcap b \cdot \top) \cdot \top$$

The proof of injectivity of f in a weakly modular KA (WMKA) then looks as follows:

Proof.
$(j \sqcap 1) \cdot \top \leq j \cdot \top = j = j \sqcap 1 \cdot \top \leq (j \cdot \top \sqcap 1) \cdot \top = (j \sqcap 1) \cdot \top$, thus: $(j \sqcap 1) \cdot \top = j$ which immediately shows injectivity of f. Symmetrically $j = \top \cdot (j \sqcap 1)$, so that $(j \sqcap 1)$ is a scalar. □

We now also try to eliminate the need for a meet operator by searching for conditions equal to the restricted modular law without using meets. In the case of weakly modular KAs there is a closed formula for the domain operator:

Lemma 11. *Assume a WMKA, then*

$$\ulcorner a = a \cdot \top \sqcap 1$$

Proof. By the modular laws $a = a \sqcap 1 \cdot \top \leq (a \cdot \top \sqcap 1) \cdot \top$ holds, so $\ulcorner a \leq a \cdot \top \sqcap 1$ follows immediately from the Galois connection for the domain operator. On the other hand $a \cdot \top \sqcap 1 = \ulcorner a \cdot a \cdot \top \sqcap 1 = \ulcorner a \cdot a \cdot \top \sqcap \ulcorner a \leq \ulcorner a$ □

As a consequence the operation $f(a) = a \sqcap 1$ on ideals simplifies to $f(j) = j \sqcap 1 = j \cdot \top \sqcap 1 = \lceil j$. This is an operation that we also have in Kleene algebra. So we can ask, if it is possible to establish a correspondence between ideals and scalars by using domain.

Lemma 12. *The following conditions are equivalent in SKAs with domain:*

1. $\lceil a \leq a \cdot \top$
2. $\lceil a \cdot \top = a \cdot \top$
3. $\lceil a = a \cdot \top \sqcap 1$

Proof.

1. $\Rightarrow$ 2.**:** $\lceil a \cdot \top \leq a \cdot \top \cdot \top = a \cdot \top = \lceil a \cdot a \cdot \top \leq \lceil a \cdot \top \cdot \top = \lceil a \cdot \top$
2. $\Rightarrow$ 3.**:** $\lceil a = \lceil a \sqcap 1 \leq \lceil a \cdot \top \sqcap 1 = a \cdot \top \sqcap 1 = \lceil a \cdot a \cdot \top \sqcap 1 = \lceil a \cdot a \cdot \top \sqcap \lceil a \leq \lceil a$
3. $\Rightarrow$ 1.**:** $\lceil a = a \cdot \top \sqcap 1 \leq a \cdot \top$ □

It is easy to show that the reverse implication from 2. to 1. also holds, so that the first two equations are equivalent even in KAs with domain.

Lemma 13. *Symmetrically the following formulas are equivalent in SKAs with domain.*

1. $a\rceil \leq \top \cdot a$
2. $\top \cdot a\rceil = \top \cdot a$
3. $a\rceil = \top \cdot a \sqcap 1$

Motivated by equations 12.1 and 13.1 we will call Lemmas 12 and 13 subordination of domain respectively codomain. An alternative but to Lemma 12 equivalent definition that looks more symmetrical could be given by the condition:

$$\lceil a \cdot b \leq \lceil b \cdot a \cdot \top$$

Proof. " $\Rightarrow$ " : $\lceil a \cdot b = \lceil a \cdot \lceil b \cdot b = \lceil b \cdot \lceil a \cdot b \leq \lceil b \cdot \lceil a \cdot \top \overset{12.2}{=} \lceil b \cdot a \cdot \top$

" $\Leftarrow$ " : $\lceil a = \lceil a \cdot 1 \leq \lceil 1 \cdot a \cdot \top = a \cdot \top$ □

and symmetrically for the codomain conditions. Nevertheless we will use the laws from Lemmas 12 and 13 due to their simplicity. By adding one at a time of the characterizations from Lemmas 12 and 13 above we can show some more properties of ideals that are needed in later derivation steps (we only show the ones using domain):

Lemma 14. *Assume again subordination of domain, then for $j \in \mathcal{J}$*

1. $j = \lceil j \cdot \top$
2. $\lceil j = j\rceil$
3. $\lceil j \cdot \top = \top \cdot \lceil j$

Proof. 1. $\lceil j \cdot \top = j \cdot \top = j$

2. $\lceil j \leq j \cdot \top = j \Rightarrow (\lceil j)\rceil \leq j\rceil \Leftrightarrow \lceil j \leq j\rceil$. Symmetrically $j\rceil \leq \lceil j$

3. $\lceil j \cdot \top = j \cdot \top = \top \cdot j = \top \cdot j\rceil = \top \cdot \lceil j$ □

Equation 2. shows, that it does not matter if one uses domain or codomain to map ideals to scalars. This mimics the fact, that one is also free to choose composition with top either from the left or right to map a scalar to its corresponding ideal. Subordination of domain also is the key to be able to show that the domain operation on ideals is injective:

Lemma 15. *Assume subordination of domain, then $\lceil$ is injective on ideals.*

Proof. Assume $\ulcorner j = \ulcorner k$, then $j \overset{14.1}{=} \ulcorner j \cdot \top = \ulcorner k \cdot \top \overset{14.1}{=} k$ □

As one can see by Lemma 14.3 function $i_{\mathcal{JS}}$ really maps into the set of scalars, viz. commutes with the top element. Indeed the two functions are inverse:

Lemma 16. $i_{\mathcal{JS}}(i_{\mathcal{SJ}}(\alpha)) = i_{\mathcal{JS}}(\alpha \cdot \top) = \ulcorner(\alpha \cdot \top) = \alpha$
$i_{\mathcal{SJ}}(i_{\mathcal{JS}}(j)) = i_{\mathcal{SJ}}(\ulcorner j) = \ulcorner j \cdot \top = j$

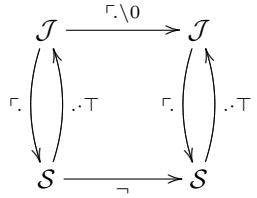
By the now established bijection between scalars and ideals it is immediately clear that the set of ideals also forms a Boolean lattice. The in the presence of residuals often used pseudo complement construction $a\backslash 0$ now coincides for ideals with the real Boolean complement. So we are able to give a closed formula of the converse operation on ideals by:

Lemma 17. *The elements j and $\overline{j} = \ulcorner j\backslash 0 = \neg\ulcorner j \cdot \top$ are complements in $\mathcal{J}$.*

Proof. By definition of $\backslash$ it holds that $\ulcorner j\backslash 0 = 0 + \neg\ulcorner j \cdot \top$ and

- $j + \overline{j} = j + \neg\ulcorner j \cdot \top \overset{14.1}{=} \ulcorner j \cdot \top + \neg\ulcorner j \cdot \top = \top$
- $j \cdot \overline{j} = j \cdot (\neg\ulcorner j \cdot \top) = j \cdot j\urcorner \cdot \neg\ulcorner j \cdot \top \overset{14.2}{=} j \cdot \ulcorner j \cdot \neg\ulcorner j \cdot \top = 0$ □

Summarizing we have the following relations between scalars and ideals (here with the use of domain and composition on the right):



2.6 About locality

An important rule that does not follow from the axiomatization - neither the Kleene algebraic, nor the S-Kleene algebraic one - is (left/right) locality [30]. This describes the fact that composition only depends on the domains of the elements on the respective side.

Definition 9 (localality). *A Kleene algebra shows left locality if*

$$\ulcorner b = \ulcorner c \Rightarrow \ulcorner(a \cdot b) = \ulcorner(a \cdot c)$$

Right locality is defined symmetrically. The definition of locality is equivalent to

$$\ulcorner(a \cdot b) = \ulcorner(a \cdot \ulcorner b)$$

which implies also immediately:

$$\ulcorner(\ulcorner a \cdot b) = \ulcorner a \cdot \ulcorner b$$

Left locality holds in Kleene algebra extended with one of the equations from Lemma 12:

Lemma 18. *Assume subordination of domain, then $\ulcorner(a \cdot \ulcorner b) = \ulcorner(a \cdot b)$*

Proof. $\ulcorner(a \cdot \ulcorner b) \leq \ulcorner(a \cdot b \cdot \top) \overset{12.2}{=} \ulcorner(\ulcorner(a \cdot b) \cdot \top) = \ulcorner(a \cdot b)$ and the opposite direction holds in all Kleene algebras. □

Conversely, right locality follows from one of the properties in Lemma 13. So a KA with subordinated domain and codomain shows locality.

2.7 Updates

To be able to change elements in certain parts we define a selective update operator. Selective here means that the update is performed with respect to the domain of the involved elements. The updated element is preserved exactly where the update is not defined.

Definition 10 (update). *Element b overwrites a by*

$$b \mid a \stackrel{\text{def}}{=} b + \neg\ulcorner b \cdot a$$

The following properties are easy to see:

Lemma 19. *1.* $b \leq b \mid a$
2. $b = \ulcorner b \cdot (b \mid a)$
3. $a \mid a = a$
4. $c \mid (a + b) = c \mid a + c \mid b$
5. $\ulcorner(b \mid a) = \ulcorner b + \ulcorner a$

Proofs of these lemmas can be found in [13] where the update operator is examined in more detail.

2.8 Images

In concrete applications of Kleene algebra, as for example pointer structures, the nodes that are reachable from some node set play an essential rôle. Here the calculation of nodes that are direct successors of other nodes is of great importance. What we need is an operator for the image of a node under an element which again returns a set of nodes. This is an instance of a Peirce [5] product. For an abstract axiomatization of the image operator in Kleene algebra see [15]. There, it is also shown that the pure definition of Kleene modules using this abstract setting is not of useful expressive power because there are too few properties connecting the two sorts of the module. We can find a remedy by identifying the Boolean sort of the module with the predicates. Together with the image operator this is a special view of a dynamic algebra [33] were the second sort is embedded as predicates into the Kleene algebraic part:

Definition 11 (image). *We define the image of s under a by:*

$$s : a \stackrel{\text{def}}{=} (s \cdot a)^{\urcorner}$$

Here predicate s could be seen as model of the addresses and a represents the pointer-linked data structure. We use the convention that $\cdot$ binds more tightly than $:$ to avoid parentheses if possible. As the image operator is a composition of two universally disjunctive functions $\cdot$ and $^{\urcorner}$, it follows that image is monotone in both arguments and

Corollary 2. *Image distributes through joins:*

$$s : (a + b) = (s : a) + (s : b)$$
$$(s + t) : a = (s : a) + (t : a)$$

Locality is directly inherited by the image operator. The corresponding equality is:

Lemma 20. *Local composition of the image operator*

$$(s : a) : b = s : (a \cdot b)$$

Proof. $(s : a) : b = ((s \cdot a)^{\urcorner} \cdot b)^{\urcorner} = (s \cdot (a \cdot b))^{\urcorner} = s : (a \cdot b)$ □

The following lemmas are immediate by definitions and the corresponding laws for composition:

Lemma 21.

1. $s : t = s \cdot t$
   Immediately:
   (a) $s : 1 = s$
   (b) $s : 0 = 0$
2. $0 : a = 0$
3. $1 : a = a^{\neg}$
4. $(s : a) \cdot b = 0 \Leftrightarrow (s : a) : b = 0$
5. $s : a^* = 0 \Leftrightarrow s = 0$
6. $s : a^* = s + (s : a) : a^*$
7. $s : (a \cdot t) = (s : a) \cdot t$
8. $s : (t \cdot a) = (s \cdot t) : a$

As there is an induction principle for the star operator in Kleene algebra (Definitions 1.(14) and 1.(15)), we also have such a rule for the image under a starred Kleene element.

Lemma 22. *A generalized induction principle for the image operator is*

$$s : a + t : b \leq t \Rightarrow s : (a \cdot b^*) \leq t$$

Proof.

$s : (a \cdot b^*) \leq t$

$\Leftrightarrow$ {[definition of : and $^{\neg}$]}

$s \cdot a \cdot b^* \leq \top \cdot t$

$\Leftarrow$ {[star induction principle]}

$s \cdot a + \top \cdot t \cdot b \leq \top \cdot t$

$\Leftrightarrow$ {[definition of $^{\neg}$, : and distributivity]}

$s : a + (\top \cdot t \cdot b)^{\neg} \leq t$

$\Leftrightarrow$ {[codomain version of Lemma 6.4 and definition of :]}

$s : a + t : b \leq t$

□

This lemma is an instance of the well-known μ-fusion rule from fixed point theory.

$$f \circ g \leq h \circ f \Rightarrow f \circ \mu g \leq \mu h$$

(see e.g. [2]) with definitions

$$f(x) = x^{\neg} \qquad g(x) = s \cdot a + x \cdot b \qquad h(x) = t$$

The least scalar a predicate is included in can equationally be defined using the image operator.

Lemma 23. *The image of a predicate s under $\top$ is the least scalar α with $s \leq \alpha$*

Proof. $s : \top = (s \cdot \top)^{\neg} = ((\top \cdot s)^{\neg} \cdot \top)^{\neg} = (\top \cdot s \cdot \top)^{\neg}$ which by Lemma 16 is a scalar and from assumption $\beta \geq s$, follows $s : \top \leq \beta : \top = (\beta \cdot \top)^{\neg} = (\top \cdot \beta)^{\neg} = \beta$ which shows that $s : \top$ is the least scalar greater than s.

2.9 Observational equivalence

As our goal is to get an abstract framework to handle pointer structures we can not demand that there is always equality of two elements. But in some cases only the equivalence of mapping behaviour suffices.

Definition 12 (observational equivalence). *We say the Kleene elements a and b are observational equivalent, if*

$$a \equiv b \overset{\text{def}}{\Leftrightarrow} \forall s \leq 1.\ s : a = s : b$$

The scope of s can be restricted to the joined domain of the two elements.

Lemma 24. $a \equiv b \Leftrightarrow \forall s \leq (\ulcorner a + \ulcorner b).\ s : a = s : b$

Proof. $s : a = (s \cdot \ulcorner a) : a = (s \cdot \ulcorner a) : b = (s \cdot \ulcorner a \cdot \ulcorner b) : b$ and symmetrically $s : b = (s \cdot \ulcorner a \cdot \ulcorner b) : a$ and by assumption follows the claim. □

2.10 Determinacy and atomicity

As we are interested in the mapping behaviour of the used elements, we sometimes have to demand that elements are deterministic. Determinacy can be characterized in Kleene algebra [10] by:

Definition 13 (determinacy). *An element $a \in \mathcal{K}$ is called a map (deterministic) if*

$$\forall b \leq a.\ b = \ulcorner b \cdot a$$

Deterministic elements in a Kleene algebra are downclosed:

Lemma 25. $map(a) \Rightarrow \forall b \leq a.\ map(b)$

Proof. Let $c \leq b \leq a$ and $map(a)$, then $\ulcorner c \cdot b = \ulcorner c \cdot \ulcorner b \cdot a = \ulcorner c \cdot a = c$ □

To achieve a really applicable framework to deal with concrete applications there is no way around atomicity. Defining a single link from one address to another in a pointer structure is for example such an atomic concept. We show here the definitions of atomicity and its relation to determinacy but in the end we can do in most of the cases with scalar-atomicity which is defined later.

Definition 14 (atom). *An element $0 \neq a \in \mathcal{K}$ is called an atom if*

$$\forall b \leq a.\ b = 0 \vee b = a$$

To mark an element as atomic we abbreviate the previous formula by the predicate $at(a)$. Determinacy and atomicity are in a certain sense related, but not equal:

Lemma 26. *1.* $at(a) \Rightarrow map(a)$
2. $at(a) \Rightarrow at(\ulcorner a) \wedge at(a^{\urcorner})$
3. $at(s) \wedge map(a) \Rightarrow at(s \cdot a)$

Proof. 1. Assume $at(a)$, then
$$\begin{aligned} &\forall b \leq a.\ b = 0 \vee b = a \\ \Rightarrow &\forall b \leq a.\ b = 0 = \ulcorner b \cdot a \vee b = \ulcorner b \cdot b = \ulcorner b \cdot a \\ \Rightarrow &\forall b \leq a.\ b = \ulcorner b \cdot a \end{aligned}$$

2. Let $s \leq \ulcorner a$, then $s \cdot a \leq a \overset{at(a)}{\Rightarrow} s \cdot a = 0 \vee s \cdot a = a$. The first disjunct simplifies to $s = 0$ by $s \leq \ulcorner a \Rightarrow s = s \cdot \ulcorner a = \ulcorner(s \cdot a) = \ulcorner 0 = 0$ whereas the second one by $s \cdot a = a \Rightarrow \ulcorner(s \cdot a) = \ulcorner a \Leftrightarrow s \cdot \ulcorner a = \ulcorner a \Leftrightarrow \ulcorner a \leq s$ and $s \leq \ulcorner a$ implies $s = \ulcorner a$ which shows the claim for domain. Atomicity of codomain is shown symmetrically.
3. Let $b \leq s \cdot a$ and therefore also $b \leq a$, hence $\ulcorner b \leq \ulcorner(s \cdot a) = s \cdot \ulcorner a \leq s$ and by $at(s) : \ulcorner b = 0 \vee \ulcorner b = s \Leftrightarrow b = 0 \vee \ulcorner b = s \overset{map(a)}{\Rightarrow} b = 0 \vee b = \ulcorner b \cdot a = s \cdot a$ □

From 26.2 and 26.3 follows immediately:

Corollary 3. $at(s) \wedge map(a) \Rightarrow at(s : a)$

Additionally we define the concept of atomicity on scalars. This is not a scalar that is atomic in $\mathcal{K}$ but atomic in the lattice of scalars!

Definition 15 (scalar-atomic). *A scalar $0 \neq \alpha \in \mathcal{S}$ is called scalar-atomic if*

$$\forall \beta \in \mathcal{S}.\ \beta \leq \alpha \Rightarrow \beta = 0 \vee \beta = \alpha$$

In the sequel we will use the predicate $sat(\alpha)$ to express that α is scalar-atomic. This concept gives us a handle to access parts of elements in a Kleene algebra as shown in the next section.

3 Simultaneous treatment of Kleene Algebras

In this section we show how to handle several Kleene algebras in one. To achieve this we port some concepts as for example crispness from fuzzy relation algebra [22,36] to Kleene algebra. Crispness describes the total absence of uncertain information. So a crisp relation relates two elements a hundred percent or not at all. With an abstract notion of crispness we are able to calculate exactly those parts of an elements that are present in all concurrently handled algebras. To model crispness we define two new operators $^\uparrow$ and $^\downarrow$ that send an element to the least crisp element it is included in and to the greatest crisp element it includes, respectively. The effect of these operations carried over to labeled graphs is depicted in Figure 1. So for example assume we have three graphs (each a Kleene algebra) fitted together in one. To distinguish the edges that come from different graphs each is labeled with a unique identifier, say μ, ν, π. In the original graph on the left side a crisp connection exists from node A to node B as they are connected by all types of links. Applying $^\uparrow$ results in the graph in the middle, where all previously anyhow linked nodes are totally linked. Application of $^\downarrow$ yields the graph on the right side in which remain only the previously crisp parts. As $^\uparrow$ and $^\downarrow$ produce related least and greatest elements we can use a Galois connection to define them. To fully axiomatize them we need

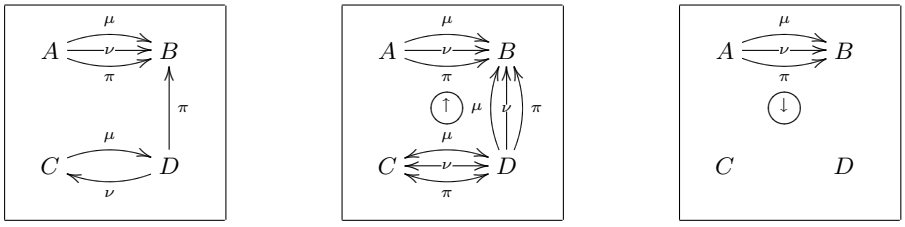


Figure 1. Example graph and application of $^\uparrow$ and $^\downarrow$

additional laws like for example Definition 16.4 which models the conversion of a fuzzy relation into its resolution form.

Definition 16 (up and down).

1. *($^\uparrow$,$^\downarrow$) form a Galois connection, e.g.* $a^\uparrow \leq b \Leftrightarrow a \leq b^\downarrow$
2. *(a)* $(a \cdot b^\downarrow)^\uparrow = a^\uparrow \cdot b^\downarrow$
   (b) $(a^\downarrow \cdot b)^\uparrow = a^\downarrow \cdot b^\uparrow$
3. *α scalar and* $\alpha \neq 0$*, then* $\alpha^\uparrow = 1$
4. $a \leq \sum_{\alpha \in \mathcal{S}} \alpha \cdot (\alpha \backslash a)^\downarrow$

Monotonicy and the cancellation laws follow directly from the Galois connection and therefore are given without proof. The interested reader may have a look at [1] for properties of Galois connections.

Corollary 4. *1. $^\uparrow$ and $^\downarrow$ are monotone*
2. $a \leq a^{\uparrow\downarrow}$ *and* $a^{\downarrow\uparrow} \leq a$
3. $a^\uparrow = a^{\uparrow\downarrow\uparrow}$ *and* $a^\downarrow = a^{\downarrow\uparrow\downarrow}$

We can now define crisp elements as the ones that are not changed by $^\uparrow$ and $^\downarrow$.

Definition 17 (crispness). *An element $a \in \mathcal{K}$ is called* crisp*, if* $a^\uparrow = a$.

As $^\uparrow$ and $^\downarrow$ return crisp elements it is evident that multiple application will not change the argument. Just as from the standard models it is also clear that 0,1 and $\top$ are crisp. As defined for scalars we will also use the term *crisp atomic* as a predicate for elements that are atoms in the lattice of crisp elements. By definition of $^\uparrow$ and $^\downarrow$ we can show:

Lemma 27.

1. $1^\uparrow = 1$
2. $a^{\downarrow\uparrow} = a^\downarrow$
3. $a^{\uparrow\downarrow} = a^\uparrow$
4. $a^{\uparrow\uparrow} = a^\uparrow$ *and* $a^{\downarrow\downarrow} = a^\downarrow$
5. $a \leq a^\uparrow$ *and* $a^\downarrow \leq a$
6. $a^\uparrow = a \Leftrightarrow a^\downarrow = a$
7. $0^\uparrow = 0$ *and* $\top^\uparrow = \top$
8. $a^\uparrow = 0 \Leftrightarrow a = 0$ *and* $a^\downarrow = \top \Leftrightarrow a = \top$ *and similar:* $s^\downarrow = 1 \Leftrightarrow s = 1$
9. $(a \cdot b^\uparrow)^\uparrow = a^\uparrow \cdot b^\uparrow = (a^\uparrow \cdot b)^\uparrow$
10. $j \neq 0$ *ideal, then* $j^\uparrow = \top$
11. $a \neq 0 \Rightarrow \top \cdot a^\uparrow \cdot \top = \top$

Proof. 1. Assume $1 \neq 0$, then apply Definition 16.3. Otherwise for all a holds $a = 1 \cdot a = 0 \cdot a = 0$ and so $1^\uparrow = 0 = 1$.
2. $a^{\downarrow\uparrow} = (1 \cdot a^\downarrow)^\uparrow = 1^\uparrow \cdot a^\downarrow = 1 \cdot a^\downarrow = a^\downarrow$
3. $a^\uparrow = a^{\uparrow\downarrow\uparrow} = a^{\uparrow\downarrow}$
4. $a^\uparrow = a^{\uparrow\downarrow} = a^{\uparrow\downarrow\uparrow} = a^{\uparrow\uparrow}$
5. $a \leq a^{\uparrow\downarrow} = a^\uparrow$ and $a^\downarrow = a^{\downarrow\uparrow} \leq a$
6. By Galois connection and Lemma 27.5
7. $0^\downarrow \leq 0$, thus $0^\downarrow = 0$ and by Lemma 27.6 follows the proposition. The second one is immediate from Lemma 27.5 as $\top \leq \top^\uparrow$.
8. - $a^\uparrow = 0 \Leftrightarrow a^\uparrow \leq 0 \Leftrightarrow a \leq 0^\downarrow \Leftrightarrow a = 0$
   - $a^\downarrow = \top \Leftrightarrow a^\downarrow \geq \top \Leftrightarrow a \geq \top^\uparrow \Leftrightarrow a = \top$
9. $(a \cdot b^\uparrow)^\uparrow \overset{27.3}{=} (a \cdot b^{\uparrow\downarrow})^\uparrow \overset{Ax.16.2}{=} a^\uparrow \cdot b^{\uparrow\downarrow} \overset{27.3}{=} a^\uparrow \cdot b^\uparrow$. The second symmetrically.
10. Assume α to be the corresponding scalar to j. Then from $j \neq 0$ follows $\alpha \neq 0$ and therefore $j^\uparrow = (\alpha \cdot \top)^\uparrow = \alpha^\uparrow \cdot \top = \top$.
11. From $\top \cdot a \cdot \top = 0$ follows $\ulcorner a \leq \top \cdot \ulcorner a \cdot \top = \top \cdot a \cdot \top = 0 \Leftrightarrow a = 0$ which is equivalent to $a \neq 0 \Rightarrow \top \cdot a \cdot \top \neq 0$. As $\top \cdot a \cdot \top$ is an ideal we can show:

$$\top \overset{27.10}{=} (\top \cdot a \cdot \top)^\uparrow \overset{27.9/27.7}{=} \top \cdot a^\uparrow \cdot \top$$

□

As we can see by Corollaries 4.1, 4.2 and Lemma 27.6 up is a closure and down an interior operator. Lemma 27.11 reflects the fact that the crisp elements form a simple Kleene algebra. Some of the laws for up (e.g. Lemma 27.9) remind us of axioms in a cylindric algebra [17]. Indeed one can see the up operator as a sort of cylindrification. An immediate consequence of Lemma 27.9 is

Corollary 5. *α scalar $\Rightarrow \alpha^\uparrow$ scalar*

Proof. $\alpha^\uparrow \cdot \top = (\alpha \cdot \top)^\uparrow = (\top \cdot \alpha)^\uparrow = \top \cdot \alpha^\uparrow$ □

3.1 Crisp algebras

In addition to crispness for single elements we will also introduce a notion of crispness with respect to the whole algebra. We will call a KA crisp, if every element $a \in \mathcal{K}$ is crisp. A first observation shows, that in fact most of the crisp elements lie outside the set of scalars. In fact there are exactly only two crisp scalars.

Lemma 28. *The only crisp scalars are* 0 *and* 1*.*

Proof. By Lemmas 27.1 and 27.7 the elements 0 and 1 are crisp scalars. Now suppose that $\alpha \in \mathcal{S}$, $\alpha \neq 0, 1$ and α crisp. Then $\alpha = \alpha^\uparrow = 1$ by Definition 16.3. □

Using this we can give a direct characterization of a crisp KA using the structure of its scalars.

Lemma 29. *A Kleene algebra is crisp if and only if* 0 *and* 1 *are the only scalars.*

Proof. First assume that 0 and 1 are the only scalars, then: $a = \sum_\alpha \alpha \cdot (\alpha \backslash a)^\downarrow = 0 \cdot (0 \backslash a)^\downarrow + 1 \cdot (1 \backslash a)^\downarrow = (a + \neg 1 \cdot \top)^\downarrow = a^\downarrow$. So for every element $a = a^\downarrow = a^\uparrow$ holds. Now assume a crisp Kleene algebra. Then all elements are crisp and therefore also the scalars. By Lemma 28 this can only be 0 and 1. □

The crisp elements of an arbitrary Kleene algebra are closed under join and composition:

Lemma 30. *Crisp elements are closed under join and composition.*

Proof. Let a, b be crisp elements, then: $(a + b)^\uparrow = a^\uparrow + b^\uparrow = a + b$

$$(a \cdot b)^\uparrow = (a \cdot b^\uparrow)^\uparrow = a^\uparrow \cdot b^\uparrow = a \cdot b$$ □

As the set of crisp elements also involves the constants $0, 1$ and $\top$ it forms a Kleene algebra.

3.2 Interaction with domain, codomain and negation

The interaction of $^\uparrow$ and $^\downarrow$ with join, composition and the constants was shown above. More interesting is the connection to domain and codomain. We will focus on the properties of the domain operator. The laws for codomain hold symmetrically. As we will see, the application of $\ulcorner$ and $^\uparrow$ can be commuted. For $\ulcorner$ and $^\downarrow$ we only can show an inequality.

Lemma 31. *1.* $\ulcorner(a^\uparrow) = (\ulcorner a)^\uparrow$
2. $\ulcorner(a^\uparrow)$ *and* $\ulcorner(a^\downarrow)$ *are crisp, e.g.* $(\ulcorner(a^\uparrow))^\uparrow = \ulcorner(a^\uparrow)$ *and* $(\ulcorner(a^\downarrow))^\uparrow = \ulcorner(a^\downarrow)$
3. $\ulcorner(a^\downarrow) \leq (\ulcorner a)^\downarrow$

Proof. 1. $\ulcorner(a^\uparrow) = \ulcorner(a^\uparrow \cdot \top) = \ulcorner((a \cdot \top)^\uparrow) \overset{12}{=} \ulcorner((\ulcorner a \cdot \top)^\uparrow) = \ulcorner((\ulcorner a)^\uparrow \cdot \top) = \ulcorner((\ulcorner a)^\uparrow) = (\ulcorner a)^\uparrow$
2. $(\ulcorner(a^\uparrow))^\uparrow \overset{1.}{=} \ulcorner(a^{\uparrow\uparrow}) \overset{27.4}{=} \ulcorner(a^\uparrow)$ and $(\ulcorner(a^\downarrow))^\uparrow \overset{1.}{=} \ulcorner(a^{\downarrow\uparrow}) \overset{27.2}{=} \ulcorner(a^\downarrow)$
3. $\ulcorner(a^\downarrow) \overset{2.}{=} (\ulcorner(a^\downarrow))^\downarrow \leq (\ulcorner a)^\downarrow$ □

By Lemma 31 it follows immediately that Axiom 16.2 and Lemma 27.9 can be lifted from compositions to images:

Corollary 6. *1.* $(s : a^\uparrow)^\uparrow = s^\uparrow : a^\uparrow = (s^\uparrow : a)^\uparrow$
2. $(s : a^\downarrow)^\uparrow = s^\uparrow : a^\downarrow$ *and* $(s^\downarrow : a)^\uparrow = s^\downarrow : a^\uparrow$

For the interaction with negation on predicates we also are only able to show inequalities:

Lemma 32. *1.* $\neg(s^\uparrow) \leq (\neg s)^\uparrow$
2. $(\neg s)^\downarrow \leq \neg(s^\downarrow)$

Proof. We only show the first proposition. The second is proven symmetrically. $s \leq s^\uparrow \Leftrightarrow \neg(s^\uparrow) \leq \neg s \Rightarrow (\neg(s^\uparrow))^\uparrow \leq (\neg s)^\uparrow$ and by $\neg(s^\uparrow) \leq (\neg(s^\uparrow))^\uparrow$ follows the proposition. □

3.3 Projection Properties

To retrieve a desired element from a graph or pointer structure we use a selector α as unique handle for an embedded subgraph. Then we have to calculate a sort of projection to get access to an element representing the mapping behavior of the embedded graph. By $\alpha \backslash a = a + \neg\alpha \cdot \top$ we can see that the residual with a scalar completes the resulting graph with links labeled with marks that are not in α. So two nodes are completely connected after the operation if and only if they before were linked via all pointers described by α. Application of $^\downarrow$ yields a graph completely

connecting all the nodes that are previously connected at least via the α links. By restricting this result to α we get a graph consisting of all the α-links of the original graph. So the projection function is

$$P_\alpha(a) = \alpha \cdot (\alpha \backslash a)^\downarrow$$

If α is scalar-atomic $P_\alpha(a)$ can be simplified:

Lemma 33. *$sat(\alpha) \Rightarrow \alpha \cdot (\alpha\backslash a)^\downarrow = \alpha \cdot a$*

Proof. One direction follows from Lemma 36.1. Immediately from $sat(\alpha)$ follows $\alpha \cdot \beta = 0 \vee \alpha \leq \beta$, so

$$\begin{aligned}\alpha \cdot a = \alpha \sum_{\beta \in \mathcal{S}} \beta \cdot (\beta\backslash a)^\downarrow = \sum_{\beta \in \mathcal{S}} \alpha \cdot \beta \cdot (\beta\backslash a)^\downarrow = \sum_{\beta \geq \alpha} \alpha \cdot (\beta\backslash a)^\downarrow \\ \leq \sum_{\beta \geq \alpha} \alpha \cdot (\alpha\backslash a)^\downarrow = \alpha \cdot (\alpha\backslash a)^\downarrow\end{aligned}$$

□

It is not clear if the opposite direction also holds. It does so in the standard model presented in Section 3.5. What we can show is:

Lemma 34. *Assume $\alpha^\downarrow$ is a scalar and $\alpha < 1$, then:*

1. $\alpha^\downarrow = 0$
2. $(\alpha \cdot \top)^\downarrow = 0$

Proof. 1. As $\alpha^\downarrow$ is a scalar and crisp $\alpha^\downarrow = 0$ or $\alpha^\downarrow = 1$. From $\alpha < 1$ follows $\alpha^\downarrow < 1$ which shows the claim.

2. $\ulcorner((\alpha \cdot \top)^\downarrow) \overset{31.3}{\leq} (\ulcorner(\alpha \cdot \top))^\downarrow = \alpha^\downarrow \overset{1.}{=} 0 \Leftrightarrow (\alpha \cdot \top)^\downarrow = 0$ □

With this the opposite direction can be shown under the assumption that $\alpha^\downarrow$ again is a scalar:

Lemma 35. *Assume for all scalars α holds that $\alpha^\downarrow$ again is a scalar. Then*

$$\alpha \cdot (\alpha\backslash a)^\downarrow = \alpha \cdot a \Rightarrow sat(\alpha)$$

Proof. Assume $0 < \beta < \alpha$ then $\beta = \alpha \cdot \beta = \alpha \cdot (\alpha\backslash\beta)^\downarrow = \alpha \cdot (\beta + \neg\alpha \cdot \top)^\downarrow \leq \alpha \cdot ((\beta + \neg\alpha) \cdot \top)^\downarrow$. By $\beta + \neg\alpha < \alpha + \neg\alpha = 1$ and Lemma 34.1 follows $\beta = 0$ which is a contradiction. □

The projection is used for calculating the image of m under $P_\alpha(a)$ which is abbreviated by $a_\alpha(m)$ in Section 4.2. This gives us all the α-successors of m. We show some properties of the projection function:

Lemma 36. *1. $P_\alpha(a) \leq \alpha \cdot a$ In particular: $P_\alpha(a) \leq a$*
2. $sat(\alpha) \wedge sat(\beta) \Rightarrow P_{(\alpha+\beta)} = P_\alpha + P_\beta$

Proof. 1. $\alpha \cdot (\alpha\backslash a)^\downarrow \leq \alpha \cdot (\alpha\backslash a) = \alpha \cdot a$
2. $(\alpha + \beta) \cdot ((\alpha + \beta)\backslash a)^\downarrow = (\alpha + \beta) \cdot a = \alpha \cdot a + \beta \cdot a = \alpha \cdot (\alpha\backslash a)^\downarrow + \beta \cdot (\beta\backslash a)^\downarrow$ □

By defining

$$(\alpha \cdot a)^\star \overset{\text{def}}{=} \alpha \cdot a^*$$

the sets $\mathcal{K}_\alpha = \{\alpha \cdot (\alpha\backslash a)^\downarrow \mid a \in \mathcal{K}\}$ of all elements of an atomic scalar α form Kleene algebras $(\mathcal{K}_\alpha, +, \cdot, 0, \alpha, ^\star)$. The corresponding ideal $j = \alpha \cdot \top$ to α forms the top element. By

$$\begin{aligned}\alpha \cdot (\alpha\backslash a)^\downarrow + \alpha \cdot (\alpha\backslash b)^\downarrow = (\alpha \cdot a) + (\alpha \cdot b) = \alpha \cdot (a + b) = \alpha \cdot (\alpha\backslash(a + b))^\downarrow \\ \alpha \cdot (\alpha\backslash a)^\downarrow \cdot \alpha \cdot (\alpha\backslash b)^\downarrow = \alpha \cdot a \cdot \alpha \cdot b = \alpha \cdot (a \cdot b) = \alpha \cdot (\alpha\backslash(a \cdot b))^\downarrow \\ \alpha \cdot (\alpha\backslash a)^\downarrow = \ulcorner j \cdot (\alpha\backslash a)^\downarrow \leq \ulcorner j \cdot \top = j \cdot \top = j\end{aligned}$$

we have shown that $\mathcal{K}_\alpha$ is closed under join and composition as well as j is the top element.

3.4 Intermediate summary

At this point we give a complete summary of the definition of an enriched Kleene algebra that supports the treatment of several KAs in one:

Definition 18 (EKA). *An enriched Kleene algebra (EKA) is a Kleene algebra* $(\mathcal{K}, +, \cdot, 0, 1, ^*)$ *with additional*

- *a top element* $\top$
- *subordinate domain and codomain* $\ulcorner a \leq a \cdot \top$ *and* $a^\urcorner \leq \top \cdot a$
- $^\uparrow$ *and* $^\downarrow$ *defined as in Definition 16*

In the sequel we will work with such EKAs and use the term Kleene algebra interchangeably.

3.5 A concrete model

To show that our definitions make sense we now will give a concrete (relational) model of such an extension of a Kleene algebra. The existence of a model ensures that the added properties do not imply a contradictory axiomatization. We use a model that relates two elements from a set $\mathcal{A}$ via several selectors.

Definition 19. *Let* $\mathcal{A}$ *be the set of addresses and* $\mathcal{S}$ *the set of selectors used in a pointer model. Then the elements of our concrete extended Kleene algebra are functions* $f : \mathcal{A} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$.

We define the operations of the model by:

1. $f_{a+b}(x,y) = f_a(x,y) \cup f_b(x,y)$
2. $f_{a \cdot b}(x,y) = \bigcup_z \{f_a(x,z) \cap f_b(z,y)\}$
3. $f_0(x,y) = \emptyset$
4. $f_\top(x,y) = \mathcal{S}$
5. $f_1(x,y) = \begin{cases} \mathcal{S} & , x = y \\ \emptyset & , \text{otherwise} \end{cases}$

From the definition we can derive all the other operations that are:

Corollary 7. *1.* $f_{\neg s}(x,y) = \begin{cases} \overline{f_s(x,y)} & , x = y \\ \emptyset & , otherwise \end{cases}$

2. $f_{\ulcorner a}(x,y) = \begin{cases} \bigcup_z \{f_a(x,z)\} & , x = y \\ \emptyset & , otherwise \end{cases}$

3. $f_{a^\uparrow}(x,y) = \begin{cases} \mathcal{S} & , f_a(x,y) \neq \emptyset \\ \emptyset & , otherwise \end{cases}$

4. $f_{a^\downarrow}(x,y) = \begin{cases} \mathcal{S} & , f_a(x,y) = \mathcal{S} \\ \emptyset & , otherwise \end{cases}$

5. $f_{a|b}(x,y) = f_a(x,y) \cup (\bigcap_z \overline{f_a(x,z)} \cap f_b(x,y))$

6. $f_{s \backslash a}(x,y) = f_a(x,y) \cup \overline{f_s(x,x)}$

One also can define predicates for the properties used in the calculus.

Corollary 8. *1.* $a \leq 1 \Leftrightarrow \forall x,y.\ f_a(x,y) = 0 \vee (x = y \wedge f_a(x,y) \subseteq \mathcal{S})$

2. $\alpha \cdot \top = \top \cdot \alpha \Leftrightarrow \forall x,y.\ f_\alpha(x,y) = \begin{cases} \mathcal{U} & , x = y \quad (\mathcal{U} \subseteq \mathcal{S}) \\ \emptyset & , otherwise \end{cases}$

3. $a^\uparrow = a \Leftrightarrow \forall x,y.\ f_a(x,y) = \mathcal{S} \vee f_a(x,y) = \emptyset$

4 Modeling Pointer Structures

As mentioned earlier the previously described EKAs can be used to model pointer structures. We consider an abstract model of pointer structures as described in [27] using several selectors to model records. The concurrently treated algebras represent the selector types via which addresses can be linked. Each scalar-atomic element is the unique identifier to get its related selector from a Kleene element. Selection of a certain selector structure is calculated using the projection presented in Section 3.3.

4.1 Addresses

We already have defined the notion of crispness. By using crisp predicates we are able to model elements that can play the part of addresses in pointer structures. The idea is, that addresses are represented by nodes that are completely connected via all selectors.

Definition 20 (address). *A crisp element $m \leq 1$ is called an* address.

In the sequel we will use letters m and n to denote addresses. As addresses are crisp, they are closed under join and composition. Additionally they are closed under complement and so form a lattice.

Lemma 37. *1. If m is an address then $\neg m$ is also an address*
2. $(a \cdot m)^\uparrow = a^\uparrow \cdot m$ and $(m \cdot a)^\uparrow = m \cdot a^\uparrow$
3. $(a : m)^\uparrow = a^\uparrow : m$ and $(m : a)^\uparrow = m : a^\uparrow$
4. If $\alpha \neq 0$, then $m \cdot \alpha = 0 \Leftrightarrow m = 0$

Proof. 1. $m + (\neg m)^\uparrow = m^\uparrow + (\neg m)^\uparrow = (m + \neg m)^\uparrow = 1^\uparrow = 1$

$$m \cdot (\neg m)^\uparrow = m^\uparrow \cdot (\neg m)^\uparrow = (m \cdot \neg m)^\uparrow = 0^\uparrow = 0$$

So $(\neg m)^\uparrow$ is the unique complement of m and therefore $(\neg m)^\uparrow = \neg m$.
2. $a^\uparrow \cdot m = a^\uparrow \cdot m^\uparrow = (a \cdot m^\uparrow)^\uparrow = (a \cdot m)^\uparrow$ The second symmetrically!
3. Follows immediately from 2.
4. $m \cdot \alpha = 0 \Rightarrow (m \cdot \alpha)^\uparrow = 0^\uparrow \Leftrightarrow m \cdot \alpha^\uparrow = 0 \Leftrightarrow m = 0$. The opposite direction is trivial. □

4.2 Ministore

To have the possibility to define single links from one address to an other we will define a ministore that models completely linked addresses from the domain to the range. So we restrict the totally linked store $\top$ at selector α to m on its domain and to n on its range:

$$m \cdot P_\alpha(\top) \cdot n = m \cdot \alpha \cdot (\alpha \backslash \top)^\downarrow \cdot n = m \cdot \alpha \cdot \top^\downarrow \cdot n = m \cdot \alpha \cdot \top \cdot n$$

Definition 21 (ministore). *Let $m, n \in \mathcal{K}$ be addresses and $\alpha \in \mathcal{S}$ a selector. Then we call the element $(m \xrightarrow{\alpha} n) \stackrel{\text{def}}{=} m \cdot \alpha \cdot \top \cdot n$ an* α-ministore *with source addresses m and target addresses n.*

If addresses m and n are atomic, an α-ministore models exactly a single pointer link from address m via selector α to address n.

It is easy to see that $(m \xrightarrow{\alpha} n)^\uparrow = m \cdot \top \cdot n = (m \xrightarrow{1} n)^\uparrow$. By construction it is also evident that the domain of a ministore equals the set of starting addresses restricted to the respective selector. As well the image of the given set of addresses under the ministore should result in all the connected addresses.

Lemma 38. *Let $\alpha \in \mathcal{S}$ a selector and $m, n \in \mathcal{K}$ be addresses*

1. $\ulcorner(m \xrightarrow{\alpha} n) = m \cdot \alpha$
2. $(m \xrightarrow{\alpha} n)\urcorner = \alpha \cdot n$
3. $m : (m \xrightarrow{\alpha} n) = \alpha \cdot n$
4. $\neg m \cdot ((m \xrightarrow{\alpha} n) \mid a) = \neg m \cdot a$
5. $m : ((m \xrightarrow{\alpha} n) \mid a)^\uparrow = n + m : (\neg\alpha \cdot a)^\uparrow$

Proof.

1. $m \cdot \alpha \overset{6.4}{=} \ulcorner(m \cdot \alpha \cdot \top) \overset{27.11}{=} \ulcorner(m \cdot \alpha \cdot \top \cdot n \cdot \top) \overset{12}{=} \ulcorner(\ulcorner(m \cdot \alpha \cdot \top \cdot n) \cdot \top) \overset{6.4}{=} \ulcorner(m \cdot \alpha \cdot \top \cdot n)$
2. Symmetrically to 1.
3. $m : (m \xrightarrow{\alpha} n) = (m \cdot (m\alpha\top n))^\neg = (m\alpha\top n)^\neg = \alpha \cdot n$
4. $\begin{aligned}\neg m \cdot ((m \xrightarrow{\alpha} n) \mid a) &= \neg m \cdot (m \xrightarrow{\alpha} n) + \neg m \cdot \neg(m \cdot \alpha) \cdot a \\ &= 0 + \neg m \cdot a + \neg m \cdot \neg\alpha \cdot a = \neg m \cdot a\end{aligned}$
5. $\begin{aligned}m : ((m \xrightarrow{\alpha} n) \mid a)^\uparrow &= (m : (m \xrightarrow{\alpha} n))^\uparrow + m : (\neg(m \cdot \alpha) \cdot a)^\uparrow \\ &= (\alpha \cdot n)^\uparrow + (m : (\neg m \cdot a + \neg\alpha \cdot a))^\uparrow = n + m : (\neg\alpha \cdot a)^\uparrow \Box\end{aligned}$

For local reasoning we often have to step exactly one link further along a selector. We will abbreviate the image of address m under selector α of store a by $a_\alpha(m)$. As we normally want the result to be an address again, we additionally define $\hat{a}_\alpha(m)$ to be the crisp image:

Definition 22 (restricted image).

1. $a_\alpha(m) \overset{\text{def}}{=} m : P_\alpha(a) = m : (\alpha \cdot (\alpha\backslash a)^\downarrow)$
2. $\hat{a}_\alpha(m) \overset{\text{def}}{=} a_\alpha(m)^\uparrow = (m : (\alpha \cdot (\alpha\backslash a)^\downarrow))^\uparrow = m : (\alpha\backslash a)^\downarrow$

By Lemma 36.1 follows

Corollary 9. $\hat{a}_\alpha(m) \leq m : a^\uparrow$

So we are in the position to show that overwriting of an α-successor with the original value leaves the store untouched. This law was denoted as $(p.k := p.k) = p$ in [30]. Nevertheless, by the more abstract model we are only able to show observational equivalence of the two terms.

Lemma 39. *Assume* $sat(\alpha)$*, then* $(m \xrightarrow{\alpha} \hat{a}_\alpha(m)) \mid a \equiv a$

Proof. Let m, n be addresses and m crisp atomic, then

$$m \cdot n \leq m \Rightarrow m \cdot n = 0 \vee m \cdot n = m$$

So we handle two cases:

$m \cdot n = 0$**:** By assumption $n : (m \xrightarrow{\alpha} \hat{a}_\alpha(m)) = 0$ and $n : ((m \cdot \alpha) \cdot a) = 0$.

$$\begin{aligned}&n : ((m \xrightarrow{\alpha} \hat{a}_\alpha(m)) \mid a) \\ &= n : (m \xrightarrow{\alpha} \hat{a}_\alpha(m)) + n : (\neg(m \cdot \alpha) \cdot a) \\ &= 0 + n : (\neg(m \cdot \alpha) \cdot a) + n : ((m \cdot \alpha) \cdot a) \\ &= n : a\end{aligned}$$

$m \cdot n = m$**:** With a first auxiliary calculation

$$\begin{aligned}n : (m \xrightarrow{\alpha} \hat{a}_\alpha(m)) = (m \xrightarrow{\alpha} \hat{a}_\alpha(m))^\neg &= \alpha \cdot \hat{a}_\alpha(m) = \alpha \cdot (m : (\alpha\backslash a)^\downarrow) \\ &\overset{21.7}{=} m : (\alpha \cdot (\alpha\backslash a)^\downarrow) \overset{sat(\alpha)}{=} m : (\alpha \cdot a) = (n \cdot m) : (\alpha \cdot a) \\ &= n : ((m \cdot \alpha) \cdot a)\end{aligned}$$

we can show: $\begin{aligned}&n : ((m \xrightarrow{\alpha} \hat{a}_\alpha(m)) \mid a) \\ &= n : (m \xrightarrow{\alpha} \hat{a}_\alpha(m)) + n : (\neg(m \cdot \alpha) \cdot a) \\ &= n : ((m \cdot \alpha) \cdot a) + n : (\neg(m \cdot \alpha) \cdot a) \\ &= n : a\end{aligned}$

$\Box$

4.3 Reachability

The most important things in pointer structures are based on reachability observations. Especially we are interested in addresses or nodes reachable from a set of starting addresses as well as the part of the store that is reachable. So we first define an operator to calculate all the reachable addresses starting from m in store a:

Definition 23 (reach). $reach(m,a) \stackrel{\text{def}}{=} m : (a^\uparrow)^*$

To avoid unnecessary parenthesis we abbreviate $(a^\uparrow)^*$ by $a^{\uparrow^*}$. Distributivity over joins is directly inherited from the image operator:

Corollary 10. $reach(m+n,a) = reach(m,a) + reach(n,a)$

We additionally will use reachability only via a certain selector defined by:

$$reach_\alpha(m,a) \stackrel{\text{def}}{=} reach(m, P_\alpha(a)) \stackrel{sat(\alpha)}{=} reach(m, \alpha \cdot a)$$

Evidently, this is equivalent to a reachability calculation in the algebra $\mathcal{K}_\alpha$.

As consequence from Definition 23 and star decomposition we can decompose *reach* in different ways. Even the very efficient calculation in Lemma 40.3 can be improved to 40.4 by only proceeding with addresses not in m.

Lemma 40. *1.* $reach(m,a) = m + reach(m,a) : a^\uparrow$
2. $reach(m,a) = m + reach(m : a^\uparrow, a)$
3. $reach(m,a) = m + reach(m : a^\uparrow, \neg m \cdot a)$
4. $reach(m,a) = m + reach((m : a^\uparrow) \cdot \neg m, \neg m \cdot a)$

Proof. 1. $reach(m,a) = m : a^{\uparrow^*} = m : (1 + a^{\uparrow^*} \cdot a^\uparrow) = m + m : (a^{\uparrow^*} \cdot a^\uparrow)$
$= m + (m : a^{\uparrow^*}) : a^\uparrow = m + reach(m,a) : a^\uparrow$

2. Symmetrically to 1.

3. The claim follows by image induction from

$$\begin{aligned}
&m + (m + reach(m : a^\uparrow, \neg m \cdot a)) : a^\uparrow \\
&= m + m : a^\uparrow + reach(m : a^\uparrow, \neg m \cdot a) : (m \cdot a^\uparrow) + reach(m : a^\uparrow, \neg m \cdot a) : (\neg m \cdot a^\uparrow) \\
&\leq m + m : a^\uparrow + 1 : (m \cdot a^\uparrow) + reach(m : a^\uparrow, \neg m \cdot a) : (\neg m \cdot a^\uparrow) \\
&\stackrel{21.8}{=} m + (m : a^\uparrow) + reach(m : a^\uparrow, \neg m \cdot a) : (\neg m \cdot a^\uparrow) \\
&\stackrel{1.}{=} m + reach(m : a^\uparrow, \neg m \cdot a)
\end{aligned}$$

4. $reach(m,a)$

$$\begin{aligned}
&\stackrel{3.}{=} m + reach(m : a^\uparrow, \neg m \cdot a) \\
&= m + (m : a^\uparrow) : (1 + \neg m \cdot a^\uparrow \cdot (\neg m \cdot a)^{\uparrow^*}) \\
&= m + (m : a^\uparrow) : (m + \neg m) + (m : a^\uparrow) : (\neg m \cdot a^\uparrow \cdot (\neg m \cdot a)^{\uparrow^*}) \\
&= m + (m : a^\uparrow) \cdot m + (m : a^\uparrow) \cdot \neg m + ((m : a^\uparrow) \cdot \neg m) : (\neg m \cdot a^\uparrow \cdot (\neg m \cdot a)^{\uparrow^*}) \\
&= m + ((m : a^\uparrow) \cdot \neg m) : (1 + (\neg m \cdot a)^\uparrow \cdot (\neg m \cdot a)^{\uparrow^*}) \\
&= m + reach((m : a^\uparrow) \cdot \neg m, \neg m \cdot a)
\end{aligned}$$

□

Reachability in a join of two stores also can be calculated recursively similar to Lemma 2:

Lemma 41. $reach(m, a+b) = reach(m,a) + reach(m,a) : (b^\uparrow \cdot (a+b)^{\uparrow^*})$

Proof.
$$
\begin{aligned}
& reach(m, a + b) \\
= \quad & \{\!\!\{ \text{ definition of } reach \text{ and distributivity } \}\!\!\} \\
& m : (a^{\uparrow} + b^{\uparrow})^* \\
= \quad & \{\!\!\{ \text{ Lemma 2 } \}\!\!\} \\
& m : (a^{\uparrow^*} \cdot (1 + b^{\uparrow} \cdot (a^{\uparrow} + b^{\uparrow})^*)) \\
= \quad & \{\!\!\{ \text{ distributivity and local composition } \}\!\!\} \\
& reach(m, a) + reach(m, a) : (b^{\uparrow} \cdot (a + b)^{\uparrow^*})
\end{aligned}
$$
□

The purpose of *reach* should be to calculate a set of addresses that are reachable via a given pointer structure from a starting set of addresses. So it would be reasonable that *reach* returns a *crisp* predicate. By definition of *reach* it is evident that the resulting element is a predicate. Additionally we can show that the calculated result of the reach operator is crisp and therefore an address:

Lemma 42. *The reach operator returns addresses:* $reach(m, a)^{\uparrow} = reach(m, a)$

Proof. By Lemma 40 and image induction it follows that:

$$
\begin{aligned}
& (m + reach(m, a)^{\downarrow} : a^{\uparrow})^{\uparrow} \leq m + reach(m, a) : a^{\uparrow} = reach(m, a) \\
\Leftrightarrow\ & m + reach(m, a)^{\downarrow} : a^{\uparrow} \leq reach(m, a)^{\downarrow} \\
\Rightarrow\ & m : a^{\uparrow^*} \leq reach(m, a)^{\downarrow} \\
\Leftrightarrow\ & reach(m, a) \leq reach(m, a)^{\downarrow} \\
\Leftrightarrow\ & reach(m, a)^{\uparrow} \leq reach(m, a)
\end{aligned}
$$

□

If we advance one step in the pointer structure it is evident, that the set of reachable nodes can not grow:

Lemma 43. $reach(a_\alpha(m), a) \leq reach(m, a)$

Proof. $reach(a_\alpha(m), a) \leq reach(m : a^{\uparrow}, a) \leq m + reach(m : a^{\uparrow}, a) = reach(m, a)$
□

The $from$ operator describes the part of a store that contains all the links and addresses that are reachable from the entry address. This is a sort of projection of the live part of the store.

Definition 24 (from). $from(m, a) \stackrel{\text{def}}{=} reach(m, a) \cdot a$

In contrast to [27] we abstract from the original definition of $from$ in that we only focus on the used store and not on the whole pointer structure, as the rest can be handled simply by pairing and comparing the starting addresses. As before we define the reachable part of a store via selector α by

$$from_\alpha(m, a) \stackrel{\text{def}}{=} reach_\alpha(m, a) \cdot a$$

Equality of the $from$ part of a pointer structure implies equality of the reachable addresses:

Lemma 44. $from(m, a) = from(m, b) \Rightarrow reach(m, a) = reach(m, b)$

Proof. The claim follows immediately from Lemma 40.1 as we know that $reach$ can be expressed by $from$: $reach(m,a) = m + reach(m,a) : a^{\uparrow} = m + (from(m,a))^{\neg\uparrow}$ □

By lifting the result from Lemma 40.4 we are also able to calculate $from$ efficiently:

Lemma 45. $from(m,a) = m \cdot a + from((m:a) \cdot \neg m, \neg m \cdot a)$

Proof.

$$
\begin{aligned}
&from(m,a)\\
&\overset{40.4}{=} m \cdot a + reach((m:a) \cdot \neg m, \neg m \cdot a) \cdot a\\
&= m \cdot a + reach((m:a) \cdot \neg m, \neg m \cdot a) \cdot (m + \neg m) \cdot a\\
&= m \cdot a + reach((m:a) \cdot \neg m, \neg m \cdot a) \cdot m \cdot a + reach((m:a) \cdot \neg m, \neg m \cdot a) \cdot \neg m \cdot a\\
&= m \cdot a + reach((m:a) \cdot \neg m, \neg m \cdot a) \cdot \neg m \cdot a\\
&= m \cdot a + from((m:a) \cdot \neg m, \neg m \cdot a)
\end{aligned}
$$

□

Another interesting point is iteration of the reachability operators $reach$ and $from$. The idempotence of $reach$ is a rather simple calculation using locality of images. Additionally we can show that

Lemma 46. *$reach$ is a closure operator, viz*

1. *Extensive:* $m \leq reach(m,a)$
2. *Idempotent:* $reach(reach(m,a),a) = reach(m,a)$
3. *Monotone:* $m \leq n \Rightarrow reach(m,a) \leq reach(n,a)$

Proof. 1. Follows immediately from 40.1.

2. $reach(reach(m,a),a) = (m : a^{\uparrow^*}) : a^{\uparrow^*} = m : (a^{\uparrow^*} \cdot a^{\uparrow^*})$
$$\overset{1.4}{=} m : a^{\uparrow^*} = reach(m,a)$$
3. By monotony of all involved operators. □

Idempotence of $from$ is a little bit more tricky. Here we have to use the image induction principle to be able to reason about the star of a $reach$:

Lemma 47. *$from$ is an interior operator, viz*

1. *Reductive:* $from(m,a) \leq a$
2. *Idempotent:* $from(m, from(m,a)) = from(m,a)$
3. *Monotone:* $a \leq b \Rightarrow from(m,a) \leq from(m,b)$

Proof. 1. Trivial

2. Let $b = reach(m,a) \cdot a$, then $reach(m,a) \leq reach(m,b)$ follows from:

$$
\begin{aligned}
&m + reach(m,b) : a^{\uparrow}\\
&= m + (reach(m,b) \cdot (reach(m,a) + \neg reach(m,a))) : a^{\uparrow}\\
&= m + (reach(m,b) \cdot reach(m,a)) : a^{\uparrow} + (reach(m,b) \cdot \neg reach(m,a)) : a^{\uparrow}\\
&\leq m + reach(m,b) : (reach(m,a) : a^{\uparrow})\\
&= m + reach(m,b) : b^{\uparrow}\\
&= reach(m,b)
\end{aligned}
$$

Then: $from(m,b) = reach(m,b) \cdot b = reach(m,b) \cdot reach(m,a) \cdot a$
$= reach(m,a) \cdot a = from(m,a)$

3. Immediately from monotonicity of $reach$. □

With this we can show, that the $from$ operator really does not change connections in the live part of the store. So the reachable addresses are equal to the ones in the original store:

Corollary 11. $reach(m, from(m, a)) = reach(m, a)$

Which follows immediately from Lemma 44 and the previous one.

4.4 Non-reachability

If we know which addresses are allocated, we are able to define a complementary operator to $reach$ that calculates all the used but not reachable records in a pointer structure. Therefore we define $recs$ that takes all elements a pointer link starts from and converts them to addresses.

Definition 25 (allocated records). $recs(a) \stackrel{\text{def}}{=} (\ulcorner a)^{\uparrow}$

As by Lemma 31 we know that $\ulcorner$ and $^{\uparrow}$ can be commuted, this is equivalent to $recs(a) = \ulcorner(a^{\uparrow})$ which we will also use if appropriate. Additivity is inherited from the involved operations. The following rules can be used to simplify expressions containing $recs$ by elimination of domain or join.

Lemma 48.

1. $recs(\ulcorner a) = recs(a)$
2. $recs(\ulcorner b \cdot a) \leq recs(b)$
3. $\alpha \neq 0 \Rightarrow recs(m \xrightarrow{\alpha} n) = m$
4. $recs(b \mid a) = recs(b) + recs(a)$
5. $recs(m \cdot a) = m \cdot recs(a)$
6. $\alpha \neq 0 \Rightarrow recs((m \xrightarrow{\alpha} n) \mid a) = m + \neg m \cdot recs(a)$

Proof. 1. $recs(\ulcorner a) = (\ulcorner(\ulcorner a))^{\uparrow} = (\ulcorner a)^{\uparrow} = recs(a)$

2. $recs(\ulcorner b \cdot a) = recs(\ulcorner(\ulcorner b \cdot a)) = recs(\ulcorner b \cdot \ulcorner a) \leq recs(\ulcorner b) \stackrel{1.}{=} recs(b)$
3. $recs(m \xrightarrow{\alpha} n) = (\ulcorner(m \xrightarrow{\alpha} n))^{\uparrow} = (m \cdot \alpha)^{\uparrow} = m$
4. $recs(b \mid a) = recs(b) + recs(\neg\ulcorner b \cdot a) \stackrel{2.}{=} recs(b) + recs(\ulcorner b \cdot a) + recs(\neg\ulcorner b \cdot a)$
   $= recs(b) + recs(a)$
5. $recs(m \cdot a) = (\ulcorner(m \cdot a))^{\uparrow} = (m \cdot \ulcorner a)^{\uparrow} = m \cdot (\ulcorner a)^{\uparrow} = m \cdot recs(a)$
6. $recs((m \xrightarrow{\alpha} n) \mid a) = recs((m \xrightarrow{\alpha} n)) + \neg m \cdot recs(a) \stackrel{3.}{=} m + \neg m \cdot recs(a)$ □

As abbreviation we define the set of addresses pointers are linked to by:

Definition 26. $links(a) \stackrel{\text{def}}{=} (a^{\urcorner})^{\uparrow}$

For symmetry reasons all the laws that hold for $recs$ hold correspondingly. Immediately from Lemma 48.5 follows that the allocated records of $from(m, a)$ are all allocated records that are reachable.

Corollary 12. $recs(from(m, a)) = reach(m, a) \cdot recs(a)$

The allocated but non-reachable records now are all these in $recs$ without the reachable ones.

Definition 27 (noreach). $noreach(m, a) \stackrel{\text{def}}{=} recs(a) \cdot \neg reach(m, a)$

The operator $noreach_{\alpha}$ of non-reachability via a certain selector works similar to $reach_{\alpha}$ and $from_{\alpha}$. The previously noticed relations between $reach$ and $from$ immediately can be applied to $noreach$.

Lemma 49. $noreach(m, a) = recs(a) \cdot \neg recs(from(m, a))$

Proof.

$$
\begin{aligned}
& recs(a) \cdot \neg recs(from(m,a)) \\
= \quad & \{\!\![\text{ Corollary 12 }]\!\!\} \\
& recs(a) \cdot \neg(reach(m,a) \cdot recs(a)) \\
= \quad & \{\!\![\text{ de Morgan and distributivity }]\!\!\} \\
& recs(a) \cdot \neg reach(m,a) + recs(a) \cdot \neg recs(a) \\
= \quad & \{\!\![\text{ definition of } noreach]\!\!\} \\
& noreach(m,a)
\end{aligned}
$$

□

By antitony and Lemma 43 for stepping in the calculation of reachable nodes we can establish a dual proposition for *noreach*:

Lemma 50. $noreach(m,a) \leq noreach(a_\alpha(m),a)$

Proof. Immediate from Lemma 43 □

We can show that the non-reachable part with respect to a fixed selector of a pointer structure after a pointer manipulation equals the non-reachable part starting from the new target address.

Lemma 51. $sat(\alpha) \Rightarrow noreach_\alpha(m,(m \xrightarrow{\alpha} n) \mid a) = noreach_\alpha(n, \neg m \cdot a)$

Proof.

$$
\begin{aligned}
& noreach_\alpha(m,(m \xrightarrow{\alpha} n) \mid a) \\
= \quad & \{\!\![\text{ def. of } noreach_\alpha \text{ and } reach_\alpha \text{ }]\!\!\} \\
& recs((m \xrightarrow{\alpha} n) \mid a) \cdot \neg reach(m, \alpha \cdot ((m \xrightarrow{\alpha} n) \mid a)) \\
= \quad & \{\!\![\text{ Lemmas 48.4 and 40.3 }]\!\!\} \\
& (m + recs(a)) \cdot \neg reach(m : (\alpha \cdot ((m \xrightarrow{\alpha} n) \mid a))^\uparrow, \neg m \cdot \alpha \cdot ((m \xrightarrow{\alpha} n) \mid a)) \\
= \quad & \{\!\![\text{ Boolean algebra, definition of } \mid \text{ and simplification }]\!\!\} \\
& \neg m \cdot recs(a) \cdot \neg reach(n, \alpha \cdot \neg m \cdot a) \\
= \quad & \{\!\![\text{ Lemma 48.5 }]\!\!\} \\
& recs(\neg m \cdot a) \cdot \neg reach(n, \alpha \cdot \neg m \cdot a) \\
= \quad & \{\!\![\text{ def. of } reach_\alpha \text{ and } noreach_\alpha \text{ }]\!\!\} \\
& noreach_\alpha(n, \neg m \cdot a)
\end{aligned}
$$

□

Incidentally we noticed a copy error on the right hand side of this lemma in [29], as we tried to prove it in the form given there. The same lemma was noted correctly in the former articles [27] and [28]. Nevertheless, in all these papers the restriction to a single selector is not mentioned.

Additionally, as a further abbreviation we define a reachability predicate that evaluates to true if a set of nodes represented by a predicate n is reachable from the pointer structure (m,a). In contrast to a point-wise definition we are only able to model sets of nodes. Therefore we define three different reachability and non-reachability predicates:

Definition 28.

1. *Every node in element n is reachable:* $(m,a) \vdash n \overset{\text{def}}{\Leftrightarrow} n \leq reach(m,a)$
2. *Some nodes in n are reachable:* $(m,a) \vdash n \overset{\text{def}}{\Leftrightarrow} 0 < reach(m,a) \cdot n < n$
3. *None of the nodes in n is reachable:* $(m,a) \nvdash n \overset{\text{def}}{\Leftrightarrow} reach(m,a) \cdot n = 0$

If n is atomic the predicate $\vdash$ evaluates to false. In this case we have the pointwise view. Each address element represents exactly one node and $\vdash$ and $\nvdash$ are complementary predicates. The validity of predicate $(m,a) \nvdash n$ immediately can be deduced from non-reachability. We can give an exact characterization when address n is in the set of non-reachable records:

Lemma 52. $n \leq noreach(m,a) \Leftrightarrow n \leq recs(a) \wedge (m,a) \nvdash n$

Proof.

$\Rightarrow$: a) $n \leq noreach(m,a) = recs(a) \cdot \neg reach(m,a) \leq recs(a)$
b) $n \cdot reach(m,a) \leq recs(a) \cdot \neg reach(m,a) \cdot reach(m,a) = recs(a) \cdot 0 = 0$
$\Leftarrow$: $n \cdot noreach(m,a) = n \cdot recs(a) \cdot \neg reach(m,a) = n \cdot \neg reach(m,a)$
$= n \cdot \neg reach(m,a) + n \cdot reach(m,a) = n$ □

4.5 Localization

Most of the expressions for pointer structures are only valid under certain reachability conditions that have to hold. If we know that the records of a certain element b are not reachable from a pointer structure (m,a), we can simplify some expressions. This means that changes of the pointer structure only have local effects. First we show some simple consequences from reachability constraints:

Lemma 53. *Assume that $(m,a) \nvdash recs(b)$ which by definition is equivalent to $reach(m,a) \cdot recs(b) = 0$, then*

1. $reach(m,a) \cdot b^{\uparrow} = 0$
2. $reach(m,a) \cdot b = 0$
3. $reach(m,a) \cdot \ulcorner b = 0$

Proof. 1. $reach(m,a) \cdot b^{\uparrow} = reach(m,a) \cdot \ulcorner(b^{\uparrow}) \cdot b^{\uparrow} = reach(m,a) \cdot recs(b) \cdot b^{\uparrow} = 0$
2. $reach(m,a) \cdot b \leq reach(m,a) \cdot b^{\uparrow} = 0$
3. $reach(m,a) \cdot \ulcorner b \leq reach(m,a) \cdot \ulcorner(b^{\uparrow}) = reach(m,a) \cdot recs(b) = 0$ □

By strictness of domain all these laws can be lifted to images. Using these prerequisites the expression $(m,a) \nvdash recs(b)$ gives us a lot of information about reachability in pointer structures. So for example we can completely leave out certain regions of the store in the calculation of reachable addresses. In other words the effects to reachability can be localized.

Lemma 54 (Localization I). *Assume $(m,a) \nvdash recs(b)$, then*

1. $reach(m,a+b) = reach(m,a)$
2. $reach(m,b \mid a) = reach(m,a)$

Proof.

1. $reach(m,a+b) \overset{41}{=} reach(m,a) + reach(m,a) : (b^{\uparrow} \cdot (a+b)^{\uparrow^*}) \overset{53.1}{=} reach(m,a)$

2. As by 1. $\leq$ is trivial, we show:

$$\begin{aligned} reach(m,a) &= reach(m, \neg\ulcorner b \cdot a + \ulcorner b \cdot a) \\ &\overset{41}{=} reach(m, \neg\ulcorner b \cdot a) + reach(m, \neg\ulcorner b \cdot a) : ((\ulcorner b \cdot a)^{\uparrow} \cdot a^{\uparrow^*}) \\ &\leq reach(m, \neg\ulcorner b \cdot a) + reach(m, a) : ((\ulcorner b \cdot a)^{\uparrow} \cdot a^{\uparrow^*}) \\ &= reach(m, \neg\ulcorner b \cdot a) \\ &\overset{54.1}{=} reach(m, b \mid a) \end{aligned}$$

□

But the previous lemma can also be lifted from *reach* to *from*so that the part reachable in the join of two stores can be simplified.

Lemma 55 (Localization II). *Assume* $(m,a) \nvdash recs(b)$*, then*

1. $from(m, a+b) = from(m,a)$
2. $from(m, b \mid a) = from(m,a)$

Proof. 1. $$\begin{aligned} from(m,a+b) &= reach(m,a+b) \cdot (a+b) \\ &\overset{54.1}{=} reach(m,a) \cdot a + reach(m,a) \cdot b \\ &\overset{53.2}{=} reach(m,a) \cdot a \\ &= from(m,a) \end{aligned}$$

2. $$\begin{aligned} from(m, b \mid a) &= reach(m, b \mid a) \cdot (b \mid a) \\ &\overset{54.2}{=} reach(m,a) \cdot b + reach(m,a) \cdot (\neg\ulcorner b \cdot a) \\ &\overset{53.3}{=} 0 + reach(m,a) \cdot (\neg\ulcorner b \cdot a) + reach(m,a) \cdot (\ulcorner b \cdot a) \\ &= from(m,a) \end{aligned}$$

□

In particular, with pointer structures $p = (m,a), q = (n,b)$ and $r = (m,b)$ we can show some of the most sophisticated rules that are needed to derive algorithms on pointer structures with selective updates.

Corollary 13. *Set* $c = (m \xrightarrow{\alpha} n) \mid b$ *and assume* $sat(\alpha) \wedge sat(\beta)$ *then*

1. $q \nvdash m \Rightarrow from(n,c) = from(q)$
2. $\alpha \cdot \beta = 0 \wedge (b_\beta(m), b) \nvdash m \Rightarrow from(c_\beta(m), c) = from(b_\beta(m), b)$

which in the notation of [27] are:

1. $q \nvdash ptr(p) \Rightarrow from((p.\alpha := q).\alpha) = from(q)$
2. $\alpha \cdot \beta = 0 \wedge r.\beta \nvdash ptr(p) \Rightarrow from((p.\alpha := q).\beta) = from(r.\beta)$

For the second proposition one needs to show that $c_\beta(m) = b_\beta(m)$ which follows from $\alpha \cdot \beta = 0$.

4.6 Correctness of pointer structures

To have an anchor for inductively defined data structures we need a special address that serves as model for nil pointers. In contrast to [19] who propose to model it by an address with all links pointing to itself we choose nil to be a special node that no link starts from. This better reflects the property that it can not be dereferenced.

Definition 29 (nil). *The special value* $\diamond$ *is an address that has no image under any store.*

$$\diamond \leq 1 \wedge \diamond^{\uparrow} = \diamond \text{ and } \diamond : a = 0 \text{ for all stores } a$$

From this definition follows that no proper addresses are reachable from $\diamond$:

Corollary 14. $reach(\diamond, a) = \diamond \qquad from(\diamond, a) = 0$

In the sequel we assume that all used stores fulfill this requirement and use it for proofs if necessary. We can also show that the definition intuitively is correct, as it implies that $\diamond$ is not in the set of allocated addresses:

Lemma 56. $recs(a) \cdot \diamond = 0$

Proof. $\diamond : a = 0 \Leftrightarrow \diamond \cdot a = 0 \Leftrightarrow \diamond \cdot \ulcorner a = 0 \Rightarrow recs(a) \cdot \diamond = \ulcorner(a^{\uparrow}) \cdot \diamond = (\ulcorner a \cdot \diamond)^{\uparrow} = 0$ □

As we have seen that the given framework enables us to model labeled graphs, we also can give the set of terminal states by calculating all reachable nodes that no further link starts from:

Definition 30 (final nodes). $final(m, a) \stackrel{\text{def}}{=} reach(m, a) \cdot \neg recs(a)$

The intuitive interpretation of final nodes - that they have no successors - immediately follows:

Corollary 15. $final(m, a) : a = final(m, a) : a^{\uparrow} = 0$

With this definition we are able to check the correctness of a store that represents concrete data structures. It is evident by definition that the reachable addresses from terminal nodes are only these nodes themselves and that $final$ is an idempotent operator

Lemma 57. *1.* $reach(final(m, a), a) = final(m, a)$
2. $final(m, from(m, a)) = final(m, a)$
3. $final(final(m, a), a) = final(m, a)$

Proof. 1. $reach(final(m, a), a) \stackrel{40.2}{=} final(m, a) + reach(final(m, a) : a^{\uparrow}, a)$
$\stackrel{Cor.15}{=} final(m, a) + reach(0, a)$
$= final(m, a)$

2. $final(m, from(m, a)) = reach(m, from(m, a)) \cdot \neg recs(from(m, a))$
$= reach(m, a) \cdot \neg recs(a)$
$= final(m, a)$

3. $final(final(m, a), a) = reach(final(m, a), a) \cdot \neg recs(a)$
$\stackrel{1.}{=} final(m, a) \cdot \neg recs(a)$
$= final(m, a)$

□

As all data representations should be terminated by nil, we can define a predicate that serves as a sort of invariant for operations on pointer structures. This says that the only final state in a pointer structure should be nil:

Definition 31 (correctness). *The store a is a correct representation of inductively defined pointer data structures if for all available entries m the condition $final(m, a) \leq \diamond$ is satisfied.*

Additionally one can demand that the store is closed and so there are no dangling links:

$$links(a) \leq recs(a) + \diamond$$

If this condition holds there are no links that point to non-allocated records but one has to assert that the record schemes match the corresponding addresses.

With respect to the store we can also define the set of sources and sinks of the graph. These are the addresses where pointer-links only start from or where they just end. With this we can define the inner nodes that have entering and leaving edges.

Definition 32 (source, sink and inner nodes).

$$\begin{aligned} src(a) &\stackrel{\text{def}}{=} recs(a) \cdot \neg links(a) \\ snk(a) &\stackrel{\text{def}}{=} links(a) \cdot \neg recs(a) \\ inner(a) &\stackrel{\text{def}}{=} recs(a) \cdot \neg src(a) = links(a) \cdot \neg snk(a) = recs(a) \cdot links(a) \end{aligned}$$

4.7 Acyclicity

A higher concept that is based on reachability is acyclicity of graphs and pointer structures. The standard way in relation algebra to define acyclicity is

Definition 33 (relational acyclicity (RA)).

$$acyclic_{RA}(a) \stackrel{\text{def}}{\Leftrightarrow} a^+ \sqcap 1 = 0$$

As there is no meet operation in EKA we have to find a different characterization. One possibility is to switch to observational equivalence. This means that the image of an arbitrary address under both sides has to be equal. So we work in the set of predicates where we have a meet (namely composition) at hand.

Definition 34 (observational acyclicity (OA)).

$$acyclic_{OA}(a) \stackrel{\text{def}}{\Leftrightarrow} \forall m.\ m \cdot (m : a^+) = 0$$

It is easy to see that this definition is quite natural by showing that it is equivalent to a reachability proposition (Note, that we assume to work in a crisp KA):

$$acyclic_{OA}(a) \Leftrightarrow \forall m.\ (m : a, a) \not\vdash m$$

Nevertheless this characterization is much stronger than acyclicity as address elements model *set* of nodes. By setting $m = 0$ one can see that $acyclic_{OA}a$ is equivalent to $a = 0$. So a logical step would be to switch to atomic address elements representing only a single node.

Definition 35 (atomic observational acyclicity (AOA)).

$$acyclic_{AOA}(a) \stackrel{\text{def}}{\Leftrightarrow} \forall at(m).\ m \cdot (m : a^+) = 0$$

An alternative characterization comes from graph theory. There one says that a graph is *progressively finite* [35] if all paths in the graph have finite length. So there are no infinite chains which says that the graph is Noetherian or well-founded.

Definition 36 (progressively finite (PF)).

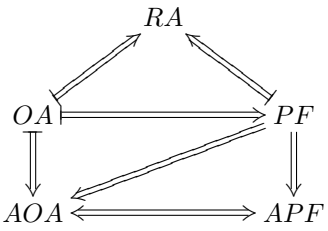
$$acyclic_{PF}(a) \stackrel{\text{def}}{\Leftrightarrow} \forall m.\ m \leq m : a^+ \Rightarrow m = 0$$

For finite graphs it is well-known that progressive finiteness and freeness of circuits are equivalent. So we also can define progressive finiteness for atoms.

Definition 37 (atomic progressively finite (APF)).

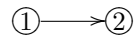
$$acyclic_{APF}(a) \stackrel{\text{def}}{\Leftrightarrow} \forall at(m).\ m \leq m : a^+ \Rightarrow m = 0$$

For these characterization candidates for acyclicity we can show the following relations:



Here an arrow with an open tail stands for an unknown connection between these two characterizations in the respective direction. A closed tail ($\mapsto$) means that the characterization on this side is strictly stronger than the one pointed to.

Proof. The unifying counter example that proves that OA neither follows from RA, AOA nor PF is the graph a with two nodes and only one connection from 1 to 2:



We choose $m = \{1, 2\}$, then $m : a^+ = \{2\}$ and $m \cdot (m : a^+) = \{2\}$. By this OA does not hold, but PF holds for all m, AOA holds for all atomic m and RA trivially holds. A counter example for $RA \Rightarrow PF$ can be found in PAT, the algebra of paths. Assume $P = \{aa\}$ a path in PAT, then $P^+ = \{aa, aaa, \ldots\}$ and therefore $P^+ \sqcap 1 = 0$. But $0 \neq \{a\} \subseteq \{a\} : P^+ = \{a\}$. For finite graphs RA and PF are equivalent (see [35]).
The implications from $OA \Rightarrow AOA$ and $PF \Rightarrow APF$ are trivial. The other implications are shown as follows:

$OA \Rightarrow PF$**:** $m = m \cdot m \leq m \cdot (m : a^+) = 0$
$AOA \Rightarrow APF$**:** Similar to $OA \Rightarrow PF$ with additional assumption $at(m)$.
$PF \Rightarrow AOA$**,**$APF \Rightarrow AOA$**:** $at(m) \Rightarrow m \cdot (m : a^+) = m \vee m \cdot (m : a^+) = 0$ and for the first term holds: $m = m \cdot (m : a^+) \leq (m : a^+) \Rightarrow m = 0$
$OA \Rightarrow RA$**:** OA holds for all addresses, so also for $m = 1$, then $0 = m \cdot (m : a^+) = 1 \cdot (1 : a^+) = (a^+)^\neg \Rightarrow a^+ = 0 \Rightarrow a^+ \sqcap 1 = 0$
$PF \Rightarrow RA$**:** $a^+ \sqcap 1 = (a^+ \sqcap 1) \cdot (a^+ \sqcap 1) \leq (a^+ \sqcap 1) \cdot a^+$

$$\Rightarrow (a^+ \sqcap 1)^\neg \leq ((a^+ \sqcap 1) \cdot a^+)^\neg$$
$$\Leftrightarrow a^+ \sqcap 1 \leq (a^+ \sqcap 1) : a^+$$
$$\Rightarrow a^+ \sqcap 1 = 0$$

In the sequel we will use characterization PF as definition of acyclicity, as OA is too strong and RA is not expressible in Kleene algebra. We note that acyclicity is a downward closed predicate:

Lemma 58. $acyclic(a) \Rightarrow \forall b \leq a.\ acyclic(b)$

Proof. Assume $m \leq m : b^+$, then $m \leq m : b^+ \leq m : a^+ \Rightarrow m = 0$ □

With the additional assumption of acyclicity we can show stronger properties of pointer algebra operation. So one can reason about reachability after having performed a step:

Lemma 59. $acyclic(a^\uparrow) \wedge m \neq 0 \Rightarrow reach(m : a, a) < reach(m, a)$

Proof. Evidently $reach(m : a, a) \leq reach(m : a^\uparrow, a) \leq m + reach(m : a^\uparrow, a) = reach(m, a)$. So assume

$$reach(m, a) \leq reach(m : a, a)$$
$$\Rightarrow m : a^{\uparrow^*} \leq (m : a) : a^{\uparrow^*} \leq (m : a^\uparrow) : a^{\uparrow^*} = (m : a^{\uparrow^*}) : a^{\uparrow^+}$$
$$\Rightarrow m : a^{\uparrow^*} = 0$$
$$\Leftrightarrow m = 0$$

which is a contradiction to $m \neq 0$. □

Standard consequences from acyclicity can also be proven. Assume an element n is in more than one step reachable from m. If the store is acyclic it follows that m is not in the part of the store reachable from n. In contrast to the corresponding lemmas in [27] we always have to demand that the involved address is not 0. This is a consequence from the set representation of addresses and assures non-emptyness.

Lemma 60. $n \neq 0 \wedge n \leq m : a^{\uparrow^+} \wedge acyclic(a^\uparrow) \Rightarrow \neg((n,a) \vdash m)$

Proof. Assume $(n,a) \vdash m$ which is equivalent to $m \leq reach(n,a)$, then

$$n \leq m : a^{\uparrow^+} \leq (n : a^{\uparrow^*}) : a^{\uparrow^+} = n : (a^{\uparrow^*} \cdot a^{\uparrow^+}) = n : a^{\uparrow^+} \overset{acycl.}{\Rightarrow} n = 0$$

which contradicts the precondition. □

If m is atomic the implication simplifies to $(n,a) \nvdash m$. By this observation specialized versions of the localization properties for singly selective updates in Corollary 13 follow from acyclicity:

Lemma 61 (Localization III). *Set $c = (m \xrightarrow{\alpha} a_\gamma(m)) \mid a)$ and assume m crisp atomic, $a_\beta(m) \neq 0$ and $acyclic(a^\uparrow)$, then*

$$from((a_\beta(m), (m \xrightarrow{\alpha} a_\beta(m)) \mid a) = from(a_\beta(m), a)$$
$$\alpha \cdot \beta = 0 \Rightarrow from(c_\beta(m), c) = from(a_\beta(m), a)$$

which again in the notation of [27] are:

$$from((p.\alpha := p.\beta).\alpha) = from(p.\beta)$$
$$\alpha \cdot \beta = 0 \Rightarrow from((p.\alpha := p.\gamma).\beta) = from(p.\beta)$$

Proof. The claims follow immediately from Lemma 60 and

$$m : a^{\uparrow^+} = m : a^\uparrow + m : (a^\uparrow \cdot a^{\uparrow^+}) \geq m : a^\uparrow \geq m : (\beta \cdot a)^\uparrow = a_\beta(m)$$

□

4.8 Sharing

Using the reachability operator from Section 4.3 we are able to define a predicate that expresses the sharing of parts of two pointer structures. As $\diamond$ is used as terminator for linked data structures it plays a special rôle. We say that two pointer structures do not share any parts if the intersection of their reachable addresses is at most $\diamond$.

Definition 38 (sharing). $\neg sharing(m,n,a) \overset{\text{def}}{\Leftrightarrow} reach(m,a) \cdot reach(n,a) \leq \diamond$

As immediate consequence from Lemma 43 follows that if two pointer structures have no nodes in common, the successor structures also do not show sharing:

Lemma 62. $\neg sharing(m,n,a) \Rightarrow \neg sharing(a_\alpha(m),n,a)$

Proof. $reach(a_\alpha(m),a) \cdot reach(n,a) \leq reach(m,a) \cdot reach(n,a) \leq \diamond$ □

By calculation with our algebra we observed, that the following lemma from [27] in fact does not need acyclicity as a precondition.

Lemma 63. *From $n \leq m : a^{\uparrow^+}$ follows $\forall o.\ \neg sharing(m,o,a) \Rightarrow \neg sharing(n,o,a)$*

Proof.

$$reach(n,a) = n : a^{\uparrow^*} \leq (m : a^{\uparrow^+}) : a^{\uparrow^*} = m : a^{\uparrow^+} \leq m + m : a^{\uparrow^+} = reach(m,a)$$

and thus $reach(m,a) \cdot reach(o,a) \leq \diamond \Rightarrow reach(n,a) \cdot reach(o,a) \leq \diamond$ □

5 Summary

We have presented an extension of Kleene algebra that can be used to model the concurrent treatment of several equally shaped KAs. Calculations like the transitive closure there can be performed simultaneously on all involved KAs. Afterwards it is possible get the result in the context of a specific KA by projection. As application we have shown how EKAs can be used as a formal basis for pointer algebra. Future tasks are the investigation of an equational axiomatization based on action algebra and the application of pointer algebra to larger problems like for example garbage collection algorithms.

6 Acknowledgement

I would like to thank B. Möller, G. Struth and M. Winter for valuable critic and discussion.

References

1. C.J. Aarts. Galois connections presented calculationally. Afstudeer verslag (Graduating Dissertation), Department of Computing Science, Eindhoven University of Technology, July 1992.
2. R. Backhouse. Galois connections and fixed point calculus. In *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction International Summer School and Workshop, Oxford, UK, April 10-14, 2000, Revised Lectures*, volume 2297 of *Lecture Notes in Computer Science*, pages 89–148. Springer-Verlag, 2002.
3. A. Bijlsma. Calculating with pointers. *Science of Computer Programming*, 12(3):191–206, September 1989.
4. R. Bornat. Proving pointer programs in Hoare logic. In R. Backhouse and J.N. Oliveira, editors, *Mathematics of Program Construction, 5th International Conference, MPC 2000*, volume 1837 of *Lecture Notes in Computer Science*, pages 102–126. Springer-Verlag, 2000.
5. C. Brink, K. Britz, and R.A. Schmidt. Peirce algebras. *Formal Aspects of Computing*, 6:1–20, 1994.
6. R.M. Burstall. Some techniques for proving correctness of programs which alter data structures. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 7*, pages 23–50. Edinburgh University Press, Edinburgh, Scotland, 1972.
7. M. Butler. Calculational derivation of pointer algorithms from tree operations. *Science of Computer Programming*, 33(3):221–260, March 1999.
8. J.H. Conway. *Regular Algebra and Finite Machines*. Chapman & Hall, London, 1971.
9. A. de Morgan. On the syllogism, no. iv, and on the logic of relations. *Transactions of the Cambridge Philosophical Society*, 10:331–358, 1864.
10. J. Desharnais and B. Möller. Characterizing determinacy in Kleene algebras. In J. Desharnais, M. Frappier, A. Jaoua, and W. MacCaull, editors, *Relational Methods in Computer Science. Int. Seminar on Relational Methods in Computer Science, Jan 9–14, 2000 in Québec*, volume 139 of *Information Sciences — An International Journal*, pages 153–273, 2001.
11. J. Desharnais, B. Möller, and G. Struth. Kleene algebra with a domain operator. Technical report 2003-7, Institut für Informatik, Universität Augsburg, 2003.
12. J. Desharnais, B. Möller, and F. Tchier. Kleene under a Demonic Star. In T. Rus, editor, *Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000*, volume 1816 of *Lecture Notes in Computer Science*, pages 355–370. Springer-Verlag, 2000.
13. T. Ehm. Properties of overwriting for updates in typed Kleene algebras. Technical report 2000-7, Institut für Informatik, Universität Augsburg, 2000.
14. T. Ehm. Pointer Kleene Algebra. Submitted to RelMiCS, 2003.
15. T. Ehm, B. Möller, and G. Struth. Kleene modules. Submitted to RelMiCS, 2003.

16. P.J. Freyd and A. Scedrov. *Categories, Allegories*, volume 39 of *North-Holland Mathematical Library*. North-Holland, Amsterdam, 1990.
17. L. Henkin, J.D. Monk, and A. Tarski. *Cylindric Algebras I*, volume 64 of *Studies in logic and the foundations of mathematics*. North-Holland, 1971.
18. C.A.R. Hoare. Proofs of correctness of data representation. *Acta Informatica*, 1:271–281, 1972.
19. C.A.R. Hoare and H. Jifeng. A trace model for pointers and objects. In R. Guerraoui, editor, *ECCOP'99 - Object-Oriented Programming, 13th European Conference, Lisbon, Portugal, June 14-18, 1999, Proceedings*, volume 1628 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 1999.
20. B. Jónsson and A. Tarski. Boolean algebras with operators, Part I. *American Journal of Mathematics*, 73:891–939, 1951.
21. B. Jónsson and A. Tarski. Boolean algebras with operators, Part II. *American Journal of Mathematics*, 74:127–167, 1952.
22. Y. Kawahara and H. Furusawa. Crispness and representation theorems in Dedekind categories. Technical report DOI-TR 143, Kyushu University, 1997.
23. G.J. Klir and T.A. Folger. *Fuzzy Sets, Uncertainty and Information*. Prentice Hall International, Englewood Cliffs (NJ), 1988.
24. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. Technical report TR90-1123, Cornell University, Computer Science Department, May 1990.
25. D. Kozen. On action algebras. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 78–88. MIT Press, 1994.
26. D. Kozen and F. Smith. Kleene algebra with tests: Completeness and decidability. Technical report TR96-1582, Cornell University, Computer Science Department, April 1996.
27. B. Möller. Calculating with pointer structures. In R. Bird and L. Meertens, editors, *Algorithmic Languages and Calculi*, pages 24–48. Proc. IFIP TC2/WG2.1 Working Conference, Le Bischenberg, Feb. 1997, Chapman & Hall, 1997.
28. B. Möller. Linked Lists Calculated. Technical report 1997-7, Institut für Informatik, Universität Augsburg, 1997.
29. B. Möller. Calculating with acyclic and cyclic lists. In A. Jaoua and G. Schmidt, editors, *Relational Methods in Computer Science. Int. Seminar on Relational Methods in Computer Science, Jan 6–10, 1997 in Hammamet*, volume 119 of *Information Sciences — An International Journal*, pages 135–154, 1999.
30. B. Möller. Typed Kleene Algebras. Technical report 1999-8, Institut für Informatik, Universität Augsburg, 1999.
31. J.M. Morris. Assignment and linked data structures. In *Theoretical Foundations of Programming Methodology*, volume 91 of *NATO Advanced Study Institutes Series C Mathematical and Physical Sciences*, pages 35–51. Dordrecht, Reidel, 1981.
32. V. Pratt. Action logic and pure induction. In J. van Benthem and J. Eijck, editors, *Proceedings of JELIA-90, European Workshop on Logics in AI*, Amsterdam, September 1990.
33. V. Pratt. Dynamic Algebras as a well-behaved fragment of Relation Algebras. In C.H. Bergman, R.D. Maddux, and D.L. Pigozzi, editors, *Algebraic Logic and Universal Algebra in Computer Science*, volume 425 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.
34. J.C. Reynolds. Intuitionistic reasoning about shared mutable data structure. In J. Davies, B. Roscoe, and J. Woodcock, editors, *Millennial Perspectives in Computer Science*, pages 303–321, Houndsmill, Hampshire, 2000. Palgrave.
35. G. Schmidt and T. Ströhlein. *Relations and Graphs, Discrete Mathematics for Computer Scientists*. EATCS-Monographs on Theoretical Computer Science. Springer-Verlag, 1993.
36. M. Winter. Relational constructions in Goguen categories. In H. de Swart, editor, *6th International Seminar on Relational Methods in Computer Science (RelMiCS)*, pages 222–236, 2001.

A Standard Kleene Algebra

Definition 39 (SKA). *A standard Kleene algebra is a sixtuple* $(\mathcal{K}, \leq, \top, \cdot, 0, 1)$ *satisfying the following properties:*

1. $(\mathcal{K}, \leq)$ *is a complete lattice with least element* 0 *and greatest element* $\top$*. The supremum of a subset* $L \subseteq \mathcal{K}$ *is denoted by* $\sum L$*.*
2. $(\mathcal{K}, \cdot, 1)$ *is a monoid.*
3. *The operation* $\cdot$ *is universally disjunctive (i.e. distributes through arbitrary suprema) in both arguments.*

We only summarize the important laws that hold in SKAs due to the existence of a meet operator.

Lemma 64. *Consider a SKA and* $s, t \in \mathcal{P}$*.*

1. $s \cdot t = s \sqcap t$
2. $s \cdot (a \sqcap b) = s \cdot a \sqcap s \cdot b$
3. $(s \sqcap t) \cdot a = s \cdot a \sqcap t \cdot a$
4. $s \cdot a \sqcap \neg s \cdot b = 0$
5. $a \sqcap s \cdot b = s \cdot a \sqcap s \cdot b$
   In particular: $a \sqcap s \cdot \top = s \cdot a$