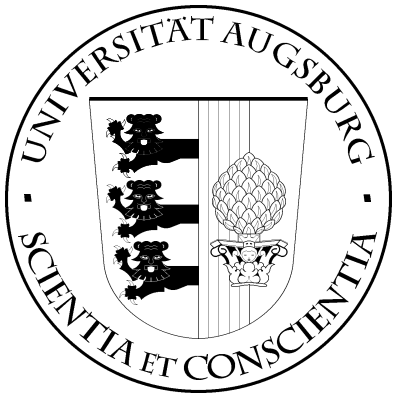
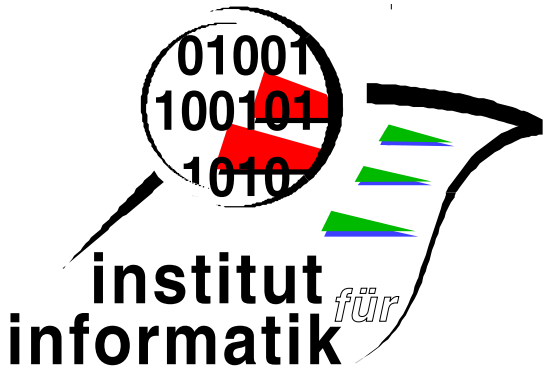
Universität Augsburg



Properties of Overwriting for Updates in Typed Kleene Algebras

Thorsten Ehm

Report 2000-7 Dezember 2000



Institut für Informatik
D-86135 Augsburg

Properties of Overwriting for Updates in Typed Kleene Algebras

Thorsten Ehm

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

Abstract. In this paper we present an abstract representation of pointer structures in Kleene algebras and the properties of a particular selective update function. These can be used as prerequisites for the definition of in-situ pointer updates and a general framework to derive in-situ pointer algorithms from their specification.
Keywords: pointer, overwriting, Kleene algebra, relational algebra.

1 Introduction

Although pointers are frequently used in implementations of software, there are only a few methods to formally describe them [2,3]. This paper is part of a project to improve upon this situation. To obtain this goal we use a relational framework to model pointer structures as described in [8]. In this approach algorithms on complex data structures are transformed from their functional specification to an executable pointer implementation. Such a process often contains a number of degrees of freedom in the choice of transformation rules. So the resulting pointer algorithms may vary a lot, depending on the applied transformation steps. But often there are certain implicit requirements which have to be fulfilled by the pointer algorithm, such as reusing memory as much as possible. That particular demand is described in this paper by the notion of in-situ update of an data structure.

The final goal of our project is to derive - possibly automatically - in-situ pointer implementation from functional specifications. To define in-situ updates on pointer structures it is necessary to know which operation changes the given structure as little as possible but preserves the specified properties. This means we have to find orders and lattices, respectively least and greatest elements of certain sets. To achieve this goal we examine the selective update function $|$, which is the only operation capable of changing pointers in our relational model. Particularly, in this paper we are interested in extremal solutions of the equation $x \mid a = b$. The model used can be abstracted to the level of Boolean Kleene algebras.

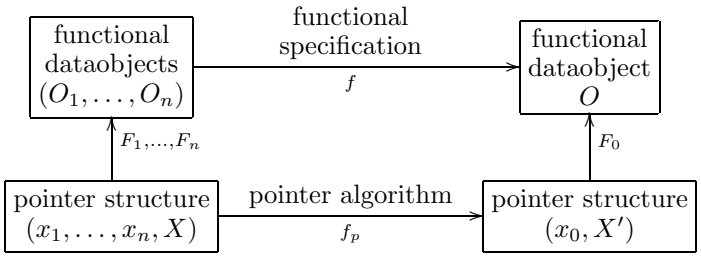
This paper is structured as follows: Section 2 defines pointer structures and how one can derive a pointer algorithm from a functional one. A definition of Kleene Algebra and some rules are introduced in Section 3. Section 4 presents the notions of domain in Typed Kleene Algebra and the concept of local composition. Differences as used in Boolean Algebras, such as for example set theory, are introduced in Section 5. The very important concept of modelling overwriting in Kleene Algebras is handled in Section 6. Essential parts of this paper are in Section 7. There a minimal solution for updates in Kleene Algebras is presented. In Section 8 an application of these results to maps is shown and Section 9 gives a short summary of the whole paper. To make this paper self-contained but also as readable as possible we defer all the proofs to Appendix A.

2 Pointer structures

Pointer structures play an essential role in all programming tasks. Though there are efforts to hide explicit use of pointers as for example in Java, they are present in almost all implementations of software. Complex data structures in object oriented programming languages are pointer structures as well as dataflow dependencies in compilers and so on.

To treat pointer structures formally we make the used memory an explicit parameter in all the calculations. So in our model a pointer structure consists of a store and a list of entries. A store is a family of relations between records (represented by their initial addresses $\mathcal{A}$) or between records and node values $\mathcal{N}_j$. Each relation represents a selector on the records like e.g. *head* and *tail* for lists with functionality $\mathcal{A} \to \mathcal{N}_j$ respectively $\mathcal{A} \to \mathcal{A}$. The entries of a pointer structure are addresses that form starting points of the modelled data structures.

Each abstract object implemented by pointer structures is represented by a pointer structure (n, P) with a single entry $n \in \mathcal{A}$. The relation between abstract and concrete levels is established by a partial abstraction function F as described in [7]. The pointer implementation f_p of a given functional operation f is now specified by the equation $f(F(p)) = F(f_p(p))$.



To derive a pointer implementation f_p from this specification one tries to transform $f(F(p))$ by equational reasoning into an expression $F(E)$ such that E does not contain F. Then we can define f_p by setting $f_p(p) = E$.

Our future goal is now to make available a set of transformation rules that leads to an in-situ pointer algorithm of the specified operation.

3 Kleene Algebras

A Kleene algebra is an algebraic structure to describe sets endowed with an ordering and a composition operation. We will use a definition first introduced in [4].

Definition 1. *A* Kleene algebra *(KA) is a sixtuple* $(K, \leq, \top, \cdot, 0, 1)$ *satisfying the following laws:*

1. $(K, \leq)$ *forms a complete lattice with least element* 0 *and greatest element* $\top$.
2. $(K, \cdot, 1)$ *is a monoid.*
3. *The operation* $\cdot$ *is universally disjunctive (i.e. distributes through arbitrary suprema) in both arguments.*

Prominent candidates of KAs are for example the algebra of formal languages

$$\mathrm{LAN} = (\mathcal{P}(A^*), \subseteq, A^*, \cdot, \emptyset, \epsilon)$$

over some alphabet A or the algebra PAT of path sets in a directed graph. More important for our goal is the algebra

$$\mathrm{REL} = (\mathcal{P}(M \times M), \subseteq, M \times M, ;, \emptyset, I)$$

of homogenous binary relations over some set M. There we can model pointers as (atomic) relations between addresses. Concrete relation algebras can be lifted to the concept of abstract relation algebra [10].

Definition 2. *An* abstract relation algebra *is a tuple* $RA = (N, \leq, \top, ;, 0, 1, \bar{}, ^\smile)$ *where*

1. $(N, \leq, \bar{}, 0, \top)$ *is a Boolean algebra.*
2. $(N, ;, 1)$ *is a monoid.*
3. *Tarski's rule* $x \neq 0 \Rightarrow \top; x; \top = \top$ *holds.*
4. *Dedekind's rule* $x; y \sqcap z \leq (x \sqcap z; y^\smile); (y \sqcap x^\smile; z)$ *is satisfied.*

Such a RA is a special case of a Boolean KA. We call a KA *Boolean*, if its underlying lattice is a Boolean lattice. The complement of an element $a \in K$ is denoted by $\overline{a}$. In the sequel we will exclusively use such Boolean KAs, and all the proven properties are inherited by relation algebras. So we can do all our calculations and reasoning over pointer structures in Boolean KAs.

3.1 Types

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. In our context the domain corresponds to the set of starting addresses of pointers or in other words to the allocated records in the store. In othere disciplines these elements are also called partial identities, monotypes or coreflexives. Though we will use the notion *t*ype as in [9].

Definition 3. *A* type *of a Kleene algebra is an element* t *with* $t \leq 1$.

We will use $\mathrm{TYP} = \{t : t \leq 1\}$ as synonym for the set of all types in a KA. Types in Kleene algebras are idempotent wrt. $\cdot$. The meet of two types coincides with $\cdot$ (see Lemma 1.1). To have a complement for elements in TYP we define

Definition 4. *The negation of a type* $t \in TYP$ *is* $\neg t = \overline{t} \sqcap 1$.

If appropriate we shall use this operation also for elements not in TYP. There are some properties which are only valid for elements of TYP.

Lemma 1. *Consider a typed KA and* $s, t \in TYP$.

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

4 Domain and Locality of Composition

To model the region pointers start from we introduce the notion of a domain. The domain is defined using Galois connections because there is no possibility in KAs to access the components of elements.

Definition 5. *(see also [1])* $\ulcorner a \leq y \Leftrightarrow a \leq y \cdot \top$

Note that y only ranges over types. This definition implies that the domain operator is universally disjunctive, strict and monotonic. The domain operator maps an element to the representation of its domain in TYP. Domains are often used to restrict elements to certain subelements. This is done by simply composing the restriction

domain with the element. But an implication of Lemma 1.3 is, that the two notions of restriction ($t \cdot a$ and $a \sqcap t \cdot \top$) to a type t are equivalent.

In general Kleene algebras one can only prove that $\ulcorner(a \cdot b) \leq \ulcorner(a \cdot \ulcorner b)$. The converse inequation does not follow from the axiomatisation. But all the examples given in Section 3 and all reasonable KAs satisfy this property. So we define:

Definition 6. *A KA has* left-local composition *if it satisfies*

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

Then the following Lemma holds:

Lemma 2. *1. A KA has left-local composition iff it satisfies* $\ulcorner(a \cdot b) = \ulcorner(a \cdot \ulcorner b)$
2. If a KA has left-local composition then $\ulcorner(\ulcorner a \cdot b) = \ulcorner a \sqcap \ulcorner b = \ulcorner a \cdot \ulcorner b$

We summarize some properties in the context of domain and left-local composition needed later:

Lemma 3. *Let* $t \in TYP$.

1. $\ulcorner a \cdot a = a$
2. $a \leq b \Rightarrow \ulcorner b \cdot a = a$
3. $\ulcorner t = t$
4. $\ulcorner(a + b) = \ulcorner a + \ulcorner b$
5. $\ulcorner(a \sqcap b) \leq \ulcorner a \sqcap \ulcorner b$
6. $\neg\ulcorner a \cdot a = 0$
7. $\neg\ulcorner a \cdot \overline{a} = \neg\ulcorner a \cdot \top$
8. $\neg\ulcorner a \cdot b \leq \neg\ulcorner a \cdot \overline{a}$
9. $\neg\ulcorner a \cdot \neg\ulcorner b = \neg(\ulcorner a + \ulcorner b)$
10. $\overline{t \cdot a} = \neg t \cdot a + \overline{a} = \neg t \cdot \top + t \cdot \overline{a}$
11. $\ulcorner(t \cdot a) = t \cdot \ulcorner a$

In the sequel we presume that all KAs treated in this paper have left-local composition.

5 Differences

To describe updates that change the given structure as little as possible it is necessary to have a "metric" on the considered elements. Such a metric has to express the difference between input and output elements. Later steps in the project have to compare different solutions and figure out the best possibly by using such a metric. But in this paper we are interested in extremal solutions of a particular update equation and so we use orders defined by differences instead of a metric.

To model simple differences in Boolean KAs we use the well-known definition from often used Boolean algebras such as set theory.

Definition 7. *The* difference *between a and b in a Boolean KA is defined by*

$$a \setminus b = a \sqcap \overline{b}$$

This operation is left distributive and right anti-distributive, but neither associative nor commutative. The following properties not using types or domains hold in all Boolean algebras.

Lemma 4.
1. $(a + b) \setminus c = a \setminus c + b \setminus c$
2. $(a \sqcap b) \setminus c = a \setminus c \sqcap b \setminus c$
3. $a \setminus (b + c) = a \setminus b \sqcap a \setminus c$
4. $a \setminus (b \sqcap c) = a \setminus b + a \setminus c$
5. $(a \setminus b) \setminus c = a \setminus (b + c)$
6. $a \setminus (b \setminus c) = a \setminus b + (a \sqcap c)$
7. $a = (a \sqcap b) + a \setminus b$
8. $a \setminus (t \cdot a) = \neg t \cdot a$
9. $(t \cdot a) \setminus a = 0$
10. $t \setminus \ulcorner a = t \cdot \neg\ulcorner a$
11. $\neg\ulcorner a \cdot (b \setminus a) = \neg\ulcorner a \cdot b$

5.1 The symmetric difference

If two elements are treated equally in the calculation of a difference we can use the symmetric difference. There are all elements not in the meet of a and b taken into account.

Definition 8. *The symmetric difference in a Boolean KA is defined as*

$$a \Delta b = a \setminus b + b \setminus a$$

All the properties of Δ are also inherited from Boolean algebra. Some of the most needed are:

Lemma 5.
1. $a \Delta b = (a + b) \setminus (a \sqcap b)$
2. $a \Delta 0 = a$
3. $a \setminus b \leq a \Delta b$
4. $(a \Delta b) \Delta c = a \Delta (b \Delta c)$
5. $a \sqcap (b \Delta c) = (a \sqcap b) \Delta (a \sqcap c)$
6. $a = b \Leftrightarrow a \Delta b = 0$

6 Overwriting

Overwriting is the only operation which alters the structure in our pointer model. It is used to model selective updates on entire domains. The area outside the domain is not touched.

Definition 9. *The overwriting "a on b" is defined as*

$$a \mid b = a + \neg \ulcorner a \cdot b$$

Lemma 6.
1. $a \leq a \mid b$
2. $a = \ulcorner a \cdot (a \mid b)$
3. $a \mid a = a$
4. $a \mid (b + c) = a \mid b + a \mid c$
5. $\ulcorner (a \mid b) = \ulcorner a + \ulcorner b$
6. $a \Delta (b \mid a) = \ulcorner b \cdot (a \Delta b)$

In the context of selective overwriting and in-situ updates on pointer structures we are particularly interested in elements changing the given structure as little as possible. So we are searching for minimal and maximal solutions of $x \mid a = b$. This represents a sort of left residue for the update operator.

7 Equivalence on the same domain

In our approach overwriting is performed on the entire domain range of the overwriting element (see definition in Section 6). This means, that it is not possible to change some and simultaneously preserve other subelements which have the same domain. This gives us the occasion to define a set $\mathcal{E}_a$ describing all elements less than a which contain all elements of a restricted to the domain of this element.

Definition 10. $\mathcal{E}_a = \{x : x \leq a \wedge \ulcorner x \cdot a = x\}$

Note that, by monotony $x \leq a \Rightarrow x \leq \ulcorner x \cdot a$ holds anyway. An important observation is:

Lemma 7. *$\mathcal{E}_a$ forms a complete lattice.*

The definition of $\mathcal{E}_a$ singles out "maximal" subelements of a. In general it is not possible to make propositions about the restriction to an arbitrary subdomain. But this sort of completeness of $\mathcal{E}_a$ in the domain leads to some properties that only hold for elements in $\mathcal{E}_a$:

Lemma 8. *Let* $x \in \mathcal{E}_a$

1. $\ulcorner x \cdot a = x$
2. $a \setminus x = \neg \ulcorner x \cdot a$
3. $\ulcorner x \cdot (a \setminus x) = 0$

In our pointer model this means, that all pointers in a that start at the same address as any pointer in $x \in \mathcal{E}_a$ are also in x. Considering overwriting and the equation $x \mid a = b$. The parts of a and b that are equal under the restriction to a certain domain are the essential parts that are not changed by an update with a minimal element x. This observation leads to the

Definition 11. *The set of domain equivalent subelements with regard to* a *and* b *is defined as:*

$$\mathcal{D}_{a,b} = \mathcal{E}_a \cap \mathcal{E}_b = \{x : x \leq a \sqcap b \wedge \ulcorner x \cdot a = x = \ulcorner x \cdot b\}$$

For example in PAT $\mathcal{D}_{a,b}$ are the paths p that are in a and b and satisfy that all paths in $a \cap b$ starting at the same node as p are also in $\mathcal{D}_{a,b}$. By definition of $\mathcal{D}_{a,b}$ we observe that

Lemma 9. $\mathcal{D}_{a,b}$ *forms a complete lattice*

as well. This provides us with the foundation to speak of minimum and maximum elements in $\mathcal{D}_{a,b}$. In the sequel we are particularly interested in the supremum of $\mathcal{D}_{a,b}$. This is the greatest element that coincides on its domain with a and b and therefore is the part that does not have to be overwritten to obtain b from a.

Definition 12. $(a \otimes b) = \bigsqcup \mathcal{D}_{a,b} = \bigsqcup \{x : x \leq a \sqcap b \wedge \ulcorner x \cdot a = x = \ulcorner x \cdot b\}$

This definition is well-formed since by Lemma 9 the supremum exists for all subsets of $\mathcal{D}_{a,b}$. The elements of $\mathcal{D}_{a,b}$ inherit all the laws given in Lemma 8 not only for a but also symmetrically for b.

7.1 Heading for a closed formula

One major drawback is, that calculating with the definition for domain equivalent subsets is very unwieldy. This lies in the set theoretic definition of $\mathcal{D}_{a,b}$ and the supremum calculation for $(a \otimes b)$. A closed formula for the greatest domain equivalent subset would be a remedy.

The analysis of KAs given in Section 3 and Lemma 11 led us to suppose that there is a close relation between the symmetric difference and domain equivalent subsets. In fact we can prove

Lemma 10. $(a \otimes b) = \neg \ulcorner (a \Delta b) \cdot (a \sqcap b)$

In addition the difference between b and $(a \otimes b)$ can be calculated simply by restricting b to the domain of a symmetric difference.

Lemma 11.
$b \setminus (a \otimes b) = \ulcorner (a \Delta b) \cdot b$

This rule also holds symmetrically for a.

Corollary 1. $a \leq b \Rightarrow a \otimes b = \neg \ulcorner (b \setminus a) \cdot a$

This is the foundation for the following investigation of overwriting in KAs.

By the definition of $\mid$ we observe that the result of an update has to be greater than the overwritten element. Even more, we can show:

Lemma 12. *Let* $\mathcal{X}$ *be the set* $\{x : x \mid a = b\}$. *Then*

1. $\mathcal{X} \neq \emptyset$ *iff* $\ulcorner a \leq \ulcorner b$
2. $\ulcorner a \leq \ulcorner b \Rightarrow b = \max \mathcal{X}$
3. $\ulcorner a \leq \ulcorner b \Rightarrow b \setminus (a \otimes b) = \min \mathcal{X}$

These results make us confident of finding a calculus to determine solutions of equations containing the update operator and will be investigated further.

8 Maps

Since we are heading for a general calculus for selective pointer updates, we are mainly interested in functional correspondences, namely maps, as it does not make sense for a pointer to point to two different objects.

There are several ways to characterize maps in Kleene algebra. In this paper we will use a domain-oriented approach described in [5]:

Definition 13. *a is a map* $\overset{def}{\Leftrightarrow} \forall b : b \leq a \Rightarrow b = \ulcorner b \cdot a$

This is equivalent to the statement $\mathcal{E}_a = \{x : x \leq a\}$. For maps in Kleene algebras some of the common properties are true:

Lemma 13. *1. a map $\wedge\, b \leq a \Rightarrow b$ map*
2. a map $\Rightarrow a \sqcap b$ map
3. a, b map $\Rightarrow a \mid b$ map

With the previous results we can give an extreme solution for $x \mid a = b$ with a and b maps.

Lemma 14. *Assume that a and b are maps.*

1. $a \otimes b = a \sqcap b$
2. The least solution of $x \mid a = b$ is $b \setminus a$ if $\ulcorner a \leq \ulcorner b$

This raises our hope of achieving a calculus and a set of transformation rules for deriving in-situ pointer updates.

9 Summary

We have presented some properties of a particular selective update function. These properties were used to investigate minimal and maximal solutions of $x \mid a = b$. In the future these results will be used to define orders on pointer algorithms to achieve a formal notion of in-situ updates.

10 Acknowledgment

I am grateful to Bernhard Möller for his previous work on pointer structures and Kleene algebras and his critical remarks and numerous suggestions for improvement of this paper.

References

1. C.J. Aarts: *Galois connections presented calculationally*. Eindhoven University of Technology, Dept. of Mathematics and Computer Science, July 1992
2. A. Bijlsma: *Calculating with pointers*. Science of Computer Programming **12**, Elsevier 1989, 191–205
3. M. Butler: *Calculational derivation of pointer algorithms from tree operations*. Science of Computer Programming **33**, Elsevier 1999, 221–260
4. J.H. Conway: *Regular algebra and finite machines*. London: Chapman and Hall 1971
5. J. Desharnais, B. Möller: *Characterizing determinacy in Kleene Algebras*. Proc. RelMiCS 2000, Québec, Technical Report 2000-5, Institut für Informatik, Universität Augsburg.
6. J. Desharnais, B. Möller: *Kleene under a demonic star*. Technical Report 2000-3, Institut für Informatik, Universität Augsburg.

7. C. A. R. Hoare: *Proofs of correctness of data representations.* Acta Informatica **1**, 1972, 271-281
8. B. Möller: *Calculating with pointer structures.* In: R. Bird, L. Meertens (eds.): Algorithmic languages and calculi. Proc. IFIP TC2/WG2.1 Working conference, Le Bischenberg, Feb. 1997. Chapman & Hall 1997, 24–48
9. B. Möller: *Typed Kleene Algebras.* Technical Report 1999-8, Institut für Informatik, Universität Augsburg.
`http://www.informatik.uni-augsburg.de/skripts/techreports`
10. G. Schmidt, T. Ströhlein. *Relations and graphs.* EATCS Monographs in Computer Science, Springer-Verlag, Berlin, 1993

A Proofs

Lemma 1

1. - $s \cdot t = s \cdot t \sqcap s \cdot t \leq s \sqcap t$
   - $s \sqcap t = (s \sqcap t) \cdot (s \sqcap t) \leq s \cdot t$
2. $$\begin{aligned} t \cdot (a \sqcap b) &= t \cdot (a \sqcap b) \sqcap t \cdot (a \sqcap b) \\ &\leq t \cdot a \sqcap t \cdot b \\ &= t \cdot ((a \sqcap b) + (a \sqcap \overline{b})) \sqcap t \cdot ((\overline{a} \sqcap b) + (a \sqcap b)) \\ &= (t \cdot (a \sqcap b) + t \cdot (a \sqcap \overline{b})) \sqcap (t \cdot (\overline{a} \sqcap b) + t \cdot (a \sqcap b)) \\ &= (t \cdot (a \sqcap b) \sqcap t \cdot (\overline{a} \sqcap b)) + t \cdot (a \sqcap b) + (t \cdot (a \sqcap \overline{b}) \sqcap t \cdot (\overline{a} \sqcap b)) + \\ &\quad (t \cdot (a \sqcap \overline{b}) \sqcap t \cdot (a \sqcap b)) \\ &\leq (a \sqcap b \sqcap \overline{a} \sqcap b) + t \cdot (a \sqcap b) + (a \sqcap \overline{b} \sqcap \overline{a} \sqcap b) + (a \sqcap \overline{b} \sqcap a \sqcap b) \\ &= t \cdot (a \sqcap b) \end{aligned}$$
3. $$\begin{aligned} (s \sqcap t) \cdot a &\leq s \cdot a \sqcap t \cdot a \\ &= (s + \neg s) \cdot (s \cdot a \sqcap t \cdot a) \\ &= s \cdot (s \cdot a \sqcap t \cdot a) + \neg s \cdot (s \cdot a \sqcap t \cdot a) \\ &\leq s \cdot t \cdot a + \neg s \cdot s \cdot a \\ &= s \cdot t \cdot a \\ &= (s \sqcap t) \cdot a \end{aligned}$$
4. $t \cdot a \sqcap \neg t \cdot b \leq t \cdot (a + b) \sqcap \neg t \cdot (a + b) = (t \sqcap \neg t) \cdot (a + b) = 0$
5. Follows from De Morgan.
6. $$\begin{aligned} a \sqcap t \cdot b &= (t + \neg t) \cdot a \sqcap t \cdot b \\ &= (t \cdot a + \neg t \cdot a) \sqcap t \cdot b \\ &= (t \cdot a \sqcap t \cdot b) + (\neg t \cdot a \sqcap t \cdot b) \\ &= t \cdot a \sqcap t \cdot b \end{aligned}$$

Lemma 2

1. - Assume left-local composition. Then $\ulcorner b = \ulcorner(\ulcorner b) \Rightarrow \ulcorner(a \cdot b) = \ulcorner(a \cdot \ulcorner b)$
   - Assume lemma holds and $\ulcorner b = \ulcorner c \Rightarrow \ulcorner(a \cdot b) = \ulcorner(a \cdot \ulcorner b) = \ulcorner(a \cdot \ulcorner c) = \ulcorner(a \cdot c)$

2. $\ulcorner(\ulcorner a \cdot b) = \ulcorner(\ulcorner a \cdot \ulcorner b) = \ulcorner(\ulcorner a \sqcap \ulcorner b) = \ulcorner a \sqcap \ulcorner b$

Lemma 3

1. - $\ulcorner a \cdot a \leq 1 \cdot a = a$
   - $a = a \sqcap a \leq a \sqcap \ulcorner a \cdot \top = \ulcorner a \cdot a$
2. $a = \ulcorner a \cdot a \leq \ulcorner b \cdot a \leq a$
3. $t \leq t \cdot \top \Rightarrow \ulcorner t \leq t$ and $t = \ulcorner t \cdot t \leq \ulcorner t$
4. Follows from $\ulcorner$ is universally disjunctive
5. Follows from monotony of $\ulcorner$.
6. $\neg\ulcorner a \cdot a = \neg\ulcorner a \cdot \ulcorner a \cdot a = 0 \cdot a = 0$
7. $\neg\ulcorner a \cdot \top = \neg\ulcorner a \cdot (\overline{a} + a) = \neg\ulcorner a \cdot \overline{a} + \neg\ulcorner a \cdot a = \neg\ulcorner a \cdot \overline{a}$
8. $\neg\ulcorner a \cdot b \leq \neg\ulcorner a \cdot \top = \neg\ulcorner a \cdot \overline{a}$
9. Follows from Lemma 1.5.
10. - $t \cdot a \sqcap (\neg t \cdot a + \overline{a}) = (t \cdot a \sqcap \neg t \cdot a) + (t \cdot a \sqcap \overline{a}) = 0$
    - $t \cdot a + (\neg t \cdot a + \overline{a}) = (t \cdot a + \neg t \cdot a) + \overline{a} = a + \overline{a} = \top$
      This proves the first equivalence
    - $\neg t \cdot \top + t \cdot \overline{a} = \neg t \cdot (a + \overline{a}) + t \cdot \overline{a} = \neg t \cdot a + \neg t \cdot \overline{a} + t \cdot \overline{a} = \neg t \cdot a + \overline{a}$
11. $\ulcorner(t \cdot a) = \ulcorner(t \cdot \ulcorner a) = \ulcorner t \cdot \ulcorner a = t \cdot \ulcorner a$

Lemma 4

1. $(a + b) \setminus c = (a + b) \sqcap \overline{c} = (a \sqcap \overline{c}) + (b \sqcap \overline{c}) = a \setminus c + b \setminus c$
2. $(a \sqcap b) \setminus c = (a \sqcap b) \sqcap \overline{c} = a \sqcap \overline{c} \sqcap b \sqcap \overline{c} = a \setminus c \sqcap b \setminus c$
3. $a \setminus (b + c) = a \sqcap \overline{(b + c)} = a \sqcap (\overline{b} \sqcap \overline{c}) = (a \sqcap \overline{b}) \sqcap (a \sqcap \overline{c}) = a \setminus b \sqcap a \setminus c$
4. $a \setminus (b \sqcap c) = a \sqcap \overline{(b \sqcap c)} = a \sqcap (\overline{b} + \overline{c}) = (a \sqcap \overline{b}) + (a \sqcap \overline{c}) = a \setminus b + a \setminus c$
5. $(a \setminus b) \setminus c = (a \sqcap \overline{b}) \sqcap \overline{c} = a \sqcap (\overline{b} \sqcap \overline{c}) = a \sqcap \overline{(b + c)} = a \setminus (b + c)$
6. $a \setminus (b \setminus c) = a \sqcap \overline{(b \sqcap \overline{c})} = a \sqcap (\overline{b} + c) = a \setminus b + a \sqcap c$
7. $(a \sqcap b) + a \setminus b = (a \sqcap b) + (a \sqcap \overline{b}) = a \sqcap (b + \overline{b}) = a \sqcap \top = a$
8. $a \setminus (t \cdot a) = a \sqcap \overline{t \cdot a} = a \sqcap (\neg t \cdot a + \overline{a}) = (a \sqcap \neg t \cdot a) + (a \sqcap \overline{a}) = \neg t \cdot a$
9. $(t \cdot a) \setminus a = t \cdot a \sqcap \overline{a} = t \cdot (a \sqcap \overline{a}) = 0$
10. $t \cdot \neg\ulcorner a = t \sqcap \neg\ulcorner a = t \sqcap (\overline{\ulcorner a} \sqcap 1) = t \sqcap 1 \sqcap \overline{\ulcorner a} = t \sqcap \overline{\ulcorner a} = t \setminus \ulcorner a$
11. $\neg\ulcorner a \cdot (b \setminus a) = \neg\ulcorner a \cdot (b \sqcap \overline{a}) = \neg\ulcorner a \cdot b \sqcap \neg\ulcorner a \cdot \overline{a} = \neg\ulcorner a \cdot b$

Lemma 5

1. $a \Delta b = a \setminus b + b \setminus a = (a \sqcap \overline{b}) + (b \sqcap \overline{a}) = (a + b) \sqcap (\overline{a} + \overline{b}) = (a + b) \sqcap \overline{(a \sqcap b)} = (a + b) \setminus (a \sqcap b)$
2. $a \Delta 0 = a \setminus 0 + 0 \setminus a = a + 0 = a$
3. Immediate from the definition
4. First we prove two auxiliary lemmata:

   $$c \setminus (a \Delta b) = c \setminus ((a + b) \setminus (a \sqcap b)) = c \setminus (a + b) + c \sqcap (a \sqcap b)$$

   $$(a \Delta b) \setminus c = ((a \setminus b) + (b \setminus a)) \setminus c = (a \setminus b) \setminus c + (b \setminus a) \setminus c = a \setminus (b + c) + b \setminus (a + c)$$

   Now we can prove the lemma itself easily:

   $$\begin{aligned}(a \Delta b) \Delta c &= (a \Delta b) \setminus c + c \setminus (a \Delta b) \\ &= a \setminus (b + c) + b \setminus (a + c) + c \setminus (a + b) + (a \sqcap b \sqcap c) \\ &= a \setminus (b + c) + (a \sqcap b \sqcap c) + b \setminus (a + c) + c \setminus (a + b) \\ &= a \setminus (b \Delta c) + (b \Delta c) \setminus a \\ &= a \Delta (b \Delta c)\end{aligned}$$

5. $(a \sqcap b) \Delta (a \sqcap c) = ((a \sqcap b) + (a \sqcap c)) \setminus (a \sqcap b \sqcap a \sqcap c) = (a \sqcap (b + c)) \setminus (a \sqcap (b \sqcap c)) = (a \sqcap (b + c)) \sqcap (\overline{a} + \overline{(b \sqcap c)}) = (a \sqcap (b + c) \sqcap (b \sqcap c)) = a \sqcap (b \Delta c)$

6. $a = b \Rightarrow a \Delta b = a \Delta a = a \setminus a + a \setminus a = 0$
   $a \Delta b = 0 \Rightarrow a = a \Delta 0 = a \Delta (a \Delta b) = (a \Delta a) \Delta b = 0 \Delta b = b$

Lemma 6

1. Immediate from the definition.
2. $\ulcorner a \cdot (a \mid b) = \ulcorner a \cdot (a + \neg \ulcorner a \cdot b) = \ulcorner a \cdot a + \ulcorner a \cdot \neg \ulcorner a \cdot b = a$
3. $a \mid a = a + \neg \ulcorner a \cdot a = a$
4. $a \mid (b+c) = a + \neg \ulcorner a \cdot (b+c) = a + \neg \ulcorner a \cdot b + \neg \ulcorner a \cdot c = a + \neg \ulcorner a \cdot b + a + \neg \ulcorner a \cdot c = a \mid b + a \mid c$
5. $\ulcorner (a \mid b) = \ulcorner (a + \neg \ulcorner a \cdot b) = \ulcorner a + \ulcorner (\neg \ulcorner a \cdot b) = \ulcorner a + \neg \ulcorner a \cdot \ulcorner b = \ulcorner a + \ulcorner a \cdot \ulcorner b + \neg \ulcorner a \cdot \ulcorner b = \ulcorner a + (\ulcorner a + \neg \ulcorner a) \cdot \ulcorner b = \ulcorner a + \ulcorner b$
6. $$\begin{aligned} a \Delta (b \mid a) &= a \Delta (b + \neg \ulcorner b \cdot a) \\ &= a \setminus (b + \neg \ulcorner b \cdot a) + (b + \neg \ulcorner b \cdot a) \setminus a \\ &= (a \setminus b \sqcap a \setminus (\neg \ulcorner b \cdot a)) + (b \setminus a + (\neg \ulcorner b \cdot a) \setminus a)) \\ &= (a \setminus b \sqcap \ulcorner b \cdot a) + (b \setminus a) \\ &= (a \Delta b) \sqcap (\ulcorner b \cdot a + b \setminus a) \\ &= (a \Delta b) \sqcap (\ulcorner b \cdot a + \ulcorner b \cdot (b \setminus a)) \\ &= (a \Delta b) \sqcap \ulcorner b \cdot (a + (b \setminus a)) \\ &= \ulcorner b \cdot ((a \Delta b) \sqcap (a + b)) \\ &= \ulcorner b \cdot (a \Delta b) \end{aligned}$$

Lemma 7 Let $X \subseteq \mathcal{E}_a$

- $\forall x \in X : x \leq a \Rightarrow \bigsqcup X \leq a$
- $\ulcorner (\bigsqcup X) \cdot a = \bigsqcup_{x \in X} \ulcorner x \cdot a = \bigsqcup_{x \in X} x = \bigsqcup X$

Lemma 8

1. Clear from definition.
2. - $x + \neg \ulcorner x \cdot a = \ulcorner x \cdot a + \neg \ulcorner x \cdot a = a$
   - $x \sqcap \neg \ulcorner x \cdot a = \ulcorner x \cdot a \sqcap \neg \ulcorner x \cdot a = (\ulcorner x \sqcap \neg \ulcorner x) \cdot a = 0$
3. $\ulcorner x \cdot (a \setminus x) = \ulcorner x \cdot a \sqcap \ulcorner x \cdot \overline{x} = x \sqcap \ulcorner x \cdot \overline{x} = \ulcorner x \cdot (x \sqcap \overline{x}) = 0$

Lemma 9 Follows immediately from Definition and Lemma 7.

Lemma 11

$$\begin{aligned} b \setminus (a \otimes b) &= b \sqcap \overline{(a \otimes b)} \\ &= b \sqcap (\ulcorner (a \Delta b) \cdot (a \sqcap b) + \overline{(a \sqcap b)}) \\ &= (b \sqcap (\ulcorner (a \Delta b) \cdot (a \sqcap b)) + (b \sqcap \overline{a}) + (b \sqcap \overline{b}) \\ &= \ulcorner (a \Delta b) \cdot (a \sqcap b) + b \sqcap \overline{a} \\ &= \ulcorner (a \Delta b) \cdot (a \sqcap b) + \ulcorner (a \Delta b) \cdot (b \setminus a) \\ &= \ulcorner (a \Delta b) \cdot ((a \sqcap b) + (b \setminus a)) \\ &= \ulcorner (a \Delta b) \cdot b \end{aligned}$$

Lemma 10 First of all $\neg \ulcorner (a \Delta b) \cdot (a \sqcap b) \in \mathcal{D}_{a,b}$:

a) $\neg \ulcorner (a \Delta b) \cdot (a \sqcap b) \leq a \sqcap b$
b) We show the second condition only for a:

$$\begin{aligned} \ulcorner (\neg \ulcorner (a \Delta b) \cdot (a \sqcap b)) \cdot a &= \neg \ulcorner (a \Delta b) \cdot \ulcorner (a \sqcap b) \cdot a \\ &= \ulcorner (a \sqcap b) \cdot \neg \ulcorner (a \Delta b) \cdot a \\ &= \ulcorner (a \sqcap b) \cdot \neg \ulcorner (a \Delta b) \cdot ((a \sqcap b) + (a \setminus b)) \end{aligned}$$

$$
\begin{aligned}
&= \ulcorner(a \sqcap b) \cdot (\neg\ulcorner(a \Delta b) \cdot (a \sqcap b) + \neg\ulcorner(a \Delta b) \cdot (a \setminus b)) \\
&= \ulcorner(a \sqcap b) \cdot \neg\ulcorner(a \Delta b) \cdot (a \sqcap b) \\
&= \neg\ulcorner(a \Delta b) \cdot \ulcorner(a \sqcap b) \cdot (a \sqcap b) \\
&= \neg\ulcorner(a \Delta b) \cdot (a \sqcap b)
\end{aligned}
$$

Next we show that $\ulcorner(a \Delta b) \cdot \ulcorner(a \otimes b) = 0$.

$$
\begin{aligned}
\ulcorner(a \Delta b) \cdot \ulcorner(a \otimes b) &= \ulcorner(a \otimes b) \cdot \ulcorner(a \Delta b) \\
&= \ulcorner(\ulcorner(a \otimes b) \cdot (a \Delta b)) \\
&= \ulcorner(\ulcorner(a \otimes b) \cdot ((a + b) \setminus (a \sqcap b))) \\
&\leq \ulcorner(\ulcorner(a \otimes b) \cdot (a + b) \setminus (a \otimes b)) \\
&= \ulcorner(\ulcorner(a \otimes b) \cdot (a \setminus (a \otimes b) + b \setminus (a \otimes b))) \\
&= \ulcorner(\ulcorner(a \otimes b) \cdot (a \setminus (a \otimes b)) + \ulcorner(a \otimes b) \cdot (b \setminus (a \otimes b))) \\
&= \ulcorner(0 + 0) \\
&= 0
\end{aligned}
$$

Hence also $\ulcorner(a \Delta b) \cdot (a \otimes b) = 0$. Now it follows that

$$
\begin{aligned}
(a \otimes b) &= \ulcorner(a \otimes b) \cdot a \sqcap \ulcorner(a \otimes b) \cdot b \\
&= \ulcorner(a \otimes b) \cdot (a \sqcap b) \\
&= (\ulcorner(a \Delta b) + \neg\ulcorner(a \Delta b)) \cdot \ulcorner(a \otimes b) \cdot (a \sqcap b) \\
&= \ulcorner(a \Delta b) \cdot \ulcorner(a \otimes b) \cdot (a \sqcap b) + \neg\ulcorner(a \Delta b) \cdot \ulcorner(a \otimes b) \cdot (a \sqcap b) \\
&= \ulcorner(a \Delta b) \cdot (a \otimes b) + \ulcorner(a \otimes b) \cdot \neg\ulcorner(a \Delta b) \cdot (a \sqcap b) \\
&= \neg\ulcorner(a \Delta b) \cdot (a \sqcap b)
\end{aligned}
$$

This last equality holds, since $\neg\ulcorner(a \Delta b) \cdot (a \sqcap b) \in \mathcal{D}_{a,b}$ and by Lemma 3.2.

Lemma 12

1. ($\Rightarrow$) Assume $\mathcal{X} \neq \emptyset$ and $x \in \mathcal{X}$:

$$\ulcorner b = \ulcorner(x \mid a) = \ulcorner x + \ulcorner a \geq \ulcorner a$$

   ($\Leftarrow$) We show that b is a solution if the righthand side holds, so assume $\ulcorner a \leq \ulcorner b$:

$$b \mid a = b + \neg\ulcorner b \cdot a = b \text{ since } \neg\ulcorner b \cdot a \leq \neg\ulcorner a \cdot a = 0$$

2. By 1. we have $b \in \mathcal{X}$, and for all $x \in \mathcal{X}$ it follows that:

$$x \leq x + \neg\ulcorner x \cdot a = x \mid a = b$$

3. Let $\ulcorner a \leq \ulcorner b$

$$
\begin{aligned}
(b \setminus (a \otimes b)) \mid a &= b \setminus (a \otimes b) + \neg\ulcorner(b \setminus (a \otimes b)) \cdot a \\
&= b \setminus (a \otimes b) + \neg\ulcorner(\neg\ulcorner(a \otimes b) \cdot b) \cdot a \\
&= b \setminus (a \otimes b) + \neg(\neg\ulcorner(a \otimes b) \cdot \ulcorner b) \cdot a \\
&= b \setminus (a \otimes b) + (\ulcorner(a \otimes b) \cdot \neg\ulcorner b) \cdot a \\
&= b \setminus (a \otimes b) + \ulcorner(a \otimes b) \cdot a + \neg\ulcorner b \cdot a \\
&= b \setminus (a \otimes b) + (a \otimes b) \\
&= b
\end{aligned}
$$

   So $b \setminus (a \otimes b) \in \mathcal{X}$.

Consider $x \in \mathcal{X}$.

$$\begin{aligned}
b \setminus (a \otimes b) &= \ulcorner(a \Delta b) \cdot b \\
&= \ulcorner(a \Delta b) \cdot (x \mid a) \\
&= \ulcorner(a \Delta b) \cdot (x + \neg\ulcorner x \cdot a) \\
&= \ulcorner(a \Delta b) \cdot x + \ulcorner(a \Delta b) \cdot \neg\ulcorner x \cdot a \\
&= \ulcorner(a \Delta b) \cdot x + \ulcorner(a \Delta (x \mid a)) \cdot \neg\ulcorner x \cdot a \\
&= \ulcorner(a \Delta b) \cdot x + \ulcorner(\ulcorner x \cdot (a \Delta x)) \cdot \neg\ulcorner x \cdot a \\
&= \ulcorner(a \Delta b) \cdot x + \ulcorner x \cdot \ulcorner(a \Delta x) \cdot \neg\ulcorner x \cdot a \\
&= \ulcorner(a \Delta b) \cdot x \\
&= \ulcorner(a \Delta b) \cdot (x \sqcap b) \\
&= \ulcorner(a \Delta b) \cdot x \sqcap b \\
&= b \setminus (a \otimes b) \sqcap x
\end{aligned}$$

So $b \setminus (a \otimes b)$ is minimal in $\mathcal{X}$

Lemma 13

1. Let $c \leq b \Rightarrow \ulcorner c \leq \ulcorner b$, then $\ulcorner c \cdot b = \ulcorner c \cdot \ulcorner b \cdot a = \ulcorner c \cdot a = c$
2. Follows directly from 1.
3.

$$\begin{aligned}
c \leq a \mid b \Rightarrow c = c \sqcap a \mid b = c \sqcap a + c \sqcap \neg\ulcorner a \cdot b = c \sqcap a + \neg\ulcorner a \cdot (c \sqcap b) \\
\Rightarrow \ulcorner c = \ulcorner(c \sqcap a) + \neg\ulcorner a \cdot \ulcorner(c \sqcap b)
\end{aligned}$$

$\ulcorner c \cdot (a \mid b) = \ulcorner c \cdot (a + \neg\ulcorner a \cdot b) = \ulcorner c \cdot a + \neg\ulcorner a \cdot \ulcorner c \cdot b = \ulcorner(c \sqcap a) \cdot a + \neg\ulcorner a \cdot \ulcorner(c \sqcap b) \cdot b$

Lemma 14

1. a, b maps $\Rightarrow \mathcal{D}_{a,b} = \{x : x \leq a \sqcap b\}$ because of remark after Definition 13.
2. Follows directly from 1. and Lemma 12.3.