Monoid Modules and Structured Document Algebra
(Extendend Abstract)

Andreas Zelend

Institut für Informatik, Universität Augsburg, Germany
zelend@informatik.uni-augsburg.de

1 Introduction

Feature Oriented Software Development (e.g. [3]) has been established in computer science as a general programming paradigm that provides formalisms, methods, languages, and tools for building maintainable, customisable, and extensible *software product lines (SPLs)* [8]. An SPL is a collection of programs that share a common part, e.g., functionality or code fragments. To encode an SPL, one can use *variation points (VPs)* in the source code. A VP is a location in a program whose content, called a *fragment*, can vary among different members of the SPL. In [2] a *Structured Document Algebra (SDA)* is used to algebraically describe modules that include VPs and their composition. In [4] we showed that we can reason about SDA in a more general way using a so called *relational predomain monoid module (RMM)*. In this paper we present the following extensions and results: an investigation of the structure of transformations, e.g., a condition when transformations commute, insights into the pre-order of modules, and new properties of predomain monoid modules.

2 Structured Document Algebra

VPs and Fragments. Let $\mathtt{V}$ denote a set of VPs at which fragments may be inserted and $\mathtt{F}(\mathtt{V})$ be the set of *fragments* which may, among other things, contain VPs from $\mathtt{V}$. Elements of $\mathtt{F}(\mathtt{V})$ are denoted by $\mathtt{f}_1, \mathtt{f}_2, \ldots$. There are two special elements, a default fragment $\mathtt{0}$ and an error $\lightning$. An error signals an attempt to assign two or more non-default fragments to the same VP within one module. The addition, or supremum operator $+$ on fragments obeys the following rules:

$$\begin{aligned} \mathtt{0} + \mathtt{x} &= \mathtt{x}\,, & \lightning + \mathtt{x} &= \lightning\,, \\ \mathtt{x} + \mathtt{x} &= \mathtt{x}\,, & \mathtt{f}_\mathtt{i} + \mathtt{f}_\mathtt{j} &= \lightning\ (\mathtt{i} \neq \mathtt{j})\,, \end{aligned}$$

where $\mathtt{x} \in \{\mathtt{0}, \mathtt{f}_\mathtt{i}, \lightning\}$. This structure forms a flat lattice with least element $\mathtt{0}$ and greatest element $\lightning$ and pairwise incomparable $\mathtt{f}_\mathtt{i}$. By standard lattice theory $+$ is commutative, associative and idempotent and has $\mathtt{0}$ as its neutral element.

Modules. A *module* is a partial function $\mathtt{m} : \mathtt{V} \rightsquigarrow \mathtt{F}(\mathtt{V})$. A VP $\mathtt{v}$ is *assigned* in $\mathtt{m}$ if $\mathtt{v} \in \mathtt{dom}(\mathtt{m})$, otherwise *unassigned* or *external*. By using partial functions rather than relations, a VP can be filled with at most one fragment *(uniqueness)*. The simplest module is the *empty module* **0**, i.e., the empty partial map.

Module Addition. The main goal of feature oriented programming is to construct programs step by step from reusable modules. In the algebra this is done by module addition (+). Addition fuses two modules while maintaining uniqueness (and signaling an error upon a conflict). Desirable properties for + are commutativity and associativity. For module addition, + on fragments is lifted to partial functions:

$$(\mathtt{m}+\mathtt{n})(\mathtt{v}) =_{df} \begin{cases} \mathtt{m}(\mathtt{v}) & \text{if } \mathtt{v} \in \mathtt{dom}(\mathtt{m}) - \mathtt{dom}(\mathtt{n})\ , \\ \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \mathtt{dom}(\mathtt{n}) - \mathtt{dom}(\mathtt{m})\ , \\ \mathtt{m}(\mathtt{v})+\mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \mathtt{dom}(\mathtt{m}) \cap \mathtt{dom}(\mathtt{n})\ , \\ \text{undefined} & \text{if } \mathtt{v} \notin \mathtt{dom}(\mathtt{m}) \cup \mathtt{dom}(\mathtt{n})\ . \end{cases}$$

If in the third case $\mathtt{m}(\mathtt{v}) \neq \mathtt{n}(\mathtt{v})$ and $\mathtt{m}(\mathtt{v}), \mathtt{n}(\mathtt{v}) \neq \mathtt{0}$ then $(\mathtt{m}+\mathtt{n})(\mathtt{v}) = \lightning$, thus signalling an error.

The set of modules forms a commutative monoid under + with the neutral element **0**.

Deletion and Subtraction. For modules $\mathtt{m}$ and $\mathtt{n}$ the *subtraction* $\mathtt{m}-\mathtt{n}$ is defined as

$$(\mathtt{m}-\mathtt{n})(\mathtt{v}) =_{df} \begin{cases} \mathtt{m}(\mathtt{v}) & \text{if } \mathtt{v} \in \mathtt{dom}(\mathtt{m}) - \mathtt{dom}(\mathtt{n})\ , \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Overriding. To allow *overriding*, an operation $\rhd\!\!-$ can be defined in terms of subtraction and addition. Module $\mathtt{m}$ overrides $\mathtt{n}$, written $\mathtt{m} \rhd\!\!- \mathtt{n}$, if

$$\mathtt{m} \rhd\!\!- \mathtt{n} \;=\; \mathtt{m} + (\mathtt{n} - \mathtt{m})$$

This replaces all assignments in $\mathtt{n}$ for which $\mathtt{m}$ also provides a value. $\rhd\!\!-$ is associative and idempotent with neutral element **0**.

Modules $\mathtt{m}$ and $\mathtt{n}$ are called *compatible*, in signs $\mathtt{m} \downarrow \mathtt{n}$, if their fragments coincide on their shared domains, i.e.,

$$\mathtt{m} \downarrow \mathtt{n} \;\Leftrightarrow_{df}\; \forall \mathtt{v} \in \mathtt{dom}(\mathtt{m}) \cap \mathtt{dom}(\mathtt{n}) :\ \mathtt{m}(\mathtt{v}) = \mathtt{n}(\mathtt{v})\ .$$

We have chosen this characterization of compatibility with regard to the non-relational abstract approach, to be pursued in Section 3, which especially needs no converse operation. For a relational treatment see [9]. All submodules of a module are pairwise compatible with each other.

2.1 Transformations

In this section we sketch an extension of SDA, intended to cope with some standard techniques in software refactoring (e.g., [1,7]). Examples of such techniques are consistent renaming of methods or classes in a large software system. To stay at the same level of abstraction as before, we realize this by a mechanism for generally modifying the fragments in SDA modules.

By a *transformation* or *modification* or *refactoring* we mean a total function $\mathtt{T} : \mathtt{F}(\mathtt{V}) \rightarrow \mathtt{F}(\mathtt{V})$. By $\mathtt{T} \cdot \mathtt{m}$ we denote the *application* of $\mathtt{T}$ to a module $\mathtt{m}$. It yields a new module defined by

$$(\mathtt{T} \cdot \mathtt{m})(\mathtt{v}) =_{df} \begin{cases} \mathtt{T}(\mathtt{m}(\mathtt{v})) & \text{if } \mathtt{v} \in \mathtt{dom}(\mathtt{m}) \\ \text{undefined} & \text{otherwise} . \end{cases}$$

We need to handle the special case of transforming the error fragment $\lightning$. Since we don't want to allow transformations to mask errors that are related to module addition, we add the requirement

$$\mathtt{T}(\lightning) = \lightning \ .$$

We use the convention that $\cdot$ binds stronger than all other operators. Note that, although $\mathtt{T}$ is supposed to be a total function on all fragments, it might well leave many of those unchanged, i.e., act as the identity on them.

2.2 Structure of Transformations

Definition 2.1 A *monoid of transformations* is a structure $\mathtt{F} = (\mathtt{F}, \circ, \mathbb{1})$, where $\mathtt{F}$ is a set of total functions $\mathtt{f} : \mathtt{X} \rightarrow \mathtt{X}$ over some set $\mathtt{X}$, closed under function composition $\circ$, and $\mathbb{1}$ the identity function. The pair $(\mathtt{X}, \mathtt{F})$ is called *transformation monoid* of $\mathtt{X}$. By $\mathtt{T}|_{\mathtt{A}}$ we denote the transformation $\mathtt{T}$ restricted to the set $\mathtt{A}$.

We call the set of transformations on fragments Γ. Then, by the above definition, $(\mathtt{F}(\mathtt{V}), \Gamma)$ is the transformation monoid of fragments $\mathtt{F}(\mathtt{V})$ which we abbreviate to Γ. Since these transformations are not necessarily invertible, in general Γ is not a transformation group. We now can extend the list of properties given in [2].

(6) $\mathtt{T} \cdot (\mathtt{m} + \mathtt{n}) = \mathtt{T} \cdot \mathtt{m} + \mathtt{n} \quad \Leftarrow \quad \mathtt{T}|_{\mathtt{ran}(\mathtt{n})} = \mathbb{1}|_{\mathtt{ran}(\mathtt{n})} \ \wedge \ \mathtt{m} \downarrow \mathtt{n}$,
(7) $\mathbb{1} \cdot \mathtt{m} = \mathtt{m}$,
(8) $\mathtt{T} \cdot \mathbf{0} = \mathbf{0}$.

$\mathbf{0}$ being an annihilator, (Property (8)) means that transformations can only change existing fragments rather than create new ones.

Furthermore we can define the *application equivalence* $\approx$ of two transformations $\mathtt{S}, \mathtt{T}$ by $\mathtt{S} \approx \mathtt{T} \Leftrightarrow_{df} \forall \mathtt{m} : \mathtt{S} \cdot \mathtt{m} = \mathtt{T} \cdot \mathtt{m}$.

It is common to undo refactorings, e.g., undo a renaming of a variable. This can be modelled by inverse transformations, denoted by $^{-1}$. When it exists, the inverse of a *stacked*, or composed transformation, is given by $(\mathtt{T} \circ \mathtt{S})^{-1} = \mathtt{S}^{-1} \circ \mathtt{T}^{-1}$. Of course the inverse of $\mathtt{T}$ or $\mathtt{S}$ might not exist, e.g., if $\mathtt{T}$ is not injective.

As stated above transformations are total functions. Since they act as the identity for fragments that should not be modified we define the set of fragments they transform as follows.

Definition 2.2 Let $\mathtt{T} : \mathtt{F}(\mathtt{V}) \rightarrow \mathtt{F}(\mathtt{V})$ be a transformation. Then we call $\mathtt{T}_{\mathtt{m}} =_{df} \{\mathtt{f} \in \mathtt{F}(\mathtt{V}) : \mathtt{T}(\mathtt{f}) \neq \mathtt{f}\}$ the *modified fragments of* $\mathtt{T}$ and $\mathtt{T}_{\mathtt{v}} =_{df} \{\mathtt{T}(\mathtt{f}) \in \mathtt{F}(\mathtt{V}) : \mathtt{T}(\mathtt{f}) \neq \mathtt{f}\} = \mathtt{ran}(\mathtt{T}|_{\mathtt{T}_{\mathtt{m}}})$ the *value set* of $\mathtt{T}$.

Restricting transformations to their modified fragments allows us to state situations in which transformations can be omitted or commute.

Lemma 2.3

1. $\mathtt{T}\cdot(\mathtt{S}\cdot\mathtt{m}) = \mathtt{S}\cdot\mathtt{m}$ *if* $\mathtt{T}_\mathtt{m} \subseteq \mathtt{S}_\mathtt{m} \wedge \mathtt{T}_\mathtt{m}\cap\mathtt{S}_\mathtt{v} = \emptyset$.
2. $\mathtt{T}$ *and* $\mathtt{S}$ *commute if* $\mathtt{T}_\mathtt{m}\cap\mathtt{S}_\mathtt{m} = \emptyset \wedge \mathtt{T}_\mathtt{m}\cap\mathtt{S}_\mathtt{v} = \emptyset \wedge \mathtt{T}_\mathtt{v}\cap\mathtt{S}_\mathtt{m} = \emptyset$.

A proof can be found in Appendix A.1.

3 Abstracting from SDA

The set $\mathtt{M}$ of modules, i.e., partial maps $\mathtt{m} : \mathtt{V} \rightsquigarrow \mathtt{F}(\mathtt{V})$, with $+$ and $-$, defined as in subsection 2, forms an algebraic structure $\mathtt{SDA} =_{df} (\mathtt{M}, +, -, \mathtt{0})$ such that $(\mathtt{M}, +, \mathtt{0})$ is an idempotent and commutative monoid and which satisfies the following laws for all $\mathtt{l}, \mathtt{m}, \mathtt{n} \in \mathtt{M}$:

1. $(\mathtt{l} - \mathtt{m}) - \mathtt{n} = \mathtt{l} - (\mathtt{m} + \mathtt{n})$,
2. $(\mathtt{l} + \mathtt{m}) - \mathtt{n} = (\mathtt{l} - \mathtt{n}) + (\mathtt{m} - \mathtt{l})$,
3. $\mathtt{0} - \mathtt{l} = \mathtt{0}$,
4. $\mathtt{l} - \mathtt{0} = \mathtt{l}$.

Definition 3.1 A *monoid module* (m-module) is an algebraic structure $(\mathtt{B}, \mathtt{M}, :)$ where $(\mathtt{M}, +, \mathtt{0})$ is an idempotent and commutative monoid and $(\mathtt{B}, +, \cdot, \mathtt{0}, \mathtt{1}, \neg)$ is a Boolean algebra in which $\mathtt{0}$ and $\mathtt{1}$ are the least and greatest element and $\cdot$ and $+$ denote meet and join. Note that $\mathtt{0}$ and $+$ are overloaded, like in classical modules or vector spaces. The restriction, or scalar product, $:$ is a mapping $\mathtt{B} \times \mathtt{M} \rightarrow \mathtt{M}$ satisfying for all $\mathtt{p}, \mathtt{q} \in \mathtt{B}$ and $\mathtt{m}, \mathtt{n} \in \mathtt{M}$:

$$(\mathtt{p} + \mathtt{q}) : \mathtt{m} = \mathtt{p} : \mathtt{m} + \mathtt{q} : \mathtt{m} \, , \quad (1) \qquad (\mathtt{p} \cdot \mathtt{q}) : \mathtt{m} = \mathtt{p} : (\mathtt{q} : \mathtt{m}) \, , \quad (4)$$

$$\mathtt{p} : (\mathtt{m} + \mathtt{n}) = \mathtt{p} : \mathtt{m} + \mathtt{p} : \mathtt{n} \, , \quad (2) \qquad \mathtt{1} : \mathtt{m} = \mathtt{m} \, , \quad (5)$$

$$\mathtt{0} : \mathtt{m} = \mathtt{0} \, , \quad (3) \qquad \mathtt{p} : \mathtt{0} = \mathtt{0} \, . \quad (6)$$

We define the natural pre-order on $(\mathtt{M}, +, \mathtt{0})$ by $\mathtt{m} \leq \mathtt{n} \Leftrightarrow_{df} \mathtt{m} + \mathtt{n} = \mathtt{n}$. Therefore $+$ is isotone in both arguments.

We have choosen the name monoid module following the notion of a module over a ring and because SDA's modules form an idempotent and commutative monoid.

Lemma 3.2

1. *Restriction* $:$ *is isotone in both arguments.*
2. $\mathtt{p} : \mathtt{m} \leq \mathtt{m}$.
3. $\mathtt{p} : (\mathtt{q} : \mathtt{m}) = \mathtt{q} : (\mathtt{p} : \mathtt{m})$

The first claim follows by distributivity, the second by isotony and (5) and the third by (4) and Boolean algebra.

The structure $\mathtt{RMM} = (\mathcal{P}(\mathtt{M}), \mathcal{P}(\mathtt{M} \times \mathtt{N}), :)$, where $:$ is restriction, i.e., $\mathtt{p} : \mathtt{m} = \{(\mathtt{x}, \mathtt{y}) \mid \mathtt{x} \in \mathtt{p} \wedge (\mathtt{x}, \mathtt{y}) \in \mathtt{m}\}$, forms a mono module. To model subtraction we extend mono modules with the predomain operator $^\ulcorner : \mathtt{M} \rightarrow \mathtt{B}$.

Definition 3.3 A *predomain monoid module* (predomain m-module) is a structure $(\mathtt{B}, \mathtt{M}, :, ^\ulcorner)$ such that$(\mathtt{B}, \mathtt{M}, :)$ is a m-module and $^\ulcorner : \mathtt{M} \rightarrow \mathtt{B}$ satisfies for all $\mathtt{p} \in \mathtt{B}$ and $\mathtt{m} \in \mathtt{M}$:

(d1) $\mathtt{m} \le {}^\ulcorner\mathtt{m} : \mathtt{m}$, (d2) ${}^\ulcorner(\mathtt{p} : \mathtt{m}) \le \mathtt{p}$.

In a predomain m-module ${}^\ulcorner\mathtt{m}$ is the least left preserver of $\mathtt{m}$ and $\neg{}^\ulcorner\mathtt{a}$ is the greatest left annihilator. To justify this we present the following lemma.

Lemma 3.4 *In a predomain m-module* $(\mathtt{B}, \mathtt{M}, :, {}^\ulcorner\ )$ *for all* $\mathtt{p} \in \mathtt{B}$ *and* $\mathtt{m} \in \mathtt{M}$*:*

(llp) ${}^\ulcorner\mathtt{m} \le \mathtt{p} \Leftrightarrow \mathtt{m} \le \mathtt{p} : \mathtt{m}$, *(gla)* $\mathtt{p} \le \neg{}^\ulcorner\mathtt{m} \Leftrightarrow \mathtt{p} : \mathtt{m} \le 0$.

Note that (llp) does not establish a Galois connection, since there is no greatest element in the monoid $(\mathtt{M}, +, 0)$ per se, cf. [5] A proof can be found in Appendix A.2.

Now we can give more useful properties of the predomain function like isotony or strictness.

Lemma 3.5 *In a predomain m-module* $(\mathtt{B}, \mathtt{M}, :, {}^\ulcorner\ )$ *for all* $\mathtt{p} \in \mathtt{B}$ *and* $\mathtt{m}, \mathtt{n} \in \mathtt{M}$*:*

1. $\mathtt{m} = 0 \Leftrightarrow {}^\ulcorner\mathtt{m} = 0$,
2. $\mathtt{m} \le \mathtt{n} \Rightarrow {}^\ulcorner\mathtt{m} \le {}^\ulcorner\mathtt{n}$,
3. $\mathtt{m} = {}^\ulcorner\mathtt{m} : \mathtt{m}$,
4. ${}^\ulcorner(\mathtt{m} + \mathtt{n}) = {}^\ulcorner\mathtt{m} + {}^\ulcorner\mathtt{n}$,
5. ${}^\ulcorner(\mathtt{p} : \mathtt{m}) : \mathtt{m} = \mathtt{p} : \mathtt{m}$,
6. ${}^\ulcorner(\mathtt{p} : \mathtt{m}) = \mathtt{p} \cdot {}^\ulcorner\mathtt{m}$.

A proof can be found in Appendix A.3.

Now it is easy to verify that the SDA laws for subtractions also hold in a predomain m-module. Note that the sides change, e.g., *right* distributivity becomes *left* distributivity.

Lemma 3.6 *Assume a predomain m-module* $(\mathtt{B}, \mathtt{M}, :, {}^\ulcorner\ )$*. Then for all* $\mathtt{l}, \mathtt{m}, \mathtt{n} \in \mathtt{M}$*:*

1. ${}^\ulcorner(\neg{}^\ulcorner\mathtt{n} : \mathtt{m}) = {}^\ulcorner\mathtt{m} \cdot \neg{}^\ulcorner\mathtt{n}$,
2. $(\neg{}^\ulcorner\mathtt{n} : 0 = 0)$,
3. $\neg{}^\ulcorner\mathtt{l} : (\mathtt{m} + \mathtt{n}) = \neg{}^\ulcorner\mathtt{l} : \mathtt{m} + \neg{}^\ulcorner\mathtt{l} : \mathtt{n}$,
4. $\neg{}^\ulcorner(\mathtt{m} + \mathtt{n}) : \mathtt{l} = \neg{}^\ulcorner\mathtt{n} : (\neg{}^\ulcorner\mathtt{m} : \mathtt{l})$,
5. $\neg{}^\ulcorner 0 : \mathtt{m} = \mathtt{m}$,
6. $\neg{}^\ulcorner\mathtt{m} : \mathtt{m} = 0$,
7. $\neg{}^\ulcorner\mathtt{n} : \mathtt{m} \le \mathtt{m}$,
8. $\mathtt{m} \le \mathtt{n} \Rightarrow \neg{}^\ulcorner\mathtt{n} : \mathtt{m} = 0$.

A proof can be found in Appendix A.4.

By defining ${}^\ulcorner\mathtt{m} =_{df} \{\mathtt{x} \mid (\mathtt{x}, \mathtt{y}) \in \mathtt{m}\}$ RMM becomes a predomain m-module and using an RMM over binary functional relations $\mathtt{R} \subseteq \mathtt{V} \times \mathtt{F}(\mathtt{V})$, i.e., $\mathtt{R}^{\smile}; \mathtt{R} \subseteq \mathsf{id}(\mathtt{F}(\mathtt{V}))$, allows us to reason about SDA. As a result, SDA's subtraction $\mathtt{m} - \mathtt{n}$ of modules is equivalent to $\neg{}^\ulcorner\mathtt{n} : \mathtt{m}$ in the corresponding RMM.

Using SDA's module addition, cf. Section 2, we can investigate the induced natural pre-order.

$$
\begin{aligned}
\mathtt{m} \leq \mathtt{n} \ \Leftrightarrow_{df}\ & \mathtt{m} + \mathtt{n} = \mathtt{n} \\
\Leftrightarrow\ & \left.\begin{cases} \mathtt{m}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} - \ulcorner\mathtt{n} \\ \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{n} - \ulcorner\mathtt{m} \\ \mathtt{m}(\mathtt{v}) + \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} \cap \ulcorner\mathtt{n} \\ \text{undefined} & \text{if } \mathtt{v} \notin \ulcorner\mathtt{m} \cup \ulcorner\mathtt{n} \end{cases}\right\} = \mathtt{n}(\mathtt{v}) \\
\Leftrightarrow\ & \begin{cases} \mathtt{m}(\mathtt{v}) = \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} - \ulcorner\mathtt{n} \\ \mathtt{n}(\mathtt{v}) = \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{n} - \ulcorner\mathtt{m} \\ \mathtt{m}(\mathtt{v}) + \mathtt{n}(\mathtt{v}) = \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} \cap \ulcorner\mathtt{n} \\ \text{true} & \text{if } \mathtt{v} \notin \ulcorner\mathtt{m} \cup \ulcorner\mathtt{n} \end{cases} \\
\Leftrightarrow\ & \begin{cases} \text{false} & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} - \ulcorner\mathtt{n} \\ \text{true} & \text{if } \mathtt{v} \in \ulcorner\mathtt{n} - \ulcorner\mathtt{m} \\ \mathtt{m}(\mathtt{v}) \leq \mathtt{n}(\mathtt{v}) & \text{if } \mathtt{v} \in \ulcorner\mathtt{m} \cap \ulcorner\mathtt{n} \\ \text{true} & \text{if } \mathtt{v} \notin \ulcorner\mathtt{m} \cup \ulcorner\mathtt{n} \end{cases} \\
\Leftrightarrow\ & \mathtt{v} \in \ulcorner\mathtt{n} - \ulcorner\mathtt{m} \vee (\mathtt{v} \in \ulcorner\mathtt{m} \cap \ulcorner\mathtt{n} \wedge \mathtt{m}(\mathtt{v}) \leq \mathtt{n}(\mathtt{v})) \vee \mathtt{v} \notin \ulcorner\mathtt{m} \cup \ulcorner\mathtt{n} \text{ for any } \mathtt{v} \in \mathtt{V}.
\end{aligned}
$$

Therefore the least element w.r.t. $\leq$ is the empty module $\mathtt{0}$ and the top element is the module $\mathtt{t}$ with $\mathtt{t}(\mathtt{v}) = \lightning$ for any $\mathtt{v} \in \mathtt{V}$.

SDA's overriding operator $\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}$ can also be defined in a predomain m-module: $\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n} \ =_{df}\ \mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{n}$. In [6] this operator, embedded into a Kleene algebra, is used to update links in pointer structures.

Lemma 3.7 *In a predomain m-module* $(\mathtt{B}, \mathtt{M}, :, \ulcorner\ )$ *for all* $\mathtt{p} \in \mathtt{B}$ *and* $\mathtt{l}, \mathtt{m}, \mathtt{n} \in \mathtt{M}$*:*

1. $\mathtt{0} \mathbin{-\!\triangleright} \mathtt{n} = \mathtt{n}$,
2. $\mathtt{m} \mathbin{-\!\triangleright} \mathtt{0} = \mathtt{m}$,
3. $\mathtt{m} \leq \mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}$,
4. $\mathtt{m} = \ulcorner\mathtt{m} : (\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n})$,
5. $\ulcorner(\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}) = \ulcorner\mathtt{m} + \ulcorner\mathtt{n}$,
6. $\ulcorner\mathtt{m} \geq \ulcorner\mathtt{n} \Rightarrow \mathtt{m} \mathbin{-\!\triangleright} \mathtt{n} = \mathtt{m}$,
7. $\mathtt{l} \mathbin{-\!\triangleright} (\mathtt{m} + \mathtt{n}) = \mathtt{l} \mathbin{-\!\triangleright} \mathtt{m} + \mathtt{l} \mathbin{-\!\triangleright} \mathtt{n}$.

A proof can be found in Appendix A.5.

Future work will focus on further properties of transformations and their incorporation into the abstract framework of predomain m-modules.

Acknowledgments I am very grateful to the anonymous referees for helpful comments and suggestions and I thank B. Möller for fruitful discussions and valuable comments.

References

1. Batory, D.: Program refactorings, program synthesis, and model-driven design. In: Krishnamurthi, S., Odersky, M. (eds.) Compiler Construction. LNCS, vol. 4420, pp. 156–171. Springer (2007)
2. Batory, D., Höfner, P., Möller, B., Zelend, A.: Features, modularity, and variation points. Tech. Rep. CS-TR-13-14, The University of Texas at Austin (2013)
3. Batory, D., O'Malley, S.: The design and implementation of hierarchical software systems with reusable components. ACM Transactions Software Engineering and Methodology 1(4), 355–398 (1992)
4. Dang, H.H., Glück, R., Möller, B., Roocks, P., Zelend, A.: Exploring modal worlds. Journal of Logical and Algebraic Methods in Programming 83(2), 135 – 153 (2014), http://www.sciencedirect.com/science/article/pii/S1567832614000058, festschrift in Honour of Gunther Schmidt on the Occasion of his 75th Birthday
5. Desharnais, J., Möller, B., Struth, G.: Kleene algebra with domain. ACM Trans. Comput. Log. 7(4), 798–833 (2006), http://doi.acm.org/10.1145/1183278.1183285
6. Ehm, T.: The Kleene Algebra of Nested Pointer Structures: Theory and Applications. Ph.D. thesis, Universität Augsburg (2005)
7. Kuhlemann, M., Batory, D., Apel, S.: Refactoring feature modules. In: Edwards, S., Kulczycki, G. (eds.) Formal Foundations of Reuse and Domain Engineering. LNCS, vol. 5791, pp. 106–115. Springer (2009)
8. Lopez-Herrejon, R., Batory, D.: A standard problem for evaluating product-line methodologies. In: Bosch, J. (ed.) GCSE '01: Generative and Component-Based Software Engineering. LNCS, vol. 2186, pp. 10–24. Springer (2001)
9. Möller, B.: Towards pointer algebra. Sci. Comput. Program. 21(1), 57–90 (1993), http://dx.doi.org/10.1016/0167-6423(93)90008-D

A Proofs for Subsection 2.2 and Section 3

A.1 Proofs for Lemma 2.3

1. $\mathtt{T_m} \cap \mathtt{S_v} = \emptyset$ implies that $\mathtt{T}$ acts as the identity on the set $\mathtt{S_v}$. Since $\mathtt{T_m} \subseteq \mathtt{S_m}$, we obtain that $\mathtt{T}$ also is the identity on the set $\mathtt{ran}(\mathtt{m}) - \mathtt{S_m}$. Therefore we have $\mathtt{T} \cdot (\mathtt{S} \cdot \mathtt{m})(\mathtt{v}) = \mathtt{T} \cdot (\mathtt{S}(\mathtt{m}(\mathtt{v}))) = \mathtt{S}(\mathtt{m}(\mathtt{v}))$.
2. We have $\mathtt{x} \in \mathtt{S_m} \Rightarrow \mathtt{T}(\mathtt{x}) = \mathtt{x} \wedge \mathtt{x} \in \mathtt{T_m} \Rightarrow \mathtt{S}(\mathtt{x}) = \mathtt{x}$ because $\mathtt{T_m} \cap \mathtt{S_m} = \emptyset$. Therefore, for an arbitrary $\mathtt{x} = \mathtt{m}(\mathtt{v})$, we get

$$(\mathtt{T} \circ \mathtt{S})(\mathtt{x}) = \mathtt{T}(\mathtt{S}(\mathtt{x})) = \begin{cases} \mathtt{S}(\mathtt{x}) \text{ if } \mathtt{x} \in \mathtt{S_m} \text{ since } \mathtt{T_m} \cap \mathtt{S_v} = \emptyset\,, \\ \mathtt{T}(\mathtt{x}) \text{ if } \mathtt{x} \in \mathtt{T_m}\,, \\ \mathtt{x} \quad\;\; \text{if } \mathtt{x} \in \overline{\mathtt{S_m}} \cap \overline{\mathtt{T_m}}\,. \end{cases}$$

On the other hand we get

$$(\mathtt{S} \circ \mathtt{T})(\mathtt{x}) = \mathtt{S}(\mathtt{T}(\mathtt{x})) = \begin{cases} \mathtt{T}(\mathtt{x}) \text{ if } \mathtt{x} \in \mathtt{T_m} \text{ since } \mathtt{S_m} \cap \mathtt{T_v} = \emptyset\,, \\ \mathtt{S}(\mathtt{x}) \text{ if } \mathtt{x} \in \mathtt{S_m}\,, \\ \mathtt{x} \quad\;\; \text{if } \mathtt{x} \in \overline{\mathtt{T_m}} \cap \overline{\mathtt{S_m}}\,. \end{cases}$$

A.2 Proofs for Lemma 3.4

(llp) $\Rightarrow$: $m \leq \ulcorner m : m \leq p : m$ by (d1) and isotony of restriction ($\ulcorner m \leq p$).
$\Leftarrow$: $m \leq p : m \Rightarrow m = p : m$ by Lemma 3.2.2 and therefore $\ulcorner m = \ulcorner(p : m) \leq p$ by (d2).

(gla) $\Rightarrow$: $p : m \leq p :(\ulcorner m : m) = (p \cdot \ulcorner m) : m$ by (d1), isotony of restriction and (4). Since $p \leq \neg\ulcorner m$, $p \cdot \ulcorner m = 0$ holds and therefore $(p \cdot \ulcorner m) : m = 0 : m = 0$ by (3).
$\Leftarrow$: $m = 1 : m = (p + \neg p) : m = p : m + \neg p : m = 0 + \neg p : m = \neg p : m$ by (5), Boolean algebra, (1) and the assumption ($p : m \leq 0$). Using (llp) we have $\ulcorner m \leq \neg p$ which is equivalent to $p \leq \neg\ulcorner m$ by shunting.

A.3 Proofs for Lemma 3.5

1. $\Rightarrow$: $\ulcorner m = \ulcorner 0 = \ulcorner(0 : 0) \leq 0$ by (3) and (d2).
   $\Leftarrow$: $m \leq \ulcorner m : m$ by (d1) and therefore $m \leq 0 : m = 0$ by (3).
2. $\neg\ulcorner n \leq \neg\ulcorner n \Rightarrow \neg\ulcorner n : n \leq 0 \Rightarrow \neg\ulcorner n : m \leq 0 \Rightarrow \neg\ulcorner n \leq \neg\ulcorner m \Rightarrow \ulcorner m \leq \ulcorner n$ by (gla), isotony of restriction, (gla) again and shunting (2 times).
3. $p \leq \neg\ulcorner(m + n) \Leftrightarrow p : (m + n) \leq 0 \Leftrightarrow p : m + p : n \leq 0 \Leftrightarrow p : m \leq 0 \wedge p : n \leq 0 \Leftrightarrow p \leq \neg\ulcorner m \wedge p \leq \neg\ulcorner n \Leftrightarrow p \leq \neg\ulcorner m \cdot \neg\ulcorner n \Leftrightarrow p \leq \neg(\ulcorner m + \ulcorner n)$. Using indirect equality we get $\neg\ulcorner(m + n) = \neg(\ulcorner m + \ulcorner m) \Leftrightarrow \ulcorner(m + n) = \ulcorner m + \ulcorner m$.
4. $m \leq \ulcorner m : m$ holds by (d1) and $\ulcorner m : m \leq m$ holds by Lemma 3.2.2.
5. $\leq$: $\ulcorner(p : m) : m \leq p : m$ holds by (d2) and isotony of restriction.
   $\geq$: $p : m \leq \ulcorner(p : m) :(p : m) = (\ulcorner(p : m) \cdot p) : m$ by (d1) and (4). Since $\ulcorner(p : m) \leq p$ by (d2) it follows that $\ulcorner(p : m) \cdot p = \ulcorner(p : m)$. In sum we have $p : m \leq \ulcorner(p : m) : m$.
6. First we have $\ulcorner m = \ulcorner(1 : m) = \ulcorner((p + \neg p) : m) = \ulcorner(p : m + \neg p : m) = \ulcorner(p : m) + \ulcorner(\neg p : m)$ by (5), Boolean algebra, and Part 4. By (d2) it holds that $\ulcorner(p : m) \leq p \wedge \ulcorner(\neg p : m) \leq \neg p$. And therefore by Boolean algebra: $p \cdot \ulcorner(p : m) = \ulcorner(p : m)$ and $p \cdot \ulcorner(\neg p : m) = 0$. In sum we conclude: $p \cdot \ulcorner m = p \cdot (\ulcorner(p : m) + \ulcorner(\neg p : m)) = p \cdot \ulcorner(p : m) + p \cdot \ulcorner(\neg p : m) = p \cdot \ulcorner(p : m) + 0 = \ulcorner(p : m)$.

A.4 Proofs for Lemma 3.6

1. $\ulcorner(\neg\ulcorner n : m) = \ulcorner m \cdot \neg\ulcorner n$ by Lemma 3.5.6 and Boolean algebra.
2. $(\neg\ulcorner n : 0 = 0)$ by (6).
3. $\neg\ulcorner 1 : (m + n) = \neg\ulcorner 1 : m + \neg\ulcorner 1 : n$ by (5).
4. $\neg\ulcorner(m + n) : 1 = \neg(\ulcorner m + \ulcorner n) : 1 = (\neg\ulcorner m \cdot \neg\ulcorner n) = \neg\ulcorner m :(\neg\ulcorner n : 1) = \neg\ulcorner m :(\neg\ulcorner n : 1)$ by Lemma 3.5.4, Boolean algebra, (4) and Lemma 3.2.3.
5. $\neg\ulcorner 0 : m = \neg 0 : m = 1 : m = m$ by (5) and $\neg\ulcorner 0 = \neg 0 = 1$ Lemma 3.5.1, Boolean algebra and (5).
6. $\neg\ulcorner m \leq \neg\ulcorner m \Rightarrow \neg\ulcorner m : m \leq 0$ by (gla).
7. $\neg\ulcorner n : m \leq m$ by Lemma 3.2.2.
8. $m \leq n \Rightarrow \ulcorner m \leq \ulcorner n \Rightarrow \neg\ulcorner n \leq \neg\ulcorner m \Rightarrow \neg\ulcorner n : m \leq 0$ by Lemma 3.5.2, Boolean algebra and (gla).

A.5 Proofs for Lemma 3.7

1. $\mathtt{0} \mathbin{-\!\triangleright} \mathtt{n} = \mathtt{0} + \neg\ulcorner\mathtt{0} : \mathtt{n} = \neg\mathtt{0} : \mathtt{n} = \mathtt{1} : \mathtt{n} = \mathtt{n}$ by Lemma 3.5.1 and (5).
2. $\mathtt{m} \mathbin{-\!\triangleright} \mathtt{0} = \mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{0} = \mathtt{m} + \mathtt{0} = \mathtt{m}$ by (6).
3. $\mathtt{m} \leq \mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{n} = \mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}$ by isotony of $+$.
4. $\ulcorner\mathtt{m} : (\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}) = \ulcorner\mathtt{m} : (\mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{n}) = \ulcorner\mathtt{m} : \mathtt{m} + \ulcorner\mathtt{m} : (\neg\ulcorner\mathtt{m} : \mathtt{n}) = \mathtt{m} + (\ulcorner\mathtt{m} \cdot \neg\ulcorner\mathtt{m}) : \mathtt{n} = \mathtt{m} + \mathtt{0} : \mathtt{n} = \mathtt{m} + \mathtt{0} = \mathtt{m}$ by (2), (4) and (3).
5. $\ulcorner(\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n}) = \ulcorner(\mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{n}) = \ulcorner\mathtt{m} + \ulcorner(\neg\ulcorner\mathtt{m} : \mathtt{n}) = \ulcorner\mathtt{m} + \neg\ulcorner\mathtt{m} \cdot \ulcorner\mathtt{n} = \ulcorner\mathtt{m} + \ulcorner\mathtt{n}$, by Lemma 3.5.4, 3.5.6 and Boolean algebra.
6. First we conclude $\ulcorner\mathtt{m} \geq \ulcorner\mathtt{n} \Rightarrow \neg\ulcorner\mathtt{m} \leq \neg\ulcorner\mathtt{n} \Rightarrow \neg\ulcorner\mathtt{m} : \mathtt{n} \leq \mathtt{0}$ by (gla). Therefore $\mathtt{m} \mathbin{-\!\triangleright} \mathtt{n} = \mathtt{m} + \neg\ulcorner\mathtt{m} : \mathtt{n} = \mathtt{m} + \mathtt{0} = \mathtt{m}$.
7. $\mathtt{l} \mathbin{-\!\triangleright} (\mathtt{m}+\mathtt{n}) = \mathtt{l}+\neg\ulcorner\mathtt{l} : (\mathtt{m}+\mathtt{n}) = \mathtt{l}+\neg\ulcorner\mathtt{l} : \mathtt{m}+\neg\ulcorner\mathtt{l} : \mathtt{n} = \mathtt{l}+\neg\ulcorner\mathtt{l} : \mathtt{m}+\mathtt{l}+\neg\ulcorner\mathtt{l} : \mathtt{n} = \mathtt{l} \mathbin{-\!\triangleright} \mathtt{m} + \mathtt{l} \mathbin{-\!\triangleright} \mathtt{n}$ by (2) and idempotence.