

MONOID MODULES AND STRUCTURED DOCUMENTS ALGEBRA

Andreas Zelend



UNA Universität
Augsburg
University

RAMiCS 15
30. September 2015

STRUCTURED DOCUMENT ALGEBRA

GOALS

- A simple yet effective algebra of structured documents such as collections of composed features
- Formal reasoning about the construction process of documents
- More general operations, notably deletion, than Feature Algebra

VARIATION POINTS AND FRAGMENTS

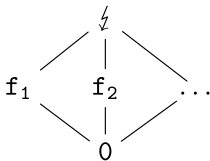
- Set V of *variation points (VPs)* at which things may be inserted
- Set $F(V)$ of *(document) fragments* which may, among other things, contain VPs from V
- every VP is a (yet unfilled) fragment by itself, i.e., $V \subseteq F(V)$
- A *text* is a fragment without VPs

VARIATION POINTS AND FRAGMENTS

- To make error handling algebraically nicer we use
 1. a default fragment 0 and
 2. an error fragment ζ
- The addition, or supremum, operator $+$ on fragments has the axioms

$$0 + x = x \quad \zeta + x = \zeta \quad f_i + f_j = \zeta (i \neq j)$$

- Together with associativity, idempotence and commutativity this structure forms a flat lattice with least element 0 and greatest element ζ



MODULES

- A *module* is a partial function $m : V \rightsquigarrow F(V)$
- VP v is *assigned* in m if $v \in \text{dom}(m)$, otherwise *unassigned* or *external*
- By using partial **functions** rather than **relations**, a VP can be filled with at most one fragment in any legal module (*uniqueness*)
- Different VPs may have assigned the same fragment to them (a module need not be an injective partial function)
- Simplest module: $\mathbf{0}$ (*empty module*)

MODULE ADDITION

- We want to construct larger modules step by step by coupling more and more VPs with fragments
- Central operation: *module addition* +
 - Fuses two modules while maintaining uniqueness (and signalling an error upon a conflict)
 - Desired properties: + should be commutative, associative and idempotent

MODULE ADDITION

- Module addition can be defined as the lifting of $+$ on fragments:

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

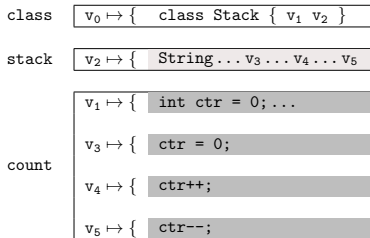
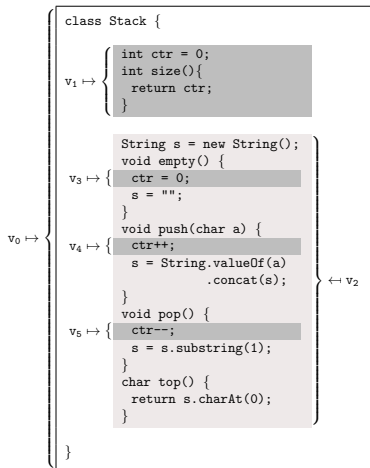
- If in the third case $m(v) \neq n(v)$ and $m(v), n(v) \neq 0$ then $(m + n)(v) = \frac{1}{2}$, thus signalling an error

MODULE ADDITION

- The set of modules with $+$ and the neutral element $\mathbf{0}$ forms a commutative monoid .
- The least element w.r.t. natural order \leq is the empty module $\mathbf{0}$ and the top element is the module t with $t(v) = \downarrow$ for any $v \in V$.
- Modules m and n are called *compatible*, in signs $m \downarrow n$, if their fragments coincide on their shared domains, i.e.,

$$m \downarrow n \iff_{df} \forall v \in \text{dom}(m) \cap \text{dom}(n) : m(v) = n(v) .$$

MODULE ADDITION



- The module addition *class + stack + count* is represented by the left module.

SUBTRACTION

- For modules m and n we define the *subtraction* – via restriction | as

$$m - n =_{df} m \upharpoonright_{\text{dom}(m) - \text{dom}(n)}$$

- This spells out to

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

ABSTRACTING FROM SDA

- Intermediate summary:
- The set M of modules with $+$ and $-$ forms an algebraic structure $SDA =_{df} (M, +, -, \mathbf{0})$ such that $(M, +, \mathbf{0})$ is an idempotent and commutative monoid and which satisfies the following laws for all $l, m, n \in M$:
 1. $(l - m) - n = l - (m + n)$,
 2. $(l + m) - n = (l - n) + (m - l)$,
 3. $\mathbf{0} - l = \mathbf{0}$,
 4. $l - \mathbf{0} = l$.

ABSTRACTING FROM SDA I

- Now we want to abstract from modules. Therefore we define *monoid modules* (m-module)
- A *monoid module* (m-module) is an algebraic structure $(B, M, :)$ where
 - $(M, +, 0)$ is an idempotent and commutative monoid,
 - $(B, +, \cdot, 0, 1, \neg)$ is a Boolean algebra in which 0 and 1 are the least and greatest element and \cdot and $+$ denote meet and join,

ABSTRACTING FROM SDA II

- The restriction, or scalar product, $:$ is a mapping $B \times M \rightarrow M$ satisfying for all $p, q \in B$ and $m, n \in M$:

$$\begin{aligned}(p + q) : m &= p : m + q : m , & (p \cdot q) : m &= p : (q : m) , \\ p : (m + n) &= p : m + p : n , & 1 : m &= m , \\ 0 : m &= 0 , & p : 0 &= 0 .\end{aligned}$$

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

MONOID MODULES

- As a consequence we have:
 1. Restriction $:$ is isotone in both arguments.
 2. $p:m \leq m$.
 3. $p:(q:m) = q:(p:m)$
- The structure $RMM = (\mathcal{P}(M), \mathcal{P}(M \times N), :)$, where $:$ is restriction, i.e., $p:m = \{(x, y) \mid x \in p \wedge (x, y) \in m\}$, forms an m -module.

PREDOMAIN MONOID MODULES

- To model subtraction we extend m-modules with the predomain operator $\lceil : M \rightarrow B$.
- A *predomain monoid module* (predomain m-module) is a structure $(B, M, :, \lceil)$ such that $(B, M, :)$ is a m-module and $\lceil : M \rightarrow B$ satisfies for all $p \in B$ and $m \in M$:

$$m \leq \lceil m : m , \qquad \lceil (p : m) \leq p .$$

PREDOMAIN MONOID MODULES

- In a predomain m -module $\ulcorner m$ is the least left preserver of m and $\neg\ulcorner m$ is the greatest left annihilator:

$$(\text{llp}) \quad \ulcorner m \leq p \iff m \leq p : m, \quad (\text{gla}) \quad p \leq \neg\ulcorner m \iff p : m \leq 0.$$

- In a predomain m -module $(B, M, :, \ulcorner)$ for all $p \in B$ and $m, n \in M$:

$$1. \quad m = 0 \iff \ulcorner m = 0,$$

$$2. \quad m \leq n \implies \ulcorner m \leq \ulcorner n,$$

$$3. \quad m = \ulcorner m : m,$$

$$4. \quad \ulcorner(m + n) = \ulcorner m + \ulcorner n,$$

$$5. \quad \ulcorner(p : m) : m = p : m,$$

$$6. \quad \ulcorner(p : m) = p \cdot \ulcorner m.$$

PREDOMAIN MONOID MODULES

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

PREDOMAIN MONOID MODULES

- SDA laws for subtractions also hold in a predomain m-module:

$$1. \quad \ulcorner (\neg \ulcorner n : m) = \ulcorner m \cdot \neg \ulcorner n$$

$$2. \quad (\neg \ulcorner n : 0 = 0)$$

$$3. \quad \neg \ulcorner l : (m + n) = \neg \ulcorner l : m + \neg \ulcorner l : n$$

$$4. \quad \neg \ulcorner (m + n) : l = \neg \ulcorner n : (\neg \ulcorner m : l)$$

$$5. \quad \neg \ulcorner 0 : m = m$$

$$6. \quad \neg \ulcorner m : m = 0$$

$$7. \quad \neg \ulcorner n : m \leq m$$

$$8. \quad m \leq n \implies \neg \ulcorner n : m = 0$$

OVERRIDING (SDA)

- Using addition and subtraction we can define *overriding* (similar to the overriding known from OOP)
- The module $m \rightarrow n$ which results from overriding n by m is defined as

$$m \rightarrow n =_{df} m + (n - m)$$

- This replaces all assignments in n for which m also provides a value
- \rightarrow is associative and idempotent with neutral element $\mathbf{0}$, but not commutative

OVERRIDING (PREDOMAIN M-MODULE)

- SDA's overriding operator $m \rightarrow n$ can also be defined in a predomain m-module: $m \rightarrow n =_{df} m + \neg \ulcorner m : n$.
- In a predomain m-module $(B, M, :, \ulcorner)$ for all $p \in B$ and $l, m, n \in M$:
 1. $0 \rightarrow n = n$,
 2. $m \rightarrow 0 = m$,
 3. $m \leq m \rightarrow n$,
 4. $m = \ulcorner m : (m \rightarrow n)$,
 5. $\ulcorner (m \rightarrow n) = \ulcorner m + \ulcorner n$,
 6. $\ulcorner m \geq \ulcorner n \Rightarrow m \rightarrow n = m$,
 7. $l \rightarrow (m + n) = (l \rightarrow m) + (l \rightarrow n)$.

TRANSFORMATIONS

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

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

- Since we don't want to allow transformations to mask errors that are related to module addition, we add the requirement

$$T(\perp) = \perp .$$

- A transformation might leave many fragments unchanged, i.e., act as the identity on them.

STRUCTURE OF TRANSFORMATIONS

- A *monoid of transformations* is a structure $F = (F, \circ, \mathbb{1})$, where F is a set of total functions $f : X \rightarrow X$ over some set X , closed under function composition \circ , and $\mathbb{1}$ the identity function.
- The pair (X, F) is called *transformation monoid* of X .

STRUCTURE OF TRANSFORMATIONS

- With this, we now can extend the list of requirements on transformations:
 1. $T \cdot (m + n) = T \cdot m + n \iff T|_{\text{ran}(n)} = \mathbb{1}|_{\text{ran}(n)} \wedge m \downarrow n$,
 2. $\mathbb{1} \cdot m = m$,
 3. $T \cdot \mathbf{0} = \mathbf{0}$.
- $\mathbf{0}$ being an annihilator means that transformations can only change existing fragments rather than create new ones.
- We define the *application equivalence* \approx of two transformations S, T by

$$S \approx T \iff_{df} \forall m : S \cdot m = T \cdot m$$

STRUCTURE OF TRANSFORMATIONS

- We define the set of fragments changed by a transformation T :
 - $T_m =_{df} \{f \in F(V) \mid T(f) \neq f\}$ the *modified fragments of T*
 - $T_v =_{df} \{T(f) \in F(V) \mid T(f) \neq f\} = \text{ran}(T|_{T_m})$ the *value set of T*
- Now we can characterise situations in which transformations can be omitted or commute:
 1. $T \cdot (S \cdot m) = S \cdot m$ if $T_m \subseteq S_m \wedge T_m \cap S_v = \emptyset$.
 2. T and S commute if $T_m \cap S_m = \emptyset \wedge T_m \cap S_v = \emptyset \wedge T_v \cap S_m = \emptyset$.

SUMMARY

- Analysed the natural order of modules
- Abstracted from SDA to a predomain monoid module
- Had a closer look at the structure of transformations
- Next step will be the addition transformations to predomain m-modules