Foundations of Coloring Algebra with Consequences for Feature-Oriented Programming

Peter Höfner[1,3], Bernhard Möller[2], Andreas Zelend[2]

[1] NICTA [2] Universität Augsburg University [3]

13th Int'l Conference on Relational and Algebraic Methods in Computer Science

20. September 2012

Motivation

The Coloring Algebra

- originally proposed by Don Batory et al.[1]
- axiomatization for feature oriented programming inspired by Kästner's Colored IDE (`CIDE`)
- no concrete models were known

Goals

- Find models of the coloring algebra
- Gain a better understanding of the algebra and its models

[1] Batory, Höfner, Kim: Feature interactions, products, and composition. (GPCE'11)

EXAMPLE (FIRE AND FLOOD CONTROL)

$$fire \times flood = fire \cdot flood + fire + flood$$

$fire \cdot flood$ indicates an interaction between the features $fire$ and $flood$ and resolves the conflicts between $fire$ and $flood$.

The Coloring Algebra

Coloring Algebra

A **CA** is a structure $(F, +, \cdot, 0)$ such that $(F, +, 0)$ is a (commutative) involutive group and $(F, \cdot)$ is a commutative, involutive semigroup. Moreover, feature interaction distributes over composition:

$$f \cdot (g + h) = f \cdot g + f \cdot h \quad \text{and} \quad (g + h) \cdot f = g \cdot f + h \cdot f$$

Hence a CA is an involutive and commutative ring without multiplicative unit.

Definition

For a given coloring algebra $(F, +, \cdot, 0)$ the **cross product** is defined by

$$f \times g =_{df} f \cdot g + f + g \ .$$

Some properties of Composition and Interaction

neutrality:	$f + 0 = f$
involution:	$f + f = 0$
annihilation (no interaction):	$f \cdot 0 = 0$
involution:	$f \cdot f = 0$

Intuition

- $+$ is feature composition, which removes all features occurring twice.
- $f \cdot g$ describes conflicts between both and offers "repairs".

Counted Stack

```
class Stack {
}
```

(a) Base

```
class Stack {
    int ctr = 0;
    int size() {
        return ctr;
    }
}
```

(c) Counter × Base

```
class Stack {

    String s = new String();
    void empty() {

        s = "";
    }
    void push(char a)  {

        s = String.valueOf(a)
               .concat(s);
    }
    void pop() {

        s = s.substring(1);
    }
    char top() {
        return s.charAt(0);
    }
}
```

(b) Stack × Base

```
      class Stack {
vp1 →   ★int ctr = 0;
         int size() {
            return ctr;
         }

vp2 →   ★String s = new String();
         void empty() {
vp3 →      ★ctr = 0;
            s = "";
         }
         void push(char a)  {
vp4 →      ★ctr++;
            s = String.valueOf(a)
                   .concat(s);
         }
         void pop() {
vp5 →      ★ctr--;
            s = s.substring(1);
         }
         char top() {
            return s.charAt(0);
         }
      }
```

(b) Counter × Stack × Base

Consequences

Lemma

A repair cannot introduce new conflicts, $i.e.$, $f \cdot g = h \Rightarrow f \cdot h = 0$.

Lemma

If $f \neq 0$ *then* $f \cdot g \neq f$, *and if* $f + g \neq 0$ *(i.e.,* $f \neq g$*) then* $f \cdot g \neq f + g$.

Hence in a full interaction $f \times g = f \cdot g + f + g$ with $f, g \neq 0$ and $f \neq g$ none of the components f, g is deleted entirely.

Lemma

Repairs are mutually exclusive: $f \cdot h_1 = g \wedge g \cdot h_2 = f \implies f = 0$.

This can be extended to finite chains. Hence Colors cannot repair each other in "cycles".

Consequences

The absence of cycles makes the divisibility relation w.r.t. $\cdot$ into a strict partial order on non-empty colors: we define

$$f < g \iff_{df} f, g \in F - \{0\} \wedge \exists h \in F : f \cdot h = g \ .$$

Lemma

Composition $+$ and interaction $\cdot$ are not isotone w.r.t. $<$.

Intuition

$f < g$ means that g is a repair of which f is a part. Hence, between repairs $<$ is a dependence relation.

Interaction Equivalence

It is useful to group colors according to their behaviour under interaction. To achieve this we define an equivalence relation $\sim$ by

$$f \sim g \iff_{df} \forall\, h : f \cdot h = g \cdot h \ .$$

The equivalence class of f under $\sim$ is denoted by $[f] =_{df} \{g \mid f \sim g\}$. Elements $f \in [0]$ are called *annihilators*, since $\forall\, g \in F : f \cdot g = 0$.

Lemma

An element of $F - \{0\}$ is an annihilator iff it is maximal w.r.t. $<$.

Interaction Equivalence

Some properties of $\sim$

(a) $\sim$ is a congruence w.r.t. $+$ and $\cdot$.

(b) $f \sim g \iff f \cdot g = 0$.

(c) Composition is cancellative w.r.t. $\sim$, i.e., $f + g \sim f + h \iff g \sim h$.

(d) $[f] = [f] + [0] =_{df} \{h + g \mid h \in [f], g \in [0]\}$.

Models—Feature Composition

- Every element has order 2 (due to involution)
- Any finite 2-group is a power of $\mathbb{Z}_2$ (the two element group).
- By the Kronecker Basis Theorem there is exactly one finite model satisfying these axioms for each of the cardinalities 2, 4, 8, $\ldots$.

Models—Feature Composition

Theorem

Every finite algebra satisfying the axioms for feature composition is isomorphic to a model that can be obtained by using symmetric difference on a power set of a finite set.

With a set B of **base colors**, $(2^B, \triangle, 0)$ satisfies the axioms for feature composition, where $\triangle$ is the symmetric difference of sets, defined, for $M, N \in 2^B$, as

$$M \triangle N =_{df} (M \cup N) - (M \cap N) .$$

A first model for CA is given by $(2^B, \triangle, \cdot, 0)$, where for sets $M, N \in 2^B$, we define $M \cdot N =_{df} 0$.

Base Colors

Definition

In general we call a color f of a CA F **base** iff it is isomorphic to a singleton set. Additionally the "empty color" 0 is a base color. The set of all base colors is again denoted by B.

If F is finitely generated, every element is a sum of base colors, i.e., for all $f \in F$

$$f = \sum_{i \in I} a_i$$

for an index set I and base colors $a_i \in B$.

Base Colors

Consequence

Due to distributivity of $\cdot$ over $+$ it is possible to reduce general interaction to that between base colors only.

$$f \cdot g \;=\; (\sum_{i \in I} a_i) \cdot (\sum_{j \in J} b_j) \;=\; \sum_{i \in I} \sum_{j \in J} (a_i \cdot b_j) \; .$$

Intuition: only the interactions (conflicts) of base features have to be considered.

A General Model for Coloring Algebra

Recapitulation

- We have a set B of base colors and set F as 2^B
- The only possibility for composition is $f + g = f \triangle g$
- Interaction need only be considered at the level of base colors

Now we assume:

- Interaction of base colors yields again a base color
- An associative interaction operator $\circ$ on B
- A special element $e \in B$ that satisfies the annihilation properties $e \circ a = e = a \circ e$ and $a \circ a = e$ for all $a \in B$

The structure $(B, \circ, e)$ is called a **base color semigroup**.

A General Model for Coloring Algebra

Based on that, feature interaction $(\cdot)$ can be defined as

$$f \cdot g \ =_{df} \ \mathop{\triangle}_{a \in f} \mathop{\triangle}_{b \in g} \iota(a \circ b) \ ,$$

where the injection $\iota : B \rightarrow F$ is given by $\iota(e) = 0$ and $\iota(a) = \{a\}$ for $a \in B - \{e\}$.

A General Model for Coloring Algebra

A concrete definition of this operation could be:

- Assume a finite set P of **pigments**.
- Base Colors are certain sets of pigments.
- Colors are then sets of base colors.

Formally, a **set of base colors** is a non-empty subset $B \subseteq 2^P$ that is downward closed: $a \in B \wedge b \subseteq a \implies b \in B$. Hence every set of base colors contains $\emptyset$. The set B is called **full** iff $B = 2^P$.

A General Model for Coloring Algebra

For two base colors (sets of pigments) $a, b \in B$ we define a no-conflict predicate $noconf$ by

$$noconf(a,b) \iff_{df} a \neq \emptyset \wedge b \neq \emptyset \wedge a \cap b = \emptyset \ .$$

The interaction $\circ : B \times B \to B$ of base colors is defined by

$$a \circ b =_{df} \begin{cases} a \cup b & \text{if } noconf(a,b) \wedge a \cup b \in B \ , \\ \emptyset & \text{otherwise.} \end{cases}$$

A General Model for Coloring Algebra

Theorem

The structure $(B, \circ, \emptyset)$ is a base color semigroup that can be used to create a CA with the feature composition $f + g = f \triangle g$ and the feature interaction $f \cdot g =_{df} \bigtriangleup_{a \in f} \bigtriangleup_{b \in g} \iota(a \circ b)$.

Example

Assume three pigments r, g, b and $B = 2^{\{r,g,b\}}$. We can define a color that consists of all base colors containing the pigment r as

$$red =_{df} \{\{r\}, \{r, g\}, \{r, b\}, \{r, g, b\}\} = r + rg + rb + rgb \ .$$

As an example how interaction $\cdot$ on colors works, consider

$$\begin{aligned} & (r + rg + rb + rgb) \cdot (b + rb + g) \\ = \ & rb + \emptyset + rg + rgb + \emptyset + \emptyset + \emptyset + \emptyset + rbg + \emptyset + \emptyset + \emptyset \\ = \ & rb + rg \ . \end{aligned}$$

A Model with Variation Points

Idea: The elements are now total functions $p : VP \to 2^{VP \cup C}$ which map each variation point in VP to a set of code fragments (C) and variation points (VP).

Now, semantically : if $p(\mathtt{vp}) \neq \emptyset$ and $\mathtt{vp} \in VP$, then the value of $p(\mathtt{vp})$ is installed at variation point $\mathtt{vp}$; else the variation point remains empty.

We only consider code fragments that commutate, e.g, entire functions or field declarations.

A Model with VariatiMon Points

We define composition and interaction pointwise:

$$(p+q)(\mathtt{vp}) =_{df} p(\mathtt{vp}) \bigtriangleup q(\mathtt{vp})$$

$$(p \cdot q)(\mathtt{vp}) =_{df} p(\mathtt{vp}) \cdot q(\mathtt{vp})$$

In the concrete model this turns into

$$(p \cdot q)(\mathtt{vp}) =_{df} \bigtriangleup_{a \in p(\mathtt{vp})} \bigtriangleup_{b \in q(\mathtt{vp})} \iota(a \circ b)$$

with the pigment set $P =_{df} VP \cup C$ and the full base color set $B =_{df} 2^P$.

From the Model to FOP

Example

To build a program from the functions $a(\mathtt{vp}), b(\mathtt{vp}), c(\mathtt{vp})$ we

- Build the cross product $p(\mathtt{vp}) = a(\mathtt{vp}) \times b(\mathtt{vp}) \times c(\mathtt{vp})$
- Replace all occurrences of each `vp` by its value (only possible if there are no cycles)
- Choose one variation point as the `start` point

From the Model to FOP

$$\mathit{white}(\mathtt{vp}) = \begin{cases} \{\,\texttt{class Stack}\{\ \mathtt{vp}_1\ \mathtt{vp}_2\ \}\,\} & \text{if } \mathtt{vp} = \texttt{start} \\ \emptyset & \text{otherwise .} \end{cases}$$

$$\mathit{green}(\mathtt{vp}) = \begin{cases} \{\,\texttt{String S = ...} \\ \quad \mathtt{vp}_3 \ldots \\ \quad \mathtt{vp}_4 \ldots \\ \quad \mathtt{vp}_5 \ldots \qquad \} & \text{if } \mathtt{vp} = \mathtt{vp}_2 \\ \emptyset & \text{otherwise .} \end{cases}$$

$$\mathit{blue}(\mathtt{vp}) = \begin{cases} \{\,\texttt{int ctr = 0; ... ctr;}\}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_1 \\ \{\,\texttt{ctr = 0;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_3 \\ \{\,\texttt{ctr++;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_4 \\ \{\,\texttt{ctr--;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_5 \\ \emptyset & \text{otherwise .} \end{cases}$$

$$p(\mathtt{vp}) = \begin{cases} \{\,\texttt{class Stack}\{\ \mathtt{vp}_1\ \mathtt{vp}_2\ \}\,\} & \text{if } \mathtt{vp} = \texttt{start} \\ \{\,\texttt{String S = ...}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_2 \\ \{\,\texttt{int ctr = 0; ... ctr;}\}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_1 \\ \{\,\texttt{ctr = 0;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_3 \\ \{\,\texttt{ctr++;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_4 \\ \{\,\texttt{ctr--;}\,\} & \text{if } \mathtt{vp} = \mathtt{vp}_5 \\ \emptyset & \text{otherwise .} \end{cases}$$

```
     class Stack {
vp1 → ★int ctr = 0;
       int size() {
         return ctr;
       }

vp2 → ★String s = new String();
       void empty() {
vp3 →    ★ctr = 0;
         s = "";
       }
       void push(char a)  {
vp4 →    ★ctr++;
         s = String.valueOf(a)
                .concat(s);
       }
       void pop() {
vp5 →    ★ctr--;
         s = s.substring(1);
       }
       char top() {
         return s.charAt(0);
       }
     }
```

Conclusion and Outlook

Conclusion

- Composition is always isomorphic to symmetric difference in a set model
- We presented a set based model of a Coloring algebra
- We presented a concrete model for Feature Orientation

Outlook

- The repair of two base colors is **always** a base color itself
- Hence the only freedom to define interaction is given by the underlying base color semigroup