

Lectures on Module Algebra

Jan Heering

Department of Software Engineering

CWI

Amsterdam

Jan.Heering@cwi.nl

www.cwi.nl/~jan

Joint work with Jan Bergstra and Paul Klint

November 12–13, 2002

Table of contents

- Introduction to/history of modularization
- Algebraic preliminaries
- Algebra of signatures
- Module algebra
- The role of the interpolation property
- Literature

Modularization

- information hiding
- abstraction
- complexity reduction
- dependency localization
- reusability
- independent development

Modularization

Fundamental subject—principles widely used in

- **mathematics** (mathematical structures)
L. Corry, Modern Algebra and the Rise of Mathematical Structures, Birkhäuser, 1996
- **(electronic) hardware** (before software, but not systematic)
- **computer software (modularization central subject in SE)**
- living beings (organs, organ transplants)
- large organizations
- business processes
- ...

Modularization

- How to modularize? (SE theory/practice)
 - may be hard
 - not necessarily unique (different perspectives)
 - not necessarily supported by language/framework
(cross-cutting concerns)
 - transfer modularization from problem domain to application software (example: [coordinate-free numerics](#), M. Haverlaen et al., *Scientific Programming* **8** (2000) 4)
- What modularization constructs are needed?
(language/framework design)
- **Semantics of modularization constructs**
(language/framework design)
- Separate processing of modules and linking (implementation)

History

Very fragmentary—huge amount of work has been/is being done

- subroutines/functions
- abstract datatypes (ADTs)
- classes \implies OOP
- components
- **module operators**
- aspects/cross-cutting concerns
- control abstraction/coordination

Some further details ...

Subroutines/functions

Oldest modularization construct in high-level PL (Fortran)

- parameter passing (by value, by reference)
- local variables (initially static—no recursion)
- libraries
- separate compilation (retain modularity at object code level)
- (static) subroutine linking

Abstract datatypes (ADTs)

- hiding of implementation/realization
- user-definable types
- packaging of elements and operations in type definition
- type definition viewed as module
- **algebraic specification** (modularization important issue)
- **mainstream: class concept**

Classes

Initially Simula, Smalltalk; later mainstream: C++, Java, C#, ...

- class instance
- method
- class library
- subclass/derived class
- inheritance
- virtual function
- parameterized class/template

2001 Turing Award for Dahl and Nygaard well-deserved, but
why so late?

Components

Hot topic: COM, JavaBeans, Corba, .NET, J2EE, ...

- independent extensibility
- language interoperability
- versioning
- globally unique identifier (GUID)
- strong name
- interface definition language (IDL)
- marshaling
- automation

See: C. Szyperski, [Component Software: Beyond Object-Oriented Programming](#), Addison-Wesley, 2nd ed., 2002

Module operators

- Σ : get interface/signature of module
signature is set of typed function declarations—defined later
- \square : export signature from module
part of signature that is not exported is hidden
- $+$: combine modules
- \cdot : rename signature elements of module
 - to avoid name clashes of signature elements
 - to force name clashes (binding)
- actualize parameter of module
- ...

Module expression

Expression built from module operators

Informal examples

$$x \square ((r_1 \cdot X) + (r_2 \cdot Y))$$

with x a signature, r_1, r_2 renamings, and X, Y modules

$$\Sigma(Y) \square (r \cdot X)$$

with r a renaming and X, Y modules

Normal form result for module expressions—we need **laws**

Normalization is **basic operation** in system for manipulating modular specifications

Laws of modularization

valid equations $X = Y$ with X, Y open module expressions

open module expression is module expression with variables

variables in equations implicitly universally quantified

laws may be conditional

- intuitively: laws largely independent of what is in modules, but this is to be investigated
- have been investigated in context of logically simpler specification languages rather than PLs
- Clear, OBJ, ASF+SDF, Larch, Maude, ...
- module algebra

Some plausible laws

$$X + Y = Y + X \quad \text{commutativity of } +$$

$$x \sqcap (y \sqcap Z) = (x \cap y) \sqcap Z \quad \cap \text{ intersection of signatures}$$

$$x \sqcap (Y + Z) = (x \sqcap Y) + (x \sqcap Z) \quad \text{distributivity of } \sqcap$$

- plausible but **not necessarily valid**—to be investigated
- might depend on
 - what logic (eql, fol, ...) is used in module and/or
 - what meaning is assigned to module for given logic and/or
 - specific axioms in module
- it turns out some laws reflect **subtle properties of the logic used (interpolation)**

Algebraic preliminaries

- signatures, expressions/terms, equations
- (conditional) equational logic (eql, ceql)
- equational theory, algebra, model, model class
- soundness and completeness
- algebraic specification
- example: module *Booleans*
- first-order logic with equality (fol)
- no need for category theory (yet)

Signatures, expressions/terms, equations

(Many-sorted) **signature** declares (abstract) expression syntax

- set of **sort** and **typed constant/function** declarations
- **constant** is function of arity 0
- **type** is sequence of ≥ 1 sorts declared in signature

sorts N, L

functions

0 : N *overloaded*

0 : L *overloaded*

S : $N \rightarrow N$ *overloaded*

i : $N \rightarrow L$

S : $L \rightarrow L$ *overloaded*

f : $L \times L \rightarrow L$

Expressions/terms

Signature defines **language of expressions/terms**

Overloading may lead to **ambiguous expressions**—attach type to each symbol to disambiguate

Some **correct** (open) expressions over example signature

$$S^{N \rightarrow N}(0^N)$$

$$S^{L \rightarrow L}(f^{L \times L \rightarrow L}(k^L, l^L))$$

with **variables** k, l

Some **incorrect** expressions

0 *not explicitly typed*

$S^{L \rightarrow L}(0^N)$ *incorrectly typed*

$f^{L \times L \rightarrow L}(0^L)$ *f not monadic*

Equations

$$S^{L \rightarrow L}(0^L) = 0^L$$

$$S^{L \rightarrow L}(i^{N \rightarrow L}(n^N)) = i^{N \rightarrow L}(S^{N \rightarrow N}(n^N))$$

$$S^{L \rightarrow L}(f^{L \times L \rightarrow L}(k^L, l^L)) = f^{L \times L \rightarrow L}(S^{L \rightarrow L}(k^L), S^{L \rightarrow L}(l^L))$$

with variables k, l, n **implicitly universally quantified** (identities)

Explicit typing **safe**, but usually largely redundant

$$S(0^L) = 0 \quad \textit{retain explicit typing of 0}$$

$$S(i(n)) = i(S(n))$$

$$S(f(k, l)) = f(S(k), S(l))$$

would be sufficient

Equational logic (eql, ceql)

Equational derivability familiar from high school algebra

Signature Σ , set of Σ -equations (axioms) E , Σ -equation e :

$E \vdash_{\text{eql}} e$ if e derivable from E using proof rules

$$t = t \quad \textit{reflexivity}$$

$$\frac{t_1 = t_2}{t_2 = t_1} \quad \textit{symmetry}$$

$$\frac{t_1 = t_2, t_2 = t_3}{t_1 = t_3} \quad \textit{transitivity}$$

$$\frac{t_1 = t_2}{C[t_1] = C[t_2]} \quad \textit{congruence}$$

$$\frac{t_1 = t_2}{\sigma(t_1) = \sigma(t_2)} \quad \textit{substitution}$$

Positive conditional Σ -equation $e_1 \& \dots \& e_n \implies e$ ($n \geq 1$)

\vdash_{ceql} needs additional rules for $\&$, \implies , and (sometimes) \exists
(later when we need them)

Equational theory, algebra, model

Signature Σ , set of (conditional) Σ -equations (axioms) E

Equational theory of (Σ, E) is set of Σ -equations derivable from E

Algebra is finite collection of **non-empty sets** (carriers) and **total functions** between them

Σ -algebra A

- sort $s \in \Sigma \longrightarrow$ carrier C_s of A
- function $f \in \Sigma \longrightarrow$ function F_f of A such that
type of f as declared in $\Sigma \longrightarrow$ type of F_f in A

Σ -algebra is **model** of (Σ, E) if the equations of E interpreted in A are valid ($A \models E$)

Group is model of group axioms

(Full) model class of (Σ, E) is class of all models of (Σ, E)

Soundness and completeness

Soundness of equational logic

If $E \vdash e$ then $A \models e$ for all models A of E

Completeness

If $A \models e$ for all models A of E then $E \vdash e$

Completeness useful for **independence proofs**

Given $e \in E$, can it be omitted, that is, $E \setminus \{e\} \vdash e$?

Try to construct model of $E \setminus \{e\}$ in which e is not valid

If successful, e is **independent**

Examples later

Algebraic specification

- Modular specification of ADTs
- Module is signature Σ + set E of (conditional) Σ -equations
- Execution mechanism: term rewriting
 - apply equations from left to right
 - may be incomplete (no proper normal forms)
 - too simple for module algebra
- Semantics
 - equational theory (or only closed equations)
 - loose semantics (some model class)
 - initial algebra (Σ -terms modulo equality generated by E)
- Study of module expressions

Example: Booleans

module *Booleans*

signature

sort *BOOL*

functions

T : *BOOL*

F : *BOOL*

\neg : *BOOL* \rightarrow *BOOL*

\vee : *BOOL* \times *BOOL* \rightarrow *BOOL*

\wedge : *BOOL* \times *BOOL* \rightarrow *BOOL*

variables X, Y, Z : *BOOL*

Functions T, F are **constants**

Booleans (cont.)

equations

$$\neg T = F$$

$$\neg F = T$$

$$T \vee X = T$$

$$F \vee X = X$$

$$X \wedge Y = \neg(\neg X \vee \neg Y)$$

end *Booleans*

$E_{Booleans} \vdash e = T$ or $E_{Booleans} \vdash e = F$ for every **closed** expression e over $\Sigma_{Booleans}$

Every closed expression over $\Sigma_{Booleans}$ has **normal form** T or F

Well-known **disjunctive normal form** result for **open** expressions requires additional laws

Example: Booleans (cont.)

Some additional laws of Boolean algebra required for disjunctive normal form result for open expressions

$$\neg\neg X = X$$

$$X \vee \neg X = T$$

$$(X \vee Y) \vee Z = X \vee (Y \vee Z) \quad \textit{associativity}$$

$$X \vee Y = Y \vee X \quad \textit{commutativity}$$

$$X \vee X = X \quad \textit{idempotency}$$

$$(X \vee Y) \wedge Z = (X \wedge Z) \vee (Y \wedge Z) \quad \textit{distributivity}$$

These are **independent of $E_{Booleans}$**

Example: Booleans (cont.)

We give model of *Booleans* in which some of them are not valid
Algebra M with elements T, F, U such that $M \models E_{Booleans}$ and

$$\neg U = U$$

$$U \vee X = U$$

$M \not\models$

$$X \vee \neg X = T$$

$$X \vee Y = Y \vee X$$

First-order logic with equality (fol)

- extension of eql: **fol = eql +**
 - \exists existential quantifier
 - $\neg, \&, \vee, \implies$ negation, conjunction, disjunction, implication
 - true, false** built-in predicates
- free vars implicitly universally quantified like in eql
- additional proof rules later when we need them
- notions of theory, model and model class similar to eql
- model class may be empty (inconsistency), for example: **false**
- ceql somewhere between eql and fol

fol specification is

signature Σ (function decls, no predicates) + set of fol Σ -sentences

Algebra of signatures

- This is where module algebra started: signature is **module without axioms**
- **Basic ingredients**
 - names
 - types
 - atomic signatures
 - renamings
- Algebra of signatures has itself a signature—**don't confuse them**
- Normal form result for closed signature expressions

Algebraic specification of signatures

First: **names (identifiers)**—just natural numbers with equality and renaming

module *Signatures*

import *Booleans*

sort *NAMES*

functions

0 : *NAMES*

N : *NAMES* \rightarrow *NAMES* *name constructor*

eq : *NAMES* \times *NAMES* \rightarrow *BOOL* *equality*

σ : *NAMES* \times *NAMES* \times *NAMES* \rightarrow *NAMES*
elementary renaming

variables l, m, n : *NAMES*

Names (cont.)

equations

$$eq(0, 0) = T$$

$$eq(0, N(l)) = F$$

$$eq(N(l), 0) = F$$

$$eq(N(l), N(m)) = eq(l, m)$$

$$\sigma(l, m, l) = m \quad | \text{ renaming is}$$

$$\sigma(l, m, m) = l \quad | \text{ permutative}$$

$$eq(l, n) = F \ \& \ eq(m, n) = F \implies$$

$$\sigma(l, m, n) = n$$

Last equation is **conditional** (ceql)

Types

Types are sequences of names—last one acts as output type

sort *TYPES*

functions

i	$: \text{NAMES} \rightarrow \text{TYPES}$	<i>injection</i>
$*$	$: \text{TYPES} \times \text{TYPES} \rightarrow \text{TYPES}$	<i>concatenation</i>
σ	$: \text{NAMES} \times \text{NAMES} \times \text{TYPES} \rightarrow \text{TYPES}$	<i>renaming</i>
\in	$: \text{NAMES} \times \text{TYPES} \rightarrow \text{BOOL}$	<i>membership</i>
eq	$: \text{TYPES} \times \text{TYPES} \rightarrow \text{BOOL}$	<i>equality</i>

variables $l, m, n : \text{NAMES}$

$t, u, v : \text{TYPES}$

σ and eq are **overloaded**, but no ambiguities

Types (cont.)

equations

$$(t * u) * v = t * (u * v)$$

$$\sigma(l, m, i(n)) = i(\sigma(l, m, n))$$

$$\sigma(l, m, t * u) = \sigma(l, m, t) * \sigma(l, m, u)$$

$$l \in i(m) = eq(l, m)$$

$$l \in (t * u) = (l \in t) \vee (l \in u)$$

$$eq(i(l), i(m)) = eq(l, m)$$

$$eq(i(l) * t, i(m) * u) = eq(l, m) \wedge eq(t, u)$$

$$eq(i(l), t * u) = F$$

$$eq(t * u, i(l)) = F$$

Atomic signatures

sort *ATSIG*

functions

S : $NAMES \rightarrow ATSIG$ *sort constructor*

F : $NAMES \times TYPES \rightarrow ATSIG$ *function constructor*

eq : $ATSIG \times ATSIG \rightarrow BOOL$ *equality*

variables $l, m : NAMES$

$t, u : TYPES$

equations

$$eq(\mathbf{S}(l), \mathbf{S}(m)) = eq(l, m)$$

$$eq(\mathbf{S}(l), \mathbf{F}(m, t)) = F$$

$$eq(\mathbf{F}(l, t), \mathbf{S}(m)) = F$$

$$eq(\mathbf{F}(l, t), \mathbf{F}(m, u)) = eq(l, m) \wedge eq(t, u)$$

Atomic renamings

sort *ATREN*

functions

$rs : NAMES \times NAMES \rightarrow ATREN$

sort renaming constructor

$rf : NAMES \times NAMES \times TYPES \rightarrow ATREN$

function renaming constructor

$\cdot : ATREN \times ATSIG \rightarrow ATSIG$

apply atomic renaming

variables $l, m, n : NAMES$

$t, u : TYPES$

Atomic renamings (cont.)

equations

$$rs(l, l) = rs(m, m) \quad | \textit{ identify}$$

$$rs(m, m) = rf(l, l, t) \quad | \textit{ all}$$

$$rf(l, l, t) = rf(m, m, u) \quad | \textit{ identity renamings}$$

$$rs(l, m) = rs(m, l)$$

$$rf(l, m, t) = rf(m, l, t)$$

$$rs(l, m) \cdot \mathbf{S}(n) = \mathbf{S}(\sigma(l, m, n))$$

$$rs(l, m) \cdot \mathbf{F}(n, t) = \mathbf{F}(n, \sigma(l, m, t))$$

$$rf(l, m, t) \cdot \mathbf{F}(n, t) = \mathbf{F}(\sigma(l, m, n), t)$$

$$eq(t, u) = F \implies$$

$$rf(l, m, t) \cdot \mathbf{F}(n, u) = \mathbf{F}(n, u)$$

$$rf(l, m, t) \cdot \mathbf{S}(n) = \mathbf{S}(n)$$

Finally: Signatures

sort *SIG*

functions

\emptyset	: SIG	<i>empty signature</i>
i	: $ATSIG \rightarrow SIG$	<i>injection</i>
$+$: $SIG \times SIG \rightarrow SIG$	<i>combination/union</i>
\mathbf{S}	: $TYPES \rightarrow SIG$	<i>convert type to set of sorts</i>
\cdot	: $ATREN \times SIG \rightarrow SIG$	<i>apply atomic renaming</i>
Σ	: $ATREN \rightarrow SIG$ signature	<i>affected by atomic renaming</i>
$\text{inv}\Sigma$: $ATREN \rightarrow SIG$ signature	<i>used by but invariant under atomic renaming</i>

Signatures (cont.)

\in : $ATSIG \times SIG \rightarrow BOOL$ *membership*

\cap : $SIG \times SIG \rightarrow SIG$ *intersection*

Δ : $ATSIG \times SIG \rightarrow SIG$ *deletion*

\supseteq : $SIG \times SIG \rightarrow BOOL$ *supersignature*

eq : $SIG \times SIG \rightarrow BOOL$ *equality*

variables $l, m : NAMES$

$t, u : TYPES$

$a : ATSIG$

$r : ATREN$

$x, y, z : SIG$

Signatures (cont.)

equations

$$x + \emptyset = x$$

$$x + x = x$$

$$x + y = y + x$$

$$(x + y) + z = x + (y + z)$$

$$i(\mathbf{F}(l, t)) = i(\mathbf{F}(l, t)) + \mathbf{S}(t) \quad \text{a function implicitly}$$

declares the sorts (≥ 1) occurring in its type

$$\mathbf{S}(i(l)) = i(\mathbf{S}(l))$$

$$\mathbf{S}(t * u) = \mathbf{S}(t) + \mathbf{S}(u)$$

Signatures (cont.)

$$r \cdot \emptyset = \emptyset$$

$$r \cdot i(a) = i(r \cdot a)$$

$$r \cdot (x + y) = (r \cdot x) + (r \cdot y)$$

$$\Sigma(rs(l, l)) = \emptyset \quad \text{this catches all identity renamings}$$

$$eq(l, m) = F \implies$$

$$\Sigma(rs(l, m)) = i(\mathbf{S}(l)) + i(\mathbf{S}(m))$$

$$eq(l, m) = F \implies$$

$$\Sigma(rf(l, m, t)) = i(\mathbf{F}(l, t)) + i(\mathbf{F}(m, t))$$

$$\text{inv}\Sigma(rs(l, m)) = \emptyset$$

$$eq(l, m) = F \implies$$

$$\text{inv}\Sigma(rf(l, m, t)) = \mathbf{S}(t)$$

Signatures (cont.)

$$a \in \emptyset = F$$

$$\mathbf{S}(l) \in i(\mathbf{S}(m)) = eq(l, m)$$

$$\mathbf{S}(l) \in i(\mathbf{F}(m, t)) = l \in t$$

$$\mathbf{F}(l, t) \in i(\mathbf{F}(m, u)) = eq(l, m) \wedge eq(t, u)$$

$$\mathbf{F}(l, t) \in i(\mathbf{S}(m)) = F$$

$$a \in (x + y) = (a \in x) \vee (a \in y)$$

$$x \cap \emptyset = \emptyset$$

$$x \cap x = x$$

$$x \cap y = y \cap x$$

$$(x \cap y) \cap z = x \cap (y \cap z)$$

Signatures (cont.)

$$\mathbf{S}(l) \in x = F \implies$$

$$i(\mathbf{S}(l)) \cap x = \emptyset$$

$$\mathbf{F}(l, t) \in x = F \implies$$

$$i(\mathbf{F}(l, t)) \cap x = \mathbf{S}(t) \cap x$$

$$a \in x = T \implies$$

$$i(a) \cap x = i(a)$$

$$(x + y) \cap z = (x \cap z) + (y \cap z)$$

$$a \in x = F \implies a\Delta x = x$$

$$a\Delta i(a) = \emptyset \quad | \textit{incorrect}$$

$$\mathbf{S}(l)\Delta i(\mathbf{S}(l)) = \emptyset \quad | \textit{correction to 1990}$$

$$\mathbf{F}(l, t)\Delta i(\mathbf{F}(l, t)) = \mathbf{S}(t) \quad | \textit{JACM article p. 344}$$

Signatures (cont.)

$$l \in t = T \implies$$

$$\mathbf{S}(l) \Delta i(\mathbf{F}(m, t)) = \mathbf{S}(l) \Delta \mathbf{S}(t)$$

$$a \Delta (x + y) = (a \Delta x) + (a \Delta y)$$

$$x = y + z \implies$$

$$x \supseteq y = T$$

$$a \in y = T \ \& \ a \in x = F \implies$$

$$x \supseteq y = F$$

$$eq(x, y) = (x \supseteq y) \wedge (y \supseteq x)$$

end *Signatures*

Signatures: Incorrect equation

Equation $a\Delta i(a) = \emptyset$ leads to (weak) inconsistency

On the one hand

$$\mathbf{F}(n, i(m))\Delta i(\mathbf{F}(n, i(m))) = \emptyset$$

On the other hand

$$\mathbf{F}(n, i(m))\Delta i(\mathbf{F}(n, i(m))) = (\text{expand with sort declaration})$$

$$\mathbf{F}(n, i(m))\Delta\{i(\mathbf{F}(n, i(m)) + \mathbf{S}(i(m)))\} = (\text{distribute } \Delta \text{ over } +)$$

$$(\mathbf{F}(n, i(m))\Delta i(\mathbf{F}(n, i(m)))) + (\mathbf{F}(n, i(m))\Delta i(\mathbf{S}(m))) =$$

$$\emptyset + (\mathbf{F}(n, i(m))\Delta i(\mathbf{S}(m))) = (\text{since } \mathbf{F}(n, i(m)) \in i(\mathbf{S}(m)) = F)$$

$$i(\mathbf{S}(m))$$

$$\text{Hence } i(\mathbf{S}(m)) = \emptyset$$

Normal form for signatures

For every **closed** signature expression e of sort SIG

$$E_{Signatures} \vdash e = \sum_{k=1}^m i(\mathbf{S}(s_k)) + \sum_{k=1}^n i(\mathbf{F}(f_k, t_k)) \quad (m, n \geq 0)$$

with $s_k \neq s_l$ ($k \neq l$), $(f_k, t_k) \neq (f_l, t_l)$ ($k \neq l$), and $s_k \notin t_l$

Only sorts not occurring in the type of any function are declared explicitly in the normal form

The other ones need not be declared because of the equation

$$i(\mathbf{F}(l, t)) = i(\mathbf{F}(l, t)) + \mathbf{S}(t)$$

Two signatures are equal if and only if their normal forms are syntactically equal modulo associativity and commutativity of $+$ and modulo associativity of $*$

Module algebra

- Postulate plausible laws and investigate their validity for
 - different logics (eql, ceql, fol) and/or
 - what meaning (textual/presentation, model class, theory) is assigned to module for given logic
- We do not consider laws that depend on the specific axioms in a module: **structure of axioms not specified (constants)**
- Logic used in modules is **parameter** of specification
 - *BMA*[eql]: basic algebra of modular algebraic specifications
 - *BMA*[fol]: basic algebra of modular first-order logic specifications
- For the time being: **forget about fol or eql**
(but we start with fol)

Basic Module Algebra

Design requirements

- (A) **Natural semantics:** All laws of $BMA[\text{fol}]$ have to hold if modules are interpreted as model classes
- (B) **Persistence of signatures:** Extending *Signatures* to $BMA[\text{fol}]$ should not lead to new signatures or to new equalities between signatures
- (C) **Normal form result:** every closed module expression must be provably equal to a normal form containing only a single instance of the export operator \square
 - multiple levels of export can be eliminated
 - makes radical form of inlining possible
- (D) **Enrichment must be special case of extension:** explained later

Specification of $BMA[\text{fol}]$

module $BMA[\text{fol}]$

import $Signatures$

sort M

functions

$\langle \phi \rangle$: M	<i>constants for each fol</i>
		<i>sentence $\phi(x)$ over signature x</i>
Σ	: $M \rightarrow SIG$	<i>signature</i>
\mathbf{T}	: $SIG \rightarrow M$	<i>injection</i>
\cdot	: $ATREN \times M \rightarrow M$	<i>apply atomic renaming</i>
$+$: $M \times M \rightarrow M$	<i>combination/union</i>
\square	: $SIG \times M \rightarrow M$	<i>export</i>

variables $r : ATREN$ $x, y : SIG$ $X, Y, Z : M$

BMA[fol] (cont.)

equations

$$\Sigma(\langle \phi \rangle) = \Sigma(\phi) \quad (\text{S1})$$

$$\Sigma(\mathbf{T}(x)) = x \quad (\text{S2})$$

$$\Sigma(X + Y) = \Sigma(X) + \Sigma(Y) \quad (\text{S3})$$

$$\Sigma(x \square Y) = x \cap \Sigma(Y) \quad (\text{S4})$$

$$\Sigma(r \cdot X) = r \cdot \Sigma(X) \quad (\text{S5})$$

$$r \cdot \langle \phi \rangle = \langle r \cdot \phi \rangle \quad (\text{R1})$$

$$r \cdot \mathbf{T}(x) = \mathbf{T}(r \cdot x) \quad (\text{R2})$$

$$r \cdot (X + Y) = (r \cdot X) + (r \cdot Y) \quad (\text{R3})$$

$$r \cdot (x \square Y) = (r \cdot x) \square (r \cdot Y) \quad (\text{R4})$$

$$r \cdot (r \cdot X) = X \quad (\text{R5})$$

$$\text{inv}\Sigma(r) \supseteq \Sigma(r) \cap \Sigma(X) = T \implies r \cdot X = X \quad (\text{R6}')$$

BMA[fol] (cont.)

$$X + Y = Y + X \quad (\text{C1})$$

$$(X + Y) + Z = X + (Y + Z) \quad (\text{C2})$$

$$\mathbf{T}(x + y) = \mathbf{T}(x) + \mathbf{T}(y) \quad (\text{C3})$$

$$X + \mathbf{T}(\Sigma(X)) = X \quad (\text{C4})$$

$$X + (y \square X) = X \quad (\text{C5})$$

$$\Sigma(X) \square X = X \quad (\text{E1})$$

$$x \square (y \square Z) = (x \cap y) \square Z \quad (\text{E2})$$

$$x \square (\mathbf{T}(y) + Z) = \mathbf{T}(x \cap y) + (x \square Z) \quad (\text{E3})$$

$$x \supseteq (\Sigma(Y) \cap \Sigma(Z)) = T \implies$$

$$x \square (Y + Z) = (x \square Y) + (x \square Z) \quad (\text{E4})$$

end BMA[fol]

$$r \cdot (x \square Y) = (r \cdot x) \square (r \cdot Y) \quad (\mathbf{R4})$$

Postulates **unrestricted** distribution of renaming over export

Permutative character of renaming crucial: let

$$X = (\mathbf{S} : A + \mathbf{F} : a : A) \square \langle a^A \neq b^A \rangle$$

(with slight change in notation of sort/function decls)

Non-permutative renaming of a to b yields inconsistent result because of **accidental name clash with hidden name b :**

$$(\mathbf{S} : A + \mathbf{F} : b : A) \square \langle b^A \neq b^A \rangle$$

Permutative renaming of a to b avoids name clash by **renaming b safely out of the way:**

$$rf(a, b, A) \cdot X = (\text{with (R4) above})$$

$$(rf(a, b, A) \cdot (\mathbf{S} : A + \mathbf{F} : a : A)) \square (rf(a, b, A) \cdot \langle a^A \neq b^A \rangle) =$$

$$(\mathbf{S} : A + \mathbf{F} : b : A) \square \langle b^A \neq a^A \rangle$$

$$\text{inv}\Sigma(r) \supseteq (\Sigma(r) \cap \Sigma(X)) = T \implies r \cdot X = X \text{ (R6')}$$

Postulates **restricted** renameability of hidden items

Condition forbids renaming of

- visible items
- hidden items causing clash between hidden and visible names

No clashes between hidden names because of **permutative character of renaming**

(See 1990 JACM article p. 347 for (R6))

$$X + (y \square X) = X \quad (\text{C5})$$

Requirement (D): enrichment must be special case of extension

Definitions:

- Y enrichment of X if Y covers more issues than X without in any way changing or constraining the meaning of X :

$$X = \Sigma(X) \square Y$$

(X abstraction of Y)

- Y extension of X if Y describes more than X in a way consistent with X and perhaps even in a more specific way than X :

$$Y = Y + X$$

Proof of requirement (D): if $X = \Sigma(X) \square Y$ then

$$Y + X = Y + (\Sigma(X) \square Y) = Y \quad (\text{with (C5) above})$$

$$x \square (\mathbf{T}(y) + Z) = \mathbf{T}(x \cap y) + (x \square Z) \quad (\mathbf{E3})$$

Postulates that **hidden parts of the signature not used in any axiom** may be deleted

Special case: $x \square \mathbf{T}(y) = \mathbf{T}(x \cap y)$

Proof: $x \square \mathbf{T}(y) = x \square \mathbf{T}(y + \emptyset) =$ (with distributive law (C3))

$x \square (\mathbf{T}(y) + \mathbf{T}(\emptyset)) =$ (with (E3) above)

$\mathbf{T}(x \cap y) + (x \square \mathbf{T}(\emptyset)) =$ (with $x \square \mathbf{T}(\emptyset) = \mathbf{T}(\emptyset)$ —see next)

$\mathbf{T}(x \cap y) + \mathbf{T}(\emptyset) =$ (with $X + \mathbf{T}(\emptyset) = X$ —see next)

$= \mathbf{T}(x \cap y)$

(E3) (cont.)

We used

$$X + \mathbf{T}(\emptyset) = X \quad (\mathbf{T}(\emptyset) \text{ is neutral element for } + \text{ on } M)$$

$$x \square \mathbf{T}(\emptyset) = \mathbf{T}(\emptyset)$$

Proof: $X + \mathbf{T}(\emptyset) =$ (with (C4))

$$(X + \mathbf{T}(\Sigma(X))) + \mathbf{T}(\emptyset) = \text{(with (C2))}$$

$$X + (\mathbf{T}(\Sigma(X)) + \mathbf{T}(\emptyset)) = \text{(with (C3))}$$

$$X + \mathbf{T}(\Sigma(X) + \emptyset) = X + \mathbf{T}(\Sigma(X)) = \text{(with (C4)) } X$$

Proof of second one: $x \square \mathbf{T}(\emptyset) =$ (with (E1))

$$x \square (\Sigma(\mathbf{T}(\emptyset)) \square \mathbf{T}(\emptyset)) = \text{(with (S2))}$$

$$x \square (\emptyset \square \mathbf{T}(\emptyset)) = \text{(with (E2))}$$

$$(x \cap \emptyset) \square \mathbf{T}(\emptyset) = \emptyset \square \mathbf{T}(\emptyset) = \mathbf{T}(\emptyset)$$

$$x \supseteq (\Sigma(Y) \cap \Sigma(Z)) = T \implies x \square (Y + Z) = (x \square Y) + (x \square Z)$$

(E4)

Postulates **restricted** distribution of \square over $+$

Unrestricted distribution **may seem plausible, but ... counterexample!**

$$x = \mathbf{S} : B + \mathbf{F} : T : B + \mathbf{F} : F : B$$

$$Y = \mathbf{T}(x + \mathbf{F} : c : B) + \langle T^B = c^B \rangle$$

$$Z = \mathbf{T}(x + \mathbf{F} : c : B) + \langle F^B = c^B \rangle$$

c^B not exported by $x \square Y$ and $x \square Z$ since $\mathbf{F} : c : B \notin x$

On the one hand $x \square (Y + Z)$ implies $T^B = F^B$ since $T^B = c^B = F^B$ and $\Sigma(T^B = F^B) \subseteq x$

On the other hand $(x \square Y) + (x \square Z)$ does **not** imply $T^B = F^B$ as one may choose $c^B = T^B$ in $x \square Y$ and $c^B = F^B$ in $x \square Z$

Hence $x \square (Y + Z) \neq (x \square Y) + (x \square Z)$

(E3) independent

There is a model of $BMA[\text{fol}] \setminus (E3)$ in which (E3) is not valid

(model class semantics allowing models with empty carriers)

In particular, (E3) is not a consequence of (E4)

Does $BMA[\text{fol}]$ satisfy its design requirements?

Laws of $BMA[\text{fol}]$ pretty convincing even without any semantics

- (A) $BMA[\text{fol}]$ valid for model class interpretation of modules and also for theory interpretation (proof later)
- (B) Persistence of signatures easily verified
- (C) We prove that every closed expression of sort M has a normal form containing a single instance of \square
Crucial: conditional distributive law (E4)
- (D) Enrichment is special case of extension by law (C5) as shown

Renaming lemma for normal form theorem

Flat module expression: no \square

$FCME[fol]$: set of flat closed module expressions

Lemma (renaming of hiddens): x, x' signatures, $Y \in FCME[fol]$

Then there is $Y' \in FCME[fol]$ such that

$$BMA[fol] \vdash x \square Y = x \square Y' = (x + x') \square Y'$$

Proof: Repeated renaming of hiddens (R6) in $x \square Y$ yields $x \square Y'$ such that all names occurring in $\Sigma(Y)$ but not in x are replaced by names not occurring in $x + x'$

$$x \square Y = x \square Y' = \text{(with (E1))}$$

$$x \square (\Sigma(Y') \square Y') = \text{(with (E2))}$$

$$(x \cap \Sigma(Y')) \square Y' = ((x + x') \cap \Sigma(Y')) \square Y' = (x + x') \square Y'$$

Proof of normal form theorem

$CME[\text{fol}]$: set of closed module expressions

\square -depth d of closed module expression:

$$d(X) = 0 \quad \text{if } X \text{ flat}$$

$$d(r.X) = d(X)$$

$$d(X + Y) = \max(d(X), d(Y))$$

$$d(x \square Y) = d(Y) + 1$$

Induction with respect to \square -depth:

$d(X) = 0$: X flat—apply (E1): $X = \Sigma(X) \square X$

Assumption: all $X \in CME[\text{fol}]$ with $d(X) \leq n$ can be brought in normal form

Induction step: let $X \in CME[\text{fol}]$ with $d(X) = n + 1$

Proof of normal form theorem (cont.)

Without loss of generality

$$X = \sum_{i=1}^k (u_i \sqcap X_i) \quad (k \geq 1, d(X_i) \leq n)$$

since

- flat summands can be brought into the form $u_i \sqcap X_i$ by (E1)
- renamings encompassing any outermost \sqcap -operators can be moved inward by means of (R3) and (R4) without changing the \sqcap -depth

Apply induction assumption:

$$X_i = v_i \sqcap Y_i \quad (Y_i \text{ flat}, 1 \leq i \leq k)$$

Substitute and apply (E2)

$$X = \sum_{i=1}^k (u_i \sqcap (v_i \sqcap Y_i)) = \sum_{i=1}^k ((u_i \cap v_i) \sqcap Y_i)$$

$k = 1$: normal form—**finished**

Proof of normal form theorem (cont.)

$k \geq 2$: take $y = \sum_{i=1}^k (u_i \cap v_i)$

Apply renaming lemma:

$$(u_i \cap v_i) \sqsupset Y_i = y \sqsupset Y'_i \quad (1 \leq i \leq k)$$

Hence

$$X = \sum_{i=1}^k (y \sqsupset Y'_i)$$

If each term of the form $(y \sqsupset Z_1) + (y \sqsupset Z_2)$ can be written as $y \sqsupset Z_3$ with Z_1, Z_2, Z_3 flat, the desired normal form can be obtained

Consider

$$Z = (y \sqsupset Z_1) + (y \sqsupset Z_2)$$

Conditional distributive law (E4) not directly applicable since in general

$$y \not\supseteq \Sigma(Z_1) \cap \Sigma(Z_2)$$

Proof of normal form theorem (cont.)

Apply renaming lemma: transform Z_2 into $Z'_2 \in FCME[\text{fol}]$ such that

$$y \sqcap Z_2 = y \sqcap Z'_2 \quad \text{and} \quad y \sqcap Z'_2 = (y + \Sigma(Z_1)) \sqcap Z'_2$$

Take signature of both sides of last eq:

$$y \cap \Sigma(Z'_2) = (y + \Sigma(Z_1)) \cap \Sigma(Z'_2) \supseteq \Sigma(Z_1) \cap \Sigma(Z'_2)$$

so

$$y \supseteq \Sigma(Z_1) \cap \Sigma(Z'_2)$$

Apply (E4):

$$Z = (y \sqcap Z_1) + (y \sqcap Z_2) = (y \sqcap Z_1) + (y \sqcap Z'_2) = y \sqcap (Z_1 + Z'_2) = y \sqcap Z_3$$

(end of proof)

Semantics of $BMA[\text{fol}]$

Model of $BMA[\text{fol}]$ is a **module algebra**

To be investigated: **What could elements of carrier M be?**

- **Presentations** of fol modules modulo equality generated by $BMA[\text{fol}]$
 - **initial model** of $BMA[\text{fol}]$
 - **exists**, unique, close to syntax
 - **two closed module expressions equal iff normal form the same**
 - **elements are normal forms of closed module expressions**
- **Full model classes** of fol modules—show it is proper semantics
- **Theories** of fol modules—show it is proper semantics

Existence of model class and theory semantics

1. Define interpretations for $BMA[\text{fol}]$ operators on model classes and theories

Combination, renaming, export, ... on model classes and theories

2. Prove that laws of $BMA[\text{fol}]$ hold for model classes and theories with these operators

Example: (E3)

$$x \square (\mathbf{T}(y) + Z) = \mathbf{T}(x \cap y) + (x \square Z)$$

becomes in theory interpretation Th

$$x \square (Th(\mathbf{T}(y)) + Th(Z)) = Th(\mathbf{T}(x \cap y)) + Th(x \square Z)$$

- $+$, \square combination and export on theories (step 1)
- Why not $Th(x)$, $Th(y)$?

Signatures always just presentation semantics—no “junk”

Model class semantics

$Alg(x)$	class of all algebras with signature x
$Alg(x, \phi)$	class of all x -algebras satisfying fol sentence ϕ over x
$r \cdot A$	A renamed by r for $A \in Alg(x)$ this yields $B \in Alg(r \cdot x)$
$r \cdot K$	$\{r \cdot A \mid A \in K\}$ for $K \subseteq Alg(x)$ this yields $L \subseteq Alg(r \cdot x)$
$x \sqcap A$	restriction of A to $x \cap y$ for $A \in Alg(y)$ hidden carriers and functions deleted from A
$x \sqcap K$	$\{x \sqcap A \mid A \in K\}$ for $K \subseteq Alg(y)$
$K + L$	$\{A \in Alg(x_1 + x_2) \mid x_1 \sqcap A \in K, x_2 \sqcap A \in L\}$ for $K \subseteq Alg(x_1), L \subseteq Alg(x_2)$ if $x_1 = x_2$ then $K + L = K \cap L$

Model class semantics (cont.)

Inductive definition of *Mod* interpretation:

$$\text{Mod}(\langle \phi \rangle) = \text{Alg}(\Sigma(\langle \phi \rangle), \phi)$$

$$\text{Mod}(\mathbf{T}(x)) = \text{Alg}(x)$$

$$\text{Mod}(r \cdot X) = r \cdot \text{Mod}(X)$$

$$\text{Mod}(X + Y) = \text{Mod}(X) + \text{Mod}(Y)$$

$$\text{Mod}(x \square Y) = x \square \text{Mod}(Y)$$

Two closed module expressions equal iff signatures and model classes the same

Not hard to show that laws of *BMA*[fol] hold

Example: we verify special case of (E3)

Model class semantics (cont.)

$$\text{Mod}(x \sqcap \mathbf{T}(y)) = \text{Mod}(\mathbf{T}(x \cap y))$$

- (1) \subseteq : Restriction of y -algebra to x is $x \cap y$ -algebra
- (2) \supseteq : Enrich $x \cap y$ -algebra with sorts and functions from y and restrict to x

Subtle: carriers not empty (by definition of algebra)

Otherwise:

- Enrichment to y -algebra may fail
- New function to empty carrier cannot be added
- (E3) fails

Theory semantics

Theories **logically closed** (closed under \vdash_{fol})

$Th(x)$ set of all derivable fol sentences over signature x
 = smallest x -theory

$Th(x, \phi)$ idem, but derivable from ϕ
 = smallest x -theory containing ϕ

$r \cdot T$ T renamed via r
 for x -theory T this yields $(r \cdot x)$ -theory T'

$T + U$ set of all $(x_1 + x_2)$ -sentences derivable from $T \cup U$
 for x_1 -theory T and x_2 -theory U

$x \square T$ **restriction** of T to signature $x = T \cap \{\text{sentences over } x\}$
axioms over hidden signature can be used in proofs
fol with hiddens stronger than fol!

Theory semantics (cont.)

Inductive definition of Th interpretation:

$$Th(\langle \phi \rangle) = Th(\Sigma(\langle \phi \rangle), \phi)$$

$$Th(\mathbf{T}(x)) = Th(x)$$

$$Th(r \cdot X) = r \cdot Th(X)$$

$$Th(X + Y) = Th(X) + Th(Y)$$

$$Th(x \square Y) = x \square Th(Y)$$

Two closed module expressions equal iff signatures and theories the same

Stronger than model class equality— see 1990 JACM article

Much harder to show that laws of $BMA[\text{fol}]$ hold

(E3) and (E4) equivalent to interpolation property of fol

Interpolation

- fol-sentences p and q with $\vdash_{\text{fol}} p \implies q$
always have **interpolant**: fol-sentence r with

$$\Sigma(r) \subseteq \Sigma(p) \cap \Sigma(q)$$

$$\vdash_{\text{fol}} p \implies r$$

$$\vdash_{\text{fol}} r \implies q$$

- Equivalently, by **deduction theorem for fol**:
if $p \vdash_{\text{fol}} q$ there always is an interpolant r with

$$\Sigma(r) \subseteq \Sigma(p) \cap \Sigma(q)$$

$$p \vdash_{\text{fol}} r \vdash_{\text{fol}} q$$

more suitable for possible generalization to eql (no \implies)

(E3) holds in theory semantics

$$Th(x \sqcap (\mathbf{T}(y) + Z)) = Th(\mathbf{T}(x \cap y) + (x \sqcap Z))$$

Proof:

$$\begin{aligned} \Sigma : \Sigma(x \sqcap (\mathbf{T}(y) + Z)) &= x \cap (\Sigma(\mathbf{T}(y)) + \Sigma(Z)) = \\ &(x \cap y) + (x \cap \Sigma(Z)) = \Sigma(\mathbf{T}(x \cap y) + (x \sqcap Z)) \end{aligned}$$

 \supseteq : Easy

$$p \in Th(\mathbf{T}(x \cap y) + (x \sqcap Z))$$

Choose $q \in Th(x \sqcap Z)$ with $q \vdash p$

$$q \in Th(x \sqcap (\mathbf{T}(y) + Z))$$

Theories logically closed: $p \in Th(x \sqcap (\mathbf{T}(y) + Z))$

(E3) holds in theory semantics (cont.)

\subseteq : Hard—left-hand side might be larger!

$$p \in Th(x \square (\mathbf{T}(y) + Z))$$

Choose $q \in Th(Z)$ with $q \vdash p$

Apply interpolation property: interpolant r with

$$\Sigma(r) \subseteq \Sigma(q) \cap \Sigma(p) \subseteq \Sigma(Z) \cap x \cap (y + \Sigma(Z)) = x \cap \Sigma(Z)$$

$$q \vdash r \text{ and } r \vdash p$$

Hence $r \in Th(x \square Z)$ and $p \in Th(\mathbf{T}(x \cap y) + (x \square Z))$

(end of proof)

Conversely, (E3) implies interpolation

Proof:

$p \vdash q$ and $x = \Sigma(q)$

Apply (E3): $q \in Th(x \square (T(x) + \langle p \rangle)) = Th(T(x) + (x \square \langle p \rangle))$

Hence there is $r \in Th(x \square \langle p \rangle)$ such that $r \vdash q$

Hence $p \vdash r \vdash q$ and $\Sigma(r) \subseteq \Sigma(q) \cap \Sigma(p)$

r interpolant

(end of proof)

(E4) holds in theory semantics

$$x \supseteq (\Sigma(Y) \cap \Sigma(Z)) \implies Th(x \square (Y + Z)) = Th((x \square Y) + (x \square Z))$$

Proof:

$$\begin{aligned} \Sigma : \Sigma(x \square (Y + Z)) &= x \cap (\Sigma(Y) + \Sigma(Z)) = \\ &(x \cap \Sigma(Y)) + (x \cap \Sigma(Z)) = \Sigma((x \square Y) + (x \square Z)) \end{aligned}$$

\supseteq : Again easy

$$p \in Th((x \square Y) + (x \square Z))$$

Choose $q \in Th(x \square Y), r \in Th(x \square Z)$ with $q \& r \vdash p$

$$q, r \in Th(x \square (Y + Z))$$

$$\text{Hence } p \in Th(x \square (Y + Z))$$

(E4) holds in theory semantics (cont.)

\subseteq : Hard—lhs might again be larger

$$p \in Th(x \square (Y + Z))$$

Choose $q_1 \in Th(Y)$, $q_2 \in Th(Z)$ with $q_1 \& q_2 \vdash p$

Apply deduction theorem: $q_1 \vdash q_2 \implies p$

Apply interpolation property: interpolant r with

$$\Sigma(r) \subseteq \Sigma(q_1) \cap \Sigma(q_2 \implies p) \subseteq$$

$$\Sigma(Y) \cap (\Sigma(Z) + (x \cap (\Sigma(Y) + \Sigma(Z)))) =$$

$$\Sigma(Y) \cap (\Sigma(Z) + (x \cap \Sigma(Y)) + (x \cap \Sigma(Z))) =$$

$$(\Sigma(Y) \cap \Sigma(Z)) + (x \cap \Sigma(Y)) + (x \cap \Sigma(Y) \cap \Sigma(Z)) \subseteq x$$

$$q_1 \vdash r \text{ and } r \vdash q_2 \implies p$$

Apply deduction theorem: $q_1 \vdash r$ and $q_2 \vdash r \implies p$

$$r \in Th(x \square Y) \text{ and } r \implies p \in Th(x \square Z)$$

Hence $p \in Th((x \square Y) + (x \square Z))$ (end of proof)

Conversely, (E4) implies interpolation

- Like (E3), (E4) implies the interpolation property (no proof)
- fol theory semantics: (E3) and (E4) equivalent
- No longer true for eql

Example revisited

Earlier example showing that condition $x \supseteq \Sigma(Y) \cap \Sigma(Z)$ of (E4) is **essential**:

$$x = \mathbf{S} : B + \mathbf{F} : T : B + \mathbf{F} : F : B$$

$$Y = \mathbf{T}(x + \mathbf{F} : c : B) + \langle T^B = c^B \rangle$$

$$Z = \mathbf{T}(x + \mathbf{F} : c : B) + \langle F^B = c^B \rangle$$

$$x \not\supseteq \Sigma(Y) \cap \Sigma(Z) = x + \mathbf{F} : c : B$$

$Th(x \square (Y + Z)) \neq Th((x \square Y) + (x \square Z))$ since

- $Th(x \square (Y + Z))$ contains $T^B = F^B$
- $Th((x \square Y) + (x \square Z))$ does **not** contain $T^B = F^B$

BMA[eql]

Summary of results—1990 JACM article has the details

What about interpretation of algebraic specifications (closed eql-expressions) as

- equational theories?
- initial algebras?

Unfortunately, no good!

BMA[eq] (cont.)

Interpretation *EqTh* of alg spec as equational theory?

- eq does **not have interpolation** property for individual eqs
(problem: no conjunction)
- interpolation property does hold for **sets of eqs**
(Rodenburg and van Glabbeek, 1988)
- E, F sets of eqs with $E \vdash_{\text{eq}} F$ have **interpolant** I
(also set of eqs)

$$\Sigma(I) \subseteq \Sigma(E) \cap \Sigma(F)$$

$$E \vdash_{\text{eq}} I \vdash_{\text{eq}} F$$

- **equivalent to (E3)**

BMA[eql] (cont.)

- In general, (E4) stronger interpolation property than (E3)
(but not for fol!)
- (E4) not valid (problem: no \exists)
- Proof of normal theorem depends on (E4)
- Proof of normal form theorem fails
- Normal form theorem may still be valid
 - normalization much more complex
 - involves transformation of axioms (not only renaming)

BMA[eql] (cont.)

Interpretation *Init* of alg spec as **initial algebra**?

- don't confuse with initial algebra of *BMA*[fol] itself
- alg spec has **unique initial algebra**
- (most) fol-specifications do **not** have initial algebra

No hope for +-operator on initial algebras:

$$X_n = \{\mathbf{S} : A + \mathbf{F} : a : A + \mathbf{F} : i : A \rightarrow A + \langle i(a) = a \rangle + \langle i^n(x) = x \rangle\}$$

$$\mathit{Init}(X_n) = \mathit{Init}(X_m) \text{ (single element)}$$

$$Y = \{\mathbf{S} : A + \mathbf{F} : b : A\}$$

$$\mathit{Init}(X_n + Y) \text{ has } n + 1 \text{ elements } a, b, \dots, i^{n-1}(b)$$

$$\mathit{Init}(X_n + Y) \neq \mathit{Init}(X_m + Y) \text{ (} n \neq m \text{)}$$

Generalization necessary—see Rodenburg (1994)

Literature

- J.A. Bergstra, J. Heering, and P. Klint (eds), [Algebraic Specification](#), ACM Press/Addison-Wesley, 1989
- J.A. Bergstra, J. Heering, and P. Klint, [Module algebra](#), *Journal of the ACM* **37** (1990) 335–372
- A. Brogi, P. Mancarella, D. Pedreschi, and F. Turini, [Modular logic programming](#), *ACM Transactions on Programming Languages and Systems* **16** (1994) 1361–1398
- L. Corry, [Modern Algebra and the Rise of Mathematical Structures](#), Birkhäuser, 1996
- F. Durán, [A reflective module algebra with applications to the Maude language](#), Ph.D. Thesis, 1999,
maude.csl.sri.com/papers/abstract/Dmodalg_1999.html

Literature (cont.)

- L.M.G. Feijs and Y. Qian, [Component algebra](#), *Science of Computer Programming* **42** (2002) 173–228
- P.H. Rodenburg and R.J. van Glabbeek, [An interpolation theorem in equational logic](#), Report CS-R8838, CWI, Amsterdam, 1988
- P. Rodenburg, [Module algebra for initial algebra semantics](#), Report P9407, Programming Research Group, University of Amsterdam, info.science.uva.nl/pub/programming-research/reports/1994/P9407.ps.gz