PROSPECTRA
an ESPRIT Project

Notes on TRAFOLA, II

The Objects of the Transformation Language and the operations upon them

Reinhold Heckmann

Universität des Saarlandes
6600 Saarbrücken
Bundesrepublik Deutschland

PROSPECTRA Study Note S.1.6-SN-7.0

1987-05-19

Distribution: public

ABSTRACT

This note tries to treat formally several features introduced in the Study Note *′A Proposal for the Syntactic Part of the PROSPECTRA Transformation Language′*, referred to as [1] in the sequel. At first we consider *values* and the operations on them. The term fragments of the study note contained exactly one hole, this will be generalized. Different kinds of syntactic insertion (inserting values into holes of another value) will be introduced, and their algebraic properties will be investigated. At last, we shall consider the number of partitions of a given term into an upper and a lower fragment.

At the end of this document, we give a summary of all definitions and theorems contained in it as a quick reference.

Public

Reinhold Heckmann
Universität des Saarlandes

in the Project

PROgram development by
SPECification and
TRAnsformation

sponsored by the

Commission of the European Communities

under the

European
Strategic
Programme for
Research and development in
Information
Technology

Project Ref. No. 390

1. Definition of general values

First we define general values Val', later we shall define some useful subsets: well formed values Val, closed values CVal, well grouped values GVal, and ungrouped values UVal.

Definition 1

Let *Const* be a finite set of constants (operators of arity 0) and *Op* be a finite set of other operators containing one special element < >, the grouping operator. Then

Val' = Val' (Const, Op) = ({ [] } ∪ Const ∪ (Op X Val'))*

General values are finite strings of primitive constituents (this fact is denoted by the Kleene star in the definition). There are three kinds of primitive constituents: holes [], constants, and trees i.e. a root operator together with a new value. This new value is to be understood as children list of the tree; its constituents are the children.

We shall denote values by *()* - the empty value - resp. (v_1, ... , v_n) - finite sequence of primitive constituents v_i. If n is 1, the parentheses may be omitted. Primitive constituents being trees are denoted by *op v* where $op \in Op$ and $v \in Val'$. The empty value *()* is sometimes denoted by ε.

In the sequel, we shall adopt the convention that *u*, *v*, *w*, *u'* etc. denote values, *c*, *d* etc. denote constants, and *op* operators.

Examples for values:
add ([], sub (c, [])) ()
(add [], sub (c, d, e, []), mul ())

For the moment, we do not restrict the set of values by means of a tree grammar; instead we allow each operator to have an arbitrary number of arbitrary children.

According to the definition of *Val'*, recursive definitions and inductive proofs over values will always contain the following cases:

(1) v = ()
(2) v = (v_1, ... , v_n)
(3) v = []
(4) v = c ∈ Const
(5) v = op v'

For values containing the grouping operator '<>', we propose an alternative paraphrasing:

<> ()	<> or <()>
<> v	<v>
(<v_1>, ... , <v_n>)	<v_1; ... ; v_n>
(<v>, <>)	<v; >

The virtues of this notation will be seen later when the insertion Δ is defined.

At last some notions being useful when reasoning about values.
A value *op v* is called *op* value, and a value <*v*> is called ' <>' value or grouped value.
The relation '*is subvalue of*' is the reflexive transitive closure of
- v_i is subvalue of (v_1, ... , v_n) for all i with $1 \le i \le n$
- *v* is subvalue of *op v*

If a value *op v'* is a subvalue of *v*, then *op* **occurs** in *v*, and if '[]' is a subvalue of *v*, then '[]' occurs in *v*.
If $v = (v_1, \ldots, v_n)$ and $v_i = op\ v$ for some i, we say *op* occurs in *v* at top level; analogously for '[]'.
If *v* has a subvalue $u = op\ (u_1, \ldots, u_n)$ and $u_i = op'\ u'$ for some i, then *op'* occurs in *v* under *op*.
Instead of '*x* occurs in *v*', we also say '*v* contains (an occurrence of) *x*'.

2. Subsets of values

Now we shall define some subsets of arbitrary values with respect to the occurrences of holes and grouping operators.

Definition 2

Well formed values: Val := {v ∈ Val’ | ′ <>′ occurs in v
only under ′ <>′ or at top level}
Ungrouped values: UVal := {v ∈ Val’ | v does not contain ′ <>′}
Closed values: CVal := {v ∈ Val | ′[]′ does not occur in v at top level}
Grouped values: GVal’ := {v ∈ Val’ |
(a) the primitive constituents of v are grouped values
(b) ′ <>′ occurs in v only at top level }
= {v ∈ Val | v = ($<u_1>$, ... , $<u_n>$) ($n \geq 0$) (a) where $u_i \in$ UVal (b)}
Well grouped values: GVal := {v ∈ GVal’ | (c) v has no primitive constituent <()> }

We shall often refer to the conditions (a), (b), and (c).

Examples:

′add (1, 2)′ is in Val, CVal, and UVal, but not in GVal’ or GVal.
′<1, <2, 3>, 4>′ is in Val and CVal, but not in UVal, GVal’, or GVal.
′<1; 2, 3; >′ is in Val, CVal, and GVal’, but not in UVal or GVal.
′<1; 2, 3>′ is in Val, CVal, GVal’, and GVal, but not in UVal.
′(add (1, 2), [])′ is in UVal and Val, but not in GVal’, GVal, or CVal.
′add (<1>, 2)′ is in Val’, but neither in Val nor in any other of the defined sets.

Proposition 3

(1) GVal ⊂ GVal’ ⊂ CVal ⊂ Val ⊂ Val’, and UVal ⊂ Val ⊂ Val’
(2) GVal ∩ UVal = GVal’ ∩ UVal = {ε}. ε is in all of those sets.
(3) If u is a subvalue of v and v is in Val (UVal), then u is in Val (UVal), too.
(4) CVal, GVal’, and GVal do not satisfy a property such as (3).

Proof:

(1) GVal’ ⊆ Val: ′ <>′ occurs only at top level.
GVal’ ⊆ CVal: since GVal’ ⊆ Val and the constituents of a value in GVal’ are grouped - thus they are not holes.
CVal ⊆ Val ⊆ Val’, UVal ⊆ Val, GVal ⊆ GVal’: by definition
For the proper inclusions see the examples above.

(2) ε does not contain any occurrence of ′ <>′, thus it is in UVal, and it satisfies the conditions (a) through (c) of the definition of GVal. Conversely, if a value is both in GVal’ and UVal, its primitive constituents are grouped and don’t contain ′ <>′. This contradiction implies that the value consists of no primitive constituents, therefore it is ε.

(3) Val: If ′ <>′ occurred in u under an operator op ≠ <>, it would occur in v under op, too.
UVal: If ′ <>′ occurred in u, then it would also occur in v.

(4) CVal: u = [] is subvalue of v = op [], v is in CVal, but u is not.
GVal, GVal’: u = c is subvalue of v = <c>, v in GVal’, u not in GVal.

The difference between (3) and (4) is due to the fact that the property of being well formed or being ungrouped depends on the whole value, whereas being closed and being well grouped are properties of the top level of a value.

Val is introduced since the grouping operator is meaningless under another operator; only at top level or under another occurrence of itself it is useful (see [1] - 5.7. ′Preventing concatenation′, and syntactic insertion below). Prop. 3 (3) and the fact that *Val* will be closed under concatenation and insertion imply that we may drop the values not in *Val* and restrict ourselves to well formed values.

The notion of closed values is important since the top level of a closed value is not affected when other values are inserted in its holes.

When we define the syntactic insertion ′u Δ v′ later, *v* will be required to be well grouped. The constituents (groups) of *v* are inserted in the holes of *u* by stripping their ′ <>′ operator (thus condition (a) is needed). Conditions (b) and (c) are needed to assert that the result ′u Δ v′ is well formed, that insertion Δ is associative, and that only a finite number of decompositions of a given value exist.

We shall show what would go wrong if the conditions (b) and (c) were omitted. Therefore we shall introduce another insertion ′u Δ’ v′ where v must only be in GVal’. This operation will neither be associative, nor the number of partitions of a given value will be finite.

Ungrouped values are needed when we shall discuss alternative definitions of insertion.

3. Concatenation, upper and lower length

In the set Val’ = ({ [] } ∪ Const ∪ (Op X Val’))* we introduce the operation of concatenation ′ · ′ with neutral element ε = () and call the number of primitive constituents of a value *UL* meaning *upper length*. The number of holes in a value will be denoted as *LL* meaning *lower length*.

Formal recursive definition:

Definition 4

() · v = v · () = v

$(v_1, \ldots, v_n) \cdot (w_1, \ldots, w_m) = (v_1, \ldots, v_n, w_1, \ldots, w_m)$

UL () = 0	$UL (v_1, \ldots, v_n) = n$	
LL () = 0	$LL (v_1, \ldots, v_n) = LL (v_1) + \ldots + LL (v_n)$	
LL ([]) = 1	LL (c) = 0	LL (op v) = LL (v)

The primitive constituents u_i of a value $u = (u_1, \ldots, u_n)$ have upper length 1. In the sequel, we shall call a value whose upper length is 1, primitive. Primitives are similar to prime numbers in number theory. If we write $(u_1, \ldots, u_n)$, the values u_i will be primitive.

From the definition, the following properties are obvious:

Proposition 5

u · (v · w) = (u · v) · w	v · ε = ε · v = v
u · v = u · v’ implies v = v’	u · v = u’ · v implies u = u’
UL (ε) = LL (ε) = 0	UL (v) = 0 iff v = ε
UL (u · v) = UL (u) + UL (v)	LL (u · v) = LL (u) + LL (v)

Val, UVal, CVal, GVal’, and GVal are closed under concatenation, even
u · v in Val (UVal, CVal, GVal(’)) iff u and v in Val (UVal, CVal, GVal(’))

Val’ is a monoid under concatenation, and *UL* and *LL* are monoid homomorphisms from (Val’, ·) to ($\mathbf{N}_o$, +). (UVal, ·) etc. are submonoids of (Val’, ·). *UL* and *LL* may independently vary in $\mathbf{N}_o$ except that UL(v) = 0 implies LL(v) = 0.

4. Insertion

Now we want to introduce syntactic insertion of values into holes. First, we shall define two similar operations Δ and Δ’ that will only differ in the domain where they are defined. *u* Δ’ *v* resp. *u* Δ *v* shall denote the value obtained by inserting the constituents of *v* into the holes of *u*. This may be done only if *v* is (well) grouped (v in GVal’ resp. GVal), and if the number of constituents of *v* equals the number of holes of *u*: *UL (v) = LL (u)*. *v* is split into its grouped constituents, and these are inserted into the holes of *u* by stripping the ′ <>′ operator.

Definition 6

Δ’ and Δ: Val’ X Val’ → Val’ are partial mappings.
u Δ’ v is defined iff v ∈ GVal’ and LL (u) = UL (v).
() Δ’ v = () (v must equal () to make this defined)
$(u_1, \ldots, u_n)\ \Delta'\ v = (u_1\ \Delta'\ v_1) \cdot \ldots \cdot (u_n\ \Delta'\ v_n)$
where u_i primitive and $LL (u_i) = UL (v_i)$ and $v_1 \cdot \ldots \cdot v_n = v$
For short, we call this: *v is partitioned into* $v_1, \ldots, v_n$ *according to u*
[] Δ’ <w> = w (Note that the <> operator is stripped here)

c Δ’ v = c (v must equal () to make this defined)
(op u) Δ’ v = op (u Δ’ v) specially <u> Δ’ v = <u Δ’ v>
u Δ is defined iff v in GVal and LL (u) = UL (v).
Then u Δ v = u Δ’ v holds.

Examples:

op [] Δ <v> = op v
<op []> Δ <v> = <op v>
if ([], t, e) Δ <c> = if (c, t, e)
([], list (a1, [], a2, a3, [])) Δ’ <b1, b2; c1, c2, c3; > =
(**b1, b2**, list (a1, **c1, c2, c3**, a2, a3))
add ([], []) Δ’ <1, 2> is undefined (too many holes)
add [] Δ’ <1; 2> is undefined (too many groups)
[] Δ’ c is undefined (c is not grouped (a))
[] Δ’ <<c>> is undefined (condition (b) is violated)
[] Δ < > is undefined (not well grouped due to condition (c))
[] Δ’ < > = ()

From the first example, we learn that even with alternative paraphrasing - *op* means *op [] - op v* is different from *op* Δ *v*.

Values may be thought of as trapezoids with height 1, length of the upper edge *UL* and of the lower edge *LL*. Then, concatenation corresponds to a horizontal combination of trapezoids, Δ*’* to a vertical combination that is only possible if the lengths agree.

The only difference between Δ and Δ’ is that Δ’ is more defined than Δ. But this induces important differences in the algebraic properties of the operations.

Proposition 7

Let u and v be values (elements of Val’) such that u Δ’ v is defined.
Then v is in GVal’ and
u is in Val (UVal) iff u Δ’ v is in Val (UVal),
if u is in CVal (GVal’), then u Δ’ v is in CVal (GVal’), the inverse is not true.
u in GVal neither implies u Δ’ v in GVal, nor vice versa.
u Δ’ v = ε iff u = $[]^k$ and v = $<>^k$ for some k ≥ 0.

Proof: Simultaneous induction by u.

Case u = () or u = c: Then v must be ε, and u Δ’ v equals u. Thus the statements trivially hold (k = 0 in the statement about u Δ’ v = ε).

Case u = []: Then v = <v’> holds and u Δ’ v = v’. [] is in Val and UVal, but not in CVal, GVal’, or GVal.

CVal: v’ may be closed or not, thus the ′iff′ property does not hold for CVal.

Val: Since v = <v’> is in GVal’, it is also in Val, and thus its subvalue v’ is in Val, too, by Prop. 3(3).

UVal: If v’ were not in UVal, it would contain an occurrence of ′ <>′. This occurrence would not be at top level in v, and v would not be grouped (condition b). Thus v’ is in UVal.

ε: u Δ’ v = ε iff v’ = ε iff v = <>. This is the case k = 1.

Case u = op u’ where ′op′ is not ′ <>′: Then u Δ’ v = op (u’ Δ’ v).

Val, UVal: u is in Val (UVal) iff u’ is in Val (UVal)
iff u’ Δ’ v in Val (UVal) iff op (u’ Δ’ v) in Val (UVal).
CVal: op u’ and op (u’ Δ’ v) are both in CVal.
GVal’: Both are not in GVal’.
ε: Both don’t equal ε.

Case u = <u’>, then u Δ’ v = <u’ Δ’ v>.

Val: analogous to case ′op u’′
UVal: <u’> and <u’ Δ’ v> are not in UVal.
CVal: They are both closed.
ε: Both don’t equal ε.

GVal': Cond. (a) is satisfied by both values.
<u'> satisfies (b) iff u' in UVal
iff u' Δ' v in UVal iff <u' Δ' v> satisfies (b)

Case $u = (u_1, \ldots, u_n)$, n > 1.
Then $v = v_1 \cdot \ldots \cdot v_n$ and $u\ \Delta'\ v = (u_1\ \Delta'\ v_1) \cdot \ldots \cdot (u_n\ \Delta'\ v_n)$.
u is in Val (UVal, CVal, GVal') iff all u_i are in Val (UVal, CVal, GVal'), and u Δ' v is in Val (UVal, CVal, GVal') iff all $u_i\ \Delta'\ v_i$ are in Val (UVal, CVal, GVal'), thus the statements are true by induction.
$u\ \Delta'\ v = \varepsilon$ iff all $u_i\ \Delta'\ v_i = \varepsilon$ iff $u_i = []^{ki}$ and $v_i = <>^{ki}$ iff
$u = []^k$ and $v = <>^k$ where $k = k_1 + \ldots + k_n$.

Examples:

1) [] is not in CVal, but [] Δ' <c> = c is.
2) u = [] is neither in GVal nor in GVal'. Let v = <>, then u Δ' v = () is in both GVal and GVal'.
3) u = <[]> is in GVal. Let v = <>. u Δ' v = <> is not in GVal.
4) Assume condition (b) in the definition of GVal' would not exist. Then op [] in Val, but op [] Δ' <<c>> = op <c> is not, [] is in UVal, but not in GVal', whereas [] Δ' <<c>> = <c> is not in UVal, but in GVal.

Proposition 8

Let u and v be values (elements of Val') such that u Δ v is defined.
Then v is in GVal and
u is in Val (UVal, GVal', GVal) iff u Δ v is in Val (UVal, GVal', GVal),
if u is in CVal, then u Δ v is in CVal, the inverse is not true.
$u\ \Delta\ v = \varepsilon$ iff $u = v = \varepsilon$

Note the differences: ε may be written as u Δ' v in infinitely many ways, but as u Δ v in exactly one way. The statements about GVal and GVal' are stronger, this will be important for associativity.

Proof:

If u Δ v is defined, then u Δ' v is also defined and equals u Δ v. Thus the statements for Val, UVal, and CVal directly follow from Prop. 7.

ε: $[]^k\ \Delta\ <>^k$ is defined iff k = 0, thus $u = v = \varepsilon$ holds.

GVal, GVal': Induction on u. Induction is needed since there are no correspondent properties of Δ'.

Case u = () or u = c: Then v must be ε, and u Δ v equals u. Thus the statements trivially hold.

Case u = []: Then v = <v'> holds and u Δ v = v'. [] is neither in GVal nor in GVal', but it is in UVal, hence v' is in UVal, too.
Due to condition (c) of GVal, v' is not ε, and since UVal ∩ GVal' = $\{\varepsilon\}$, v' is not in GVal', and not in GVal.

Case u = op u' where ′op′ is not ′ <>′: Then u Δ v = op (u' Δ v).
Both are not in GVal or GVal'.

Case u = <u'>, then u Δ v = <u' Δ v>.
GVal': Cond. (a) is satisfied by both values.
<u'> satisfies (b) iff u' in UVal iff
u' Δ v in UVal iff <u' Δ v> satisfies (b)
GVal: <u'> satisfies (c) iff $u' \neq \varepsilon$ iff $u'\ \Delta\ v \neq \varepsilon$ iff <u' Δ v> satisfies (c)

Case $u = (u_1, \ldots, u_n)$, n > 1.
Then $v = v_1 \cdot \ldots \cdot v_n$ and $u\ \Delta\ v = (u_1\ \Delta\ v_1) \cdot \ldots \cdot (u_n\ \Delta\ v_n)$.
u is in GVal(') iff all u_i are in GVal('), and u Δ v is in GVal(') iff all $u_i\ \Delta\ v_i$ are in GVal('), thus the statements are true by induction.

Syntactic insertion does not allow for omitting common operands in equations:
u Δ v = u Δ v' does not imply v = v', and
u Δ v = u' Δ v does not imply u = u'.
Examples: ([], []) Δ <a, b; c> = (a, b, c) = ([], []) Δ <a; b, c>

(c, []) Δ <c> = (c, c) = ([], c) Δ <c>

5. The X-category properties

In this section, we shall check that $'\cdot'$ and $'\Delta'$ satisfy the axioms of an X-category (see [2]). We shall use two conventions in our propositions and theorems:

1) All formulae about values are to be understood as preceded by an all quantifier over all free variables occurring in it. Variables *u*, *v*, *w*, *u'* etc. denote values, *c*, *d* etc. denote constants, and *op* operators.
2) Let *e* and *f* be two expressions over values.
   e = *f* means: *e* is defined iff *f* is, and if both are defined, they are equal.
   e = *f* means: if *e* is defined, then *f* is also defined and both are equal.
   $e \doteq f$ means: if both *e* and *f* are defined, they are equal.

Example: $u\ \Delta\ v\ \acute{=}\ u\ \Delta'\ v$

Clearly, $e = f$ implies $e \acute{=} f$, and this implies $e \doteq f$. $'='$ is a congruence relation, but $'\acute{=}'$ is not symmetric, and $'\doteq'$ is not transitive e.g. $1 \doteq 1/0 \doteq 2$, but $1 \doteq 2$ is false.

Proposition 9

$LL\ (u\ \Delta\ v)\ \acute{=}\ LL\ (u\ \Delta'\ v)\ \acute{=}\ LL\ (v)$
i.e. u Δ(') v contains as many holes as v.

Due to our convention, this is an abbreviation for
For all u in Val' and v in Val', where u Δ v is defined,
LL (u Δ v) = LL (u Δ' v) holds, and for all u , v in Val',
where u Δ' v is defined, LL (u Δ' v) = LL (v) holds.

Proof: Induction by u:

LL (() Δ' v) = LL (()) = LL (v) since v = ()
LL ($(u_1, ..., u_n)$ Δ' v) = LL ($(u_1\ \Delta'\ v_1) \cdot ... \cdot (u_n\ \Delta'\ v_n)$)
= LL $(u_1\ \Delta'\ v_1)$ + ... + LL $(u_n\ \Delta'\ v_n)$ by Prop. 5
= LL (v_1) + ... + LL (v_n) by induction hypothesis
= LL $(v_1 \cdot ... \cdot v_n)$ = LL (v)
LL ([] Δ' <w>) = LL (w) = LL (<w>)
LL (c Δ' v) = LL (c) = 0 = LL (v) since v = ()
LL (op u Δ' v) = LL (op (u Δ' v)) = LL (u Δ' v) = LL (v) by induction

A corresponding proposition for the upper length - $UL\ (u\ \Delta'\ v)\ \acute{=}\ UL\ (u)$ - is not generally true, e.g. UL ([]) = 1, but UL ([] Δ <c, d>) = UL (c, d) = 2, and UL ([] Δ' <>) = 0. But it holds if *u* is closed:

Proposition 10

If u Δ v is defined, then UL (u Δ v) ≥ UL (u).
If u is in CVal, then $UL\ (u\ \Delta\ v)\ \acute{=}\ UL\ (u\ \Delta'\ v)\ \acute{=}\ UL\ (u)$ holds.

Proof:

Case u = (): UL (() Δ' v) = UL (())
Case u = $(u_1, ... , u_n)$:
UL $(u_1, ... , u_n)$ = n,
UL (u Δ' v) = UL ($(u_1\ \Delta'\ v_1) \cdot ... \cdot (u_n\ \Delta'\ v_n)$) =
UL $(u_1\ \Delta'\ v_1)$ + ... + UL $(u_n\ \Delta'\ v_n)$.
We have to show that UL $(u_i\ \Delta'\ v_i) \geq 1$ for u_i = [] and v_i in GVal,
and = 1 if $u_i \neq$ [].
u_i = c: UL $(u_i\ \Delta'\ v_i)$ = UL (c) = 1
u_i = op u': UL $(u_i\ \Delta'\ v_i)$ = UL (op (u' Δ' v_i)) = 1
u_i = []: Then v_i = <v'>. UL $(u_i\ \Delta\ v_i)$ = UL (v') ≥ 1
due to condition (c) of the definition of GVal (v' ≠ ε).

The operations Δ resp. Δ' do not possess a neutral element, but for each length there is a partial neutral element:

Proposition 11

Let $e = \langle[]\rangle$ and $e^k = e \cdot \ldots \cdot e$ (k times), $e^0 = ()$.
Then e^k is in GVal, and UL (e^k) = LL (e^k) = k holds, and
for all u in Val' with LL (u) = k, $u \,\Delta\, e^k = u$ holds, and
for all v in GVal with UL (v) = k, $e^k \,\Delta\, v = v$ holds.

Examples: $\langle [] ; [] \rangle \,\Delta\, \langle v; w\rangle = \langle v; w\rangle$
op ([], a, []) Δ $\langle [] ; [] \rangle$ = op ([], a, [])

Proof:

The first part is proved by induction on u:

$() \,\Delta\, e^0 = ()$
$(u_1, ..., u_n) \,\Delta\, e^k = (u_1 \,\Delta\, e^{LL(u1)}) \cdot\ldots\cdot (u_n \,\Delta\, e^{LL(un)})$
$= u_1 \cdot \ldots \cdot u_n = (u_1, ..., u_n)$ by induction
$[] \,\Delta\, e^1 = [] \,\Delta\, \langle[]\rangle = []$
$c \,\Delta\, e^0 = c$
op $u \,\Delta\, e^k$ = op $(u \,\Delta\, e^k)$ = op u

Proof of the second part:

If UL (v) = 0, then v = () and k = 0, and $() \,\Delta\, () = ()$ holds.
Let UL(v) = k > 0:
$e^k \,\Delta\, v = \langle[]; ...; []\rangle \,\Delta\, \langle v_1; ... ; v_k\rangle$ since v is well grouped
$= \langle [] \,\Delta\, \langle v_1\rangle \rangle \cdot\ldots\cdot \langle [] \,\Delta\, \langle v_k\rangle \rangle$
$= \langle v_1\rangle \cdot\ldots\cdot \langle v_k\rangle = \langle v_1; ... ; v_k\rangle = v$

The neutral elements of Δ are compatible with concatenation:

Proposition 12: $e^{i+j} = e^i \cdot e^j$

Proof: trivial

The horizontal and vertical combinations are interchangeable:

Proposition 13

$(u \,\Delta\, v) \cdot (u' \,\Delta\, v') \doteq (u \cdot u') \,\Delta\, (v \cdot v')$ (same for Δ')

Scheme:

$$\begin{array}{ccc} u & \cdot & u' \\ \Delta & & \Delta \\ v & \cdot & v' \end{array}$$

Proof: (Δ' is analogous to Δ)

The right hand side is defined since v, v' in GVal(') implies $v \cdot v'$ in GVal(')
and LL $(u \cdot u')$ = LL(u) + LL(u') = UL(v) + UL(v') = UL $(v \cdot v')$

Case $u = \varepsilon$: Then v must be ε, too.
$(\varepsilon \,\Delta\, v) \cdot (u' \,\Delta\, v') = \varepsilon \cdot (u' \,\Delta\, v') = u' \,\Delta\, v' =$
$(\varepsilon \cdot u') \,\Delta\, (\varepsilon \cdot v') = (u \cdot u') \,\Delta\, (v \cdot v')$.

Case $u' = \varepsilon$: analogous

Case $u = (u_1, ... , u_n)$, $u' = (u_1', ... , u_m')$ where n, m > 0:

Let v be partitioned into $v_1, ... , v_n$ according to u, and v' into $v_1', ... , v_m'$ according to u'. Then $v_1, ... , v_n, v_1', ... , v_m'$ is a partition of $v \cdot v'$ according to $u \cdot u'$.
$(u \,\Delta\, v) \cdot (u' \,\Delta\, v')$
$= (u_1 \,\Delta\, v_1) \cdot\ldots\cdot (u_n \,\Delta\, v_n) \cdot (u_1' \,\Delta\, v_1') \cdot\ldots\cdot (u_m' \,\Delta\, v_m')$
$= (u \cdot u') \,\Delta\, (v \cdot v')$.

The operation Δ is associative, but Δ' is not:

Proposition 14

(u Δ’ v) Δ’ w $\acute{=}$ u Δ’ (v Δ’ w)
If v in GVal’, then ′=′ holds above.
(u Δ v) Δ’ w $\dot{=}$ u Δ (v Δ’ w)
(u Δ’ v) Δ w = u Δ’ (v Δ w)
(u Δ v) Δ w = u Δ (v Δ w)

Proof:

Case (Δ’, Δ’):
Left hand side defined
iff v and w in GVal’ and LL(u Δ’ v) = UL(w) and LL(u) = UL(v)
iff v and w in GVal’ and LL(u) = UL(v) and LL(v) = UL(w) (1)
Right hand side defined
iff v Δ’ w and w in GVal’ and
LL(u) = UL(v Δ’ w) and LL(v) = UL(w) (2)
(1) implies (2): v in GVal’ implies v Δ’ w in GVal’ by Prop. 7
and v in GVal’ ⊆ CVal implies UL(v Δ’ w) = UL(v) by Prop. 10
(2) implies (1) if v in GVal’: same argument as above

Cases (Δ(’), Δ): Then we have
Left hand side defined
iff v in GVal(’) and w in GVal and LL(u) = UL(v) and LL(v) = UL(w) (1)
Right hand side defined
iff v Δ w in GVal(’) and w in GVal and
LL(u) = UL(v Δ w) and LL(v) = UL(w) (2)
′v in GVal(’)′ and ′v Δ w in GVal(’)′ are equivalent due to Prop. 8,
and v in GVal(’) implies UL (v Δ w) = UL (v) by Prop. 10.

The remainder of the proof is done for case (Δ, Δ), the other cases are analogous.
Assume both sides are defined. Then the law is proved by induction on u:
() Δ (v Δ w) = () = () Δ v = (() Δ v) Δ w
$(u_1, ..., u_n)$ Δ (v Δ w):
Let v be partitioned into $v_1, ..., v_n$ such that $LL(u_i) = UL(v_i)$,
and let w be partitioned into $w_1, ..., w_n$ such that $LL(v_i) = UL(w_i)$.
Then $v \Delta w = (v_1 \Delta w_1) \cdot ... \cdot (v_n \Delta w_n)$ holds by Prop. 13,
and this is a partition of v Δ w according to u since $UL(v_i \Delta w_i) = UL(v_i)$.
Thus: $(u_1, ..., u_n) \Delta (v \Delta w)$
$= (u_1 \Delta (v_1 \Delta w_1)) \cdot ... \cdot (u_n \Delta (v_n \Delta w_n))$
$= ((u_1 \Delta v_1) \Delta w_1) \cdot ... \cdot ((u_n \Delta v_n) \Delta w_n)$ by induction hypothesis
$= ((u_1 \Delta v_1) \cdot ... \cdot (u_n \Delta v_n)) \Delta w$ by Prop. 13
= (u Δ v) Δ w by Prop. 13 again or by definition of Δ
[] Δ (<v’> Δ w) = [] Δ <v’ Δ w> = v’ Δ w = ([] Δ <v’>) Δ w
c Δ (v Δ w) = c = c Δ v = (c Δ v) Δ w
(op u’) Δ (v Δ w) = op (u’ Δ (v Δ w)) = op ((u’ Δ v) Δ w)
= (op (u’ Δ v)) Δ w = ((op u’) Δ v) Δ w

Examples:

() Δ ([] Δ’ <>) = () Δ () = (), but (() Δ []) Δ’ <> is not defined, since ’[]’ is not grouped and the lengths don’t agree.
This example works as well if Δ is replaced by Δ’.
[] Δ (<[]> Δ’ <>) = [] Δ <> is not defined due to condition (c),
but ([] Δ <[]>) Δ’ <> = [] Δ’ <> = ().
Condition (b) (operator ′ <>′ only at top level) is as important as condition (c) (no <()> constituent) for associativity: without (b), [] Δ’ ([] Δ’ <<c>>) would be [] Δ’ <c> = c, whereas ([] Δ’ []) Δ <<c>> would not be defined since condition (a) is violated by the value [].

All the propositions of this chapter taken together mean that both *GVal’* together with the operations ′ · ′ and Δ’ and *GVal* together with the operations ′ · ′ and Δ are X-categories (see [2]).

6. An alternative definition of insertion

In the previous section, we have defined insertion Δ only for values where the number of holes of the upper value equals the number of groups of the lower one. Now we want to define insertion for arbitrary grouped values (Δ") to obtain a total operation on *GVal*, and we shall show that this operation is unsatisfactory.

If there are more holes than groups, some holes remain unfilled, and if there are less holes than groups, some groups are placed unaltered onto the top level of the resulting value.

We cannot delete the superfluous groups since we later want to define patterns performing the inverse operation of insertion, i.e. from a value *w*, we want to obtain the set of all pairs *(u, v)* such that $u \, \Delta'' \, v = w$. If subvalues were deleted during insertion, this set might be infinite, and the semantics of patterns would be incomputable.

Naturally, we must determine which holes remain resp. which groups are placed onto the top level. We decide that the process of inserting groups into holes proceeds from left to right; this introduces an ugly asymmetry into the new operation.

Examples:

Too many holes: $\mathrm{add}\ ([], \mathrm{sub}\ ([], []))\ \Delta''\ <a; b> = \mathrm{add}\ (a, \mathrm{sub}\ (b, []))$

Too many groups: $\mathrm{add}\ ([], 3)\ \Delta''\ <2; 5> = (\mathrm{add}\ (2, 3), <5>)$

$<\mathrm{add}\ ([], 3)>\ \Delta''\ <2; 5> = <\mathrm{add}\ (2, 3); 5>$

Formal definition: $u \, \Delta'' \, v$ is defined iff v is well grouped ($v \in \mathrm{GVal}$)

Let $e = <[]>$

(1) If $LL(u) > UL(v)$ then $u \, \Delta'' \, v = u \, \Delta \, (v \cdot e^{LL(u) - UL(v)})$

(2) If $LL(u) = UL(v)$ then $u \, \Delta'' \, v = u \, \Delta \, v$

(3) If $LL(u) < UL(v)$ then $u \, \Delta'' \, v = (u \cdot e^{UL(v) - LL(u)}) \, \Delta \, v$

If there are too many holes (case 1), then the lower value *v* is extended to the right by as many groups <[]> as there are superfluous holes. Those groups are inserted into these holes by Δ such that the holes seem to remain unfilled. If there are too many groups (case 3), then the upper value *u* is extended to the right by as many holes <[]> as there are superfluous groups in *v*.

Remarks to the definition:

Case (2) is a special case of both (1) and (3) since $e^0 = \varepsilon$.

Δ is always defined in it:

Case (1): $UL\ (v \cdot e^{LL(u) - UL(v)}) = UL(v) + LL(u) - UL(v) = LL(u)$

Case (3): $LL\ (u \cdot e^{UL(v) - LL(u)}) = LL(u) + UL(v) - LL(u) = UL(v)$

The operation Δ" is a total, associative operation in *GVal* with neutral element ε. The relations to *UL* and *LL* are more complex than the relations of Δ:

$LL\ (u \, \Delta'' \, v) = LL(v) + \max\ (0, LL(u) - UL(v))$

$UL\ (u \, \Delta'' \, v) = UL(u) + \max\ (0, UL(v) - LL(u))$

Most of these properties may be proved straightforward, only the proof of the associativity is very tedious.

The property $(u \, \Delta'' \, v) \cdot (u' \, \Delta'' \, v') = (u \cdot u') \, \Delta'' \, (v \cdot v')$ is not true.

Example:

$u = <\mathrm{add}\ ([], [])>$ $\qquad u' = <\mathrm{sub}\ ([], [])>$

$v = <1>$ $\qquad v' = <2>$

$(u \, \Delta'' \, v) \cdot (u' \, \Delta'' \, v') = <\mathrm{add}\ (1, [])> \cdot <\mathrm{sub}\ (2, [])>$

$= <\mathrm{add}\ (1, []); \mathrm{sub}\ (2, [])>$

$(u \cdot u') \, \Delta'' \, (v \cdot v') = <\mathrm{add}\ ([], []); \mathrm{sub}\ ([], [])>\ \Delta''\ <1; 2>$

$= <\mathrm{add}\ (1, 2); \mathrm{sub}\ ([], [])>$

Advantages of the alternative insertion Δ" are its totality and the existence of a real neutral element, but severe drawbacks are its inherent asymmetry, its complex relations with *UL* and *LL*, and the missing compatibility with concatenation. Moreover, we think that Δ" is less natural than the original operation Δ such that we shall not consider it any longer.

7. Insertion one by one

The insertion Δ defined above is "many to many": it allows for inserting many (≥ 1) primitives into each of many (≥ 0) holes. It requires that the second operand is well grouped such that the groups of primitives to be inserted into one hole can be recognized.

Now we want to define a "one to many" insertion that inserts exactly one primitive into each hole. Therefore the second operand need not be grouped. We shall define the new operation in terms of Δ and thus need a mapping to introduce ′ $\langle\rangle$′ into ungrouped values.

Definition 15 (grouping mapping)

$\text{group } () = ()$
$\text{group } (v_1, \ldots , v_n) = (\langle v_1\rangle, \ldots , \langle v_n\rangle) = \langle v_1; \ldots ; v_n\rangle$

Proposition 16

′group′ is an injective, but not surjective monoid homomorphism
from UVal to GVal.

group: UVal $\rightarrow$ GVal	u in UVal iff group (u) in GVal
group (ε) = ε	group ($u \cdot v$) = group (u) $\cdot$ group (v)
group (u) = group (v) implies u = v	
UL (group v) = UL (v)	LL (group v) = LL (v)

If u in CVal then group (u Δ(’) v) = group (u) Δ(’) v
For u not in CVal, this equation does not generally hold.
If UL (u Δ v) = UL (u) then group (u Δ v) = group (u) Δ v
A corresponding statement for Δ’ is not true.

Proof:

The first, third, fourth, and fifth line is trivial. ′group′ is not surjective since values such as $\langle a, b; c\rangle$ are not in its image.
Let $u = (u_1, \ldots , u_n)$, then group (u) = $(\langle u_1\rangle, \ldots , \langle u_n\rangle)$. group (u) always satisfies conditions (a) and (c) of GVal, (c) since the u_i are primitive and thus they are not ε. Condition (b) is equivalent to ′all u_i are in UVal′, i.e. u in UVal.
group (u Δ(’) v) = group (u) Δ(’) v:
Examples: $\text{group } ([]\ \Delta(')\ \langle a, b\rangle) = \text{group } (a, b) = \langle a; b\rangle$
whereas $\text{group } ([])\ \Delta(')\ \langle a, b\rangle = \langle []\rangle\ \Delta(')\ \langle a, b\rangle = \langle a, b\rangle$.
$\text{UL } ([], []) = 2$ and $\text{UL } (([], [])\ \Delta'\ \langle a, b; \rangle) = \text{UL } (a, b) = 2$,
but $\text{group } ([], [])\ \Delta'\ \langle a, b; \rangle = \langle []; []\rangle\ \Delta'\ \langle a, b; \rangle = \langle a, b; \rangle$,
$\text{group } (([], [])\ \Delta'\ \langle a, b; \rangle) = \text{group } (a, b) = \langle a; b\rangle$.
Proof for Δ:
u = (): group (u Δ v) = group () = () = () Δ v = group (u) Δ v.
$u = (u_1, \ldots , u_n)$: Let v be partitioned into $v_1, \ldots , v_n$ according to u.
If u is in CVal, all u_i are in CVal, and thus $\text{UL } (u_i\ \Delta\ v_i) = \text{UL } (u_i) = 1$.
If UL (u Δ v) = UL (u), then we may conclude $\text{UL } (u_i\ \Delta\ v_i) =$
$\text{UL } (u_i) = 1$, since $\text{UL } (u_i\ \Delta\ v_i) \geq \text{UL } (u_i)$ holds due to Prop. 10.
Therefore, the values $u_i\ \Delta\ v_i$ are the primitive constituents of u Δ v.
$\text{group } (u\ \Delta\ v) = (\langle u_1\ \Delta\ v_1\rangle, \ldots , \langle u_n\ \Delta\ v_n\rangle) =$
$(\langle u_1\rangle\ \Delta\ v_1, \ldots , \langle u_n\rangle\ \Delta\ v_n) = \text{group } (u)\ \Delta\ v$.
For Δ’ instead of Δ, the argument for case ′u in CVal′ works analogously, but the UL argument does not work.

Definition 17

For u, v in Val’ we define u A v = u Δ group (v)

Proposition 18

A: Val’ X Val’ $\rightarrow$ Val’ is a partial mapping.
u A v is defined iff v $\in$ UVal and LL (u) = UL (v).
() A v $\acute{=}$ () (v must equal () to make this defined)
(u_1, ..., u_n) A v = (u_1 A v_1) $\cdot$ ... $\cdot$ (u_n A v_n)
where u_i primitive and LL (u_i) = UL (v_i) and $v_1 \cdot ... \cdot v_n$ = v
[] A w $\acute{=}$ w
c A v $\acute{=}$ c (v must equal () to make this defined)
(op u) A v = op (u A v) specially <u> A v = <u A v>

Proof:
u Δ group (v) is defined iff group (v) in GVal and LL (u) = UL (group (v))
iff v in UVal and LL (u) = UL (v).
The other properties are a direct consequence of the definition of Δ, except
′[] A w $\acute{=}$ w′ that is a little bit more complex:
if [] A w is defined, UL (w) = 1 holds, and group (w) = <w>.
<[] A w> = <[]> Δ group (w) = group (w) = <w>,
thus [] A w = w.

Examples:

op [] A v = op v (only if UL (v) = 1)
if ([], t, e) A c = if (c, t, e)
([], list (a1, [], a2, a3, [])) A (b, c, d) = (**b**, list (a1, **c**, a2, a3, **d**))
add ([], []) A 1 is undefined (too many holes)
add [] A (1, 2) is undefined (too many values to insert)
[] A <c> is undefined (<c> is not ungrouped)

Now we give a list of the properties of A. Most properties are inherited from Δ, but there are some more properties since A is more restrictive than Δ.

Proposition 19

(1) Let u and v be values (elements of Val’) such that u A v is defined.
Then v is in UVal and
u is in Val (UVal, GVal’, GVal) iff u A v is in Val (UVal, GVal’, GVal),
if u is in CVal, then u A v is in CVal, the inverse is not true.
u A v = ε iff u = ε and v = ε

(2) u A v = u A v’ does not imply v = v’.
Let u, u’, v be values such that u A v and u’ A v are both defined.
Then u A v = u’ A v implies u = u’.

(3) LL (u A v) $\acute{=}$ LL (v) (4) UL (u A v) $\acute{=}$ UL (u)
(5) group (u A v) = group (u) A v = group (u) Δ group (v)
(6) $[]^k$ A v $\acute{=}$ v u A $[]^k$ $\acute{=}$ u
(7) (u A v) $\cdot$ (u’ A v’) $\acute{=}$ (u $\cdot$ u’) A (v $\cdot$ v’)
(8) (u $\square$ v) A w = u $\square$ (v A w) for $\square$ = Δ’, Δ, A
(9) (u A v) Δ w $\doteq$ u A (v Δ w)
(10) (u A v) Δ’ w may be different from u A (v Δ’ w)

Note that properties (2), (4) and (5) differ from those of Δ; in (4) and (5), the precondition ′u in CVal′ is not needed.

Proof:
(1) directly from Prop. 8
(2) Example: ([], c) A c = (c, c) = (c, []) A c
The positive statement must be proved by induction on u.
Case u = () or u = c:
Then v = v’ = () must hold to make u A v and u A v’ be defined.

Case u = []: v = [] A v = [] A v' = v'
Case u = op u':
(op u') A v = (op u') A v' implies op (u' A v) = op (u' A v'),
thus u' A v = u' A v', and hence v = v' by induction.
Case u = (u_1, ... , u_n): Then u A v = (u_1 A v_1) · ... · (u_n A v_n),
u A v' analogous with v_i replaced by v_i'.
Due to (4) that will be proved soon, we may conclude
UL (u_i A v_i) = UL (u_i) = 1, and UL (u_i A v_i') = 1.
(This conclusion is not possible for Δ since (4) does not hold generally.)
Therefore we obtain u_i A v_i = u_i A v_i' for all i,
whence v_i = v_i' by induction, and thus v = v'.

(3) LL (u A v) = LL (u Δ group (v)) $\acute{=}$ LL (group (v)) = LL (v)

(4) Since the correspondent property for Δ does not hold, we must prove this by induction on u. The proof is analogous to that of Prop. 10 except for case u = []. To make [] A v defined, UL (v) = 1 must hold, and thus
UL ([] A v) = UL (v) = 1 = UL ([]).   (For Δ, we had only '≥').

(5) The second equation holds due to the definition of A. The first equation may be proved analogously to the proof of Prop. 16 except that the line 'If u is in CVal ...' may be dropped due to (4).

(6) u A $[]^k$ = u Δ group ($[]^k$) = u Δ < [] $>^k$ $\acute{=}$ u   by Prop. 11
group ($[]^k$ A v) = group ($[]^k$) A v   (by (4))
= < [] $>^k$ Δ group (v) $\acute{=}$ group (v)   by Prop. 11
Since 'group' is injective, this implies $[]^k$ A v $\acute{=}$ v.

(7) (u A v) · (u' A v') = (u Δ group (v)) · (u' Δ group (v')) $\acute{=}$
(u · u') Δ (group (v) · group (v')) = (u · u') A (v · v')   by Prop. 13

(8) Case (A, A):
(u A v) A w = (u Δ group (v)) Δ group (w)
= u Δ (group (v) Δ group (w))   by Prop. 14
= u Δ (group (v) A w) = u Δ group (v A w)   by (5)
= u A (v A w)
Case (Δ, A):
(u Δ v) A w = (u Δ v) Δ group (w) = u Δ (v Δ group (w)) = u Δ (v A w)
Case (Δ', A): analogous

(9) (u A v) Δ w $\doteq$ u A (v Δ w)
If the left hand side is defined, then LL (u) = UL (v) holds, and the definedness of the right hand side implies LL (u) = UL (v Δ w). If both sides are defined, we may conclude UL (v Δ w) = UL (v) and Prop. 16 is applicable.
(u A v) Δ w = (u Δ group (v)) Δ w
= u Δ (group (v) Δ w) = u Δ group (v Δ w)   by Prop. 16
= u A (v Δ w)
'$\doteq$' cannot be improved to '$\acute{=}$' or '=':
(u A []) Δ <c, d>
u A ([] Δ <c, d>) = u A (c, d)
The first term is defined iff LL (u) = 1, and the second one iff LL (u) = 2. Now choose u = [] resp. u = ([], []) and you see that '$\acute{=}$' or '=' do not hold.

(10) The two terms (u A v) Δ' w and u A (v Δ' w) may be both defined and yield different results.
(([], a, []) A ([], [])) Δ' <b, c; > = ([], a, []) Δ' <b, c; > = (b, c, a)
([], a, []) A (([], []) Δ' <b, c; >) = ([], a, []) A (b, c) = (b, a, c)

These properties imply that UVal together with the operations ' · ' and 'A' is also an X-category. The mapping 'group' is an injective X-category homomorphism from (UVal, ·, A) to (GVal, ·, Δ) due to Prop. 19(5).

8. Simple insertion

In the previous section, we have restricted the general insertion Δ (many (≥ 0) holes, many (≥ 1) primitives into each hole) to a new operation A (many holes, one primitive into each hole). Another kind of restriction would be one hole, many (≥ 1) primitives into it. A suitable definition would be $u \Lambda v = u \Delta < v >$. But we shall see that it is possible without drawbacks (loss of associativity, infinitely many partitions) to allow for filling ε into the one hole, such that the new operation will be defined as $u \Lambda v = u \Delta' <v>$. Note that this is exactly the operation that we have used in [1].

Definition 20

For u, v in Val' let $u \Lambda v = u \Delta' <v>$

Proposition 21

Λ: Val' X Val' $\rightarrow$ Val' is a partial mapping.
$u \Lambda v$ is defined iff $v \in$ UVal and LL (u) = 1.
$() \Lambda v$ and $c \Lambda v$ are not defined since LL (()) = LL(c) = 0.
$[] \Lambda v \doteq v$
$(op\ u) \Lambda v = op\ (u \Lambda v)$ specially $<u> \Lambda v = <u \Lambda v>$
$(u_1, ..., u_n) \Lambda v = (u_1, \ldots, u_{i-1}) \cdot (u_i \Lambda v) \cdot (u_{i+1}, \ldots, u_n)$
where LL (u_i) = 1 and LL (u_j) = 0 for $j \neq i$.
(u_i is the primitive containing the hole of u).

Proof:
$u \Lambda$ is defined iff $u \Delta' <v>$ is defined iff
$<v>$ in GVal' and LL (u) = UL ($<v>$) iff v in UVal and LL (u) = 1.
The other properties are proved straightforward;
$<[] \Lambda v> = <[] \Delta' <v>> = <[]> \Delta' <v> = <v>$
by Prop. 11; this implies $[] \Lambda v = v$.

Examples:

op $[] \Lambda v$ = op v
add $[] \Lambda (1, 2)$ = add (1, 2)
if $(c, []) \Lambda (t, e)$ = if (c, t, e)
list $(a1, [], a2, a3) \Lambda ()$ = list (a1, a2, a3)
add $([], []) \Lambda v$ is undefined (too many holes)
$[] \Lambda <c>$ is undefined ($<c>$ is not ungrouped)

Now we give a list of the properties of Λ. Most properties are inherited from Δ', but there are some more properties since Λ is more restrictive than Δ'.

Proposition 22

(1) Let u and v be values (elements of Val') such that $u \Lambda v$ is defined.
Then v is in UVal and
u is in Val (UVal) iff $u \Lambda v$ is in Val (UVal),
if u is in CVal (GVal'), then $u \Lambda v$ is in CVal (GVal'), the inverse is not true.
u in GVal does not imply $u \Lambda v$ in GVal, and vice versa.
$u \Lambda v = \varepsilon$ iff $u = []$ and $v = \varepsilon$

(2) $u \Lambda v = u \Lambda v'$ does not imply $v = v'$.
Let u, u', v be values such that $u \Lambda v$ and $u' \Lambda v$ are both defined.
Then $u \Lambda v = u' \Lambda v$ implies $u = u'$.

(3) LL ($u \Lambda v$) $\doteq$ LL (v)
(4) UL ($u \Lambda v$) $\doteq$ if u in CVal then UL (u) else UL (u) + UL (v) - 1
(5) if UL (u) = UL ($u \Lambda v$) then group ($u \Lambda v$) = group (u) Λ v
(6) $[] \Lambda v \doteq v$ $u \Lambda [] \doteq u$
(7) if UL (w) = 0 then (for $\square = \Delta', \Delta$, A, Λ)
$(u \square v) \cdot w = (u \cdot w) \square v$ $w \cdot (u \square v) = (w \cdot u) \square v$

(8) $(u \,\Lambda\, v) \square w \;=\; u \,\Lambda\, (v \square w)$ for $\square = \Delta', \Delta, A, \Lambda$
(9) $(u \square v) \,\Lambda\, w \;\doteq\; u \square (v \,\Lambda\, w)$ for $\square = \Delta, A$
(10) $(u \,\Delta'\, v) \,\Lambda\, w \;\acute{=}\; u \,\Delta'\, (v \,\Lambda\, w)$

Proof:

(1) directly from Prop. 7 for Δ'
The examples given there may be translated into examples for here:
$[] \,\Lambda\, c = c \qquad [] \,\Lambda\, () = () \qquad \langle[]\rangle \,\Lambda\, () = \langle\rangle$
$u \,\Lambda\, v = \varepsilon$ iff $u \,\Delta'\, v = \varepsilon$
iff $u = []^k$ and $\langle v\rangle = \langle\rangle^k$ iff $u = []$ and $\langle v\rangle = \langle\rangle$
iff $u = []$ and $v = \varepsilon$

(2) Example: $([], c) \,\Lambda\, c \;=\; (c, c) \;=\; (c, []) \,\Lambda\, c$
The positive statement is proved by induction on u.
Case $u = ()$ or $u = c$: impossible
Case $u = []$: $v = [] \,\Lambda\, v = [] \,\Lambda\, v' = v'$
Case $u = op\ u'$:
$(op\ u') \,\Lambda\, v = (op\ u') \,\Lambda\, v'$ implies $op\ (u' \,\Lambda\, v) = op\ (u' \,\Lambda\, v')$,
thus $u' \,\Lambda\, v = u' \,\Lambda\, v'$, and hence $v = v'$ by induction.
Case $u = (u_1, \dots, u_n)$: Then $u \,\Lambda\, v = (u_1, \dots, u_{i-1}) \cdot (u_i \,\Lambda\, v) \cdot (u_{i+1}, \dots, u_n)$,
$u \,\Lambda\, v'$ analogous with v replaced by v'.
It follows $u_i \,\Lambda\, v = u_i \,\Lambda\, v'$ and $v = v'$ by induction.

(3) $LL\ (u \,\Lambda\, v) \;=\; LL\ (u \,\Delta'\, \langle v\rangle) \;\acute{=}\; LL\ (\langle v\rangle) \;=\; LL\ (v)$

(4) Let u in CVal. Then
$UL\ (u \,\Lambda\, v) \;=\; UL\ (u \,\Delta'\, \langle v\rangle) \;\acute{=}\; UL\ (u)$.
Now assume u is not in CVal and $u \,\Lambda\, v$ is defined, i.e. $LL\ (u) = 1$.
Then $u = (u_1, \dots, u_n)$ with $u_i = []$ for some i and $LL\ (u_j) = 0$ for $j \neq i$.
By Prop. 21, we obtain $u \,\Lambda\, v = (u_1, \dots, u_{i-1}) \cdot v \cdot (u_{i+1}, \dots, u_n)$,
and thus $UL\ (u \,\Lambda\, v) = UL\ (u) - 1 + UL\ (v)$.

(5) $u \,\Lambda\, v = (u_1, \dots, u_{i-1}) \cdot (u_i \,\Lambda\, v) \cdot (u_{i+1}, \dots, u_n)$
$UL\ (u \,\Lambda\, v) = UL\ (v)$ implies $UL\ (u_i \,\Lambda\, v) = 1$.
Thus the primitives of $u \,\Lambda\, v$ are $u_1, \dots, u_{i-1}, u_i \,\Lambda\, v, u_{i+1}, \dots, u_n$,
and this implies the statement together with $\langle u_i \,\Lambda\, v\rangle = \langle u_i\rangle \,\Lambda\, v$.

(6) $u \,\Lambda\, [] \;=\; u \,\Delta'\, \langle[]\rangle \;=\; u$
$[] \,\Lambda\, v \;=\; v$ by Prop. 21

(7) Case Δ', Δ, A:
$(u \square v) \cdot w \;=\; (u \square v) \cdot (w \square ()) \;\acute{=}\; (u \cdot w) \square (v \cdot ()) \;=\; (u \cdot w) \square v$
Both sides are equally defined since $LL\ (u \cdot w) = LL\ (u) + LL\ (w) = LL\ (u)$
Case Λ: $(u \,\Lambda\, v) \cdot w \;=\; (u \,\Delta'\, \langle v\rangle) \cdot w \;=\; (u \cdot w) \,\Delta'\, \langle v\rangle \;=\; (u \cdot w) \,\Lambda\, v$

(8-10) We have to prove $'='$ for $(\Lambda, \square)$, $'\doteq'$ for (Δ, Λ) and (A, Λ),
and $'\acute{=}'$ for (Δ', Λ).
Case (Δ', Λ): $(u \,\Delta'\, v) \,\Lambda\, w \;=\; (u \,\Delta'\, v) \,\Delta'\, \langle w\rangle$
$\acute{=}\; u \,\Delta'\, (v \,\Delta'\, \langle w\rangle) \;=\; u \,\Delta'\, (v \,\Lambda\, w)$
Ex.: $(() \,\Delta'\, []) \,\Lambda\, ()$ is undefined, but $() \,\Delta'\, ([] \,\Lambda\, ()) = () \,\Delta'\, () = ()$
Case (Δ, Λ): analogously replied to (Δ, Δ') with $'\acute{=}'$ replaced by $'\doteq'$.
Ex.: The example above remains valid if Δ' is replaced by Δ.
$([] \,\Delta\, \langle[]\rangle) \,\Lambda\, () = [] \,\Lambda\, () = ()$, but $[] \,\Delta\, (\langle[]\rangle \,\Lambda\, ()) = [] \,\Delta\, \langle\rangle$ is undefined.
Case (A, Λ): analogous to the case (A, Δ) in Prop. 19(9) using (5).
Ex.: $(u\ A\ []) \,\Lambda\, (c, d)$ is defined iff $LL\ (u) = 1$,
$u\ A\ ([] \,\Lambda\, (c, d)) = u\ A\ (c, d)$ is defined iff $LL\ (u) = 2$.
Case $(\Lambda, \square)$:
$(u \,\Lambda\, v) \square w$ is defined iff
v in UVal and $LL\ (u) = 1$ and w in X and $LL\ (u \,\Lambda\, v) = Y$ iff
$v \square w$ in UVal and $LL\ (u) = 1$ and w in X and $LL\ (v) = Y$ iff
$u \,\Lambda\, (v \square w)$ defined
Here, X is a set of values depending on $\square$, and Y is a number

depending on □ and UL (w).
We have just seen that both sides are equally defined.
They are really equal since
(u Λ v) □ w = (u Δ' <v>) □ w $\doteq$ u Δ' (<v> □ w) =
u Δ' <v □ w> = u Λ (v □ w).
This proof uses some properties that are shared by the four insertions:
<v> □ w = <v □ w> Def. 6, Prop. 18, Prop. 21
v in UVal iff v □ w in UVal Prop. 7, 8, 19(1), 22(1)

The virtue of the simple insertion is its less complexity, the '=' sign in the associative law, and its relation to the application of an operator to a value: op [] Λ v = op v. Furthermore, it is the only kind of insertion except Δ' allowing for filling ε into a hole. For patterns, we shall introduce operations that are inverse to concatenation ·, general insertion Δ, A, and simple insertion Λ. The distinction between general and simple insertion makes sense since the inverse of simple insertion will be probably easier to implement than the inverse of general insertion.

9. Number of partitions

Finally, we investigate how many partitions into two values a given value has.

9.1. Definition

Let □ be a partial operation in Val', and let w be an element of Val'. Then we denote the number of pairs u, v such that u □ v is defined and equals w, by $n_{\square}$ (w).

9.2. Concatenation

$n_{\cdot}$ (w) = UL (w) + 1
since $(w_1, \ldots, w_n) = (w_1, \ldots, w_i) \cdot (w_{i+1}, \ldots, w_n)$ for $0 \le i \le n$

9.3. Most general insertion

$n_{\Delta'}$ (w) = ∞
since $w = (w \cdot [\,]^k)\ \Delta'\ (<[\,]>^{LL(w)} \cdot <>^k)$ for all $k \ge 0$.

9.4. Insertion of one primitive into each hole

First, we consider n_A since it is easier to calculate than n_Δ and n_Λ.
n_A (()) = 1: () = () A ()
n_A ([]) = 1: [] = [] A []
n_A (c) = 2: c = c A () = [] A c
n_A (op w) = (if (op w) in UVal then 1 else 0) + n_A (w)
op w = (op u) A v = [] A (op w)
The first partition is possible whenever w = u A v, and the second one is defined only if (op w) in UVal.
$n_A (w_1, \ldots, w_n) = n_A (w_1) \cdot \ldots \cdot n_A (w_n)$
If $w_i = u_i$ A v_i, then w = $(u_1 \cdot \ldots \cdot u_n)$ A $(v_1 \cdot \ldots \cdot v_n)$.
Let vice versa w = u A v. Then UL (u) = UL (u A v) = n, and thus
u = $(u_1, \ldots, u_n)$ and w = $(u_1$ A $v_1) \cdot \ldots \cdot (u_n$ A $v_n)$.
Since UL (u_i A v_i) = UL (u_i) = 1, we have $w_i = u_i$ A v_i.
Examples:
n_A (add (a, b)) = 1 + n_A (a, b) = 1 + n_A (a) · n_A (b) = 1 + 2·2 = 5.
n_A (if ([], add (a, b), sub (a, b))) = 1 + 1·5·5 = 26
n_A (if (eq (a, b), add (a, b), sub (a, b))) = 1 + 5·5·5 = 126

9.5. General insertion

Remark: For all w in Val', n_A (w) ≤ n_Δ (w) holds.
Proof: w = u A v implies w = u Δ group (v); 'group' is injective.

We shall show that n_Δ is not essentially greater than n_A.

n_Δ (()) = 1:   () = () Δ ()
n_Δ ([]) = 1:   [] = [] Δ <[]>
n_Δ (c) = 2:   c = c Δ () = [] Δ <c>

n_Δ (op w) = (if (op w) in UVal then 1 else 0) + n_Δ (w)
op w = (op u) Δ v = [] Δ <op w>
The first partition is possible whenever w = u Δ v, and the second one is defined only if (op w) in UVal.

Case w = (w_1, ... , w_n):
This case is more difficult then the respective one of n_A since some w_i may be extracted together e.g. (a, b, c, d) = (a, [], d) Δ <b, c>. This sample partition is only possible if b and c are ungrouped such that we have to distinguish two cases:

Case w not in UVal. Let w_k be not in UVal.
Then we may extract subvalues of w_k (as far as they are ungrouped), but we cannot extract the whole value w_k. Therefore, any extracted group
(w_i, ... , w_j) cannot contain w_k, and we obtain:
n_Δ (w_1, ... , w_n) = n_Δ (w_1, ... , w_{k-1}) · n_Δ (w_k) · n_Δ (w_{k+1}, ... , w_n).
Corollary: n_Δ (<w_1; ... ; w_n>) = n_Δ (w_1) · ... · n_Δ (w_n)
Proof: Note that <w_1; ... ; w_n> = (<w_1>, ... , <w_n>) and
n_Δ (<w_i>) = n_Δ (w_i).

Case w in UVal i.e. all w_i are in UVal.
Consider the last primitive w_n. Either only some subvalues of w_n (may be () or w_n itself) are extracted, or w_n is extracted as part of a group
(w_i, ... , w_n) where i < n.
w = (u · u') Δ (v · v') where u Δ v = (w_1, ... , w_{n-1}) and u' Δ v' = w_n
or w = (u · []) Δ (v · <w_{k+1}, ... , w_n>)
where k + 1 < n and u Δ v = (w_1, ... , w_k).
Thus: n_Δ (w) = n_Δ (w_1, ... , w_{n-1}) · n_Δ (w_n) + n_Δ (w_1, ... , w_{n-2}) +
n_Δ (w_1, ... , w_{n-3}) + ... + n_Δ (()).
We shall discuss this recursive formula further after some examples.

Examples:
Assume u, v, w are ungrouped values.
n_Δ (u, v) = n_Δ (u) · n_Δ (v) + n_Δ (()) = n_Δ (u) · n_Δ (v) + 1
n_Δ (u, v, w) = n_Δ (u, v) · n_Δ (w) + n_Δ (u) + n_Δ (()) =
n_Δ (u) · n_Δ (v) · n_Δ (w) + n_Δ (u) + n_Δ (w) + 1
n_Δ (add (a, b)) = 1 + n_Δ (a, b) = 1 + 2 · 2 + 1 = 6
n_Δ (if ([], add (a, b), sub (a, b))) = 1 + 1·6·6 + 1 + 6 + 1 = 45
n_Δ (if (eq (a, b), add (a, b), sub (a, b))) = 1 + 6·6·6 + 6 + 6 + 1 = 230

Discussion of the formula for n_Δ (w_1, ... , w_n) where all the w_i are unbound:

(1) $n_\Delta (w_1, \ldots, w_n) = n_\Delta (w_1, \ldots, w_{n-1}) \cdot n_\Delta (w_n) + \sum_{k=0}^{n-2} n_\Delta (w_1, \ldots, w_k)$

Assume n > 1, and express n_Δ (w_1, ... , w_{n-1}) by formula (1) applied to (n-1) instead of n, and take the difference of both equations. We obtain
n_Δ (w_1, ... , w_n) - n_Δ (w_1, ... , w_{n-1}) = n_Δ (w_1, ... , w_{n-1}) · n_Δ (w_n) -
n_Δ (w_1, ... , w_{n-2}) · n_Δ (w_{n-1}) + n_Δ (w_1, ... , w_{n-2})
and thus

(2) $n_\Delta (w_1, \ldots, w_n) = (1 + n_\Delta (w_n)) \cdot n_\Delta (w_1, \ldots, w_{n-1}) + (1 - n_\Delta (w_{n-1})) \cdot n_\Delta (w_1, \ldots, w_{n-2})$

Formula (1) implies that n_Δ (w_1, ... , w_n) monotonically depends on n_Δ (w_i). Therefore, we have the following property:

(3) $B \le n_\Delta (w_i) \le C$ for all i implies $F_n (B) \le n_\Delta (w_1, \ldots, w_n) \le F_n (C)$
where $F_0 (X) = 1$ and $F_1 (X) = X$, and
$F_n (X) = (1 + X) \cdot F_{n-1} (X) + (1 - X) \cdot F_{n-2} (X)$

Formula (3) is immediately derived from (2). The numbers F_n (X) may be explicitly calculated by the

same method as applied to the Fibonacci sequence.
Result:

(4) Let $\rho = \sqrt{(X-1)^2 + 4}$

$\alpha = \frac{1}{2}(X + 1 + \rho)$ $\qquad\qquad \beta = \frac{1}{2}(X + 1 - \rho)$

$a = \frac{1}{2}(1 + \frac{X-1}{\rho})$ $\qquad\qquad b = \frac{1}{2}(1 - \frac{X-1}{\rho})$

Then $F_n(X) = a\alpha^n + b\beta^n = \lceil a\alpha^n \rceil$ for natural X

$\lceil\ \rceil$ means rounding upward to the next integer.

(5) Estimations: (for $X \geq 1$)

$0 \leq b \leq \frac{1}{X+1}$ $\qquad\qquad 0 \leq \beta \leq 1$

$1 - \frac{1}{X+1} \leq a \leq 1$ $\qquad\qquad X + \frac{1}{X} \leq \alpha \leq X + \min(1, \frac{1}{X-1})$

Note that the estimations imply $0 \leq b\beta^n < 1$ and since $F_n(X)$ is an integer if X is natural, the rounding to the ceiling is correct.

Proof of the estimations:

$X - 1 = \sqrt{(X-1)^2} \leq \rho \leq \sqrt{(X-1)^2 + 4X} = X + 1$

If this estimation for ρ is inserted into the definitions of a, b, α, and β, the formulae above result except $X \leq \alpha \leq X + 1$. Note that $(\alpha\text{-}X)(\alpha\text{-}1) = 1$, and hence $\alpha = X + \frac{1}{\alpha - 1}$. If the first estimation for α is applied to the occurrence of α on the right hand side of the very last formula, another estimation for α results. The final one is obtained by combining these two estimations.

9.6. Insertion into one hole

It will turn out that the formulae for n_Λ are more complex than the other ones.

n_Λ (()) = 1: $\qquad$ () = () Λ ()

n_Λ ([]) = 1: $\qquad$ [] = [] Λ []

n_Λ (c) = 3: $\qquad$ c = [] Λ c = ([], c) Λ () = (c, []) Λ ()

We call the last two partitions degenerated.

n_Λ (op w) = $\qquad$ (if LL (w) = 0 then 2 else 0) +
(if (op w) in UVal then 1 else 0) + n_Λ (w)

op w = (op u) Λ v = [] Λ (op w) = ([], op w) Λ () = (op w, []) Λ ()

The first partition is possible whenever w = u Λ v, the second one is defined iff (op w) in UVal, and the two degenerated ones are defined iff LL (w) = 0.

Case w = (w_1, ... , w_n):

This case is more difficult then the respective one of n_Δ since some w_i may be extracted together e.g. (a, b, c, d) = (a, [], d) Λ (b, c), but this partition is only possible if b and c are ungrouped and a and d do not contain holes.

A partition w = u Λ v must obey the following two rules:

1) v must be in UVal i.e. all grouping operators occurring in w must remain in u.
2) u must contain exactly one hole i.e. all holes occurring in w must be extracted into v.

Case w not in UVal. Let w_k be not in UVal.

Let $w^{(1)} = (w_1, \ldots, w_{k-1})$, $w^{(2)} = w_k$, and $w^{(3)} = (w_{k+1}, \ldots, w_n)$. Thus, $w = w^{(1)} \cdot w^{(2)} \cdot w^{(3)}$, UL $(w^{(2)}) = 1$, $w^{(2)}$ not in UVal.

If at least two of the three values $w^{(i)}$ contain holes, n_Λ (w) = 0 holds, since we had to extract at least those values containing holes such that only one hole is remaining in u, but we cannot extract the whole value $w^{(2)}$ since it is not in UVal.

If exactly one of the three values, namely $w^{(j)}$, contains a hole, then $n_\Lambda (w) = n_\Lambda (w^{(j)})$ since the partitioning must cut off the part containing the hole, but cannot go beyond the value $w^{(j)}$ because the middle value $w^{(2)}$ is not ungrouped.

If w does not contain holes,

$n_\Lambda (w) = n_\Lambda (w^{(1)}) + n_\Lambda (w^{(2)}) + n_\Lambda (w^{(3)}) - 2$

since the grouping operator in $w^{(2)}$ cannot be extracted into v and thus splits w into three regions.

The partition may be performed either in $w^{(1)}$ or in $w^{(2)}$ or in $w^{(3)}$, therefore their respective partition numbers must be added. The two degenerated partitions $w = w^{(1)} \cdot [] \cdot w^{(2)} \cdot w^{(3)} \Lambda ()$ and $w = w^{(1)} \cdot w^{(2)} \cdot [] \cdot w^{(3)} \Lambda ()$ are counted twice and must be subtracted.

Example showing the inner consistency of the formula:

Let w = <w'> such that $w^{(1)} = w^{(3)} = ()$ and $w^{(2)} = w$. Assume w does not contain holes.

$n_\Lambda (w) = n_\Lambda (()) + n_\Lambda (w) + n_\Lambda (()) - 2 = n_\Lambda (w)$

Case w in UVal, $w = (w_1, \dots, w_n)$.

Case w contains holes.

Let k be the minimal index and l the maximal index such that
$LL (w_k) > 0$ resp. $LL (w_l) > 0$.

If $k < l$, then we must extract at least the subvalue $(w_k, \dots, w_l)$, such that $w = (w_1, \dots, w_{i-1}, [], w_{j+1}, \dots, w_n) \Lambda (w_i, \dots, w_j)$ where $1 \le i \le k$ and $l \le j \le n$. These are $k \cdot (n + 1 - l)$ possibilities.

If $k = l$, then we may extract a part of w_k or a subvalue containing w_k.
(1) $w = (w_1, \dots, w_{k-1}, u, w_{k+1}, \dots, w_n) \Lambda v$ where $u \Lambda v = w_k$.
(2) $w = (w_1, \dots, w_{i-1}, [], w_{j+1}, \dots, w_n) \Lambda (w_i, \dots, w_j)$
where $1 \le i \le k \le j \le n$.
(1) are $n_\Lambda (w_k)$ possibilities, (2) gives $k \cdot (n + 1 - k)$ ones. Case u = [], $v = w_k$ of (1) and case $i = j = k$ of (2) are identical, thus we obtain
$n_\Lambda (w) = n_\Lambda (w_k) + k \cdot (n + 1 - k) - 1$

Case LL (w) = 0 i.e. w does not contain holes.

(1) $w = (w_1, \dots, w_{i-1}, u, w_{i+1}, \dots, w_n) \Lambda v$ where $u \Lambda v = w_i$ and
UL (u) = 1 (the degenerated partitions of w_i are excluded).
(2) $w = (w_1, \dots, w_i, [], w_{i+1}, \dots, w_n) \Lambda ()$ where $0 \le i \le n$
(3) $w = (w_1, \dots, w_{k-1}, [], w_{l+1}, \dots, w_n) \Lambda (w_k, \dots, w_l)$
where $1 \le k < l \le n$.

Numbers of partitions:
(1) $n_\Lambda (w_1) + \dots + n_\Lambda (w_n) - 2n$ (the degenerated partitions)
(2) n + 1
(3) n - 1 for k = 1, n - 2 for k = 2, ... , 1 for k = n - 1.
Summing up results in:
$n_\Lambda (w) = n_\Lambda (w_1) + \dots + n_\Lambda (w_n) + \frac{1}{2} (n-2)(n-1)$

Examples:

$n_\Lambda (\text{add}(a, [])) = 1 + n_\Lambda (a, []) = 1 + n_\Lambda ([]) + 2 \cdot (2 + 1 - 2) - 1 =$
$1 + 1 + 2 - 1 = 3$

$n_\Lambda (\text{add}(a, b)) = 3 + n_\Lambda (a, b) =$
$3 + n_\Lambda (a) + n_\Lambda (b) + \frac{1}{2} (2-2)(2-1) = 3 + 3 + 3 + 0 = 9$

$n_\Lambda (\text{if}(\text{eq}(a, b), \text{add}(a, []), \text{sub}(b, []))) = 1 + 2 \cdot (3 + 1 - 3) = 1 + 2 = 3$

$n_\Lambda (\text{if}(\text{eq}(a, []), \text{add}(a, b), \text{sub}(a, b)))$
$= 1 + n_\Lambda (\text{eq}(a, [])) + 1 \cdot (3 + 1 - 1) - 1 = 1 + 3 + 3 - 1 = 6$

$n_\Lambda (\text{if}(\text{eq}(a, b), \text{add}(a, b), \text{sub}(a, b)))$
$= 3 + n_\Lambda (\text{eq}(a, b)) + n_\Lambda (\text{add}(a, b) + n_\Lambda (\text{sub}(a, b)) + \frac{1}{2} (3-2)(3-1) =$
$= 3 + 9 + 9 + 9 + 1 = 31$

10. Conclusion

At last, we shall summarize the properties of the four kinds of insertion: Δ', Δ, A, Λ. We use the symbol □ to denote the insertion operators.

e = f	e is defined iff f is defined, both are equal
e = f	if e is defined then f is defined and both are equal
$e \doteq f$	if both e and f are defined, they are equal.

Informal description

	Δ’	Δ	A	Λ
Number of holes:	≥ 0	≥ 0	≥ 0	1
Number of primitives per hole:	≥ 0	≥ 1	1	≥ 0

Typical examples
list ([], a1, [], a2, []) Δ’ <b1, b2; c1, c2; > = list (**b1, b2**, a1, **c1, c2**, a2)
list ([], a1, [], a2) Δ <b1, b2; c1, c2> = list (**b1, b2**, a1, **c1, c2**, a2)
list ([], a1, [], a2) A (b, c) = list (**b**, a1, **c**, a2)
list (a1, [], a2) Λ (b, c) = list (a1, **b, c**, a2)
list (a1, [], a2) Λ () = list (a1, a2)

Relations among the operations (Def. 6, Def. 17, Def. 20)
u Δ v ≐ u Δ’ v u A v = u Δ group (v) u Λ v = u Δ’ <v>

Domains of definedness (Def. 6, Prop. 18, Prop. 21)

u Δ’ v	is defined iff v in GVal’	and LL (u) = UL (v)
u Δ v	is defined iff v in GVal	and LL (u) = UL (v)
u A v	is defined iff v in UVal	and LL (u) = UL (v)
u Λ v	is defined iff v in UVal	and LL (u) = 1

Subsets of values (Prop. 3)
ε ∈ GVal ⊂ GVal’ ⊂ CVal ⊂ Val ⊂ Val’
ε ∈ UVal ⊂ Val ⊂ Val’
GVal’ ∩ UVal = {ε}

Closure properties of the subsets (Prop. 7, 8, 19(1), 22(1))

u in X implies u □ v in X holds for some sets X and operations □

	Val	UVal	CVal	GVal’	GVal
Δ’ and Λ	+	+	+	+	-
Δ and A	+	+	+	+	+

u □ v in X implies u in X holds for some sets X and operations □

	Val	UVal	CVal	GVal’	GVal
Δ’ and Λ	+	+	-	-	-
Δ and A	+	+	-	+	+

When does ε result? (Prop. 7, 8, 19(1), 22(1))
u Δ’ v = ε iff u = $[]^k$ and v = $<>^k$ for some k ∈ $\mathbf{N}_0$
u Δ v = ε iff u = v = ε
u A v = ε iff u = v = ε
u Λ v = ε iff u = [] and v = ε

Special shapes of the operands (Def. 6, Prop. 18, Prop. 21)
() □ v ≐ () c □ v ≐ c
(op u) □ v = op (u □ v) <u> □ v = <u □ v>
u □ () ≐ u
[] □ <w> ≐ w for Δ’, Δ [] □ w ≐ w for A, Λ

Neutral elements (Prop. 11, 19(6), 21(6))

Δ’ and Δ:	$< [\,] >^k$	$< [\,] >^k \;\square\; v \doteq v$	$u \;\square < [\,] >^k \doteq u$
A:	$[\,]^k$	$[\,]^k \;A\; v \doteq v$	$u \;A\; [\,]^k \doteq u$
Λ:	$[\,]$	$[\,] \;\Lambda\; v \doteq v$	$u \;\Lambda\; [\,] \doteq u$

Omitting common operands (End of chapter 4, Prop. 19(2), 22(2))

$u \square v = u' \square v$ does not imply $u = u'$ for any of the four operations

$u \square v = u \square v'$ implies $v = v'$: true for A and Λ, false for Δ’ and Δ

Relations to lengths (Prop. 9, 10, 19(3), 19(4), 22(3), 22(4))

LL $(u \square v) \doteq$ LL (v) — for Δ’, Δ, A, Λ

if u in CVal then UL $(u \square v) \doteq$ UL (v) — for Δ’, Δ, A, Λ

UL $(u \;A\; v) \doteq$ UL (v)

UL $(u \;\Lambda\; v) \doteq$ if u in CVal then UL (u) else UL (u) + UL (v) - 1

if $u \;\Delta\; v$ is defined, then UL $(u \;\Delta\; v) \geq$ UL (u)

Compatibility with grouping (Prop. 16, 19(5), 22(5))

if u in CVal then group $(u \square v)$ = group $(u) \square v$ — for Δ’, Δ, A, Λ

group $(u \;A\; v)$ = group $(u) \;A\; v$ = group $(u) \;\Delta\;$ group (v)

if UL $(u \square v) \doteq$ UL (u) then group $(u \square v)$ = group $(u) \square v$ — for Δ, A, Λ

Compatibility with concatenation (Prop. 13, 19(7), 22(7))

if LL (w) = LL (w’) = 0 then

$w \cdot (u \square v) \cdot w' = (w \cdot u \cdot w') \square v$ — for Δ’, Δ, A, Λ

$(u \square v) \cdot (u' \square v') \doteq (u \cdot u') \square (v \cdot v')$ — for Δ’, Δ, A

Associativity (Prop. 14, 19(8-10), 22(8-10))

$(u \square v) \square w = u \square (v \square w)$ — for Δ, A, Λ

$(u \;\Delta'\; v) \;\Delta'\; w \doteq u \;\Delta'\; (v \;\Delta'\; w)$ — if v in GVal’ then ′=′

$(u \square_1 v) \square_2 w \;\;?\;\; u \square_1 (v \square_2 w)$

	Δ’	Δ	A	Λ
Δ’	=	=	=	=
Δ	$\doteq$	=	=	$\doteq$
A	≠	$\doteq$	=	$\doteq$
Λ	=	=	=	=

X-categories

X1 = (UVal, ·, A) X2 = (GVal, ·, Δ) X3 = (GVal’, ·, Δ’)

′group′ is an injective X-category homomorphism from X1 to X2, and X2 is a sub-X-category of X3.

Number of partitions (Chapter 9)

$n_\square (w) = |\{ \text{pair } u, v \mid u \square v = w \}|$

$n_\cdot (w) = \text{UL}(w) + 1$

$n_{\Delta'} (w) = \infty$

For □ = A and Δ:

$n_\square (()) = 1$ $\quad n_\square ([\,]) = 1$

$n_\square (c) = 2$

$n_\square (op\ w) = (\text{if } (op\ w) \text{ in UVal then 1 else 0}) + n_\square (w)$

$n_A (w_1, \ldots, w_n) = n_A (w_1) \cdot \ldots \cdot n_A (w_n)$

For w in UVal and n > 1:

$n_\Delta (w_1, \ldots, w_n) =$

$(1 + n_\Delta (w_n)) \cdot n_\Delta (w_1, \ldots, w_{n-1}) + (1 - n_\Delta (w_{n-1})) \cdot n_\Delta (w_1, \ldots, w_{n-2})$

Estimations see section (9.5)

n_Λ (()) = 1 $\quad$ n_Λ ([]) = 1
n_Λ (c) = 3
n_Λ (op w) = (if (op w) in UVal then 1 else 0) + (if LL (w) = 0 then 2 else 0) + n_Λ (w)
For w = (w_1, ... , w_n) in UVal:
  Case LL (w) > 0:
    Let k minimal and l maximal such that LL (w_k) > 0 resp. LL (w_l) > 0.
    k < l: $n_\Lambda (w) = k \cdot (n + 1 - l)$
    k = l: $n_\Lambda (w) = n_\Lambda (w_k) + k \cdot (n + 1 - k) - 1$
  Case LL (w) = 0: $n_\Lambda (w) = n_\Lambda (w_1) + ... + n_\Lambda (w_n) + \frac{1}{2} (n - 2) (n - 1)$
For w not in UVal see section (9.6)

References

[1] Heckmann, R.: A Proposal for the Syntactic Part of the PROSPECTRA Transformation Language, [S.1.6-SN-6.0]

[2] Hotz, G.: Schaltkreistheorie, Walter de Gruyter & Co, (1974)