Discussiones Mathematicae
General Algebra and Applications 25 (2005) 165–219

A SEMANTIC CONSTRUCTION OF TWO-ARY INTEGERS

Gabriele Ricci

Universitá di Parma
Dipartimento di Matematica,
I–43100 Parma, Italy

e-mail: gabriele.ricci@unipr.it

Abstract

To binary trees, two-ary integers are what usual integers are to natural numbers, seen as unary trees. We can represent two-ary integers as binary trees too, yet with leaves labelled by binary words and with a structural restriction. In a sense, they are simpler than the binary trees, they relativize. Hence, contrary to the extensions known from Arithmetic and Algebra, this integer extension does not make the starting objects more complex.

We use a *semantic construction* to get this extension. This method differs from the algebraic ones, mainly because it is able to find equational features of the extended objects. Two-ary integers turn out to form the free algebra corresponding to the Jónsson–Tarski's "paradoxical" equations. This entails that they have a "sum" operation as well as other operations of higher dimensions.

Two-ary integers can provide LISP memories with convenient direct access jumps and the above low complexity hints at feasible hardware implementations.

Keywords: universal matrix, analytic monoid, LISP, semantics, jump.

CCS codes: E1, Fm.

2000 Mathematics Subject Classification: 08B20, 68P05.

0. Preliminaries

0.0. Extending a generalization

We will define two-ary integers as an indirect generalization of integer numbers: as the latter integers *extend* natural numbers, the former have to extend some *generalization* of natural numbers in a similar way. Then, the latter generalization will identify the former.

We can generalize natural numbers in several ways depending on how we see them. For instance, if we see a natural number as a word on a singleton alphabet, then a larger alphabet, let us say a binary one, will define such a generalization as binary words. This is to say that we replace the single unary operation of successor with two unary ones, while keeping a single zero and a single partial predecessor. Then, the extension will concern such words.

M. Servi introduced such an approach in [22] by his “clans”. They inhabit the single infinite binary tree corresponding to the words to be extended: its root is the zero (or empty word) and its finite sub-paths from the root are such words. Clans consist of relational structures with one predecessor and two successors as for the words they extend. Their ensuing theory reached results of an order-theoretic interest [23]–[25].

Our approach, on the contrary, comes from seeing a natural number as a unary tree, a finite chain with the zero leaf. Then, binary trees will exemplify the generalization we want to extend: we now start from a single binary successor operation with two partial predecessors and any number of “zeroes”.

Natural numbers are “unary”, because, according to Peano’s axioms, we can think of them as the terms of a singleton unary species with a single unknown (the zero). Hence, one might well think of the “two-ary” natural numbers as the terms for the singleton binary species on arbitrary unknowns (the “zeroes”). In spite of this algebraic view, the algebraic methods cannot provide them with their “integer” extension.

0.1. The extension problem

The usual construction of integers by algebraic extension starts from $\boldsymbol{N}\times\boldsymbol{N}$, the set of pairs $\langle m,n\rangle$ of natural numbers, and divides it by the equivalence $\equiv$, such that $\langle m',n'\rangle \equiv \langle m'',n''\rangle$ iff $m'+n''=m''+n'$. In the two-ary case, this does not work because the “sum” does not merely involve two binary trees, but two (indexed) forests of them (see the term matrix product in 1.6 and 1.7 (B)).

Then, one might resort to the Universal Algebra construction of free algebras from their equational specifications. In fact, one could use it also for the usual integers from the two equations stating that a unary successor is the inverse of a (total) unary predecessor.

Unfortunately, this abstract construction is not an extension. It looses the link with the unary or binary trees concerned.

Furthermore, one has to guess both the freedom and the new equations. On the contrary, they are what we want to prove and find in the two-ary case. Then, we need a third way to construct our two-ary integers.

0.2. Semantic constructions

Are intermediate between algebraic extensions and equational constructions. As the former, they start from the objects we want to extend, yet they do not involve their "sums" (universal matrix products [20]). As the latter, they require to guess the species of the extended algebra, yet neither its freedom nor its equations.

To introduce this new idea let us outline how it could work in the familiar case of usual integers. We do from the point of view of a programmer, who considers an integer $j - i$ as the jump from a present address i to a new address j in a direct access standard memory. This also serves to introduce the LISP "addressing" problem of 0.4.

Without the direct access capability, our programmer has a sequential memory: a semi-infinite Turing tape. He can go from address i to any j through some finite succession of two "atomic" jumps: an upward one, $+1$ for successor, and a downward one, -1 for predecessor, where $+1\colon \boldsymbol{N} \to \boldsymbol{N}$, while -1 only is a partial function.

Formally, all such successions form the minimal set $\boldsymbol{T}$ such that it contains the empty succession, $\emptyset$, and is closed under atomic jumps, $\{+1, -1\} \times \boldsymbol{T} \subseteq \boldsymbol{T}$. Hence, they are the terms (words) for a binary species of unary "symbols". We use such functions as "symbols" also because different practices, as the ones of Algebra, can reach inconsistencies as 1.5 will recall.

These binary word have a natural functional "semantics". In fact, we can define *the semantics* s_w *of* word $w \in \boldsymbol{T}$ by word induction as

$$\begin{aligned} s_\emptyset &= \boldsymbol{i_N}\ , \\ s_{\langle +1,v\rangle} &= +1 \cdot s_v \ \text{ and} \\ s_{\langle -1,v\rangle} &= -1 \cdot s_v\ , \end{aligned} \tag{1}$$

where $\cdot$ denotes functional composition as in 0.5. For instance, $s_{\langle +1, \emptyset \rangle} = +1$ and in general any semantics is a partial function, $s_w \subseteq \boldsymbol{N} \times \boldsymbol{N}$.

Then, we consider the domain $\Delta_w \subseteq \boldsymbol{N}$ of the semantics of any word w, called the *domain induced* by w, which is a co-finite segment. For instance, by equation (1) $\Delta_{\langle -1, \emptyset \rangle} \subset \boldsymbol{N}$ is the set of positive natural numbers, because we cannot backtrack our tape below its beginning. Clearly, such domains form a semilattice with zero ($\boldsymbol{N}$ itself) under containment.

Two words can have different, yet equivalent, semantics. For instance, $\emptyset$ has the identity on all natural numbers, whereas for $\langle +1, \langle -1, \emptyset \rangle \rangle$ by equation (1) the identity is on the positive ones only. Clearly, this equivalence is defined by the relation $\asymp$ on $\boldsymbol{T}$ such that $v \asymp w$ iff $s_v(i) = s_w(i)$ for all $i \in \Delta_v \cap \Delta_w$, which forgets the above finite domain differences.

Trivially, we also got a congruence of the term algebra. Then, its congruence classes are made of binary words, not of the pairs $\langle n, m \rangle$ for $\equiv$ in 0.1. Yet, the two equivalences easily identify each other. The canonical representatives of the latter classes, pairs as $\langle 0, m \rangle$ or $\langle n, 0 \rangle$, correspond to similar canonical representatives: all words on a single "symbol".

Canonical representatives are the only interesting ones for our programmer. Among several other properties, they have maximal induced domains in each congruence class, i.e. the maximal addressing ability.

Such a semantic construction of integers works as the one of algebraic extension, yet it did not use sums. Moreover, from it we can easily *prove,* not just guess, the freedom of the quotient algebra, which *comes from* the numbers under extension, as well as the two inversion equations of 0.1. This now does not matter, as usual integers are well-known. It will do with our new integers.

0.3. Universal matrices

Seemingly, the semantic construction has a drawback: the quotient of the term algebra does not provide the new numbers with their (new) "sum", contrary to what the algebraic extension did through the old sum. Actually, this pertains to the new corresponding arithmetic, a development beyond the present step, yet the existence of sum(s) will still come from freedom.

In the unary case, integer sum arithmetic provides direct access memories with the most of their gains. It allows our programmer to compute an address *before using it.* Without it, he could merely access fixed locations bypassing other locations, whereas address computations allow him memory relocations, addressing in many-dimensional arrays, hashing and so on.

However, once a semantic construction has reached a free algebra, another unconventional tool, the universal matrices [15]–[21], always assure that a new arithmetic exists with at least some minimal features. This tool stems from the Geometry of the XIXth century closely following the ideas in [30], namely it markedly diverges from the abstract Universal Algebra of the past century.

Universal ("square") matrices are to *any universal* based algebra what usual square matrices are to a finitely based vector space: there always is a new binary associative operation (the "matrix product") on them, together with properties relevant to the original algebra, and an heterogeneous one (the "Menger system" corresponding to the product of a vector times a matrix). In the case of a one-dimensioned vector space, the two operations are almost the same: the product of the underlying field.

For unary integers, the term algebra only has a singleton base and again both operations essentially coincide. We get the integer sum, while its algebra relevant properties are the ones of integer arithmetic.

In case of two-ary integers, on the contrary, freedom refers to the equational class of Jónsson–Tarski in [8]. Its uncommon dimensionality features (bases of any positive cardinality) will provide them both an associative sum that extends word catenation and infinite matrix products of higher dimensions.

Then, our quest for an extended generalization of integers also got an extension of words. This might hint us that insisting on universal matrices could reach further extensions. (Work in progress concerns the prefix codes.)

0.4. Set-interpretative LISP semantics

While natural numbers can be addresses for actual memories, binary trees are the basic structure for the virtual memories of some high level programming languages, the earliest of which is LISP. Then two-ary integers will serve to "jump" from one such tree to another or from one their forest to another in the same way a usual integer does from a standard memory location to another.

Actually, such jumps will appear as wide memory reorganizations. To help programmers to grasp them, we borrowed from LISP some basic terminology. Yet, their corresponding notions are set-theoretical.

Our semantic construction will also provide a small part of LISP with a semantics, oriented more to its set-theoretical interpretation than to its set-theoretical formalization. This "set-interpretative" semantics does use set-theoretical formalism, like the set-theoretical semantics of (a wider part of) LISP in [11], yet it does not formalize how an actual computer has to behave under LISP instructions. It merely tries to interpret the set-theoretical meaning of a few LISP structures and instructions.

The recalled set-theoretical semantics succeeded in providing present day LISP with predictability results for computations on present day computers. Our set-interpretative one, on the contrary, hints at possible small extensions of LISP and of computer architecture: a new data type, the “jump”, and a new arithmetical register, the “two-ary” adder. Jumps are canonical representatives of two-ary integers. Two-ary adders should perform “sums” on them to enable index registers to directly “access” (reorganize) a tree structured memory.

Anyway, our set-theoretical notions merely formalize intensional ideas: they ideally come from Combinatory Logic as LISP did. In 1.9 we will hint a possible further connection with Combinatory Logic that also concerns Algorithmic Information Theory.

0.5. Notation

We give up any efficient functional notation for the one of Calculus, where repeated functional applications alternate subscripting and right parenthesizing. In spite of this choice of conventional notation, the foundation chosen here is the *pure* set–theoretical one not the conventional algebraic one. Some flaws of the latter (see 0.6 in [20]) concern our treatment.

Hence, we conform to [13], but for the following few differences. We denote the set-theoretical pair $\{\{a\},\{a,b\}\}$ by $\langle a,b\rangle$, yet we still simplify $f(\langle a,b\rangle)$ into $f(a,b)$ and $\langle\langle x,y\rangle,z\rangle$ into (x,y,x) as in [13]. For instance, we write the basic property of pairs of **I**.1.33 ibid. as

$$\langle t',t''\rangle = \langle v',v''\rangle \text{ iff } t'=v' \text{ and } t''=v'' \tag{2}$$

and from the regularity axiom of Set Theory in **I**.1.18 of [13] we get

$$a,b \neq \langle a,b\rangle\ . \tag{3}$$

$\mathsf{P}X$ denotes the set of subsets of set X and $\boldsymbol{i}_X$ its identity function.

We also cannot follow [13] as far as functional composition is concerned, because of the dangers shown in [18]. We merely consider it as the restriction of relational composition, here denoted by $\cdot$, namely $f\cdot g$ is “the composition of g and f” and $(f\cdot g)(x) = f(g(x))$. Accordingly, we perform the restriction of a function f to some set S merely by functional composition: $f\cdot\boldsymbol{i}_S$.

As usual, we write $f\colon A \to B$ to say that f is a function with arguments in the whole set A and values in B, $f\colon A \rightarrowtail B$ or $f\colon A \twoheadrightarrow B$ to say that it also is one to one or onto B and $f\colon A \rightarrowtail\!\!\!\twoheadrightarrow B$ to say it is a bijection onto B. Yet, as motivated in 0.6 of [20], we will forget that "function – domain" and "family – index" are pairwise synonymic and we avoid the notation $\{a_i\}_{i\in I}$ or $(a_i \mid i \in I)$. Within informal comments we will replace "function" with "labelling", to emphasize arguments, and with "indexing", to emphasize values. Also, we denote the set-theoretical power ${}^{A}B = \{f \mid f : A \to B\}$ as the arithmetic one B^A. (The latter will not occur here.)

1. S-terms and search terms

1.0. Definitions

Given any set U, consider the class of the sets T' containing it and closed under pairing. It has a smallest set, $T = \bigcap\{T' \mid U \subseteq T' \text{ and } T' \times T' \subseteq T'\}$.

In fact, this intersection is one of such sets T',

$$U \subseteq T \qquad \text{and} \tag{4}$$

$$T \times T \subseteq T \ , \tag{5}$$

as one could easily check by considering any $u \in U$ and any $\langle t', t''\rangle \in T \times T$. Often, we deal with subsets of T and, to get $T' = T$, the trivial consequence

$$T' \supseteq T \tag{6}$$

can replace the full induction principle that we will state in 1.3 (A).

Then, let $D = T \times T$, which by equation (5) is a subset of T. We easily get

$$U \cup D = T \ . \tag{7}$$

In fact, $U \cup D \subseteq T$ follows from $U, D \subseteq T$, while $U \cup D \supseteq T$ from $U \cup D$ being such a superset T', because the closure under pairing follows from equations (4) and (5) by the distributivity of $\times$ with respect to $\cup$ (see **I**.3.13 (e) in [13]) through the preservations of inclusions by cartesian products.

If

$$U \cap D = \emptyset \ , \tag{8}$$

then we say that U is a set of *unknowns for pairing* and that T is the set of *two-ary natural numbers on U.* We will denote T by $\boldsymbol{N}_2(U)$ to specify its unknowns.

For example, $\boldsymbol{N}_2(1)$ is the set of two-ary natural numbers on the single unknown $\emptyset$, because $U = \{\emptyset\} = 1$ satisfies equation (8), since a pair cannot be empty whereas our unknown $u = \emptyset$ is.

One might well think of two-ary natural numbers as terms of a species with only one binary operation symbol. Since it is the only operation symbol, we do not need to specify it.

We will use several sets U. We choose one of them as a "main" U, we call *atoms* its elements and *S-terms on U* the corresponding two-ary natural numbers. Hereinafter, we reserve notation T for their set, unless otherwise stated, whereas for any other U' we use $\boldsymbol{N}_2(U')$. We call *dyads* the pairs forming D.

The "S" in "S-terms" recalls the "S-expressions" of LISP [29]. Actually, the letter "**D**" and the word "dyads" are a notational abuse with respect to Combinatory Logic [7], where such pairs correspond to the terms composed by combinatory application, whereas "**D**" and "dyads" denote the combinators modelling set-theoretical pairs.

LISP assumes an enumerable infinite set U of atoms: variable-length words. For now, we do not.
We only assume $U \neq \emptyset$, because in the opposite case $T = \emptyset$ and all the following theory becomes very trivial.

Under this assumption, T cannot be finite: its proper subset $D = T \times T$ should have more elements than T. Hence, hereinafter $U \neq \emptyset$ and by equations (8) and (7) U and D define a bipartition of T.

We will also use some subsets of T. For every natural number n we define the set of *full terms of height n*, denoted by $\lfloor n \rfloor$, by arithmetic induction as

$$\lfloor 0 \rfloor \ = U \ \text{ and} \tag{9}$$

$$\lfloor m+1 \rfloor \ = \ \lfloor m \rfloor \times \lfloor m \rfloor \ . \tag{10}$$

(It is easy to show that $\lfloor n \rfloor \subset T$.) Among them we also consider the ones built up from a single atom $u \in U \neq \emptyset$. We define their set $\lfloor n \rfloor_u \subseteq \lfloor n \rfloor \subseteq T$, called the set of the *full u-terms (of height n),* by replacing equation (9):

(11) $$\lfloor 0 \rfloor_u \;=\; \{u\} \quad \text{and}$$

(12) $$\lfloor m+1 \rfloor_u \;=\; \lfloor m \rfloor_u \times \lfloor m \rfloor_u \;.$$

Clearly, every $\lfloor n \rfloor_u$ is singleton.

1.1. Definitions

Now, let us idealize the cons, car and cdr commands of LISP by respectively defining the functions *dyad of, left of* and *right of*

(13) $$\boldsymbol{d}\colon T \times T \mapsto\!\!\!\twoheadrightarrow\!\!\!\!\rightarrowtail D \;, \quad \boldsymbol{l}, \boldsymbol{r}\colon D \twoheadrightarrow\!\!\!\!\rightarrowtail T \;, \text{ i.e. } \boldsymbol{d}, \boldsymbol{l}, \boldsymbol{r}\colon T \times T \to T$$

by

(14) $$\boldsymbol{d}(t', t'') \;=\; \langle t', t'' \rangle \;,$$

(15) $$\boldsymbol{l}(t', t'') \;=\; t' \qquad \text{and}$$

(16) $$\boldsymbol{r}(t', t'') \;=\; t'' \qquad \text{for all} \quad t', t'' \in T \;.$$

Then, while $\boldsymbol{l}$ and $\boldsymbol{r}$ are nontrivial functions (the ones that extract the components of a pair), $\boldsymbol{d}$ merely is the identity on D. Of course, this is not the way any LISP transducer (either compiler or interpreter) behaves, since among other things it has to exploit memory efficiently: the car and cdr are reading instructions on already filled cells, whereas cons, a write instruction, fills a new cell when needed. This allows a LISP program to get any S-expression without wasting memory by a runtime dynamic allocation of cells.

However, if, for a while, one assumes to have a "large" ROM memory, filled with all written cells one will use, then one could implement cons too as a read instruction merely by (the computation on pointers corresponding to) our identity, seen as a binary operation. One might think to *ideally* replace the actual runtime dynamic allocation of cells with a static "expensive" one.

We also do not need to distinguish between a command identifier and what the command does, as 1.3 (C) and 1.5 will show, and, for the purposes of this work, one might well call the three functions $\boldsymbol{d}, \boldsymbol{l}$ and $\boldsymbol{r}$ "search commands". Yet, they are dependent on our U. For later purposes, independent search commands will be convenient. Then, we refer to the set of atoms $U = 1$ and we denote the three corresponding functions by $\mathbf{d}$, $\mathbf{l}$ and $\mathbf{r}$.

Therefore, we call the three functions $\mathbf{d}, \mathbf{l}, \mathbf{r} \colon \boldsymbol{N}_2(1) \times \boldsymbol{N}_2(1) \to \boldsymbol{N}_2(1)$ *search commands.* Anyway we boldfaced them, as the "operation symbols" of [6] in order to help algebraic readers, unfamiliar with 1.5.

The (last two) search commands define the set $\boldsymbol{T}$ of the *(unary) search terms* by

$$\{\emptyset\} \subseteq \boldsymbol{T} \quad \text{and} \tag{17}$$

$$(\boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}) \times \boldsymbol{T} \subseteq \boldsymbol{T} , \tag{18}$$

namely, $\boldsymbol{T}$ is the minimal set (as T was in 1.0) satisfying them or satisfying the following ones:

$$\emptyset \in \boldsymbol{T} , \tag{19}$$

$$\boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T} \quad \text{imply} \quad \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \in \boldsymbol{T} \quad \text{and} \tag{20}$$

$$\boldsymbol{t} \in \boldsymbol{T} \quad \text{implies} \quad \langle \mathbf{l}, \boldsymbol{t} \rangle, \langle \mathbf{r}, \boldsymbol{t} \rangle \in \boldsymbol{T} . \tag{21}$$

We call "terms" such objects because of the motivations in 1.3 (C), 1.4 and 1.5, where we show that they behave as the terms for a species with one (omitted) binary symbol and two unary ones. There, we also show that the set of "search unknowns" is the singleton $1 = \{\emptyset\}$ as for $\boldsymbol{N}_2(1)$.

Hence, here one might call "unary" such terms. The proof that larger arities are not necessary pertains to (later) arithmetical developments. In the same way we got equation (7), we still have

$$\{\emptyset\} \cup (\boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}) \times \boldsymbol{T} = \boldsymbol{T} . \tag{22}$$

Also, notice that

$$\mathbf{l} \neq \mathbf{r} \quad \text{and } \mathbf{l}, \mathbf{r} \notin \boldsymbol{T} . \tag{23}$$

In fact, by equation (3) $\mathbf{l}(\emptyset, \langle \emptyset, \emptyset \rangle) = \emptyset \neq \langle \emptyset, \emptyset \rangle = \mathbf{r}(\emptyset, \langle \emptyset, \emptyset \rangle)$ and the functions $\mathbf{l}$ and $\mathbf{r}$ are infinite sets as their common domain $\boldsymbol{N}_2(1) \times \boldsymbol{N}_2(1)$ is, whereas by equation (22) every $\boldsymbol{t} \in \boldsymbol{T}$ is finite. (The same holds for $\boldsymbol{l}$ and $\boldsymbol{r}$, since $U \neq \emptyset$.)

We also define two subsets of search terms. The set $\boldsymbol{N}_2(1) \subseteq \boldsymbol{T}$ of the *upward paths* is the one of the search terms we get by (19) and (20) only and the set $W \subseteq \boldsymbol{T}$ of the *downward paths* or *(search) words* by (19) and (21) only. To define both subsets we used the usual minimality assumptions, which also imply such inclusions.

Hereinafter, "word" will only denote such binary words, unless otherwise stated. As usual, the *length of* a word is the number of the "induction steps generating it". (We are omitting their formal definition that here would occur only after 1.3 (D).)

Given a word, we consider its letters *construction–ordered,* not ordered by reading them, e.g. in $(\mathbf{l}, \mathbf{r}, \emptyset) = \langle \mathbf{l}, \langle \mathbf{r}, \emptyset\rangle\rangle$ its first letter is $\mathbf{r}$, while $\mathbf{l}$ is the last. In our notation we will also keep the generator $\emptyset$, though necessary only with the empty word. When words occur in search terms, it is a useful landmark that improves reading.

Then, we define another identity function

$$\tau\colon (\boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}) \times \boldsymbol{T} \to \boldsymbol{T} \quad \text{by} \quad \tau(\boldsymbol{x}, \boldsymbol{t}) = \langle \boldsymbol{x}, \boldsymbol{t}\rangle \ , \tag{24}$$

for all $\boldsymbol{x} \in \boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}$ and $\boldsymbol{t} \in \boldsymbol{T}$. As we will show in the proof of 1.3 (C), one might well think of it as the definition of three operations on $\boldsymbol{T}$, called *τ-operations,*

$$\tau'_{\mathbf{d}}\colon \boldsymbol{T} \times \boldsymbol{T} \to \boldsymbol{T}, \qquad \tau'_{\mathbf{l}}, \tau'_{\mathbf{r}}\colon \boldsymbol{T} \to \boldsymbol{T} \tag{25}$$

by

$$\tau'_{\mathbf{d}}(\boldsymbol{t}', \boldsymbol{t}'') = \langle \boldsymbol{t}', \boldsymbol{t}''\rangle \ , \ \ \tau'_{\mathbf{l}}(\boldsymbol{t}) = \langle \mathbf{l}, \boldsymbol{t}\rangle \ \text{ and } \ \tau'_{\mathbf{r}}(\boldsymbol{t}) = \langle \mathbf{r}, \boldsymbol{t}\rangle \ . \tag{26}$$

for all $\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$.

1.2. Examples

The formal notation for search terms allows us to see their structure as long as they are small: $\emptyset$, $\langle\emptyset, \emptyset\rangle$, $\langle \mathbf{l}, \emptyset\rangle$, $\langle \mathbf{r}, \emptyset\rangle$. Bigger terms, as $\langle(\mathbf{l}, \mathbf{l}, \emptyset), \langle \mathbf{r}, \emptyset\rangle\rangle$ and $\langle(\mathbf{l}, \mathbf{l}, \emptyset), \langle(\mathbf{l}, \mathbf{r}, \emptyset), (\mathbf{r}, \mathbf{r}, \emptyset)\rangle\rangle$, needs some graphical support, as in Figure 1 and Figure 2 respectively. There, we adapted the standard diagrams for algebraic terms (vertex-labelled trees). For later purposes, more proper diagrams are the directed acyclic graphs we get from them by collapsing all identical subterms, e.g. Figure 2 will become Figure 3.

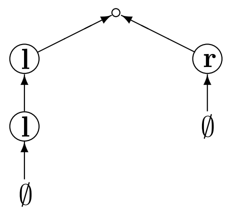

Figure 1

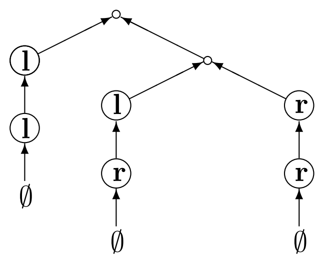

Figure 2

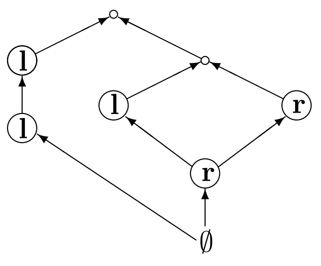

Figure 3

1.3. Lemmata

(A) *The sets U and T and the binary operation $\boldsymbol{d}$ satisfy the three Peano properties*

$$\boldsymbol{d}(t',t'') \notin U \quad \textit{for all} \quad t',t'' \in T \ , \tag{27}$$

$$\boldsymbol{d}\colon T \times T \mathrel{\rightarrowtail\!\!\twoheadrightarrow} D \ , \quad \textit{i.e.} \ \ \boldsymbol{d}^{-1}\colon T \setminus U \mathrel{\rightarrowtail\!\!\twoheadrightarrow} T \times T \ , \tag{28}$$

and, for all sets T',

$$T' \supseteq T \ \textit{iff} \ \begin{cases} U \subseteq T' & \textit{and,} \quad \textit{for all} \quad t',t'' \in T \ , \\ t',t'' \in T' & \textit{imply} \quad \boldsymbol{d}(t',t'') \in T' \ . \end{cases} \tag{29}$$

(B) *The left and right functions provide the dyad function with a partial "binary" inverse*

$$\boldsymbol{d}(\boldsymbol{l}(t),\boldsymbol{r}(t)) = t \qquad \textit{for all} \ \ t \in D \ , \tag{30}$$

$$\boldsymbol{l}(\boldsymbol{d}(t',t'')) = t' \qquad \textit{and} \tag{31}$$

$$\boldsymbol{r}(\boldsymbol{d}(t',t'')) = t'' \quad \textit{for all} \ \ t',t'' \in T \ . \tag{32}$$

(C) *The sets $1 = \{\emptyset\}$ and $\boldsymbol{T}$ and the function τ satisfy the Peano properties*

(33) $$\tau^{-1}: \boldsymbol{T} \setminus \{\emptyset\} \rightarrowtail\!\!\twoheadrightarrow (\boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}) \times \boldsymbol{T},\ \textit{hence}$$

(34) $$\tau(\boldsymbol{x}, \boldsymbol{t}) \neq \emptyset\ ,\textit{for all}\ \ \boldsymbol{x} \in \boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}\textit{and}\ \ \boldsymbol{t} \in \boldsymbol{T},$$

and, for all sets T',

(35) $$T' \supseteq \boldsymbol{T}\ \textit{iff}\ \begin{cases} 1 \subseteq T' & \textit{and, for all}\ \ \boldsymbol{x}\ \textit{and}\ \ \textit{all}\ \ \boldsymbol{t} \in \boldsymbol{T}, \\ \boldsymbol{x} \in (\boldsymbol{T} \cap T') \cup \{\mathbf{l}, \mathbf{r}\}\ \textit{and}\ \ \boldsymbol{t} \in T' & \textit{imply}\ \ \tau(\boldsymbol{x}, \boldsymbol{t}) \in T' \end{cases}$$

(that one could well restate by means of the operations $\tau'_{\mathbf{d}}$, $\tau'_{\mathbf{l}}$ and $\tau'_{\mathbf{r}}$).

(D) *For all sets T',*

$$T' \supseteq W\ \textit{iff}\ \begin{cases} 1 \subseteq T' & \textit{and, for all}\ \ \boldsymbol{t} \in W\ , \\ \boldsymbol{t} \in T' & \textit{implies}\ \ \langle \mathbf{l}, \boldsymbol{t} \rangle, \langle \mathbf{r}, \boldsymbol{t} \rangle \in T'\ . \end{cases}$$

Proofs.

(A) Condition (27) is a restatement of (8). As motivated in 1.4 (A), it corresponds to Peano's first axiom. Condition (28), comes from defining $\boldsymbol{d}$ as an identity on set-theoretical pairs. It corresponds to Peano's axioms about successor and uniqueness of predecessor, since we can rewrite it as $\boldsymbol{d}: T \times T \rightarrowtail T$.

The last condition corresponds to Peano's induction axiom. Its (only if) part easily follows from (4) and (5) by the transitivity of $\subseteq$. To prove its (if) consider $T'' = T \cap T'$. Our premises get $T'' \supseteq T$ as it was for T' in (6). Hence, $T' \supseteq T$.

(B) To prove (30), consider that by (28) for every $t \in D$ there are $t', t'' \in T$ such that $\langle t', t'' \rangle = t$. Then, $\boldsymbol{d}(\boldsymbol{l}(t), \boldsymbol{r}(t)) = \boldsymbol{d}(\boldsymbol{l}(\langle t', t'' \rangle), \boldsymbol{r}(\langle t', t'' \rangle)) = \boldsymbol{d}(t', t'') = \langle t', t'' \rangle = t$ by (14)–(16). Such definitions also prove (31), $\boldsymbol{l}(\boldsymbol{d}(t', t'')) = \boldsymbol{l}(t', t'') = t'$, as well as (32) (after replacing $\boldsymbol{r}$ for $\boldsymbol{l}$).

(C) At now, contrary to (A), we do not need a disjunctive assumption, as (8), to prove (34). In fact, $\emptyset$ can never be any of such terms, which are composed by pairs, because pairs never are empty. See **I**.1.31 in [13] and 3.1 in [16].

The identity argument, used in (A) for (28), still holds for (33). However, now (33) enables any nonempty term $\boldsymbol{t}$ to have a single decomposition within three sorts ($\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle, \langle \mathbf{l}, \boldsymbol{t}' \rangle, \langle \mathbf{r}, \boldsymbol{t}' \rangle$) that are disjoint because of (23). This allows us to see τ as the τ' of (26). Yet, we cannot think to always replace τ or τ' by an "absolutely free algebra" for one binary and two unary operations, because of an inconsistency we are going to show in 1.5.

Finally, one could prove the induction principle in (35) in the same way we did in (A).

(D) Same proof as in (C), but for the case $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ that drops out. ■

1.4. Inductions

(A) When we replace a (properly chosen) unary operator, say $\mathcal{S}$, for our (binary) dyad operator in (5) and a proper singleton set for U, Definition 1.0 becomes the one of a model of natural numbers, where (8) or (27) corresponds to the first Peano's axiom, while (28) to the second. Then, the proper choice of $\mathcal{S}$ will provide Peano's induction axiom with a proof within the model (e.g. see 2.4.1 in [3]). Hence, the properties in 1.3 (A) are to S-terms what Peano's axioms are to natural numbers.

We can restate this correspondence for 1.3 (C), but for two details. The former is the occurrence of two other (unary) "successors", $\tau'_{\mathbf{l}}$ and $\tau'_{\mathbf{r}}$, in addition to the binary one $\tau'_{\mathbf{d}}$ replacing the $\boldsymbol{d}$ of 1.3 (A). The latter is that we replace the given set 1, as the recalled model did, for the undeterminate one U. This allowed us to omit an assumption as (8). In fact, $\{\emptyset\} \cap (\boldsymbol{T} \cup \{\mathbf{l}, \mathbf{r}\}) \times \boldsymbol{T} = \emptyset$, because pairs never are empty.

In 1.3 (D) things are as in 1.3 (C). There, we omitted the statements corresponding to the initial Peano's axioms, because they were trivial cases (restrictions) of the ones in 1.3 (C). Yet, contrary to a practice widespread in abstract Algebra, all such statements, as well as (28), are mathematically necessary as the counterexample in 1.5 will show.

(B) We will call the Peano induction principle in (29) *T-induction,* or in case of upward paths *upward induction,* the one in (35) *$\boldsymbol{T}$-induction* or *bold induction,* and the one in 1.3 (D) for words *word induction.* We will introduce another induction for a third subset of $\boldsymbol{T}$ in 4.0 and 4.1 (E). Seldom, we will also exploit Peano's induction, which we call *arithmetic induction.*

We will use the above Peano induction properties, as well as the ones we will introduce, both for proving statements and for defining functions (or predicates). In the former case the sets T' of 1.3 are the ones where a statement holds. In the latter, they are the "sets where a function is being defined" (not the domains of a defined function). They are not set-theoretical sets, since our Set Theory does not define "to define".

Therefore, in the latter case one should prove the existence of the function to be defined. One could do it by proving corollaries of the general recursion principle in **2**.13.1 of [13]. Such proofs contain several uninteresting details (but one) and we will follow the algebraic practice of taking "algebraic recursion" as granted.

Yet, we cannot follow Algebra in calling our inductions "recursions". In fact, they will correspond to the arithmetical case of "induction", which is intermediate between primitive recursion and iteration, where the function defining the *induction* step is constant with respect to the induction variable.

We can expect that in two-ary arithmetics the cases corresponding to primitive recursion and recursion will occur. Hence, to keep the word "induction" as in (unary) Arithmetic serves to save "recursion" for future proper uses. Here, we will not state any general recursion principle for any of our terms. The only recursion, we will use, concerns words, yet it will come from word induction through word reversals, without such a general principle.

Furthermore, we cannot disregard one of the details of the corollaries we are omitting, as Algebra does. It concerns a well-founded relation occurring in the above mentioned recursion principle. To get this condition of being well-founded we stated the firsts of Peano axioms.

Without them one cannot define predicates nor functions on the terms concerned, for the very reason one cannot do on natural numbers: without such Peano's axioms the base part of a definition contradicts its induction part, unless the defined object is a constant.

1.5. Counterexample

We can see how the above contradiction rises, when we deny (8), i.e. within two-ary natural numbers. Assume we define the *depth* $d(t)$ of a term t by (29) as

$$\begin{aligned} d(u) &= 0\ , \text{ for all } \ u \in U\ , \\ d(t', t'') &= 1 + \max(d(t'), d(t''))\ , \text{ for all } \ t = \langle t', t'' \rangle \in D\ . \end{aligned}$$

Let $u \in U$. Without (8), U can well contain $\langle u, u \rangle$. In such a case $d(u,u) = 0$ by the former step as well as $d(u,u) = 1$ by the latter. Hence, the *function* d was misdefined.

If we keep (4), this misdefinition has to occur even if we change the construction in the induction step (5): just replace $\langle u, u\rangle$ by the new corresponding object. We have to stress this, because sometimes people in Algebra and Logic (but for [7]) try to weaken (or omit) (8) through such changes. For instance, a proper choice of operation symbols should allow any set to play the rôle of a set of "variables" and get a "symbolic term" functor.

It does not matter whether the construction in an induction step uses "good operation symbols" (nor whether one fails to specify such a construction, as when one states that a composed term is a word, without defining words). If the construction is not trivial, it will anyway give us new terms other than the generators and any such term is a possible extra element for another "inconsistent" set of generators, as $\langle u, u\rangle$ was.

Algebraically, such inconsistency is the lack of independence within the algebra of terms, while set-theoretically it prevents the recursion principle mentioned in (B) to have its well-founded relation. Disjunction assumptions, like (8), cannot become weaker.

Hence, there are not functions sending an arbitrary set of generators into a nontrivial set of terms on such generators: "symbolic term" functors *cannot exist.* A formal proof of this is in Lemma 7.3 (A) and (B) of [14].

On the contrary, there are sets that are made only of unknowns for whatever algebraic species we choose, as shown in 3.1 in [16]: while one cannot freely choose generators, then we always can freely choose operation symbols, contrary to some opposite belief.

The only hope to use an arbitrary U as a set of algebraic "variables" might be to give up the inclusion assumptions as (4). For instance [9] claims to be able to do so by replacing the injection $\boldsymbol{i}_U : U \rightarrowtail T$, as in (4), by another (burdensome) injection $j : U \rightarrowtail T$. Yet, it does use (4) wherever.

Anyway, whatever j we choose to replace $\boldsymbol{i}_U$, this is tantamount to replace $U \subseteq T$ by $V \subseteq T$, where V is the j-image of U. Then, the set of generators is V that is not arbitrary. The only way to use arbitrary sets U is not to use their elements as *generators*, contrary to what we want. In *any free* algebra one cannot choose a base at will.

Within Set Theory we always have to (carefully) define unknowns for any nonempty species, whereas operation symbols (including the nullary ones) can be freely chosen depending on the application. As the latter freedom is consistent, in 1.1 we are exploiting it for the unary "symbols" and — in a sense — for the omitted binary one.

1.6. Definitions

Given two search terms $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{T}$, define $\boldsymbol{u} \oplus \boldsymbol{v}$, the *catenation of* $\boldsymbol{v}$ *with* $\boldsymbol{u}$*,* by bold induction on $\boldsymbol{v}$, as

$$\boldsymbol{u} \oplus \emptyset \; = \; \boldsymbol{u}, \tag{36}$$

$$\boldsymbol{u} \oplus \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \; = \; \langle \boldsymbol{u} \oplus \boldsymbol{t}', \boldsymbol{u} \oplus \boldsymbol{t}'' \rangle \ , \tag{37}$$

$$\boldsymbol{u} \oplus \langle \mathbf{l}, \boldsymbol{t} \rangle' \; = \; \langle \mathbf{l}, \boldsymbol{u} \oplus \boldsymbol{t}' \rangle \quad \text{and} \quad \boldsymbol{u} \oplus \langle \mathbf{r}, \boldsymbol{t}' \rangle = \langle \mathbf{r}, \boldsymbol{u} \oplus \boldsymbol{t}' \rangle \ , \tag{38}$$

for all $\boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$. Hence, we defined a binary operation $\oplus\colon \boldsymbol{T} \times \boldsymbol{T} \to \boldsymbol{T}$. As 1.7 (A) will show $\oplus$ and $\emptyset$ define a monoid on search terms.

Later on, we also will use the *(reversed) catenation monoid* that we define by commuting $\oplus$, namely there the monoid composition of $\boldsymbol{v}$ with $\boldsymbol{u}$ is $\boldsymbol{v} \oplus \boldsymbol{u}$.

Catenation allows us to define the *reversed* $\overleftrightarrow{w}$ *of* every word $w \in W$ by word induction, $\overleftrightarrow{\emptyset} = \emptyset$, $\overleftrightarrow{\langle \mathbf{l}, v \rangle} = \langle \mathbf{l}, \emptyset \rangle \oplus \overleftrightarrow{v}$ and $\overleftrightarrow{\langle \mathbf{r}, v \rangle} = \langle \mathbf{r}, \emptyset \rangle \oplus \overleftrightarrow{v}$. We omit the proof that reversal is a permutation, $\hookleftarrow\colon W \rightarrowtail\!\!\!\twoheadrightarrow W$. Yet, we will use this when, while defining or proving something about any $\overleftrightarrow{w}$ inductively, we will say that we do about w by *backward induction* on w.

By catenation we also define a *search increment function* $\mathring{\eta}\colon \boldsymbol{T} \to \boldsymbol{T}^{\boldsymbol{T}}$ by

$$\mathring{\eta}_{\boldsymbol{u}} \, (\boldsymbol{t}) = \boldsymbol{u} \oplus \boldsymbol{t} \ , \ \text{for all} \ \ \boldsymbol{t}, \boldsymbol{u} \in \boldsymbol{T} \ . \tag{39}$$

Here and whenever the letter η will occur, it denotes a function that extends its argument to some endomorphism. For the present occurrence, this will be shown in 1.7 (A).

On S-terms and two-ary natural numbers, on the contrary, we define a *heterogeneous* operation $\hat{+}\colon T^U \times T \to T$, we call *upward addition,* by T-induction on the latter argument: for all $M\colon U \to T$, $u \in U$ and $\langle t', t'' \rangle \in D$,

$$M \,\hat{+}\, u \; = \; M(u), \tag{40}$$

$$M \,\hat{+}\, \langle t', t'' \rangle \; = \; \langle M \,\hat{+}\, t', M \,\hat{+}\, t'' \rangle \ . \tag{41}$$

By it we also define the (bottom) *term increment function* $\hat{\eta}\colon T^U \to T^T$ by

$$\hat{\eta}_M(t) = M \hat{+} t\ , \quad \text{for all}\ \ M\colon U \to T\ \text{and}\ \ t \in T\ , \tag{42}$$

where we call function M the *increment matrix,* and the *term matrix product* $\diamond\colon T^U \times T^U \to T^U$ by

$$(M \diamond L)(u) = M \hat{+} L(u)\ , \quad \text{for all}\ \ L, M\colon U \to T\ \text{and}\ \ u \in U\ . \tag{43}$$

Again, we call $M \diamond L$ the product *of L times M.* ([18] shows the necessity of such reversed readings.) As 1.7 (B) will show, $\diamond$ is the composition of a monoid with unit $\boldsymbol{i}_U$. Hence, from this product we can define a *term matrix power* by iteration:

$$M^{\langle 0\rangle} = \boldsymbol{i}_U \tag{44}$$

$$M^{\langle m+1\rangle} = M^{\langle m\rangle} \diamond M \quad \text{or} \quad M^{\langle m+1\rangle} = M \diamond M^{\langle m\rangle}\ . \tag{45}$$

We can see the tree of $\hat{\eta}_M(t)$ and $M \hat{+} t$ as the S-term tree we get from the one of t by replacing each leaf $u \in U$ in it with the tree of $M(u)$. When we have a matrix $L\colon U \to T$, we see an "indexed forest" of such trees t and the product of L and M corresponds to the indexed forest we get by sprouting each tree t of the former in the same way as above. Figure 4 shows such sprouts from circled nodes for $U = \{u, v\}$.

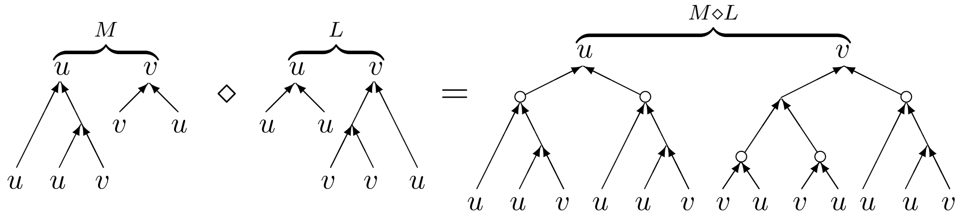

Figure 4

Given any natural number n, the product $\prod_{u\in U} \lfloor n \rfloor_u$ is singleton, since by (11) and (12) every factor is. We call its element $C_n\colon U \to T$ the *full matrix of height n.* Hence, by (11) and (12) respectively

$$C_0 = \boldsymbol{i}_U, \tag{46}$$

$$C_{n+1}(u) = \langle C_n(u), C_n(u)\rangle\ . \tag{47}$$

When $M = C_n$ is a full matrix of some positive height, $n > 0$, we call its increment $\hat{\eta}_M\colon T \to T$ a *full increment.*

A matrix $M\colon U \to T$ can have a fixed value $t = M(u)$, for all $u \in U$. We call it a *constant matrix* and we denote it as $M = \mathrm{K}_t$, where function K is a *generator of constants* defined by

(48) $$\mathrm{K}_t(u) = t \ \ ,$$

for all $t \in T$ and $u \in U$ and, clearly, $K\colon T \Vdash\!\!\to T^U$. Later on, we will use other generators of constants (where t and u belongs to other sets), yet we will not repeat (48).

Consider a function $h\colon \boldsymbol{T} \to \boldsymbol{T}$ such that, for all $\boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$,

(49) $$h(\boldsymbol{t}', \boldsymbol{t}'') \;=\; \langle h(\boldsymbol{t}'), h(\boldsymbol{t}'')\rangle \ ,$$

(50) $$h(\mathbf{l}, \boldsymbol{t}') = \langle \mathbf{l}, h(\boldsymbol{t}')\rangle \quad \text{and} \quad h(\mathbf{r}, \boldsymbol{t}') = \langle \mathbf{r}, h(\boldsymbol{t}')\rangle \ .$$

We call h a *search endomorphism* and denote the set of all such h by $\mathcal{E}_\tau \subseteq \boldsymbol{T}^{\boldsymbol{T}}$.

Clearly, $\mathcal{E}_\tau$ is the carrier of a monoid of functional composition ($\boldsymbol{i}_{\boldsymbol{T}} \in \mathcal{E}_\tau$ and $h'' \cdot h' \in \mathcal{E}_\tau$ for all $h', h'' \in \mathcal{E}_\tau$) that we call the *search endomorphism monoid.* Then, define the *search endomorphism representation* $\overset{\circ}{r}\colon \mathcal{E}_\tau \to \boldsymbol{T}$ by

(51) $$\overset{\circ}{r}(h) = h(\emptyset) \ , \text{for all} \ \ h \in \mathcal{E}_\tau \ .$$

Finally, in case of S-terms, we define the *term endomorphisms* as the functions $l\colon T \to T$ such that

(52) $$l(t', t'') = \langle l(t'), l(t'')\rangle \ , \text{for all} \ \ t', t'' \in T$$

and we denote their set by $\hat{\mathcal{E}}$. Again, we easily get a monoid of functional composition on it, which we call the *endomorphism monoid of S-terms.* Then, define the *term endomorphism representation* $\hat{r}\colon \hat{\mathcal{E}} \to T^U$ by $\hat{r}(l)_u = l(u)$, for all $l \in \hat{\mathcal{E}}$ and $u \in U$, namely by

(53) $$\hat{r}(l) = l \cdot \boldsymbol{i}_U \ , \text{for all} \ \ l \in \hat{\mathcal{E}} \ .$$

Clearly, there are functions $l\colon T \to T$ which are not term endomorphisms, i.e. not the endomorphism of $\boldsymbol{d}\colon T \times T \to T$. On the contrary, any $f\colon T \to T$ is an endomorphism of $\boldsymbol{l}, \boldsymbol{r}\colon T \times T \to T$, because by 1.1 for instance we have

$$\boldsymbol{l}(f(t'), f(t'')) = f(t') = f(\boldsymbol{l}(t', t'')) \text{ , for all } t', t'' \in T \ . \tag{54}$$

Then, $\hat{\mathcal{E}}$ also is the set of endomorphisms of all $\boldsymbol{d}, \boldsymbol{l}, \boldsymbol{r}: T \times T \to T$.

1.7. Theorems

(A) *The search increment function is the inverse of the search endomorphism representation and the image under such a representation of the composition of search endomorphisms is catenation. Hence,* $\overset{\circ}{r}: \mathcal{E}_{\mathcal{T}} \rightarrowtail\!\!\!\!\twoheadrightarrow \boldsymbol{T}$*,* $\overset{\circ}{\eta}: \boldsymbol{T} \rightarrowtail\!\!\!\!\twoheadrightarrow \mathcal{E}_{\mathcal{T}}$ *and for all* $\boldsymbol{t}, \boldsymbol{u} \in \boldsymbol{T}$

$$(\boldsymbol{v} \oplus \boldsymbol{u}) \oplus \boldsymbol{t} = \boldsymbol{v} \oplus (\boldsymbol{u} \oplus \boldsymbol{t}) \textit{ and} \tag{55}$$

$$\emptyset \oplus \boldsymbol{t} = \boldsymbol{t} \oplus \emptyset = \boldsymbol{t} \ . \tag{56}$$

(B) *The term increment function is the inverse of the term endomorphism representation and the image under such a representation of term endomorphism composition is term matrix product. Hence,* $\hat{r}: \hat{\mathcal{E}} \rightarrowtail\!\!\!\!\twoheadrightarrow T^U$*,* $\hat{\eta}: T^U \rightarrowtail\!\!\!\!\twoheadrightarrow \hat{\mathcal{E}}$ *and for all* $L, M: U \to T$ *and* $t \in T$

$$(M \diamond L) \mathbin{\hat{+}} t = M \mathbin{\hat{+}} (L \mathbin{\hat{+}} t) \textit{ and} \tag{57}$$

$$\boldsymbol{i}_U \mathbin{\hat{+}} t = t \ . \tag{58}$$

Proofs.

(A) At first, let us show that $\overset{\circ}{\eta} \cdot \overset{\circ}{r} = \boldsymbol{i}_{\mathcal{E}_{\mathcal{T}}}$. We prove $\overset{\circ}{\eta}_{\overset{\circ}{r}(h)}(\boldsymbol{t}) = h(\boldsymbol{t})$, i.e. we prove

$$h(\emptyset) \oplus \boldsymbol{t} = h(\boldsymbol{t}) \ , \tag{59}$$

for all $h \in \mathcal{E}_{\mathcal{T}}$ and all $\boldsymbol{t} \in \boldsymbol{T}$ by bold induction. The basis step, $h(\emptyset) \oplus \emptyset = h(\emptyset)$, immediately follows from (36).

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and $h(\emptyset) \oplus h(\boldsymbol{t}') = h(\boldsymbol{t}')$ and $h(\emptyset) \oplus \boldsymbol{t}'' = h(\boldsymbol{t}'')$, by (37) and (49) $h(\emptyset) \oplus \boldsymbol{t} = \langle h(\emptyset) \oplus \boldsymbol{t}', h(\emptyset) \oplus \boldsymbol{t}'' \rangle = \langle h(\boldsymbol{t}'), h(\boldsymbol{t}'') \rangle = h(\boldsymbol{t})$. When $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $h(\emptyset) \oplus h(\boldsymbol{t}') = h(\boldsymbol{t}')$, by (38) and (50) $h(\emptyset) \oplus \boldsymbol{t} = \langle \mathbf{l}, h(\emptyset) \oplus \boldsymbol{t}' \rangle = \langle \mathbf{l}, h(\boldsymbol{t}') \rangle = h(\boldsymbol{t})$. When $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$, merely replace $\mathbf{l}$ with $\mathbf{r}$.

Now, let us prove that $\overset{\circ}{r}$ is onto $\boldsymbol{T}$. The endomorphism conditions (49) and (50) allow (35) to define any $h \in \mathcal{E}_\tau$ on the whole $\boldsymbol{T}$ by bold induction, once we have only chosen a value for $h(\emptyset)$. Hence, given any $\boldsymbol{t} \in \boldsymbol{T}$, we get an $h \in \mathcal{E}_\tau$ such that $\overset{\circ}{r}(h) = h(\emptyset) = \boldsymbol{t}$ by merely choosing $h(\emptyset) = \boldsymbol{t}$.

Finally, we get $\overset{\circ}{r}(h'') \oplus \overset{\circ}{r}(h')$ as the representation $\overset{\circ}{r}(h'' \cdot h')$ of composition, for all $h'', h' \in \mathcal{E}_\tau$, merely by (59). In fact, by (51) this is to prove that $h''(\emptyset) \oplus h'(\emptyset) = h''(h'(\emptyset))$ and, if we set $\boldsymbol{t} = h'(\emptyset)$ and $h = h''$ in (59), we find that we already proved it by bold induction on $h'(\emptyset)$. Notice also that the consequence $\overset{\circ}{\eta} = \overset{\circ}{r}{}^{-1}$allows us to rewrite $\overset{\circ}{r}(h'') \oplus \overset{\circ}{r}(h') = \overset{\circ}{r}(h'' \cdot h')$ as $\overset{\circ}{\eta}_{\boldsymbol{v} \oplus \boldsymbol{u}} = \overset{\circ}{\eta}_{\boldsymbol{v}} \cdot \overset{\circ}{\eta}_{\boldsymbol{u}}$, where $\boldsymbol{u} = \overset{\circ}{r}(h')$ and $\boldsymbol{v} = \overset{\circ}{r}(h'')$. Hence, we get (55) by (39) after applying $\boldsymbol{t}$ to both sides, while (56) comes from $\overset{\circ}{\eta}_\emptyset = \boldsymbol{i}_{\boldsymbol{T}} \in \mathcal{E}_\tau$.

(B) Let us show that $\hat{\eta} \cdot \hat{r} = \boldsymbol{i}_{\hat{\mathcal{E}}}$.

We prove $\hat{\eta}_{\hat{r}(l)}(t) = l(t)$, i.e. we prove

$$\hat{r}(l) \,\hat{+}\, t = l(t) \; , \tag{60}$$

for all $l \in \hat{\mathcal{E}}$ and all $t \in T$ by T-induction. The basis step, $\hat{r}(l) \,\hat{+}\, u = \hat{r}(l)_u$, immediately follows from (40).

When $t = \langle t', t'' \rangle$ where $\hat{r}(l) \,\hat{+}\, t' = l(t')$ and $\hat{r}(l) \,\hat{+}\, t'' = l(t'')$, by (41) and (52) $\hat{r}(l) \,\hat{+}\, t = \langle \hat{r}(l) \,\hat{+}\, t', \hat{r}(l) \,\hat{+}\, t'' \rangle = \langle l(t'), l(t'') \rangle = l(t)$. Hence, we proved the induction step too.

Now, let us prove that $\hat{r}$ is onto T^U. The endomorphism axiom (52) allows (29) to define any $l \in \hat{\mathcal{E}}$ on the whole T by T-induction, once we have only chosen any value for $l \cdot \boldsymbol{i}_U \colon U \to T$. Hence, given any $t \colon U \to T$, we get an $l \in \hat{\mathcal{E}}$ such that $\hat{r}(l) = l \cdot \boldsymbol{i}_U = t$ by merely choosing $l \cdot \boldsymbol{i}_U = t$.

Finally, we get $\hat{r}(l'') \diamond \hat{r}(l')$ as the representation $\hat{r}(l'' \cdot l')$ of composition, for any $l', l'' \in \hat{\mathcal{E}}$, merely by (60). In fact, by (53) this is to prove that $(l'' \cdot \boldsymbol{i}_U) \diamond (l' \cdot \boldsymbol{i}_U) = l'' \cdot l' \cdot \boldsymbol{i}_U$ and, if we set $t = l'(u)$ and $l = l''$ in (60), we find that we already proved it by T-induction on $l'(u)$, namely $(l'' \cdot \boldsymbol{i}_U) \,\hat{+}\, l'(u) = l''(l'(u))$ for all $u \in U$. Notice also that we can rewrite $\hat{r}(l'') \diamond \hat{r}(l') = \hat{r}(l'' \cdot l')$ as $\hat{\eta}_{M \diamond L} = \hat{\eta}_M \cdot \hat{\eta}_L$, where $L = \hat{r}(l')$ and $M = \hat{r}(l'')$, since $\hat{\eta}$ is the inverse of $\hat{r}$. Hence, we get (57) by (42) after applying t to both sides and (58) comes from $\boldsymbol{i}_T \in \hat{\mathcal{E}}$. ■

1.8. Lemmata

(A) *The values of full increments are dyads:* $\hat{\eta}_M \colon T \to D$ *, for every* $M = C_n$ *with* $n > 0$.

(B) *The height of a full matrix is the exponent of the term matrix power of C_1 giving it: $C_n = C_1^{\langle n\rangle}$.*

(C) *Full increments are one to one.*

(D) *Full increments do not have fixed points.*

(E) *The restrictions of catenation to upward and downward paths are sub-operations: for all $\boldsymbol{u}, \boldsymbol{t} \in \boldsymbol{N}_2(1)$ and $\boldsymbol{v}, \boldsymbol{w} \in W$, $\boldsymbol{t} \oplus \boldsymbol{u} \in \boldsymbol{N}_2(1)$ and $\boldsymbol{w} \oplus \boldsymbol{v} \in W$, where $\boldsymbol{t} \oplus \boldsymbol{u} = (t \diamond u)(\emptyset)$, when $t, u\colon 1 \to \boldsymbol{N}_2(1)$, $t(\emptyset) = \boldsymbol{t}$ and $u(\emptyset) = \boldsymbol{u}$. Namely, the generator of constants $\mathrm{K}\colon T \twoheadrightarrowtail T^1$ is an isomorphism from the former restriction of $\oplus$ onto $\diamond\colon T^1 \times T^1 \to T^1$, the term matrix product for the singleton $U' = 1$.*

Proofs.

(A) Consider $C_n \hat{+} t$ for $n > 1$ and all $t \in T = U \cup D$. When $t = u \in U$, by (40) it is $C_n(u) \in D$ because of (47). When $t = \langle t', t''\rangle \in D$, by (41) it is $\langle C_n \hat{+} t', C_n \hat{+} t''\rangle \in D$.

(B) Arithmetic induction on n. The basis step is in (46) and (44). The induction step comes from (45), (43), (47), (46), (41), (40) and (47): $C_1^{\langle m+1\rangle}(u) = (C_m \diamond C_1)(u) = C_m \hat{+} C_1(u) = C_m \hat{+} \langle C_0(u), C_0(u)\rangle = C_m \hat{+} \langle u, u\rangle = \langle C_m \hat{+} u, C_m \hat{+} u\rangle = \langle C_m(u), C_m(u)\rangle = C_{m+1}(u)$ for all $u \in U$.

(C) By (B) and 1.7 (B) any full increment is a positive power of functional composition for the full increment of C_1. Hence, we can merely show that the latter increment is one to one. By T-induction we show that for all $t \in T$, if $C_1 \hat{+} t = C_1 \hat{+} d$ for some $d \in T$, then $t = d$.

When $t = u \in U$, $C_1 \hat{+} d = C_1 \hat{+} t = \langle u, u\rangle \in U \times U$ by (40), (47) and (46). This implies that $d \notin D$, since $d \in D$ by (41) contradicts (A). Then, by (8) $d \in U$ as t did and $C_1 \hat{+} d = \langle d, d\rangle$ that by (2) gets $t = d$. Notice that we get this also from the symmetric assumption $d \in U$.

When $t = \langle t', t''\rangle$ where both t' and t'' satisfy our implication, $d \notin U$ as $d \in U$ implies $t = d \in U$. Then, $d = \langle d', d''\rangle \in D$ and the implication premise gets $\langle C_1 \hat{+} t', C_1 \hat{+} t''\rangle = \langle C_1 \hat{+} d', C_1 \hat{+} d''\rangle$ by (41), i.e. by (2) $C_1 \hat{+} t' = C_1 \hat{+} d'$ and $C_1 \hat{+} t'' = C_1 \hat{+} d''$. Hence, $t = \langle t', t''\rangle = \langle d', d''\rangle = d$.

(D) We show that, when $n > 0$, $C_n \hat{+} t \neq t$ by T-induction on t.

When $t \in U$, $C_n \hat{+} t = C_n(t) \in D$ by (40) and (A), whereas $t \in U$ and $U \cap D = \emptyset$. When $t = \langle t', t''\rangle$ and both $C_n \hat{+} t' \neq t'$ and $C_n \hat{+} t'' \neq t''$, by (41) and (2) $C_n \hat{+} t = \langle C_n \hat{+} t', C_n \hat{+} t''\rangle \neq \langle t', t''\rangle = t$.

(E) The trivial either upward or word induction on $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively can exploit (36) and either (37) or (38). In the former case (37) becomes (41) through (43). ■

1.9. Combinatory translators

Readers acquainted with Combinatory Logic will likely find an immediate example of term increments in the substitutions of combinatory terms for variables. When U is the set of atoms for (a set-theoretical model of) combinatory terms with reducible constants $\mathbf{K}$ and $\mathbf{S}$ as in [7], this occurs with any increment matrix M that keeps such constants:

$$M(\mathbf{K}) = \mathbf{K} \text{ and } M(\mathbf{S}) = \mathbf{S}.$$

However, term increments make sense even without variables. For instance, when, $U = \{\mathbf{K}, \mathbf{S}\}$, namely for pure combinators, we can keep $\mathbf{K}$ only and replace $\mathbf{S}$ by $\mathbf{B}(\mathbf{BW})(\mathbf{BBC})$ in combinatory notation. Within our K–S system, we now see a translation of this system to the B–C–K–W system of chapter 5 of [4].

In fact, this increment sends every combinator into an extensionally equal combinator that only sprouts the combinators $\mathbf{B}$, $\mathbf{C}$, $\mathbf{K}$, and $\mathbf{W}$. If U' denotes the set of these four terms (trees) on U, the combinator we reach always belongs to $\boldsymbol{N}_2(U')$, where according to 1.5 U' is made of unknowns, as well as to $\boldsymbol{N}_2(U)$.

In general, U will contain more than two reducible constants and the matrix $M\colon U \to T$ more non trivial replacements. Then, $\hat{\eta}_M\colon T \to T$ will be a "translator" between combinatory systems, namely an idealized compiler or translator between ideal programming languages.

This opens the quest for possible "two-ary asymptotic" invariants, i.e. for properties that do not change under such translations. In fact, at a soft level, our translators perform what emulations between universal Turing machines do at the hard level.

(The formalization of G.J. Chaitin [1] implements such emulations by "unary increments", viz. by catenations with a word prefix that identifies the Turing machine under emulation. Contrary to other attempts, it succeeded in proving the Minsky–Solomonoff's conjecture [12] neatly: asymptotic invariance under such emulations defines *algorithmic complexity*, the quantity of "intrinsic information". Among many results, a major result of Logic of the past century [2] followed from this.)

2. Semantics of search terms

2.0. Definition
Given any set U of atoms, the search commands $\mathbf{d}$, $\mathbf{l}$ and $\mathbf{r}$ get their "execution meaning" merely from the functions $\boldsymbol{d}$, $\boldsymbol{l}$ and $\boldsymbol{r}$ of (13) that we can also see as partial functions in $\mathsf{P}(T \times T)$, since $T \times T \subset T$.

Even the search terms become so through a function $\sigma{:}\, \boldsymbol{T} \to \mathsf{P}(T \times T)$ that provides each search term $\boldsymbol{t}$ with its *(pure) semantics* $\sigma_{\boldsymbol{t}} \subseteq T \times T$, which is the function we define $\boldsymbol{T}$-inductively for all $\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$ by

(61) $$\sigma_{\emptyset} = \boldsymbol{i}_T \ ,$$

(62) $$\sigma_{\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle}(t) = \langle \sigma_{\boldsymbol{t}'}(t), \sigma_{\boldsymbol{t}''}(t) \rangle \text{ , for all } \ t \in \mathrm{Dom}\ \sigma_{\boldsymbol{t}'} \cap \mathrm{Dom}\ \sigma_{\boldsymbol{t}''} \ ,$$

(63) $$\sigma_{\langle \mathrm{l}, \boldsymbol{t} \rangle} = \boldsymbol{l} \cdot \sigma_{\boldsymbol{t}} \quad \text{and} \quad \sigma_{\langle \mathbf{r}, \boldsymbol{t} \rangle} = \boldsymbol{r} \cdot \sigma_{\boldsymbol{t}} \ .$$

When $\Sigma \subseteq \mathsf{P}(T \times T)$ denotes the σ-image of $\boldsymbol{T}$, $\sigma{:}\, \boldsymbol{T} \twoheadrightarrow \Sigma$, each $f \in \Sigma$ is a partial function from T to T indexed by some $\boldsymbol{t} \in \boldsymbol{T}$, $f = \sigma_{\boldsymbol{t}}$. It represents the "pure address jump" that enables any S-term of its domain to reach another S-term by the search term $\boldsymbol{t}$. Hence, this direct addressing disregards the departure S-terms, yet it does so only within some domain. To denote it, we define the *induced domain* function $\Delta{:}\, \boldsymbol{T} \to PT$ by $\Delta_{\boldsymbol{t}} = \mathrm{Dom}\ \sigma_{\boldsymbol{t}}$, for all $\boldsymbol{t} \in \boldsymbol{T}$.

This domain is the whole T by (61), when $\boldsymbol{t} = \emptyset$. Since by (62) $\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ keeps the common domain of $\boldsymbol{t}'$ and $\boldsymbol{t}''$, $\Delta_{\boldsymbol{t}'} \cap \Delta_{\boldsymbol{t}''}$, it will remain the whole T, as long as we build up search terms by $\tau'_{\mathbf{d}}$ only as in (20),

(64) $$\Delta_{\boldsymbol{u}} = T \ \text{ for all } \ \boldsymbol{u} \in \boldsymbol{N}_2(1) \ .$$

By $\tau'_{\mathbf{l}}$ or $\tau'_{\mathbf{r}}$, on the contrary, it can narrow:

(65) $$\Delta_{\langle \mathrm{l}, \boldsymbol{t} \rangle} = \Delta_{\langle \mathbf{r}, \boldsymbol{t} \rangle} \subseteq \Delta_{\boldsymbol{t}} \ .$$

In fact, since by (13) the domains of $\boldsymbol{l}$ and $\boldsymbol{r}$ are the proper subset D of T, (63) states that only the S-terms t, such that $\sigma_{\boldsymbol{t}}(t) \in D$, stay in this (narrower) domain. For example, $\Delta_{\langle \mathrm{l}, \emptyset \rangle} = D \subset T = \Delta_{\emptyset}$. From (62) we also get for every $\boldsymbol{t} \in \boldsymbol{T}$

(66) $$\Delta_{\langle \boldsymbol{t}, \boldsymbol{t} \rangle} = \Delta_{\boldsymbol{t}} \ .$$

We can denote all upward paths by S-terms through the *unlabel function* $ł\colon T \to \boldsymbol{N}_2(1)$ defined by T-induction as

(67) $$ł(u) = \emptyset \ \text{, for all } \ u \in U \ \text{, and}$$

(68) $$ł(t', t'') = \langle ł(t'), ł(t'') \rangle \ \text{, for all } \ t', t'' \in T \ .$$

Clearly,

(69) $$ł\colon T \twoheadrightarrow \boldsymbol{N}_2(1) \ .$$

Since $\boldsymbol{N}_2(1) \subseteq \boldsymbol{T}$, this allows a two-ary natural number or S-term $v \in T$ to have a *term semantics* $\sigma_{ł(v)}$, namely this defines a *term semantic function* that by (64) is $\sigma \cdot ł\colon T \to T^T$.

We can also denote some increment matrices, the constant matrices of 1.6, by S-terms through the generator of constants in (48). This allows an S-term $v \in T$ to be uniformly incremented by another S-term $t \in T$, namely we define the *growth function* $\gamma\colon T \to T^T$ by the *growth of* v *by* t, $\gamma_v(t) = \hat{\eta}_{\mathrm{K}(t)}(v)$, for all $t, v \in T$.

With respect to term incrementing as in 1.6, this uniform growth looses something, due to the proper injectivity of the generator of constants ("proper" in case of more than one atom). On the other hand, with respect to the semantic function, term semantics also does, because by (67) the unlabel function collapses all the atoms in U onto $\emptyset$. As 2.3 (B) will show, the two different losses get the same function.

Since all semantics in $\Sigma \subseteq \mathsf{P}(T \times T)$ are (partial) functions, we can consider the functional composition of any two of them. By Theorem 2.2 (B) it is again a semantics. This composition and $\boldsymbol{i}_T$ as in (61) form a monoid that we call the *complete semantic monoid.* In fact, 4.5 and 4.6 will introduce a "smaller" monoid that is more relevant to an effective semantics. Recall that by 0.4 the composition notation is reversed.

2.1. Example

Consider the search terms $\boldsymbol{t'} = \langle (\mathbf{l}, \mathbf{l}, \emptyset), \langle \mathbf{r}, \emptyset \rangle \rangle$ and $\boldsymbol{t''} = \langle (\mathbf{l}, \mathbf{l}, \emptyset), \langle (\mathbf{l}, \mathbf{r}, \emptyset), (\mathbf{r}, \mathbf{r}, \emptyset) \rangle \rangle$ of Figure 1 and 2. Clearly, they work the same by (62), (63) and (30), but only the latter requires that the right of its argument be a dyad as the diamond in Figure 3 depicts. Hence, their semantics is the function that gives us the same S-term $\langle t', t'' \rangle$, once we provide it with arguments of the same form $\langle \langle t', t \rangle, t'' \rangle$ for some $t, t' \in T$, yet respectively with $t'' \in T$ and $t'' \in D$.

In Figure 5 $\sigma_{\boldsymbol{t}'}$ sends both boxed S-terms (among others) into the two circled ones (bold arrows), whereas $\sigma_{\boldsymbol{t}''}$ sends the rightmost only. The above dyad requirement determines two properly inclusive induced domains, which correspond to the concave areas at the left and top of the dotted lines.

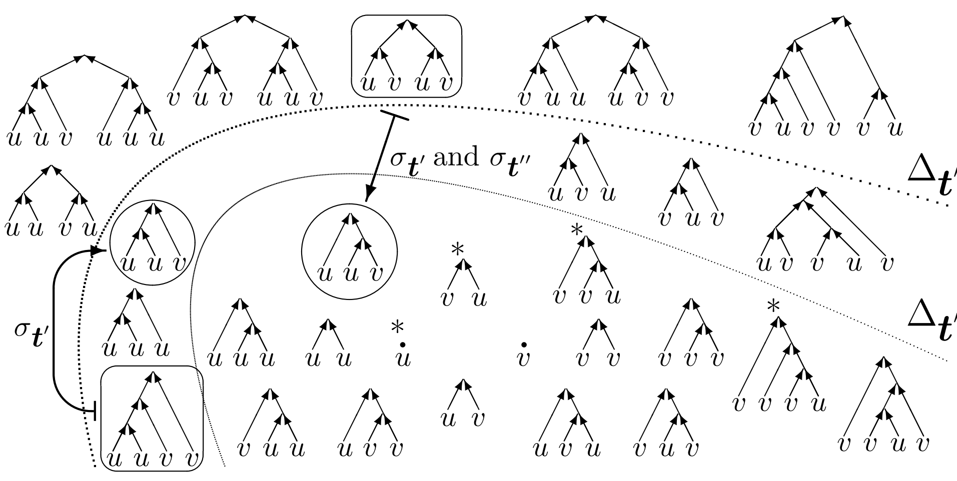

Figure 5

The equality of these two semantics on the common part of their different domains hints how Section **3** will define the notion of two-ary integers semantically. This will overcome the failures of conventional algebraic extensions we mentioned in 0.1.

Clearly, both induced domains are infinite. We starred four trees outside them. Their left to right sequence hints that also the complement of an induced domain can be infinite, contrary to the co-finiteness of induced domains of unary integers we recalled in 0.2.

Still, 4.4 (C) will extend the characterization of the latter domains (co-finite *segments* of natural numbers) to the two-ary integers, while this section will introduce certain "cartesian sets" that extend this characterizing feature to the two-ary case.

2.2. Theorems

(A) *Any semantics preserves term increments: for every $\boldsymbol{t} \in \boldsymbol{T}$ and $M\colon U \to T$, $\sigma_{\boldsymbol{t}} \cdot \hat{\eta}_M = \hat{\eta}_M \cdot \sigma_{\boldsymbol{t}}$, namely for all $t \in \Delta_{\boldsymbol{t}}$*

(70) $$\sigma_{\boldsymbol{t}}(M \hat{+} t) = M \hat{+} \sigma_{\boldsymbol{t}}(t)$$

or also by (43) $\sigma_{\boldsymbol{t}} \cdot (M \diamond L) = M \diamond (\sigma_{\boldsymbol{t}} \cdot L)$, *for every* $L{:}\, U \to \Delta_{\boldsymbol{t}}$. *(Notice that this implies* $\hat{\eta}_M(t) \in \Delta_{\boldsymbol{t}}$ *for all* $t \in \Delta_{\boldsymbol{t}}$*.)*

(B) *The semantic function is a homomorphism from the catenation monoid onto the complete semantic one, namely together with* (61) *we have* $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}} = \sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}}$, *for all* $\boldsymbol{t}, \boldsymbol{u} \in \boldsymbol{T}$.

Proofs.

(A) Bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, by (61) we get $\hat{\eta}_M \cdot \sigma_{\boldsymbol{t}} = \hat{\eta}_M \cdot \boldsymbol{i}_T = \boldsymbol{i}_T \cdot \hat{\eta}_M = \sigma_{\boldsymbol{t}} \cdot \hat{\eta}_M$, since in 1.6 $\hat{\eta}_M{:}\, T \to T$.

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\sigma_{\boldsymbol{t}'}(\hat{\eta}_M(t)) = \hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t))$ for all $t \in \Delta_{\boldsymbol{t}'}$ and $\sigma_{\boldsymbol{t}''}(\hat{\eta}_M(t)) = \hat{\eta}_M(\sigma_{\boldsymbol{t}''}(t))$ for all $t \in \Delta_{\boldsymbol{t}''}$, by (62), 1.7 (B), (52) and (62) we get $\hat{\eta}_M(\sigma_{\boldsymbol{t}}(t)) = \hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t), \sigma_{\boldsymbol{t}''}(t)) = \langle \hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t)), \hat{\eta}_M(\sigma_{\boldsymbol{t}''}(t)) \rangle = \langle \sigma_{\boldsymbol{t}'}(\hat{\eta}_M(t)), \sigma_{\boldsymbol{t}''}(\hat{\eta}_M(t)) \rangle = \sigma_{\boldsymbol{t}}(\hat{\eta}_M(t))$ for all $t \in \Delta_{\boldsymbol{t}'} \cap \Delta_{\boldsymbol{t}''} = \Delta_{\boldsymbol{t}}$.

Now, assume $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $\sigma_{\boldsymbol{t}'}(\hat{\eta}_M(t')) = \hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t'))$ for all $t' \in \Delta_{\boldsymbol{t}'}$. When $t \in \Delta_{\boldsymbol{t}}$, by (63) and (65) $\sigma_{\boldsymbol{t}'}(t) \in D$. Hence, by (42) and (41) $\hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t))$ too is a dyad and we have the left of it. By our induction assumption, (65) and (63)

(71) $$\boldsymbol{l}(\hat{\eta}_M(\sigma_{\boldsymbol{t}'}(t))) = \boldsymbol{l}(\sigma_{\boldsymbol{t}'}(\hat{\eta}_M(t))) = \sigma_{\boldsymbol{t}}(\hat{\eta}_M(t)) \ .$$

Besides, for all $\langle s', s'' \rangle \in D$, by 1.7 (B) $\hat{\eta}_M(s', s'') = \langle \hat{\eta}_M(s'), \hat{\eta}_M(s'') \rangle$ and by (54) $\boldsymbol{l}(\hat{\eta}_M(s', s'')) = \boldsymbol{l}(\hat{\eta}_M(s'), \hat{\eta}_M(s'')) = \hat{\eta}_M(\boldsymbol{l}(s', s''))$. Then, for $\langle s', s'' \rangle = \sigma_{\boldsymbol{t}'}(t) \in D$ by (71) and (63) $\sigma_{\boldsymbol{t}}(\hat{\eta}_M(t)) = \hat{\eta}_M(\boldsymbol{l}(\sigma_{\boldsymbol{t}'}(t))) = \hat{\eta}_M(\sigma_{\boldsymbol{t}}(t))$.

Finally, the step for $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$ comes from the preceding step by replacing $\boldsymbol{l}$ as usual.

(B) Use bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, by (36) $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}} = \sigma_{\boldsymbol{u}}$ and $\sigma_{\boldsymbol{t}} = \boldsymbol{i}_T$ by (61). Hence, $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}} = \boldsymbol{i}_T \cdot \sigma_{\boldsymbol{u}} = \sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}}$.

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}'} = \sigma_{\boldsymbol{t}'} \cdot \sigma_{\boldsymbol{u}}$ and $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}''} = \sigma_{\boldsymbol{t}''} \cdot \sigma_{\boldsymbol{u}}$, by (37) and (62) $\sigma_{\boldsymbol{u} \oplus \boldsymbol{t}}(t) = \sigma_{\langle \boldsymbol{u} \oplus \boldsymbol{t}', \boldsymbol{u} \oplus \boldsymbol{t}'' \rangle}(t) = \langle \sigma_{\boldsymbol{u} \oplus \boldsymbol{t}'}(t), \sigma_{\boldsymbol{u} \oplus \boldsymbol{t}''}(t) \rangle = \langle \sigma_{\boldsymbol{t}'}(\sigma_{\boldsymbol{u}}(t)), \sigma_{\boldsymbol{t}''}(\sigma_{\boldsymbol{u}}(t)) \rangle = \sigma_{\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle}(\sigma_{\boldsymbol{u}}(t))$, for all $t \in \Delta_{\boldsymbol{u} \oplus \boldsymbol{t}} = \Delta_{\boldsymbol{u} \oplus \boldsymbol{t}'} \cap \Delta_{\boldsymbol{u} \oplus \boldsymbol{t}''}$.

Also, by the induction premises and (62) $\Delta_{\boldsymbol{u}\oplus\boldsymbol{t}'} \cap \Delta_{\boldsymbol{u}\oplus\boldsymbol{t}''} = \{t \mid \sigma_{\boldsymbol{u}}(t) \in \Delta_{\boldsymbol{t}'}\} \cap \{t \mid \sigma_{\boldsymbol{u}}(t) \in \Delta_{\boldsymbol{t}''}\} = \{t \mid \sigma_{\boldsymbol{u}}(t) \in \Delta_{\boldsymbol{t}'} \cap \Delta_{\boldsymbol{t}''}\} = \{t \mid \sigma_{\boldsymbol{u}}(t) \in \Delta_{\langle \boldsymbol{t}',\boldsymbol{t}''\rangle}\} = \{t \mid \sigma_{\boldsymbol{u}}(t) \in \Delta_{\boldsymbol{t}}\} = \mathrm{Dom}\,(\sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}})$. Hence, $t \in \Delta_{\boldsymbol{u}\oplus\boldsymbol{t}}$ iff $t \in \mathrm{Dom}\,(\sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}})$ and by (62) $\sigma_{\boldsymbol{u}\oplus\boldsymbol{t}}(t) = \sigma_{\langle \boldsymbol{t}',\boldsymbol{t}''\rangle}(\sigma_{\boldsymbol{u}}(t)) = \sigma_{\boldsymbol{t}}(\sigma_{\boldsymbol{u}}(t)) = (\sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}})(t)$, for all such t, namely $\sigma_{\boldsymbol{u}\oplus\boldsymbol{t}} = \sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}}$.

When $\boldsymbol{t} = \langle \mathrm{l}, \boldsymbol{t}' \rangle$ and $\sigma_{\boldsymbol{u}\oplus\boldsymbol{t}'} = \sigma_{\boldsymbol{t}'} \cdot \sigma_{\boldsymbol{u}}$, by (38) and (63) $\sigma_{\boldsymbol{u}\oplus\boldsymbol{t}} = \sigma_{\boldsymbol{u}\oplus\langle \mathrm{l},\boldsymbol{t}'\rangle} = \sigma_{\langle \mathrm{l},\boldsymbol{u}\oplus\boldsymbol{t}'\rangle} = \boldsymbol{l} \cdot \sigma_{\boldsymbol{u}\oplus\boldsymbol{t}'} = \boldsymbol{l} \cdot (\sigma_{\boldsymbol{t}'} \cdot \sigma_{\boldsymbol{u}}) = (\boldsymbol{l} \cdot \sigma_{\boldsymbol{t}'}) \cdot \sigma_{\boldsymbol{u}} = \sigma_{\langle \mathrm{l},\boldsymbol{t}'\rangle} \cdot \sigma_{\boldsymbol{u}} = \sigma_{\boldsymbol{t}} \cdot \sigma_{\boldsymbol{u}}$, since the composition of partial functions in $P(T \times T)$ is associative. When $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$, merely replace $\boldsymbol{l}$ with $\boldsymbol{r}$. ■

2.3. Lemmata

(A) *The semantics of any upward path is one to one, for all* $\boldsymbol{u} \in \boldsymbol{N}_2(1)$

$$\sigma_{\boldsymbol{u}} \colon T \rightarrowtail T \ . \tag{72}$$

(B) *The growth function is the term semantic function,* $\gamma = \sigma \cdot \ell$.

(C) *For all full terms* $z \in \lfloor m \rfloor$ *and* $w \in \lfloor l \rfloor$,

$$\gamma_z(w) \in \lfloor l + m \rfloor \ . \tag{73}$$

(D) *If* $t \in \Delta_{\boldsymbol{t}}$, *then* $C_1 \hat{+} t \in \Delta_{\langle \mathrm{l},\boldsymbol{t}\rangle}, \Delta_{\langle \mathrm{r},\boldsymbol{t}\rangle}$.

(E) *Induced domains are "bottom-cofinite", namely for each* $\boldsymbol{t} \in \boldsymbol{T}$ *there is a full matrix* $C_n \colon U \to T$, *for some natural number* n, *such that its increment of every* $t \in T$, $v = C_n \hat{+} t$, *reaches the domain of* $\sigma_{\boldsymbol{t}}$, $v \in \Delta_{\boldsymbol{t}}$.

(F) *Catenating any search term* $\boldsymbol{t} \in \boldsymbol{T}$ *with a single letter word restricts the induced domain* $D = T \times T$ *of the word to*

$$\Delta_{\langle \mathrm{l},\emptyset\rangle\oplus\boldsymbol{t}} = \Delta_{\boldsymbol{t}} \times T \quad \textit{or} \tag{74}$$

$$\Delta_{\langle \mathrm{r},\emptyset\rangle\oplus\boldsymbol{t}} = T \times \Delta_{\boldsymbol{t}} \ . \tag{75}$$

Proofs.

(A) Use upward induction on $\boldsymbol{u}$. When $\boldsymbol{u} = \emptyset$, by (61) $\sigma_{\boldsymbol{u}} = \boldsymbol{i}_T$ is one to one. When $\boldsymbol{u} = \langle \boldsymbol{u}', \boldsymbol{u}'' \rangle$ and $\sigma_{\boldsymbol{u}'}$ and $\sigma_{\boldsymbol{u}''}$ are one to one, for all $t', t'' \in T$,

$\sigma_{\boldsymbol{u}}(t') = \sigma_{\boldsymbol{u}}(t'')$ implies by (62) that $\langle \sigma_{\boldsymbol{u}'}(t'), \sigma_{\boldsymbol{u}''}(t')\rangle = \langle \sigma_{\boldsymbol{u}'}(t''), \sigma_{\boldsymbol{u}''}(t'')\rangle$. Hence, by (2) and the induction premises $t' = t''$.

(B) We prove $\hat{\eta}_{\mathrm{K}(t)}(v) = \sigma_{t(v)}(t)$ for all $t, v \in T$ by T-induction on v. By (42) this induction has to prove $\mathrm{K}_t \hat{+} v = \sigma_{t(v)}(t)$. The basis step for all $v = u \in U \subset T$ follows from (40), (48), (61) and (67), $\mathrm{K}_t \hat{+} u = \mathrm{K}_t(u) = t = \sigma_{\emptyset}(t) = \sigma_{t(u)}(t)$.

When $v = \langle v', v''\rangle$ and both $\mathrm{K}_t \hat{+} v' = \sigma_{t(v')}(t)$ and $\mathrm{K}_t \hat{+} v'' = \sigma_{t(v'')}(t)$, by (41), (62) and (68) $\mathrm{K}_t \hat{+} \langle v', v''\rangle = \langle \mathrm{K}_t \hat{+} v', \mathrm{K}_t \hat{+} v''\rangle = \langle \sigma_{t(v')}(t), \sigma_{t(v'')}(t)\rangle$ $= \sigma_{\langle t(v'), t(v'')\rangle}(t) = \sigma_{t(v)}(t)$.

Hence, we proved the induction step too.

(C) We prove (73) by arithmetic induction on m. When $m = 0$, $z \in U$ and $\gamma_z(w) = \mathrm{K}_w \hat{+} z = \mathrm{K}_w(z) = w \in \lfloor l \rfloor = \lfloor l + m \rfloor$ by (42),(40) and (48). Assume $z = \langle z', z''\rangle \in \lfloor m+1 \rfloor$ with $z', z'' \in \lfloor m \rfloor$, according to (10), and $\gamma_{z'}(w), \gamma_{z''}(w) \in \lfloor l + m \rfloor$. Then, by (42),(41) and sum associativity, $\gamma_z(w) = \mathrm{K}_w \hat{+} \langle z', z''\rangle = \langle \mathrm{K}_w \hat{+} z', \mathrm{K}_w \hat{+} z''\rangle = \langle \gamma_{z'}(w), \gamma_{z''}(w)\rangle \in \lfloor (l+m) + 1 \rfloor = \lfloor l + (m+1) \rfloor$.

(D) When $t \in \Delta_{\boldsymbol{t}}$, by 2.2 (A) $C_1 \hat{+} t \in \Delta_{\boldsymbol{t}}$ and $\sigma_{\boldsymbol{t}}(C_1 \hat{+} t) = C_1 \hat{+} \sigma_{\boldsymbol{t}}(t) \in D$ by (70) and 1.8 (A). Hence, by (63) and (13) $C_1 \hat{+} t \in \Delta_{\langle \mathbf{l}, \boldsymbol{t}\rangle}, \Delta_{\langle \mathbf{r}, \boldsymbol{t}\rangle}$.

(E) Use bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, it comes from $\Delta_{\boldsymbol{t}} = T$ as in (61). In fact, when we take $C_0 = \boldsymbol{i}_U$, by (42) and (58) for all such t, $v = \boldsymbol{i}_U \hat{+} t = t \in \Delta_{\boldsymbol{t}}$.

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}''\rangle$ and there are $C_{n'}$ and $C_{n''}$ such that $v' = C_{n'} \hat{+} t \in \Delta_{\boldsymbol{t}'}$ and $v'' = C_{n''} \hat{+} t \in \Delta_{\boldsymbol{t}''}$ for all $t \in T$, we take $n = \max(n', n'')$ and get $v \in \Delta_{\boldsymbol{t}}$ from (62) because both $v \in \Delta_{\boldsymbol{t}'}$ and $v \in \Delta_{\boldsymbol{t}''}$. In fact, assume $n = n'$ and let $m = n - n''$, then $v = v' \in \Delta_{\boldsymbol{t}'}$ and $v = (C_m \diamond C_{n''}) \hat{+} t = C_m \hat{+} v'' \in \Delta_{\boldsymbol{t}''}$ by 1.8 (B), (57) and 2.2 (A). The same for the assumption $n = n''$.

When $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}'\rangle$ and there is $C_{n'}$ such that $v' = C_{n'} \hat{+} t \in \Delta_{\boldsymbol{t}'}$ for all $t \in T$, take $n = n' + 1$. Then, $v = C_n \hat{+} t = (C_1 \diamond C_{n'}) \hat{+} t = C_1 \hat{+} v' \in \Delta_{\boldsymbol{t}}$ by 1.8 (B), (57) and (D). Finally,we prove the step $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}'\rangle$ by replacing $\boldsymbol{l}$ with $\boldsymbol{r}$ as usual.

(F) For (74) notice that by 2.2 (B), (63) and (61) $\sigma_{\langle \mathbf{l}, \emptyset\rangle \oplus \boldsymbol{t}} = \sigma_{\boldsymbol{t}} \cdot \sigma_{\langle \mathbf{l}, \emptyset\rangle} = \sigma_{\boldsymbol{t}} \cdot \boldsymbol{l}$. Hence, its domain is the set $\{t \mid t \in T \times T$ and $\boldsymbol{l}(t) \in \Delta_{\boldsymbol{t}}\} = \{t \mid t \in T \times T$ and $t \in \Delta_{\boldsymbol{t}} \times T\} = T \times T \cap (\Delta_{\boldsymbol{t}} \times T) = \Delta_{\boldsymbol{t}} \times T$ by (13), the distributivity in **I**.3.13 (f) of [13] and because $\Delta_{\boldsymbol{t}} \subseteq T$. For (75) merely replace $\boldsymbol{l}$ with $\boldsymbol{r}$ and commute these cartesian products. ■

2.4. Corollary
No induced domain can be empty, $\Delta_{\boldsymbol{t}} \neq \emptyset$ *for all* $\boldsymbol{t} \in \boldsymbol{T}$*. Furthermore, every* $\Delta_{\boldsymbol{t}}$ *is infinite.*

Proof.
At least the S-terms v of 2.3 (E) are in $\Delta_{\boldsymbol{t}}$. They are infinite because T, the domain of any (full) increment, is and because of 1.8 (C). ■

2.5. Definition
For every two-ary natural number $\boldsymbol{t} \in \boldsymbol{N}_2(1)$ we define the $\boldsymbol{t}$*–structured cartesian set* $\mathsf{X}(\boldsymbol{t})$ by upward induction on $\boldsymbol{t}$ as

$$\mathsf{X}(\emptyset) = T \text{ and} \tag{76}$$

$$\mathsf{X}(t', t'') = \mathsf{X}(t') \times \mathsf{X}(t'') \ . \tag{77}$$

By (5) such an induction also shows that every $\boldsymbol{t}$–structured cartesian set is made of S-terms,

$$\mathsf{X}(\boldsymbol{t}) \subseteq T = \boldsymbol{N}_2(U) \ . \tag{78}$$

If a set of S-terms is $\boldsymbol{t}$–structured for some such a $\boldsymbol{t}$, then we say that it is a *cartesian set* and we denote their set by $\mathcal{T} = \{\mathsf{X}(\boldsymbol{t}) \mid \boldsymbol{t} \in \boldsymbol{N}_2(1)\} \subset PT$. Notice that $U \notin \mathcal{T}$, since the only cartesian set containing it is $T \neq U$. Hence, we got the *cartesian structure function* $\mathsf{X}\colon \boldsymbol{N}_2(1) \twoheadrightarrow \mathcal{T}$.

We define the *branch (left) opposite* $\neg w$ *of* a word w recursively by the backward induction of 1.6 as

$$\neg\emptyset = \emptyset \ ,$$

$$\neg(\langle \mathbf{l}, \emptyset\rangle \oplus v) = \langle \neg v, \emptyset\rangle \quad \text{and} \quad \neg(\langle \mathbf{r}, \emptyset\rangle \oplus v) = \langle \emptyset, \neg v\rangle \quad .$$

For instance, this backward induction gets $\neg(\langle \mathbf{r}, \langle \mathbf{l}, \langle \mathbf{l}, \emptyset\rangle\rangle\rangle) = \langle\langle\langle \emptyset, \emptyset\rangle, \emptyset\rangle, \emptyset\rangle$. By backward induction also we clearly get $\neg\colon W \to \boldsymbol{N}_2(1)$.

For each $\boldsymbol{z} \in \boldsymbol{N}_2(1)$ we say that $\boldsymbol{t} \in \boldsymbol{N}_2(1)$ is *younger or equal* to $\boldsymbol{z}$ and we write $\boldsymbol{t} \leq \boldsymbol{z}$, when we can relate them by the minimal preorder such that

$$\emptyset \leq \langle \emptyset, \emptyset\rangle \qquad \text{and, for all } t', t'', z', z'' \in \boldsymbol{N}_2(1), \tag{79}$$

$$t' \leq z',\ t'' \leq z'' \qquad \text{imply} \quad \langle t', t''\rangle \leq \langle z', z''\rangle \ . \tag{80}$$

We could easily make these conditions (together with reflexivity and transitivity) into the definition of the Galois connection for the *age* relation $\leq$ by upward induction on $\boldsymbol{t}$. As 2.7 (C) will show $\leq$ to be a partial order, for any set $V \subseteq \boldsymbol{N}_2(1)$ with a g.l.b $v = \bigwedge V$, we say that v is its *youngest term,* when $v \in V$.

These age related names refer to the sprouting in botanical trees, which contrary to mathematical ones grow bottom up. They hint readers to distinguish $\leq$ from the preorder $\preceq$ such that $t', t'' \preceq \langle t', t'' \rangle$.

2.6. Lemmata

(A) *If* $Y, Z \in \mathcal{T}$, *then* $Y \cap Z \in \mathcal{T}$.

(B) *No cartesian set is finite.*

(C) *Cartesian sets identify their structures, namely the cartesian structure function is a bijection* $\mathsf{X}\colon \boldsymbol{N}_2(1) \rightarrowtail\!\!\!\!\twoheadrightarrow \mathcal{T}$.

(D) $\mathsf{X}\cdot \text{ł}\colon T \twoheadrightarrow \mathcal{T}$ *is a homomorphism from the dyad operation* $\boldsymbol{d}\colon T \times T \to T$ *onto the cartesian product of cartesian sets* $\times\colon \mathcal{T} \times \mathcal{T} \to \mathcal{T}$, *such that* $\mathsf{X}(\text{ł}(u)) = T$ *for* $u \in U$ *and* $t \in \mathsf{X}(\text{ł}(t))$ *for all* $t \in T$.

(E) $\emptyset$ *is the youngest term of* $\boldsymbol{N}_2(1)$.

Proofs.

(A) Given Y, let $Z = \mathsf{X}(\boldsymbol{z})$. Then we can prove our statement by upward induction on $\boldsymbol{z}$. When $\boldsymbol{z} = \emptyset$, $Z = T$ by (76) and, since $Y \subseteq T$, $Y \cap Z = Y \in \mathcal{T}$. If $\boldsymbol{z} = \langle \boldsymbol{z}', \boldsymbol{z}'' \rangle$ with $Y \cap \mathsf{X}(\boldsymbol{z}'), Y \cap \mathsf{X}(\boldsymbol{z}'') \in \mathcal{T}$, then there are $t', t'' \in \boldsymbol{N}_2(1)$ such that $Y \cap \mathsf{X}(\boldsymbol{z}') = \mathsf{X}(t')$ and $Y \cap \mathsf{X}(\boldsymbol{z}'') = \mathsf{X}(t'')$. Hence, by (77) and the usual distributivity in **I**.3.13 (f) of [13] we get $Y \cap Z = Y \cap (\mathsf{X}(\boldsymbol{z}') \times \mathsf{X}(\boldsymbol{z}'')) = (Y \cap \mathsf{X}(\boldsymbol{z}')) \times (Y \cap \mathsf{X}(\boldsymbol{z}'')) = \mathsf{X}(t') \times \mathsf{X}(t'') = \mathsf{X}(t', t'') \in \mathcal{T}$.

(B) Trivial upward induction.

(C) We prove that $\mathsf{X}(\boldsymbol{z}) = \mathsf{X}(\boldsymbol{t})$ implies $\boldsymbol{z} = \boldsymbol{t}$ by upward induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, by (4) and (76) $U \subseteq T = \mathsf{X}(\boldsymbol{t}) = \mathsf{X}(\boldsymbol{z})$, which by (77), (8) and (34) implies that $\boldsymbol{z}$ cannot be a dyad, $\boldsymbol{z} = \emptyset = \boldsymbol{t}$.

Now, consider $\boldsymbol{t} = \langle t', t'' \rangle$, where, for all $z \in \boldsymbol{N}_2(1)$, both $\mathsf{X}(z) = \mathsf{X}(t')$ implies $t' = z$ and $\mathsf{X}(z) = \mathsf{X}(t'')$ implies $t'' = z$. By (77) and (8) $\mathsf{X}(\boldsymbol{t}) = \mathsf{X}(\boldsymbol{z})$ cannot contain U. Hence, $\boldsymbol{z}$ is a dyad $\langle z', z'' \rangle$.

Then, (77) rewrites the assumption $\mathsf{X}(\boldsymbol{z}) = \mathsf{X}(\boldsymbol{t})$ as an equality of cartesian products, $\mathsf{X}(z') \times \mathsf{X}(z'') = \mathsf{X}(t') \times \mathsf{X}(t'')$. This implies two equalities on their factors, which by (B) are not empty: $\mathsf{X}(z') = \mathsf{X}(t')$ and $\mathsf{X}(z'') = \mathsf{X}(t'')$. Therefore, by the induction premises $\boldsymbol{t} = \boldsymbol{z}$, because of (2).

(D) $\mathsf{X}(\text{ŧ}(u)) = T$ for $u \in U$ immediately follows from (67) and (76). $\mathsf{X}(\text{ŧ}(\boldsymbol{d}(t', t''))) = \mathsf{X}(\text{ŧ}(t')) \times \mathsf{X}(\text{ŧ}(t''))$ for all $t', t'' \in T$ from 1.1, (68) and (77). Finally, to get $t \in \mathsf{X}(\text{ŧ}(t))$, we use T-induction on t and the recalled definitions.

(E) We prove $\emptyset \leq \boldsymbol{z}$ by upward induction on $\boldsymbol{z}$. When $\boldsymbol{z} = \emptyset$, we use reflexivity. When $\boldsymbol{z} = \langle z', z'' \rangle$ with $\emptyset \leq z'$ and $\emptyset \leq z''$, by (80) we get $\langle \emptyset, \emptyset \rangle \leq \boldsymbol{z}$ and by (79) and transitivity $\emptyset \leq \boldsymbol{z}$. ■

2.7. Theorems

(A) *Containment between cartesian sets is a semilattice with zero.* (This is not the main motivation of our choice of $\supseteq$ and $\cap$ as the order and join respectively, instead of the dual choice. The main one will appear in 4.5.)

(B) *The branch opposite of any word $w \in W$ defines the structure of its induced domain by $\Delta_w = \mathsf{X}(\rightharpoondown w)$. Hence, the induced domains of words are cartesian sets.*

(C) *The cartesian structure function $\mathsf{X}\colon \boldsymbol{N}_2(1) \Vdash\!\!\twoheadrightarrow \mathcal{T}$, namely the function $\mathsf{X}\cdot \text{ŧ}\colon T \Vdash\!\!\twoheadrightarrow \mathcal{T}$ for $U = 1$, is an isomorphism from the age relation $\leq$ onto the containment of* (A)*,*

$$\boldsymbol{t} \leq \boldsymbol{z} \ \text{ iff } \ \mathsf{X}(\boldsymbol{t}) \supseteq \mathsf{X}(\boldsymbol{z}) \ \ \textit{for all} \ \ \boldsymbol{t}, \boldsymbol{z} \in \boldsymbol{N}_2(1) \ , \tag{81}$$

where for $U = 1$ each argument is the youngest term of its cartesian set: for all $\boldsymbol{t} \in \boldsymbol{N}_2(1)$, $\boldsymbol{t} \in \mathsf{X}(\boldsymbol{t})$ and $\boldsymbol{t} \leq \boldsymbol{z}$ for all $\boldsymbol{z} \in \mathsf{X}(\boldsymbol{t})$, hence X defines a closure system.

Proofs.

(A) The join closure is in 2.6 (A). The zero is in (76) because of (78).

(B) Use backward induction on w. By 2.5 and 1.6 this is to prove that

$$\Delta_\emptyset = T \ ,$$

$$\Delta_{\langle \mathrm{l}, \emptyset \rangle \oplus \boldsymbol{w}} = \Delta_{\boldsymbol{w}} \times T \ \text{and} \tag{82}$$

$$\Delta_{\langle \mathrm{r}, \emptyset \rangle \oplus \boldsymbol{w}} = T \times \Delta_{\boldsymbol{w}} \ . \tag{83}$$

The base of this recursion is in (64). For the induction step notice that (82) and (83) are cases of (74) and (75) respectively.

(C) After 2.6 (C), to get the isomorphism, we only have to prove (81). (Only if) Since $\supseteq$ on $\mathcal{T}$ is a preorder as $\leq$ is, we only prove the cases corresponding to (79) and (80) by upward induction on $\boldsymbol{t}$: the basis step for (79) comes from (5) with $U = 1$ through (76) and (77), while case (80) comes from the induction premises through (77), because for all sets X', X'', Y' and Y'', if $X' \supseteq Y'$ and $X'' \supseteq Y''$ then $X' \times X'' \supseteq Y' \times Y''$.

(If) Upward induction on $\boldsymbol{z}$. When $\boldsymbol{z} = \emptyset$, by (76) $\mathsf{X}(\boldsymbol{z}) = T$ and by (78) $\mathsf{X}(\boldsymbol{t}) = \mathsf{X}(\boldsymbol{z})$, which by 2.6 (C) implies $\boldsymbol{t} = \boldsymbol{z}$ in agreement with age reflexivity.

Then, assume that $\boldsymbol{z} = \langle z', z''\rangle$, where $\mathsf{X}(t') \supseteq \mathsf{X}(z')$ implies $t' \leq z'$, while $\mathsf{X}(t'') \supseteq \mathsf{X}(z'')$ implies $t'' \leq z''$. Either $\boldsymbol{t} = \emptyset$, which by 2.6 (E) implies $\boldsymbol{t} \leq \boldsymbol{z}$, or $\boldsymbol{t} = \langle t', t''\rangle$ for some $t', t'' \in \boldsymbol{N}_2(1)$. In the latter case we easily get both premises of the induction premises from $\mathsf{X}(\boldsymbol{t}) \supseteq \mathsf{X}(\boldsymbol{z})$. Hence, by (80) we get $\boldsymbol{t} \leq \boldsymbol{z}$.

Finally, we get $\boldsymbol{t} \in \mathsf{X}(\boldsymbol{t})$ from 2.6 (D) with $U = 1$ and $\boldsymbol{t} \leq \boldsymbol{z}$ for $\boldsymbol{z} \in \mathsf{X}(\boldsymbol{t})$ by upward induction on $\boldsymbol{t}$. In fact, the basis step comes from (76) and 2.6 (E). Then, assume $\boldsymbol{t} = \langle t', t''\rangle$ with $t' \leq z'$ and $t'' \leq z''$ for all $z' \in \mathsf{X}(t')$ and $z'' \in \mathsf{X}(t'')$. Since $\boldsymbol{z} \in \mathsf{X}(t', t'')$, by (77) $\boldsymbol{z}$ is a dyad $\langle z', z''\rangle$ such that $z' \in \mathsf{X}(t')$ and $z'' \in \mathsf{X}(t'')$. Hence, the induction premises by (80) imply $\boldsymbol{t} \leq \boldsymbol{z}$. ■

3. Two-ary integers

3.0. Definition

Consider two search terms $\boldsymbol{t}, \boldsymbol{u} \in \boldsymbol{T}$. When

$$\sigma_{\boldsymbol{t}}(t) = \sigma_{\boldsymbol{u}}(t) \quad \text{for all} \quad t \in \Delta_{\boldsymbol{t}} \cap \Delta_{\boldsymbol{u}} \ , \tag{84}$$

namely when the semantics of $\boldsymbol{t}$ and of $\boldsymbol{u}$ agree whenever both of them are defined, we say that the two search terms are *locally equal* and write $\boldsymbol{t} \asymp \boldsymbol{u}$.

3.1. Recursivity

From 2.1 we can fairly guess that the domain of a semantics is recursive and that a semantics is a (recursive restriction of a) recursive function. (We omit the technical details.) Yet, the domain intersection in (84) is infinite by 2.4 and (62). Hence, as far as the computability of local equality is concerned, now we can only guess that local inequality is enumerable.

This rises the problem whether local equality is recursive or not. Its affirmative answer will come from the characterization of local equality in 4.3. This characterization will use a recursive procedure defined in 4.0.

3.2. Lemmata

(A) *τ-operations preserve $\asymp$. Namely, for all $\boldsymbol{t}, \boldsymbol{u}, \boldsymbol{t}', \boldsymbol{u}', \boldsymbol{t}'', \boldsymbol{u}'' \in \boldsymbol{T}$*

$$\boldsymbol{t} \asymp \boldsymbol{u} \quad \textit{implies} \quad \langle \mathbf{l}, \boldsymbol{t}\rangle \asymp \langle \mathbf{l}, \boldsymbol{u}\rangle\,,\ \langle \mathbf{r}, \boldsymbol{t}\rangle \asymp \langle \mathbf{r}, \boldsymbol{u}\rangle \quad \textit{and} \tag{85}$$

$$\boldsymbol{t}' \asymp \boldsymbol{u}',\, \boldsymbol{t}'' \asymp \boldsymbol{u}'' \quad \textit{imply} \quad \langle \boldsymbol{t}', \boldsymbol{t}''\rangle \asymp \langle \boldsymbol{u}', \boldsymbol{u}''\rangle\ . \tag{86}$$

(B) *Relation $\asymp$ is an equivalence on $\boldsymbol{T}$.*

(C) *Local equality on words becomes equality, for all $\boldsymbol{v}, \boldsymbol{w} \in W$, $\boldsymbol{v} \asymp \boldsymbol{w}$ implies $\boldsymbol{v} = \boldsymbol{w}$.*

Proofs.

(A) Consider (85) , then by (65) the domains of $\sigma_{\langle \mathbf{l}, \boldsymbol{t}\rangle}$ and $\sigma_{\langle \mathbf{l}, \boldsymbol{u}\rangle}$ respectively are subsets of $\Delta_{\boldsymbol{t}}$ and $\Delta_{\boldsymbol{u}}$. Hence their intersection is a subset of $\Delta_{\boldsymbol{t}} \cap \Delta_{\boldsymbol{u}}$, where $\boldsymbol{l} \cdot \sigma_{\boldsymbol{t}}$ and $\boldsymbol{l} \cdot \sigma_{\boldsymbol{u}}$ coincide by (84). By (63) $\sigma_{\langle \mathbf{l}, \boldsymbol{t}\rangle}$ and $\sigma_{\langle \mathbf{l}, \boldsymbol{u}\rangle}$ too coincide, as required. The same holds when we replaces $\mathbf{l}$ by $\mathbf{r}$.

To prove (86) let us introduce some more notation. Let $\Delta_{\boldsymbol{t}'} = x$, $\Delta_{\boldsymbol{t}''} = y$, $\Delta_{\boldsymbol{u}'} = u$ and $\Delta_{\boldsymbol{u}''} = v$, We have to show that $\sigma_{\langle \boldsymbol{t}', \boldsymbol{t}''\rangle}$ and $\sigma_{\langle \boldsymbol{u}', \boldsymbol{u}''\rangle}$ agree on $\Delta_{\langle \boldsymbol{t}', \boldsymbol{t}''\rangle} \cap \Delta_{\langle \boldsymbol{u}', \boldsymbol{u}''\rangle} = (x \cap y) \cap (u \cap v) = (x \cap u) \cap (y \cap v)$, once we know that $\sigma_{\boldsymbol{t}'}$ agrees with $\sigma_{\boldsymbol{u}'}$ on $x \cap u$ and that $\sigma_{\boldsymbol{t}''}$ agrees with $\sigma_{\boldsymbol{u}''}$ on $y \cap v$. This comes from $(x \cap u) \cap (y \cap v) \subseteq x \cap u\,,\ y \cap v$, because of (62) and (2).

(B) Symmetry and the domain are trivial. Hence, we only show that, for all $\boldsymbol{t}, \boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{T}$, $\boldsymbol{t} \asymp \boldsymbol{v}$ and $\boldsymbol{v} \asymp \boldsymbol{w}$ imply $\boldsymbol{t} \asymp \boldsymbol{w}$. Take any $t \in \Delta_{\boldsymbol{t}} \cap \Delta_{\boldsymbol{w}}$. By Lemma 2.3 (E) there is $M = C_n\colon U \to T$ such that $v = \hat{\eta}_M(t) \in \Delta_{\langle \boldsymbol{t}, \langle \boldsymbol{v}, \boldsymbol{w}\rangle\rangle}$. Hence, by (62)$v \in \Delta_{\boldsymbol{t}} \cap \Delta_{\boldsymbol{v}} \cap \Delta_{\boldsymbol{w}}$. Therefore, by (84) the premises imply that $\sigma_{\boldsymbol{t}}(v) = \sigma_{\boldsymbol{v}}(v) = \sigma_{\boldsymbol{w}}(v)$.

From $\sigma_{\boldsymbol{t}}(v) = \sigma_{\boldsymbol{w}}(v)$ 2.2 (A) gets $\hat{\eta}_M(\sigma_{\boldsymbol{t}}(t)) = \hat{\eta}_M(\sigma_{\boldsymbol{w}}(t))$. When $n = 0$, this already is $\sigma_{\boldsymbol{t}}(t) = \sigma_{\boldsymbol{w}}(t)$ by (58), while for $n > 0$ we get it from Lemma 1.8 (C). Hence, $\boldsymbol{t} \asymp \boldsymbol{w}$.

(C) At first, notice that the only word locally equal to the empty one is the empty word: for all $\boldsymbol{w} \in W$, $\boldsymbol{w} \asymp \emptyset$ implies $\boldsymbol{w} = \emptyset$. In fact, by (61) this premise implies $\sigma_{\boldsymbol{w}}(t) = t$ for all $t \in \Delta_{\boldsymbol{w}}$, where $\Delta_{\boldsymbol{w}} \neq \emptyset$ by 2.4. By the definition of pair in 0.4 it is trivial to show that this violates the regularity axiom of Set Theory in **I**.1.18 of [13], as in theorem **I**.1.19 ibid., unless the sequence of compositions of $\boldsymbol{l}$ and $\boldsymbol{r}$ in $\sigma_{\boldsymbol{w}}$ is empty. Hence, $\boldsymbol{w} = \emptyset$.

Now, it is enough to prove our statements for nonempty words $\boldsymbol{v}, \boldsymbol{w} \in W$ and we do it by contradiction.

Assume that $\boldsymbol{v}$ is such that $\boldsymbol{v} \asymp \boldsymbol{w}$ and $\boldsymbol{v} \neq \boldsymbol{w}$. Then, there are $\boldsymbol{v}', \boldsymbol{w}' \in W$ and words of length one $\boldsymbol{v}'', \boldsymbol{w}'' = \langle \mathbf{l}, \emptyset \rangle, \langle \mathbf{r}, \emptyset \rangle$ such that

$$\boldsymbol{v} = \boldsymbol{v}'' \oplus \boldsymbol{v}' \ \text{ and } \ \boldsymbol{w} = \boldsymbol{w}'' \oplus \boldsymbol{w}' \ . \tag{87}$$

Let us consider the four possible choices of $\boldsymbol{v}''$ and $\boldsymbol{w}''$.

(Case $\boldsymbol{v}'' = \langle \mathbf{l}, \emptyset \rangle \neq \boldsymbol{w}'' = \langle \mathbf{r}, \emptyset \rangle$) By 2.3 (E) there is a natural number n such that, for all $t \in T$, $C_n \hat{+} t \in \Delta_{\langle \boldsymbol{v}, \boldsymbol{w} \rangle} = \Delta_{\boldsymbol{v}} \cap \Delta_{\boldsymbol{w}}$. Hence, $\sigma_{\boldsymbol{v}}(C_n \hat{+} t) = \sigma_{\boldsymbol{w}}(C_n \hat{+} t)$ and by 2.2 (B) (87) gets $\sigma_{\boldsymbol{v}'}(\sigma_{\boldsymbol{v}''}(C_n \hat{+} t)) = \sigma_{\boldsymbol{w}'}(\sigma_{\boldsymbol{w}''}(C_n \hat{+} t))$. When $t \in D$, by 2.2 (A) this implies

$$\sigma_{\boldsymbol{v}'}(C_n \hat{+} \sigma_{\boldsymbol{v}''}(t)) = \sigma_{\boldsymbol{w}'}(C_n \hat{+} \sigma_{\boldsymbol{w}''}(t)) \ . \tag{88}$$

Now, take $t = \langle \langle u, u \rangle, u \rangle$ for some $u \in U$. Then, this choice of $\boldsymbol{v}''$ and $\boldsymbol{w}''$ gets $C_n \hat{+} \sigma_{\boldsymbol{v}''}(t) = C_n \hat{+} \langle u, u \rangle = \langle C_n \hat{+} u, C_n \hat{+} u \rangle = \langle C_n(u), C_n(u) \rangle = C_{n+1}(u) = C_1 \hat{+} C_n(u)$ by (41), (40), (47), 1.8 (B), (57) and (45) as well as $C_n \hat{+} \sigma_{\boldsymbol{w}''}(t) = C_n \hat{+} u = C_n(u)$ by (40). Flip our S-term, $t = \langle u, \langle u, u \rangle \rangle$, and get $C_n \hat{+} \sigma_{\boldsymbol{w}''}(t) = C_1 \hat{+} C_n(u)$ and $C_n \hat{+} \sigma_{\boldsymbol{v}''}(t) = C_n(u)$ in the same way.

Hence, from (88) the former t by 2.3 (E) and (70) gets

$$\sigma_{\boldsymbol{v}'}(C_1 \hat{+} C_n(u)) = C_1 \hat{+} \sigma_{\boldsymbol{v}'}(C_n(u)) = \sigma_{\boldsymbol{w}'}(C_n(u)) \ , \tag{89}$$

while the latter gets

$$\sigma_{\boldsymbol{v}'}(C_n(u)) = \sigma_{\boldsymbol{w}'}(C_1 \hat{+} C_n(u)) = C_1 \hat{+} \sigma_{\boldsymbol{w}'}(C_n(u)) \ . \tag{90}$$

This implies $\sigma_{\boldsymbol{v}'}(C_n(u)) = C_1 \hat{+} (C_1 \hat{+} \sigma_{\boldsymbol{v}'}(C_n(u))) = (C_1 \diamond C_1) \hat{+} \sigma_{\boldsymbol{v}'}(C_n(u))$ $= C_2 \hat{+} \sigma_{\boldsymbol{v}'}(C_n(u))$ by (57) and 1.8 (B). Thus, $\sigma_{\boldsymbol{v}'}(C_n(u))$ is a fixed point of a full increment, contrary to 1.8 (D).

(Case $\boldsymbol{v}'' = \langle \mathbf{r}, \emptyset \rangle \neq \boldsymbol{w}'' = \langle \mathbf{l}, \emptyset \rangle$) In all the preceding argument we can exchange $\mathbf{l}$ with $\mathbf{r}$ and $\boldsymbol{v}$ with $\boldsymbol{w}$. Hence, the fixed point violation takes place again.

(Case $\boldsymbol{v}'' = \langle \mathbf{l}, \emptyset \rangle = \boldsymbol{w}''$) Here, flipping the S-terms t does not work. We resort to the usual minimal length contradiction argument. Assume that $\boldsymbol{v}$ is of a minimal length.

Then, by 2.2 (B), (63) and (87) our premise $\boldsymbol{v} \asymp \boldsymbol{w}$ becomes $\sigma_{\boldsymbol{v}'}(\boldsymbol{l}(t)) = \sigma_{\boldsymbol{w}'}(\boldsymbol{l}(t))$ for all $t \in \Delta_v \cap \Delta_w$. By (74) $\Delta_v \cap \Delta_w = (\Delta_{\boldsymbol{v}'} \times T) \cap (\Delta_{\boldsymbol{w}'} \times T) = (\Delta_{\boldsymbol{v}'} \cap \Delta_{\boldsymbol{w}'}) \times T$ because of the distributivity of the set operations involved, e.g. see **I**.3.13 (f) of [13].

Moreover, for any $T' \subseteq T$, $t \in T' \times T$ iff $t' = \boldsymbol{l}(t) \in T'$ as it follows from 1.1. Hence, $\sigma_{\boldsymbol{v}'}(t') = \sigma_{\boldsymbol{w}'}(t')$ for all $t' \in \Delta_{\boldsymbol{v}'} \cap \Delta_{\boldsymbol{w}'}$, namely $\boldsymbol{v}' \asymp \boldsymbol{w}'$, where $\boldsymbol{v}'$ is shorter than $\boldsymbol{v}$ of one letter contrary to our minimality assumption.

(Case $\boldsymbol{v}'' = \langle \mathbf{r}, \emptyset \rangle = \boldsymbol{w}''$) In the preceding argument replace $\boldsymbol{l}$ with $\boldsymbol{r}$ and (74) with (75). ■

3.3. Definition

The relation $\asymp$ we have defined in 3.0 is an equivalence relation on $\boldsymbol{T}$ by 3.2 (B). Hence, by dividing $\boldsymbol{T}$ by $\asymp$ we get a set $\boldsymbol{I}_2$ that we call of the *two-ary integers*. In fact, $\boldsymbol{I}_2$ generalizes the usual integers, in the way introduced by 0.0 and 0.2. In another way, Lemma 3.2 (C) hints it might well generalize words. (Besides, 4.2 (E) will extend the uniqueness property in this lemma to the upward paths in $\boldsymbol{N}_2(1)$.) We will confirm this hint in 4.2 and 5.1 (A).

By Lemma 3.2 (A) and (B) $\asymp$ also is a congruence of the τ–operations. Then, by an algebra division one might get an *abstract* algebra on $\boldsymbol{I}_2$. We do not because we want to peer into it.

For instance, the abstract algebra of usual integers, the equivalence classes of pairs $\langle m, n \rangle$ of natural numbers we recalled in 0.1, is of little use. The actual algebra is the one of the *reduced* representatives of such abstract elements, the ones of the forms $\langle m, 0 \rangle$ or $\langle 0, n \rangle$.

It gets an effective representation of usual integers: "natural numbers with sign".

Here also, we will modify our two-ary natural numbers (S-terms) to get an effective concrete representation of two-ary integers. In fact, Section **4** will define a reduction procedure that gets certain S-terms. As in the unary case, this procedure has a functional nature: no need of any Church-Rosser property.

3.4. Lemma

For all $\boldsymbol{t}, \boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$,

$$\langle \langle \mathbf{l}, \boldsymbol{t} \rangle, \langle \mathbf{r}, \boldsymbol{t} \rangle \rangle \quad \asymp \quad \boldsymbol{t} \qquad \textit{and} \tag{91}$$

$$\langle \mathbf{l}, \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \rangle \asymp \boldsymbol{t}' \quad \textit{and} \quad \langle \mathbf{r}, \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \rangle \asymp \boldsymbol{t}'' \ , \tag{92}$$

(While the inversion equations of 1.3 (B) do not always hold for two-ary natural numbers because of the quantification in (30), this states they could become identities for all two-ary integers, if we had introduced their algebra.)

Proof.
When in (30) we replace $\sigma_{\boldsymbol{t}}(t)$ for t, (14), (62) and (63) prove that $\sigma_{\langle\langle\mathbf{l},\boldsymbol{t}\rangle,\langle\mathbf{r},\boldsymbol{t}\rangle\rangle}(t) = \sigma_{\boldsymbol{t}}(t)$ holds for all t such that $\sigma_{\boldsymbol{t}}(t) \in D$. Let T' be their set and $T'' = \Delta_{\boldsymbol{t}}$ be the domain of $\sigma_{\boldsymbol{t}}$. Then $T' \subseteq T''$, because $\sigma_{\boldsymbol{t}}(t) \notin D$ for $t \notin \Delta_{\boldsymbol{t}}$ (see **I**.1.43 (c) in [13]). Since T' also is the domain of both $\sigma_{\langle\mathbf{l},\boldsymbol{t}\rangle}$ and $\sigma_{\langle\mathbf{r},\boldsymbol{t}\rangle}$, $\Delta_{\langle\mathbf{l},\boldsymbol{t}\rangle} = \Delta_{\langle\mathbf{r},\boldsymbol{t}\rangle}$, by (62) it is the domain of $\sigma_{\langle\langle\mathbf{l},\boldsymbol{t}\rangle,\langle\mathbf{r},\boldsymbol{t}\rangle\rangle}$, hence, the intersection required in (84) for (91).

For (92) it is enough to prove the former equivalence, as the latter comes from replacing **r** for **l**. Now, we replace $\sigma_{\boldsymbol{t}'}(t)$ for t' and $\sigma_{\boldsymbol{t}''}(t)$ for t'' in (31). Then, we use (14), (63) and (62) and get the intersection required in (84), immediately from (62). ■

4. Jumps and reduction

4.0. Definitions

At first, notice that (23) allows us to define two functions $p_1, p_2 \subseteq \boldsymbol{T} \times \boldsymbol{T}$ such that

(93) $$p_1(\boldsymbol{t}', \boldsymbol{t}''), p_1(\mathbf{l}, \boldsymbol{t}'), p_1(\mathbf{r}, \boldsymbol{t}') = \boldsymbol{t}' \text{ and } p_2(\boldsymbol{t}', \boldsymbol{t}'') = \boldsymbol{t}'' ,$$

for all $\boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$, which pick the first and possible second search term respectively in a composed search term. Trivially, (3) prevents any fixed point:

(94) $$p_1(\boldsymbol{t}), p_2(\boldsymbol{t}) \neq \boldsymbol{t} \quad \text{, for all } \ \boldsymbol{t} \in \boldsymbol{T} \ .$$

By such a notation we say that $\boldsymbol{t}', \boldsymbol{t}'' \in \boldsymbol{T}$ are *confluent,* when $\boldsymbol{t}' = \langle \mathbf{l}, p_1(\boldsymbol{t}')\rangle$ and $\boldsymbol{t}'' = \langle \mathbf{r}, p_1(\boldsymbol{t}')\rangle$, see the bottom of the diamond in Figure 3, and that $\boldsymbol{t} \in \boldsymbol{T}$ is a **d**-*term,* when $\boldsymbol{t} = \langle p_1(\boldsymbol{t}), p_2(\boldsymbol{t})\rangle$. (Confluence is antisymmetric.)

Then, by bold induction we can define a *reduction (function)*

$\rho{:}\, \boldsymbol{T} \to \boldsymbol{T}$ as

(95) $$\rho(\emptyset) = \emptyset \ ;$$

$$(96)\quad \rho(\boldsymbol{t}',\boldsymbol{t}'') = \begin{cases} p_1(\rho(\boldsymbol{t}')) & \text{when } \rho(\boldsymbol{t}') \text{ and } \rho(\boldsymbol{t}'') \text{ are confluent,} \\ \\ \langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle & \text{otherwise} \end{cases} \quad ;$$

$$(97)\quad \rho(\mathbf{l},\boldsymbol{t}) = \begin{cases} p_1(\rho(\boldsymbol{t})) & \text{when } \rho(\boldsymbol{t}) \text{ is a } \mathbf{d}\text{-term,} \\ \\ \langle \mathbf{l}, \rho(\boldsymbol{t}) \rangle & \text{otherwise} \end{cases} \quad ;$$

$$(98)\quad \rho(\mathbf{r},\boldsymbol{t}) = \begin{cases} p_2(\rho(\boldsymbol{t})) & \text{when } \rho(\boldsymbol{t}) \text{ is a } \mathbf{d}\text{-term,} \\ \\ \langle \mathbf{r}, \rho(\boldsymbol{t}) \rangle & \text{otherwise} \end{cases} \quad .$$

We also define the set A of the *jumps* or *reduced terms* as the minimal set such that

$$(99)\qquad W \subseteq A \qquad \text{and}\,,$$

$$(100)\qquad \text{if}\quad a', a'' \in A \;\text{ are not confluent, then } \langle a', a''\rangle \in A\ .$$

Clearly, A satisfies both conditions. We allow two different names for its elements, because of the next two different inclusions in 4.1 (B) and (C). Also, because of the latter inclusion, we will sometimes boldface the letters denoting them. When a jump is a **d**-term, we call it a *composed jump* and we denote the set of them by D'. The restriction of the premise in (100) only concerns a pair of words (atoms) a' and a'', since no composed jump can begin with **l** or **r**: for all $a', a'' \in A$

$$(101)\qquad \langle a', a''\rangle \notin A \;\text{ implies }\; a', a'' \in W\ .$$

Lemma 4.1 (E) is going to state a (full) induction principle for reduced terms that we call *reduced induction.* Since its basis step requires to check a superset of W and this in most cases involves a word induction or a backward one, it is not "reduced" at all: often it will require two induction steps.

4.1. Lemmata

(A) *(Binary) search words are unknowns for pairing.*

(B) *Jumps are two-ary natural numbers on (search) words: $A \subseteq \boldsymbol{N}_2(W)$.*

(C) *Reduced terms are search terms: $A \subseteq \boldsymbol{T}$.*

(D) *We can partition jumps into words and composed jumps: $A = W \cup D'$ and $W \cap D' = \emptyset$.*

(E) *For all sets A', $A' \supseteq A$ iff $W \subseteq A'$ and, for all $\boldsymbol{t}', \boldsymbol{t}'' \in A \cap A'$ that are not confluent, $\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \in A'$.*

(F) *The first and second search term of a jump are jumps: $p_1(a) \in A$, for $\emptyset \neq a \in A$, and $p_2(a) \in A$, for all $a \in D'$.*

(G) *The first and second search term of a composed jump cannot be confluent: $\langle a', a'' \rangle \in A$ and $a' = \langle \mathbf{l}, a \rangle$ imply $a'' \neq \langle \mathbf{r}, a \rangle$.*

(H) *For every reduced term $\boldsymbol{a} \in A$ there is a natural number n such that, for each word $\boldsymbol{w} \in W$ of a length not less than n, $\boldsymbol{a} \oplus \boldsymbol{w} \asymp \boldsymbol{v}$ for some word $\boldsymbol{v} \in W$ and, for every such a $\boldsymbol{v}$, $\boldsymbol{v} = \boldsymbol{a} \oplus \boldsymbol{w}$ iff $\boldsymbol{a} \in W$.*

(I) *The catenation of any reduced term $\boldsymbol{a} \in A$ with any word $w \in W$ is reduced, $w \oplus \boldsymbol{a} \in A$.*

Proofs.

(A) When in (4) $U = W$, the elements of T in (7) are finite sets, because both words and the pairs in $D = T \times T$ are. The latter pairs neither can be the empty word nor can have an infinite component as any other word does. In fact, such nonempty words are pairs $\boldsymbol{w} = \langle \mathbf{l}, \boldsymbol{v} \rangle$ or $\boldsymbol{w} = \langle \mathbf{r}, \boldsymbol{v} \rangle$, where both $\mathbf{l}$ and $\mathbf{r}$ are infinite sets.

(B) Since $T' = \boldsymbol{N}_2(W)$ satisfies (4), with $U = W$, and (5), it will also satisfy (99) and (100). Hence, $\boldsymbol{N}_2(W) \supseteq A$ by the minimality of A.

(C) Again , we use the minimality of A, but for (4)–(5) replaced by (17)–(18).

(D) Since $D' \subseteq \boldsymbol{N}_2(W) \times \boldsymbol{N}_2(W)$, $W \cap D' = \emptyset$ has been shown in (A). Trivially, $W \cup D' \subseteq A$. To see that $A' = W \cup D' \supseteq A$, since $W \subseteq A$, we merely consider $a', a'' \in A'$ and show that $\langle a', a'' \rangle \in D' \subseteq A'$, when they are not confluent. This follows from $\langle a', a'' \rangle$ being a $\mathbf{d}$–term that must belong to all sets containing W and closed under our restricted pairing.

(E) As for 1.3 (A), the (only if) comes from (99) and (100), while the (if) from $A' \cap A \subseteq A'$ being a superset of A.

(F) and (G) In the opposite cases the minimality of A fails, because one could delete the corresponding jumps.

(H) Use reduced induction on $\boldsymbol{a}$. For $\boldsymbol{a} \in W$ take $n = 0$. Hence, for each $\boldsymbol{w} \in W$, $\boldsymbol{a} \oplus \boldsymbol{w} \in W$ by 1.8 (E) and, by the reflexivity of $\asymp$, we get $\boldsymbol{a} \oplus \boldsymbol{w} \asymp \boldsymbol{v}$ with $\boldsymbol{v} = \boldsymbol{a} \oplus \boldsymbol{w} \in W$.

Let $\boldsymbol{a} = \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle$, where $\boldsymbol{a}', \boldsymbol{a}'' \in A$ (are not confluent and) have natural numbers n' and n'' such that, for all words $\boldsymbol{w}'$ of a length not less than n' and $\boldsymbol{w}''$ of a length not less than n'', $\boldsymbol{a}' \oplus \boldsymbol{w}' \asymp \boldsymbol{v}'$ and $\boldsymbol{a}'' \oplus \boldsymbol{w}'' \asymp \boldsymbol{v}''$ for some $\boldsymbol{v}', \boldsymbol{v}'' \in W$. Then, take $n = 1 + \max(n', n'')$.

Clearly, for every $\boldsymbol{w} \in W$ of length not less than n, there is a one-letter word $\boldsymbol{u} = \langle \mathbf{l}, \emptyset \rangle, \langle \mathbf{r}, \emptyset \rangle$ such that $\boldsymbol{w} = \boldsymbol{u} \oplus \boldsymbol{z}$ for some word $\boldsymbol{z} \in W$ of a length not less than both n' and n''. Hence, by (55) $\boldsymbol{a} \oplus \boldsymbol{w} = (\boldsymbol{a} \oplus \boldsymbol{u}) \oplus \boldsymbol{z}$, where by (38) and (36) $\boldsymbol{a} \oplus \boldsymbol{u} = \langle \mathbf{l}, \boldsymbol{a} \rangle, \langle \mathbf{r}, \boldsymbol{a} \rangle$. In both cases the induction premises get

$$\boldsymbol{a} \oplus \boldsymbol{w} = \begin{cases} \langle \mathbf{l}, \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle \rangle \oplus \boldsymbol{z} \asymp \boldsymbol{a}' \oplus \boldsymbol{z} \asymp \boldsymbol{v}' & \text{for some } \boldsymbol{v}' \in W \\ \\ \langle \mathbf{r}, \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle \rangle \oplus \boldsymbol{z} \asymp \boldsymbol{a}'' \oplus \boldsymbol{z} \asymp \boldsymbol{v}'' & \text{for some } \boldsymbol{v}'' \in W \end{cases}$$

by (92), namely $\boldsymbol{a} \oplus \boldsymbol{w} \asymp \boldsymbol{v}$ for some $\boldsymbol{v} \in W$.

(iff) When $\boldsymbol{a} \in W$, by 1.8 (E) $\boldsymbol{a} \oplus \boldsymbol{w} \in W$ and by 3.2 (C) $\boldsymbol{a} \oplus \boldsymbol{w} = \boldsymbol{v}$, which conversely gets $\boldsymbol{a} \in W$ by an easy word induction on $\boldsymbol{w}$: trivial for $\boldsymbol{w} = \emptyset$ while the induction step comes from (38) and (21).

(I) Reduced induction on $\boldsymbol{a}$. When $\boldsymbol{a} \in W$, by 1.8 (E) $w \oplus \boldsymbol{a} \in W \subseteq A$. When $\boldsymbol{a} = \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle$, where $\boldsymbol{a}', \boldsymbol{a}'' \in A$ are not confluent and $w \oplus \boldsymbol{a}', w \oplus \boldsymbol{a}'' \in A$, even $w \oplus \boldsymbol{a}'$ and $w \oplus \boldsymbol{a}''$ cannot be confluent. In fact, either both $\boldsymbol{a}', \boldsymbol{a}'' \in W$ and by (38) $w \oplus \boldsymbol{a}', w \oplus \boldsymbol{a}'' \in W$ respectively end as $\boldsymbol{a}'$ and $\boldsymbol{a}''$ do or some of them is composed and by (37) even its corresponding catenation with w does, contrary to (101). Therefore, by (37) and (100) $w \oplus \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle \in A$. ■

4.2. Theorems

Jumps are "canonical" two-ary integers, namely:

(A) *reduction preserves local equality: for all $\boldsymbol{t} \in \boldsymbol{T}$, $\boldsymbol{t} \asymp \rho(\boldsymbol{t})$;*

(B) *reduced terms are irreducible, $\rho \cdot \boldsymbol{i}_A = \boldsymbol{i}_A$, i.e. $\rho(\boldsymbol{a}) = \boldsymbol{a}$ for all $\boldsymbol{a} \in A$;*

(C) *conversely, if a search term is irreducible, $\rho(\boldsymbol{t}) = \boldsymbol{t}$, then it is reduced, $\boldsymbol{t} \in A$;*

(D) *all reductions of search terms are reduced terms and conversely: $\rho\colon \boldsymbol{T} \twoheadrightarrow A$;*

(E) *among reduced terms local equality is identity: for all* $\boldsymbol{a}, \boldsymbol{u} \in A$

$$\boldsymbol{a} \asymp \boldsymbol{u} \text{ implies } \boldsymbol{a} = \boldsymbol{u} \,. \tag{102}$$

Proofs.

(A) Use bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, it comes from (95), since $\asymp$ is reflexive.

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\boldsymbol{t}' \asymp \rho(\boldsymbol{t}')$ and $\boldsymbol{t}'' \asymp \rho(\boldsymbol{t}'')$, we set $\boldsymbol{u}' = \rho(\boldsymbol{t}')$ and $\boldsymbol{u}'' = \rho(\boldsymbol{t}'')$ in (86) and get $\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \asymp \langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle$. By (96) this immediately implies that $\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle \asymp \rho(\boldsymbol{t}', \boldsymbol{t}'')$, when $\rho(\boldsymbol{t}')$ and $\rho(\boldsymbol{t}''))$ are not confluent. When they are, $\langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle \asymp p_1(\rho(\boldsymbol{t}'))$ by (91). Hence, $\langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle \asymp \rho(\boldsymbol{t}', \boldsymbol{t}'')$ by (96) and transitivity.

When $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $\boldsymbol{t}' \asymp \rho(\boldsymbol{t}')$, we set $\boldsymbol{u} = \rho(\boldsymbol{t}')$ in (85) and gets $\langle \mathbf{l}, \boldsymbol{t}' \rangle \asymp \langle \mathbf{l}, \rho(\boldsymbol{t}') \rangle$. By (97) this immediately implies that $\langle \mathbf{l}, \boldsymbol{t}' \rangle \asymp \rho(\mathbf{l}, \boldsymbol{t}')$, when $\rho(\boldsymbol{t}')$ is not a **d**-term. When it is, $\langle \mathbf{l}, \rho(\boldsymbol{t}') \rangle \asymp p_1(\rho(\boldsymbol{t}'))$ by (92). Hence, $\langle \mathbf{l}, \boldsymbol{t}' \rangle \asymp \rho(\mathbf{l}, \boldsymbol{t}')$ by (97) and transitivity. Finally, we prove the step $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$ by replacing $\mathbf{l}$ with $\mathbf{r}$ (and p_1 with p_2) as usual.

(B) Consider the steps (96), (97) and (98). When $\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle, \langle \mathbf{l}, \boldsymbol{t} \rangle, \langle \mathbf{r}, \boldsymbol{t} \rangle \in A$, so are $\boldsymbol{t}'$, $\boldsymbol{t}''$ and $\boldsymbol{t}$ by 4.1 (F). Moreover, $\boldsymbol{t}'$ and $\boldsymbol{t}''$ cannot be confluent by 4.1 (G), while $\boldsymbol{t} \in W$ by 4.1 (D) and the minimalities of A and W. Hence. in all three steps the former case cannot occur and this restriction of ρ is an identity by a trivial reduced induction.

(C) Use bold induction. For $\boldsymbol{t} = \emptyset$, $\boldsymbol{t} \in A$, since $\emptyset \in W$. When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\rho(\boldsymbol{t}') = \boldsymbol{t}'$ implies $\boldsymbol{t}' \in A$ and $\rho(\boldsymbol{t}'') = \boldsymbol{t}''$ implies $\boldsymbol{t}'' \in A$, we use the premise in (C) and (94) to find that $\boldsymbol{t}'$ and $\boldsymbol{t}''$ cannot be confluent nor reducible. Then the premises in the induction premise and (100) to get $\boldsymbol{t} \in A$. The cases $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$ easily follow from 4.1 (D).

(D) After (B), we only have to prove $\rho : \boldsymbol{T} \to A$, namely that the reduction of a search term is a reduced term, $\rho(\boldsymbol{t}) \in A$ for all $\boldsymbol{t} \in \boldsymbol{T}$, by bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, $\rho(\boldsymbol{t}) = \emptyset \in A$ by (95), (19) for W and (99).

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \in A$, $\rho(\boldsymbol{t}')$ and $\rho(\boldsymbol{t}'')$ cannot be confluent, unless they are in W as in (101).

In the former circumstance, $\rho(\boldsymbol{t}) \in A$ by (100) and the latter case of (96). In the latter, it does by (99) and the former case of (96), since $\rho(\boldsymbol{t}') \in A$ implies $p_1(\rho(\boldsymbol{t}')) \in A$ by 4.1 (F).

When $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $\rho(\boldsymbol{t}') \in A$, either $\rho(\boldsymbol{t}') \in W$ or it is a **d**-term.

In the former circumstance, $\rho(\boldsymbol{t}) \in W \subset A$ by (99), (21) for W and the latter case of (97). In the latter, it does by the former case of (97), again because $\rho(\boldsymbol{t}') \in A$ implies $p_1(\rho(\boldsymbol{t}')) \in A$. A similar proof holds for $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$.

(E) The proof of (102) will use reduced induction on $\boldsymbol{a}$. Yet, at now we prove a lesser statement that has a proof useful both in its basis step and induction step. We only prove that *(102) holds for all $\boldsymbol{a} \in A$ and all $\boldsymbol{u} \in W$* and we do it by reduced induction on $\boldsymbol{a}$.

This basis step is for $\boldsymbol{a} \in W$ and 3.2 (C) provides it with the proof of (102). The induction step is $\boldsymbol{a} = \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle$ for some $\boldsymbol{a}'$ and $\boldsymbol{a}''$ such that, for all $\boldsymbol{u}', \boldsymbol{u}'' \in W$, $\boldsymbol{a}' \asymp \boldsymbol{u}'$ implies $\boldsymbol{a}' = \boldsymbol{u}'$, while $\boldsymbol{a}'' \asymp \boldsymbol{u}''$ implies $\boldsymbol{a}'' = \boldsymbol{u}''$. Then, by (85) $\boldsymbol{a} \asymp \boldsymbol{u}$ implies $\langle \mathbf{l}, \boldsymbol{a} \rangle \asymp \langle \mathbf{l}, \boldsymbol{u} \rangle$. By (92) this implies $\boldsymbol{a}' \asymp \langle \mathbf{l}, \boldsymbol{u} \rangle$, where $\boldsymbol{u}' = \langle \mathbf{l}, \boldsymbol{u} \rangle \in W$ by (21) for W. Hence, our former induction premise gets $\boldsymbol{a}' = \langle \mathbf{l}, \boldsymbol{u} \rangle$, while, after replacing $\mathbf{l}$ with $\mathbf{r}$, the latter gets $\boldsymbol{a}'' = \langle \mathbf{r}, \boldsymbol{u} \rangle$. This contradicts the confluence free assumption of 4.1 (G) and makes the implication, we are proving, trivially true, as there are not such pairs $\langle \boldsymbol{a}', \boldsymbol{a}'' \rangle$ in A.

We can now prove (102) with its full quantification. The basis step, for $\boldsymbol{a} \in W$, comes from the preceding proof after flipping $\boldsymbol{a}$ and $\boldsymbol{u}$, namely by a reduced induction on $\boldsymbol{u}$ nested in the present induction on $\boldsymbol{a}$. For the induction step, $\boldsymbol{a} = \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle$, the induction premises now hold for all $\boldsymbol{u}', \boldsymbol{u}'' \in A$. Hence, when $\boldsymbol{u}', \boldsymbol{u}'' \in W$, the proof is the one of the preceding induction step, while, when $\boldsymbol{u} = \langle \boldsymbol{v}', \boldsymbol{v}'' \rangle$ for some $\boldsymbol{v}', \boldsymbol{v}'' \in A$, we still get $\boldsymbol{a}' \asymp \langle \mathbf{l}, \boldsymbol{u} \rangle$ and $\boldsymbol{a}'' \asymp \langle \mathbf{r}, \boldsymbol{u} \rangle$. From this, by (92) $\boldsymbol{a}' \asymp \boldsymbol{v}'$ and $\boldsymbol{a}'' \asymp \boldsymbol{v}''$. Therefore, if we take $\boldsymbol{u}' = \boldsymbol{v}'$ and $\boldsymbol{u}'' = \boldsymbol{v}''$, then the induction premises get $\boldsymbol{a}' = \boldsymbol{v}'$ and $\boldsymbol{a}'' = \boldsymbol{v}''$ and by (2) $\boldsymbol{a} = \langle \boldsymbol{a}', \boldsymbol{a}'' \rangle = \langle \boldsymbol{v}', \boldsymbol{v}'' \rangle = \boldsymbol{u}$ (now, by a non trivial implication). ■

4.3. Corollary

Local equality is the equivalence induced by reduction. Hence, every two-ary integer has a single reduced term.

Proof.

(A) To show that, for all $\boldsymbol{t}, \in \boldsymbol{T}$, $\boldsymbol{t} \asymp \boldsymbol{u}$ implies $\rho(\boldsymbol{t}) = \rho(\boldsymbol{u})$, we first use 4.2 (A) and transitivity to get $\rho(\boldsymbol{t}) \asymp \rho(\boldsymbol{u})$. Then, we use 4.2 (D) and (E). To get the converse implication we merely use 4.2 (A) and transitivity. ■

4.4. Theorems

(A) *Reduced terms have the maximal domain property: for all* $\boldsymbol{t} \in \boldsymbol{T}$, $\Delta_{\boldsymbol{t}} \subseteq \Delta_{\rho(\boldsymbol{t})}$.

(B) *Conversely, for all* $\boldsymbol{t}, \boldsymbol{u} \in \boldsymbol{T}$, $\sigma_{\boldsymbol{t}} \subseteq \sigma_{\rho(\boldsymbol{u})}$ *implies that* $\boldsymbol{t} \asymp \boldsymbol{u}$. *Yet,* $\sigma_{\boldsymbol{t}} \subseteq \sigma_{\boldsymbol{u}}$ *for all* $\boldsymbol{t} \in \boldsymbol{T}$, *such that* $\boldsymbol{t} \asymp \boldsymbol{u}$, *does not imply that* $\boldsymbol{u} \in A$.

(C) *A set of S-terms is the domain of a reduced term iff it is cartesian.*

Proofs.

(A) Use bold induction on $\boldsymbol{t}$. When $\boldsymbol{t} = \emptyset$, it comes from (95) and (64).

When $\boldsymbol{t} = \langle \boldsymbol{t}', \boldsymbol{t}'' \rangle$ and both $\Delta_{\boldsymbol{t}'} \subseteq \Delta_{\rho(\boldsymbol{t}')}$ and $\Delta_{\boldsymbol{t}''} \subseteq \Delta_{\rho(\boldsymbol{t}'')}$, by (62) we get $\Delta_{\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle} = \Delta_{\boldsymbol{t}'} \cap \Delta_{\boldsymbol{t}''} \subseteq \Delta_{\rho(\boldsymbol{t}')} \cap \Delta_{\rho(\boldsymbol{t}'')} = \Delta_{\langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle}$. By (96) this immediately implies that $\Delta_{\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle} \subseteq \Delta_{\rho(\boldsymbol{t}', \boldsymbol{t}'')}$, when $\rho(\boldsymbol{t}')$ and $\rho(\boldsymbol{t}''))$ are not confluent. When they are, $\Delta_{\rho(\boldsymbol{t}')} = \Delta_{\rho(\boldsymbol{t}'')}$ by (65). Hence, $\Delta_{\langle \rho(\boldsymbol{t}'), \rho(\boldsymbol{t}'') \rangle} = \Delta_{\rho(\boldsymbol{t}')}$ by (62), which implies $\Delta_{\langle \boldsymbol{t}', \boldsymbol{t}'' \rangle} \subseteq \Delta_{\rho(\boldsymbol{t}', \boldsymbol{t}'')}$, because $\Delta_{\rho(\boldsymbol{t}')} \subseteq \Delta_{p_1(\rho(\boldsymbol{t}'))} = \Delta_{\rho(\boldsymbol{t}', \boldsymbol{t}'')}$ by (65) and (96).

When $\boldsymbol{t} = \langle \mathbf{l}, \boldsymbol{t}' \rangle$ and $\Delta_{\boldsymbol{t}'} \subseteq \Delta_{\rho(\boldsymbol{t}')}$, we get $\Delta_{\langle \mathbf{l}, \boldsymbol{t}' \rangle} \subseteq \Delta_{\langle \mathbf{l}, \rho(\boldsymbol{t}') \rangle}$ by (65) and (63). By (97) this immediately implies that $\Delta_{\langle \mathbf{l}, \boldsymbol{t}' \rangle} \subseteq \Delta_{\rho(\mathbf{l}, \boldsymbol{t}')}$, when $\rho(\boldsymbol{t}')$ is not a $\mathbf{d}$-term. When it is, $\Delta_{\rho(\boldsymbol{t}')} = \Delta_{p_1(\rho(\boldsymbol{t}'))} \cap \Delta_{p_2(\rho(\boldsymbol{t}'))} \subseteq \Delta_{p_1(\rho(\boldsymbol{t}'))}$ by (93) and (62). Hence, $\Delta_{\boldsymbol{t}'} \subseteq \Delta_{p_1(\rho(\boldsymbol{t}')} = \Delta_{\rho(\mathbf{l}, \boldsymbol{t}')}$ by transitivity and (97). By (65) and transitivity $\Delta_{\boldsymbol{t}} = \Delta_{\langle \mathbf{l}, \boldsymbol{t}' \rangle} \subseteq \Delta_{\boldsymbol{t}'} \subseteq \Delta_{\rho(\mathbf{l}, \boldsymbol{t}')} = \Delta_{\rho(\boldsymbol{t})}$. Finally, we prove the step $\boldsymbol{t} = \langle \mathbf{r}, \boldsymbol{t}' \rangle$ by the usual replacements.

(B) $\sigma_{\boldsymbol{t}} \subseteq \sigma_{\rho(\boldsymbol{u})}$ implies $\boldsymbol{t} \asymp \rho(\boldsymbol{u})$ as in (84). Then, by 4.2 (A) $\boldsymbol{t} \asymp \boldsymbol{u}$. (Yet) Take $\boldsymbol{u} = \langle \mathbf{l}, \langle \langle \mathbf{l}, \emptyset \rangle, \emptyset \rangle \rangle$ and $\boldsymbol{a} = \langle \mathbf{l}, \emptyset \rangle = \rho(\boldsymbol{u})$. Then, $\boldsymbol{u} \asymp \boldsymbol{a}$ by 4.3, $\Delta_{\boldsymbol{u}} = D = \Delta_{\boldsymbol{a}}$ and $\boldsymbol{a} \in A$, whereas $\boldsymbol{u} \notin A$.

(C) (if) To see that every $X(\boldsymbol{t})$ is a Δ_a for some $a \in A$ use upward induction on $\boldsymbol{t}$. For $\boldsymbol{t} = \emptyset$, by (76) and (61) $X(\boldsymbol{t}) = T = \Delta_\emptyset$ and $\emptyset \in W \subseteq A$.

When $\boldsymbol{t} = \langle t, u \rangle$, where $X(t) = \Delta_a$ and $X(u) = \Delta_b$ for some $a, b \in A$, we set $a' = \langle \mathbf{l}, \emptyset \rangle \oplus a$ and $b' = \langle \mathbf{r}, \emptyset \rangle \oplus b$ and get $a', b' \in A$ again by 4.1 (I). By (23) they cannot be confluent unless $a = b = \emptyset$ and in such a case by (61) and (62) $\Delta_{\boldsymbol{t}} = T \cap T = T = \Delta_\emptyset$ again.

In all other cases $\langle a', b' \rangle \in A$ still provides $X(\boldsymbol{t})$ with the required induced domain: $\Delta_{\langle a', b' \rangle} = \Delta_{a'} \cap \Delta_{b'} = (\Delta_a \times T) \cap (T \times \Delta_b) = \Delta_a \times \Delta_b = X(t) \times X(u) = X(t, u)$ by (62), 2.3 (F), the inclusions $\Delta_a, \Delta_b \subseteq T$ and (77).

(Only if) Use reduced induction on the reduced term. The basis comes from 2.7 (B). The induction step comes from (62) and 2.6 (A). ■

5.5. Cartesian sets

At a first glance, two-ary integers look clumsier than unary integers, which can be positive, null or negative. As 5.1 (A) and (B) will detail, also in A we can distinguish three disjoint sets or *genders:* the singleton set of the zero $\{\emptyset\} = \{0\}$, the *null jump,* with the same zero of (set-theoretical) natural numbers, the set of *negative jumps,* $A^- = W \setminus \{0\}$, and the set of *positive jumps* $A^+ = \boldsymbol{N}_2(1) \setminus \{0\}$. Yet, contrary to usual integers, it is not all: e.g. $\langle\langle \mathbf{1}, \emptyset\rangle, \emptyset\rangle$ does not belong to any of the above genders. Now, we have a fourth gender: the set $A^\pm = A \setminus (\{0\} \cup A^- \cup A^+)$.

On the contrary, the theorems we just proved hint that from a semantic point of view things are not so different. By (64) positive and null jumps have the whole T as the induced domain like positive and null integers, which have the whole set of natural numbers. Now, we can say that the other jumps behave much like negative integers.

In fact, the induced domain of any negative integer is a cofinite segment of natural numbers. Such sets form a semilattice with zero, once we take $\supseteq$ as the order among them. We take it, instead of $\subseteq$, because we can identify each cofinite segment by its (finite) lowest number and such a containment becomes (directly) isomorphic to the natural order $\leq$ of natural numbers.

In case of jumps, 4.4 (C) and 2.7 (A) provide our induced domains with a semilattice, now of cartesian sets, with zero. By 2.6 (B) a cartesian set is infinite, like the cofinite segments, and 2.6 (C) always provides it again with a finite two-ary natural number as an identifier. By 2.7 (C) the age relation is the natural order among two-ary natural numbers, isomorphic to the semilattice on cartesian sets.

Furthermore, the induced domains of jumps, canonical representatives of two-ary integers, are maximal by 4.4 (A), like the ones of canonical integers in 0.2. For instance, the induced domain of $\langle -1, \emptyset\rangle$, corresponding to -1, is the segment $[1, \infty]$, whereas the one of its equivalent $\langle +1, \langle -1, \langle -1, \emptyset\rangle\rangle\rangle$ is its subset $[2, \infty]$.

This also shows that within usual integers the effective semantic monoid is not the functional one: the composition of the semantics of $\langle -1, \langle -1, \emptyset\rangle\rangle$ and the one of $\langle +1, \emptyset\rangle$ has the smaller domain $[2, \infty]$, whereas the bigger $[1, \infty]$ is the one we use, e.g. when finding the solution of equation $1 + x = 0$ "semantically". This prompt us to introduce a new semantic monoid in the next definition, by discarding any semantics that is not maximal and — accordingly — by rounding the compositions up.

4.6. Definitions

Consider the set $S' \subseteq \Sigma$ of all *maximal semantics,* namely such that $\sigma \cdot \rho \colon \boldsymbol{T} \twoheadrightarrow S'$, and by 4.2 (B) define the *prime semantics* as the function $s' \colon A \to S'$ we get by restricting the semantic function to jumps, $s' = \sigma \cdot \boldsymbol{i}_A$. By 4.4 (A), (B) and 4.3 inclusion of a semantics in a maximal semantics is a function $\varrho \colon \Sigma \twoheadrightarrow S'$: for all $f \in \Sigma$ and $g \in S'$, $f \subseteq g$ iff $\varrho(f) = g$. This defines the *rounded composition* $\odot \colon S' \times S' \to S'$ between two maximal semantics by

$$g'' \odot g' = \varrho(g'' \cdot g') \ \text{ for all } \ g', g'' \in S' \ . \tag{103}$$

Therefore,

$$g'' \cdot g' \subseteq g'' \odot g' \ , \ \text{for all } \ g', g'' \in S' \ , \tag{104}$$

and $g'' \odot g'$ is the only maximal semantics satisfying it.

By 4.7 (B) rounded composition is associative and has the unit $\boldsymbol{i}_T \in S'$ by (61) and (95). Hence, they form a monoid that we call the *prime semantic monoid (for* the set of atoms U).

4.7. Corollary

(A) *Every prime semantics is a bijection from jumps onto all maximal semantics,* $s' \colon A \rightarrowtail\!\!\!\!\twoheadrightarrow S'$.

(B) *For all* $f', f'' \in \Sigma$, $\varrho(f'' \cdot f') = \varrho(\varrho(f'') \cdot f') = \varrho(f'' \cdot \varrho(f'))$. *Hence, rounded composition is associative,* $(g'' \odot g') \odot g = g'' \odot (g' \odot g)$ *for all* $g, g', g'' \in S'$.

Proofs.

(A) 4.2 (D) and 4.4 (A) prove this ontoness. To check that s' is one to one, consider any $\boldsymbol{a}, \boldsymbol{u} \in A$. If $s'_{\boldsymbol{a}} = s'_{\boldsymbol{u}}$, then $\sigma_{\boldsymbol{a}} = \sigma_{\boldsymbol{u}}$ and $\boldsymbol{a} \asymp \boldsymbol{u}$ as in 3.0. Hence, by 4.2 (E) $\boldsymbol{a} = \boldsymbol{u}$.

(B) From 2.2 (B) and the notion of functional (or relational) composition we immediately get $\Sigma \ni f'' \cdot f' \subseteq \varrho(f'') \cdot f', f'' \cdot \varrho(f') \in \Sigma$ for all $f', f'' \in \Sigma$. Since the inclusion ϱ is a function, this implies $\varrho(f'' \cdot f') = \varrho(\varrho(f'') \cdot f') = \varrho(f'' \cdot \varrho(f'))$. Hence, $(g'' \odot g') \odot g = \varrho(\varrho(g'' \cdot g') \cdot g) = \varrho((g'' \cdot g') \cdot g) = \varrho(g'' \cdot (g' \cdot g)) = \varrho(g'' \cdot \varrho(g' \cdot g)) = g'' \odot (g' \odot g)$. ■

5. The prime representation of the jump algebra

5.0. Definition
The property in 4.2 (D) allows us to introduce the following operations on jumps. The *sum of* jump a'' *and* jump a' is the jump

(105) $$a' + a'' = \rho(a' \oplus a'') \ .$$

Hence, we defined a binary operation $+\colon A \times A \to A$ that we call *jump sum.* From it we also get the *jump increment function* $\eta'\colon A \to A^A$ by

(106) $$\eta'_{a'}(a'') = a' + a'', \text{ for all } a', a'' \in A \ .$$

The *jump cons* too is a binary operation $\alpha_{\mathbf{d}}\colon A \times A \to A$ that we define by

(107) $$\alpha_{\mathbf{d}}(a', a'') = \rho(a', a'') \ , \text{ for all } a', a'' \in A \ ,$$

whereas the *jump car* and *jump cdr* are unary, $\alpha_{\mathbf{l}}, \alpha_{\mathbf{r}}\colon A \to A$, and are respectively defined by

(108) $$\alpha_{\mathbf{l}}(a) = \rho(\mathbf{l}, a) \text{ and } \alpha_{\mathbf{r}}(a) = \rho(\mathbf{r}, a) \ , \text{ for all } a \in A \ .$$

Since it might occur that $\alpha_{\mathbf{d}}(a', a'') \neq \langle a', a''\rangle$, $\alpha_{\mathbf{l}}(a) \neq \langle \mathbf{l}, a\rangle$ or $\alpha_{\mathbf{r}}(a) \neq \langle \mathbf{r}, a\rangle$, one cannot properly call such operations jump dyad, jump left or jump right respectively. Even the LISP keywords, we are using, lack this propriety. Yet, since they are less usual, this lessens. Anyway, we call the function α we just defined the *(finite dimensional) jump algebra.* Comment 5.6 will motivate why we are discarding the notion of an algebra, $\langle A, \alpha\rangle$.

Given a set B, we also consider another function β on the domain $\{\mathbf{l}, \mathbf{r}, \mathbf{d}\}$ indexing three operations $\beta_{\mathbf{l}}, \beta_{\mathbf{r}}\colon B \to B$ and $\beta_{\mathbf{d}}\colon B \times B \to B$ and we say that a function $h\colon A \to B$ is a *homomorphism from α into β*, when for all $a, a' \in A$

(109) $$h(\alpha_{\mathbf{d}}(a, a')) \ = \ \beta_{\mathbf{d}}(h(a)), h(a')) \ ,$$

(110) $$h(\alpha_{\mathbf{l}}(a)) = \beta_{\mathbf{l}}(h(a)) \quad \text{and} \quad h(\alpha_{\mathbf{r}}(a)) = \beta_{\mathbf{r}}(h(a)) \ .$$

$H_{\alpha\beta}$ will denote the set of such h. When $\beta = \alpha$, $\mathcal{E}_\alpha$ will replace $H_{\alpha\alpha}$ and we call any h an *endomorphism of* α or a *jump endomorphism.*

We say that β is *binary invertible,* when the operations it indexes *always* satisfy the binary inversion equations partially satisfied in 2.3, namely when for all $b, b' \in B$ they satisfy the identities, due to B. Jónsson and A. Tarski [8]:

$$\beta_{\mathbf{d}}(\beta_{\mathbf{l}}(b), \beta_{\mathbf{r}}(b)) = b \quad \text{and} \tag{111}$$

$$\beta_{\mathbf{l}}(\beta_{\mathbf{d}}(b, b')) = b = \beta_{\mathbf{r}}(\beta_{\mathbf{d}}(b', b)) \; . \tag{112}$$

Then, for each β, we define $r' \colon \mathsf{H}_{\alpha\beta} \to B$ by

$$r'(h) = h(\emptyset) \; , \quad \text{for all} \;\; h \in \mathsf{H}_{\alpha\beta} \; . \tag{113}$$

As 5.4 (B) will show, when $\beta = \alpha$, r' is one of the representations of $\mathcal{E}_\alpha$. Hence, we call it the *prime (analytic) representation of* (the endomorphisms of) the jump algebra.

5.1. Lemmata

(A) *For all $v, w \in W$, $\alpha_{\mathbf{l}}(w) = \langle \mathbf{l}, w \rangle \in W$, $\alpha_{\mathbf{r}}(w) = \langle \mathbf{r}, w \rangle \in W$ and $w + v = w \oplus v \in W$. Hence, jump sum is an extension of word catenation.*

(B) *Whenever $a, a' \in A$ are not confluent, $\alpha_{\mathbf{d}}(a, a') = \langle a, a' \rangle$. Hence, $\alpha_{\mathbf{d}}(\boldsymbol{u}, \boldsymbol{t}) = \langle \boldsymbol{u}, \boldsymbol{t} \rangle$ for all $\boldsymbol{u}, \boldsymbol{t} \in \boldsymbol{N}_2(1)$.*

(C) *For all $\boldsymbol{u}, \boldsymbol{t} \in \boldsymbol{N}_2(1) \subseteq A$, $\boldsymbol{t} + \boldsymbol{u} = \mathrm{K}_{\boldsymbol{t}} \hat{+} \boldsymbol{u} = g_{\boldsymbol{u}}(\boldsymbol{t})$, where $\hat{+}$ and g are on $T = \boldsymbol{N}_2(1)$ and $\mathrm{K} \colon T \mapsto\!\!\!\twoheadrightarrow T^U$ for $U = 1$. Namely, $\mathrm{K}^{-1} \colon T^1 \mapsto\!\!\!\twoheadrightarrow T$, with $T = \boldsymbol{N}_2(1) \subseteq A$, extends term increments into jump increments, $\hat{\eta} = \eta' \cdot \mathrm{K}^{-1}$.*

(D) *We can perform jump car and cdr by sums: $\alpha_{\mathbf{l}}(a) = a + \langle \mathbf{l}, \emptyset \rangle$ and $\alpha_{\mathbf{r}}(a) = a + \langle \mathbf{r}, \emptyset \rangle$, for all $a \in A$.*

Proofs.

(A) By 4.1 (I) every $w = \emptyset + w \in W$ is reduced and is not a $\mathbf{d}$–term. Hence, by (97) and (98) $\alpha_{\mathbf{l}}(w) = \rho(\mathbf{l}, w) = \langle \mathbf{l}, w \rangle$ and $\alpha_{\mathbf{r}}(w) = \rho(\mathbf{r}, w) = \langle \mathbf{r}, w \rangle$, while from this an easy word induction on v and 1.8 (E) get $w + v = w \oplus v \in W$ by (36) and (38).

(B) It follows from 4.2 (B) and (96).

(C) By 1.9 (E) it is enough to show that $\boldsymbol{t} + \boldsymbol{u} = \boldsymbol{t} \oplus \boldsymbol{u}$. The trivial upward induction on $\boldsymbol{u}$ that uses (B) as induction step proves it.

(D) By (108), (38), (36) and (105) we get $\alpha_{\mathbf{l}}(a) = \rho(\mathbf{l}, a) = \rho(a \oplus \langle \mathbf{l}, \emptyset \rangle) = a + \langle \mathbf{l}, \emptyset \rangle$ and similarly for $\alpha_{\mathbf{r}}(a) = a + \langle \mathbf{r}, \emptyset \rangle$. ■

5.2. Theorem

α is binary invertible.

Proof. One might derive this property from the similar one of two-ary integers in 3.4. Yet, we prefer to provide it with a direct proof.

By (107) and (108) to prove $\alpha_{\mathbf{d}}(\alpha_{\mathbf{l}}(a), \alpha_{\mathbf{r}}(a)) = a$ is to prove $\rho(\rho(\mathbf{l}, a), \rho(\mathbf{r}, a)) = a$. To get this for all $a \in A$, by 4.1 (D) we can prove it for all $a \in W$ and for all $a \in D'$. When $a \in W$, we get $\rho(\rho(\mathbf{l}, a), \rho(\mathbf{r}, a)) = \rho(\langle \mathbf{l}, a \rangle, \langle \mathbf{r}, a \rangle) = p_1(\mathbf{l}, a) = a$ by 4.2 (B), the latter cases of (97) and (98), (B) again, the former case of (96) and by (93). When $a \in D'$, by 4.2 (B), the former cases of (97) and (98) and by (93) we get $\rho(\rho(\mathbf{l}, a), \rho(\mathbf{r}, a)) = \rho(p_1(a), p_2(a)) = \rho(a) = a$.

By (108) and (107) to prove $\alpha_{\mathbf{l}}(\alpha_{\mathbf{d}}(a, a')) = a$ is to prove $\rho(\mathbf{l}, \rho(a, a')) = a$. To get this for all $a \in A$, by 4.1 (D) we can first consider the case of confluent a and a'. In such a case $a, a' \in W$ and $a = \langle \mathbf{l}, p_1(a) \rangle$. Therefore, by 4.2 (B) and the former case of (96) we get $\rho(\mathbf{l}, \rho(a, a')) = \rho(\mathbf{l}, p_1(a)) = \rho(a) = a$. Then, we consider the case of non confluent a and a'.

In such a case $\langle a, a' \rangle \in D' \subseteq A$ and by the latter case of (96), 4.2 (B), the former cases of (97) and by (93) we get $\rho(\mathbf{l}, \rho(a, a')) = \rho(\mathbf{l}, \langle a, a' \rangle) = p_1(a, a') = a$.

We finally prove $\alpha_{\mathbf{r}}(\alpha_{\mathbf{d}}(a', a)) = a$ by the usual replacements, including the ones for the p_i. ■

5.3. Servi's clans

Lemma 5.1 (A) shows that jumps extend words both for the operations of affixing a letter and for catenation. Such an extension considers words as "negative" elements as in 4.5, contrary to Servi's approach, recalled in 0.0, which got clans from "nonnegative words".

Such opposite considerations lead to different properties of jumps and clans. Non-isomorphic clans are uncountably infinite (yet their partial orders are isomorphic), whereas 5.7 will show that we have a single jump algebra up to isomorphisms. The three unary operations of clans satisfy (a disjunction of) two systems of equations, whereas here we exactly got the single Jónsson–Tarski's system in (111) and (112).

In fact, while the last theorem shows that the jump algebra α is binary invertible, the next one will show that it is the most general among the binary invertible algebras. It will state that for each element b of a binary invertible algebra β there exists a homomorphism h from α into β that sends the empty jump to b, $h(\emptyset) = b$. Namely, α is a free algebra over the binary invertible class. At now, it is with the empty jump as a single free generating element. This restriction will disappear in 5.5.

5.4. Theorems

(A) *When β is binary invertible, $r'\colon \mathsf{H}_{\alpha\beta} \rightarrowtail\!\!\!\twoheadrightarrow B$.*

(B) *When $\beta = \alpha$, the jump increment function is the inverse of the prime representation $r'\colon \mathcal{E}_\alpha \rightarrowtail\!\!\!\twoheadrightarrow A$ and the image under such a representation of jump endomorphism composition is jump sum. Hence, $\eta'\colon A \rightarrowtail\!\!\!\twoheadrightarrow \mathcal{E}_\alpha$ and for all $a, b, c \in A$*

$$(c + b) + a = c + (b + a) \quad \textit{and} \tag{114}$$

$$\emptyset + a = a + \emptyset = a\ . \tag{115}$$

Proofs.

(A) (1–1) Assume $r'(h) = r'(k)$ for $h, k \in \mathsf{H}_{\alpha\beta}$, namely $h(\emptyset) = k(\emptyset)$. Then, by word induction $h(w) = k(w)$ for all $w \in W \subseteq A$.

In fact, when $w = \langle \mathbf{l}, v\rangle$ with $h(v) = k(v)$, by 5.1 (A) and (110) $h(w) = h(\alpha_{\mathbf{l}}(v)) = \beta_{\mathbf{l}}(h(v)) = \beta_{\mathbf{l}}(k(v)) = k(\alpha_{\mathbf{l}}(v)) = k(w)$ and we can rewrite the same for $w = \langle \mathbf{r}, v\rangle$.

This provides reduced induction with a base to prove $h(a) = k(a)$ for all $a \in A$. When $a = \langle a', a''\rangle$ for non confluent $a'a'' \in A$ with $h(a') = k(a')$ and $h(a'') = k(a'')$, by 5.1 (B) and (109) $h(a) = h(\alpha_{\mathbf{d}}(a', a'')) = \beta_{\mathbf{d}}(h(a'), h(a'')) = \beta_{\mathbf{d}}(k(a'), k(a'')) = k(\alpha_{\mathbf{d}}(a', a'')) = k(a)$.

(A) (onto) Given any $b \in B$, define an $h\colon A \to B$ by reduced induction as

$$h(\emptyset) = b\ ,$$

$$h(\mathbf{l}, v) = \beta_{\mathbf{l}}(h(v))\ , \qquad h(\mathbf{r}, v) = \beta_{\mathbf{r}}(h(v)) \ \text{ for all } \ v \in W \tag{116}$$

$$\text{and } \ h(a, a') = \beta_{\mathbf{d}}(h(a), h(a')) \text{ for all } \langle a, a'\rangle \in D'. \tag{117}$$

Then, we only have to prove (110) and (109).

When a is a composed jump, $a = \langle a', a'' \rangle$, by (108), (97) and 4.1 (F) $h(\alpha_{\mathbf{l}}(a)) = h(a')$, as $a' = p_1(a) \in A$.

Then, by (112), where $b = h(a')$ and $b' = h(a'')$, 4.1 (G) and (117) $h(\alpha_{\mathbf{l}}(a)) = \beta_{\mathbf{l}}(\beta_{\mathbf{d}}(h(a'), h(a''))) = \beta_{\mathbf{l}}(h(a', a'')) = \beta_{\mathbf{l}}(h(a))$. When a is a word, by (108), 5.1 (A), (97) and (116) we immediately get $h(\alpha_{\mathbf{l}}(a)) = h(\mathbf{l}, a) = \beta_{\mathbf{l}}(h(a))$. Then, by replacing $\mathbf{l}$ with $\mathbf{r}$ as usual, we get all of (110).

To get (109), first consider the case of confluent $a, a' \in A$, which implies $a, a' \in W$. By (107), (96), (111) and (116) $h(\alpha_{\mathbf{d}}(a, a')) = h(p_1(a)) = \beta_{\mathbf{d}}(\beta_{\mathbf{l}}(h(p_1(a))), \beta_{\mathbf{r}}(h(p_1(a)))) = \beta_{\mathbf{d}}(h(\mathbf{l}, p_1(a))), h(\mathbf{r}, p_1(a'))) = \beta_{\mathbf{d}}(h(a)), h(a'))$, since $p_1(a) = p_1(a')$. Then, for the case of non confluent $a, a' \in A$, by (107), 5.1 (B), (96) and (117) we immediately get $h(\alpha_{\mathbf{d}}(a, a')) = h(a, a') = \beta_{\mathbf{d}}(h(a), h(a'))$.

(B) (inverse) By (A) we know that $r' \colon \mathcal{E}_\alpha \Vdash\!\!\twoheadrightarrow A$. Hence, we only show that for all $b \in A$ the endomorphism h in (A) (onto), such that $r'(h) = b$, for $\beta = \alpha$ is η'_b. By (106) this is to show that

$$h(a) = h(\emptyset) + a = b + a \quad \text{for all} \quad a \in A \tag{118}$$

and we do by reduced induction on a. The lowest basis step follows from (105), (36) and 4.2 (B): $b + \emptyset = \rho(b \oplus \emptyset) = \rho(b) = b = h(\emptyset)$.

To get $b + \langle \mathbf{l}, v \rangle = h(\mathbf{l}, v)$ from $b + v = h(v)$ for all $v \in W$, start from $b \oplus v \asymp \rho(b \oplus v)$ as in 4.2 (A) and get $\rho(\mathbf{l}, b \oplus v) = \rho(\mathbf{l}, \rho(b \oplus v))$ by (85) and 4.3. Then, by (105), (108), (38) and (116) (with $\beta = \alpha$) get $b + \langle \mathbf{l}, v \rangle = \rho(b \oplus \langle \mathbf{l}, v \rangle) = \rho(\mathbf{l}, b \oplus v) = \rho(\mathbf{l}, \rho(b \oplus v)) = \rho(\mathbf{l}, b + v) = \alpha_{\mathbf{l}}(b + v) = \alpha_{\mathbf{l}}(h(v)) = h(\mathbf{l}, v)$. The same for $b + \langle \mathbf{r}, v \rangle = h(\mathbf{r}, v)$.

For the remaining step, $h(a, a') = b + \langle a, a' \rangle$ from two premises as in (118), we first get $\rho(\rho(b \oplus a), \rho(b \oplus a')) = \rho(b \oplus a, b \oplus a')$ as before. Then, $h(a, a') = \alpha_{\mathbf{d}}(h(a), h(a')) = \alpha_{\mathbf{d}}(b + a, b + a') = \rho(b + a, b + a') = \rho(\rho(b \oplus a), \rho(b \oplus a')) = \rho(b \oplus a, b \oplus a') = \rho(b \oplus \langle a, a' \rangle) = b + \langle a, a' \rangle$, follows from (117) (with $\beta = \alpha$), (107), (105) and (37).

(B) (jump sum) Finally, we get $r'(h'') + r'(h')$ as the representation $r'(h'' \cdot h')$ of composition, for all $h'', h' \in \mathcal{E}_\alpha$, merely by (118). In fact, by (113) this is to prove that $h''(\emptyset) + h'(\emptyset) = h''(h'(\emptyset))$ and, if we set $a = h'(\emptyset)$ and $h = h''$ in (118), we find that we already proved it by reduced induction on $h'(\emptyset)$. Since jump sum is isomorphic to the composition of endomorphisms, (114) and (115) are trivial. Anyway, we can get them as in 1.8 (A). ∎

5.5. Jump Arithmetics

The finding that the jump algebra is the free binary invertible algebra is a test for semantic constructions. In fact, both the problem of transforming binary (or LISP) trees and the latter algebra are known since half century, without any acknowledgment that the problem solution is this algebra.

Early LISP versions [29] embodied some embryonic jumps (the extensions of car and cdr to the primitives corresponding to our words) and merely observed the partial identities in 1.3 (B), forerunners of the Jónsson–Tarski's ones. Yet, later on [28] this was relinquished and no free algebra appeared.

Conversely, since [8] appeared, this free algebra was the subject of an extensive work, e.g. see [26], [5], [27] and [10]. Yet, they neither perceived its relevance to binary trees nor developed its arithmetic.

Here, the sum of jumps begins such a development. Such an operation comes from 5.4 (B), which is an instance of the general theory of Universal Matrices, recalled in 0.3. Since we are only concerned with semantic constructions, we will not develop it any further.

From the recalled extensive work, however, Universal Matrices provide us a preview of such arithmetics. E. Marczewski found that this free algebra has bases with every finite positive number of generators, see §31 in [6]. This implies that it is free with any such number of generators. It also implies that for each such a base we have a product of "(square) universal matrices" with as many "columns" as the generators.

Such products define analytic monoids of any dimension as in [20]. Every analytic monoid defines an "arithmetic" for jumps. Some of such arithmetics will collapse into a single one up to jump "transformations". For instance, with a singleton base its arithmetic is the one of our jump sum, because the isomorphism sending its generator to the empty jump provides its (one-dimensioned) analytic monoid with the required transformation.

Such collapsing transformations will generate a hierarchy of arithmetics. We do not yet know neither its structure (the theory of Universal Transformations that can enlighten it has yet to appear in print) nor the single arithmetics.

Still, two-ary integers promise to be arithmetically rich, in spite of their difference from usual integers.

5.6. Algebras as pairs

Many textbooks of Algebra and Universal Algebra denote *and define* a (homogeneous) algebra as a pair $\langle A, F\rangle$, where A denotes its carrier set and F denotes either a set of operations on A or a function indexing them. (The reason to call it “function”, instead of “family”, is in 0.6 of [20].) Our algebras, on the contrary, are not pairs: only an F occurs, in the form of an indexing function α or β, see also τ' in 1.1.

None of such textbooks motivates such pairs, while the set-theoretical point of view in 7.6 of [14] criticize them. Anyway, the theory we are using (the one of analytic monoids) does not need them.

In general, most (homogeneous total) operations determines their carrier, e.g. for $\alpha_{\mathbf{1}}\colon A \to A$ the carrier is Dom $\alpha_{\mathbf{1}}$. As far as we are concerned, this always makes our algebras equivalent to conventional algebras.

In fact, the only exception is an algebra where F contains (or indexes) nullary operations only. Then, given such an F, we have a conventional algebra $\langle A, F\rangle$ for every superset A of the set B of the corresponding constant values, whereas here we have just one algebra F. However, when we say that our algebra F has any such superset A as a carrier, such a *statement* provides us the conventional specification.

5.7. Definitions

By 5.4 (B) the sum and $\emptyset$ form a monoid that we call the *prime analytic monoid.* (Recall that by 5.0 in this monoid the composition of $\boldsymbol{u}$ with $\boldsymbol{a}$ is $\boldsymbol{a}+\boldsymbol{u}$, see also [18].) On the contrary, we call *sum monoid* the reversed one where the composition of $\boldsymbol{u}$ with $\boldsymbol{a}$ is $\boldsymbol{u}+\boldsymbol{a}$.

5.8. Corollary

The prime semantics s' is an isomorphism from the sum monoid onto the prime semantic monoid. Hence, $s' \cdot r'$ is an isomorphism from the jump endomorphism monoid onto the prime semantic monoid, up to a composition reversal.

Proof.

By 4.7 (A) $s'\colon A \rightarrowtail\!\!\!\twoheadrightarrow S'$. For all $\boldsymbol{a}, \boldsymbol{u} \in A$, $S' \ni s'_{\boldsymbol{a}+\boldsymbol{u}} = \sigma_{\boldsymbol{a}+\boldsymbol{u}} = \sigma_{\rho(\boldsymbol{a}\oplus\boldsymbol{u})} \supseteq \sigma_{\boldsymbol{a}\oplus\boldsymbol{u}} = \sigma_{\boldsymbol{u}} \cdot \sigma_{\boldsymbol{a}} \in \Sigma$, by (105), 4.2 (A), 4.4 (A) and 2.2 (B). Hence, $s'_{\boldsymbol{a}+\boldsymbol{u}} = \varrho(s'_{\boldsymbol{u}} \cdot s'_{\boldsymbol{a}}) = s'_{\boldsymbol{u}} \odot s'_{\boldsymbol{a}}$ as in 4.6. Finally, $s'_{\emptyset} = \sigma_{\emptyset} = \boldsymbol{i}_T$ by (61). Therefore, s' is the required isomorphism and by 5.4 (B) also $s' \cdot r'$ is, up to the reversal in 5.7. ■

Acknowledgments

A referee of this journal improved the language, the style and even the mathematical content of the present version. K. Głazek (T.U. of Zielona Góra), S. Martini (U. of Udine) and M. Servi (U. of Parma) provided the Author with interesting references and helpful talks. Thirty six years ago G. Degli Antoni (U. of Milan) was the first who told the Author about LISP.

References

[1] G.J. Chaitin, *Algorithmic Information Theory,* IBM J. Res. Develop. **21** (1977), 350–359, 496.

[2] G.J. Chaitin, Information-Thoretic Incompleteness (World Scientific, Singapore 1992).

[3] R. Chuaqui, Axiomatic Set Theory (Nort–Holland, Mathematics Studies, Amsterdam 1981).

[4] H.B. Curry, R. Feys and W. Craig, Combinatory Logic, **1**, (North–Holland, Amsterdam 1959).

[5] A. Goetz and C. Ryll-Nardzewski, *On bases of abstract algebras,* Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys. **8** (1960), 157–161.

[6] G. Grätzer, Universal Algebra, 2^{th} ed., (Springer–Verlag, New York 1979).

[7] J.R. Hindley and J.P. Seldin, Introduction to Combinators and λ–Calculus (Cambridge University Press, London 1986).

[8] B. Jónsson and A. Tarski, *Two general theorems concerning free algebras,* Bull. Amer. Math. Soc. **62** (1956), 554.

[9] E.G. Manes, Algebraic theories (Springer–Verlag, Berlin 1976).

[10] E. Marczewski, *Independence and homomorphisms in abstract algebras,* Fund. Math. **50** (1961–62), 45–61.

[11] I. Mason and C. Talcott, *Inferring the equivalence of functional program that mutate data,* Theoretical Computer Science **105** (1992), 167–215.

[12] M.L. Minsky, *Problems of formulation for Artificial Intelligence,* p. 35 in: R.E. Bellman, Mathematical Problems in the Biological Sciences, Proceedings of Symposia in Applied Mathematics XIV, American Mathematical Society, Providence, R.I. 1962.

[13] J.D. Monk, Introduction to Set Theory (McGraw–Hill, New York 1969).

[14] G. Ricci, *Universal eigenvalue equations,* Pure Math.and Appl. Ser. B, **3**, 2–3–4 (1992), 231–288.

(Most of the misprints appear in *ERRATA to Universal eigenvalue equations,* Pure Math.and Appl. Ser. B, **5**, 2 (1994), 241–243.)

[15] G. Ricci, A Whitehead generator, Quaderni del Dipartimento di Matematica **86**, (Universitá di Parma, Parma 1993).

[16] G. Ricci, *Two isotropy properties of "universal eigenspaces" (and a problem for DT0L rewriting systems),* p. 281–290 in: G. Pilz, Contributions to General Algebra **9** (Verlag Hölder–Pichler–Tempsky, Wien 1995 – Verlag B.G. Teubner).

[17] G. Ricci, *New characterizations of universal matrices show that neural networks cannot be made algebraic,* p. 269–291 in: D. Dorninger, G. Eigenthaler, H.K. Kaiser, H. Kautschitsch, W. Moren and W.B. Müller, Contributions to General Algebra **10** (J. Hein Verlag, Klagenfurt 1998).

[18] G. Ricci, *Boolean matrices . . . neither Boolean nor matrices,* Discuss. Math. General Algebra and Applications **20** (2000), 141–151.

[19] G. Ricci, *Isomorphisms between analytic monoids,* p. 33–33 in: *The Eighth International Workshop in Mathematics Gronów 2000, Institute of Mathematics, Technical University of Zielona Góra"*, September 25–29, 2000.

[20] G. Ricci, *Some analytic features of algebraic data,* Discrete Appl. Math. **122/1-3** (2002), 235–249.

[21] G. Ricci, *Analytic monoids versus abstract monoids,* Italian Journal of pure and Applied Mathematics, **16** (2004), 125–136.

[22] M. Servi, Classificazione dei Clan binari, Quaderni del Dipartimento di Matematica **113**, (Universitá di Parma, Parma 1995).

[23] M. Servi, *Definizione dei clan binari e loro classificazione,* Rend. Mat. Acc. Lincei (9) **9** (1998).

[24] M. Servi, Due parole sui Clan di Ordine, Quaderni del Dipartimento di Matematica **186**, (Universitá di Parma, Parma 1998).

[25] M. Servi, *I clan binari di ordine,* Riv. Mat. Univ. Parma (6) **1** (1998), 207–214.

[26] S. Świerckowski, *Algebras independently generated by every n elements,* Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys. **7** (1959), 501–502.

[27] S. Świerckowski, *Algebras which are independently generated by every n elements,* Fund. Math. **49** (1960–61), 93–104.

[28] D.S. Touretzky, Common LISP: a Gentle Introduction to Symbolic Computation (The Benjamin/Cummings Publishing Co. Inc, Redwood City 1990).

[29] C. Weissman, LISP 1.5 Primer (Dickenson, Belmont, Ca., 1967).

[30] A.N. Whitehead, A treatise on Universal Algebra with Applications, **1**, (Cambridge University Press, Cambridge 1898).

Received 3 May 2005