Paulo Veloso* e André Porto**

DOI: http://doi.org/10.32334/oqnfp.2021n49a804

The God-given Naturals: Induction and Recursion

Os Naturais que Deus nos deu: Indução e Recursão

Abstract

We discuss some basic issues underlying the natural numbers: induction and recursion. We examine recursive formulations and their use in establishing universal and particular properties.

Keywords: Naturals, induction, recursion, parameters, universal properties, particular properties.

Resumo

Nós discutimos alguns problemas básicos subjacentes aos números naturais. Examinamos as formulações recursivas e seu uso no estabelecimento de propriedades universais e particulares.

Palavras-chave: Naturais, indução, recursão, parâmetros, propriedades universais, propriedades particulares.

* COPPE. Universidade Federal do Rio de Janeiro.
** Universidade Federal de Goiás. Contato: andre.porto.ufg@gmail.com

Recebido em: 12/04/2021 Aceito em: 10/11/2021

"The good Lord created the natural numbers; all the rest is man's work."
L. Kronecker (1823-1891)

1.Initial Presentation

The present article is perhaps the last paper produced by Professor Paulo Veloso. It greatly expands on some ideas contained in (Porto, 2009) regarding Wittgenstein's treatment of inductive proofs. In that paper, an alternative parametrized version of the usual recursive formulations of arithmetical operations is introduced. For example, in the case of addition, instead of the usual two clauses:

$$\begin{cases} m + 0 = m \\ m + s(n) = s(m + n) \end{cases}$$

we substitute the second clause of the definition by its parametrized version:

$$\begin{cases} m + 0 = m \\ m + s^{\alpha}(n) = s^{\alpha}(m + n) \end{cases}$$

which allow us to move α successors in one single step. Once we have this new, parametrized recursive definition of addition, the usual inductive proof, say, of associativity of addition becomes simply:

$$a + \left(b + s^{\overline{n}}(0)\right) = a + s^{\overline{n}}(b + 0) = \; s^{\overline{n}}\left(a + (b + 0)\right) =$$

$$= s^{\overline{n}}(a + b) = s^{\overline{n}}\left((a + b) + 0\right) = \left((a + b) + s^{\overline{n}}(0)\right)$$

The main goal of professor Veloso's paper is to offer an extremely detailed comparison between usual inductive proofs with their parametrized versions, particularly with respect to universal/local validity, i.e., their validity with respect to non-standard models, and not only regarding the usual standard model. The paper retains the character of a work which was in progress. If it were not for his untimely death, professor Veloso would have certainly

clarified better the logical instruments behind his proofs (first order logic, second order logic) as well as would have probably added parallel sections regarding universal/specific properties of Multiplication.

2. Introduction

We will examine some basic issues underlying the structure of the natural numbers: induction and recursion. In Section 3, we will use some simple examples to introduce recursive formulations and induction, stressing the distinction between universal and particular properties. In the remaining sections, we will examine other recursive formulations, as well as the role of induction in establishing properties of functions with recursive formulations, considering three approaches, which we called *direct, reductive and algebraic ones*. The first approach establishes a universal property directly by induction, whereas the other two approaches derive a property from other properties. We will be particularly interested in distinguishing universal properties, valid for all models, and merely local properties, valid only for the standard model.

3. Motivation: Recursion and Induction

We now use some simple examples to introduce the issues of recursion and induction. Consider the natural numbers. They can be visualized as follows:

$$0 \rightarrow 1 \cdots n \rightarrow n + 1 \cdots$$

In this structure, we have the natural number and the (unary) successor function: s(n) = n+1. In fact, the natural numbers are generated starting from by (iterated) applications of successor:

$$0 \xrightarrow{s} 1 \cdots n \xrightarrow{s} n + 1 \cdots$$

One can give recursive formulations for some operations on naturals. For instance, a recursive formulation for addition of naturals may be as follows

$$\begin{cases} m + 0 = m & case\ zero\colon (0) \\ m + s(n) = s(m + n) & case\ non - zero\colon (s) \end{cases}$$

This recursive formulation (+) reduces addition to iterated applications of the successor operation +. One often says that it defines the operation . It is used in establishing some properties of addition, such as commutativity (a+b = b+a) and associativity (a+(b+a) = (a+b)+c).

3.1. Double of Naturals

We will first examine the case of double the naturals. Consider the case of doubling of a natural: d(n)=2·n.[1] A recursive formulation for function d reduces it to iterated double successors. It is as follows:

$$\begin{cases} \mathrm{d}(0) = 0 & case\ zero\colon (0) \\ \mathrm{d}\big(s(n)\big) = ss\big(\mathrm{d}(n)\big) & case\ non-zero\colon (s) \end{cases} \qquad (\mathrm{d})$$

With this formulation (d), one can evaluate the double of each natural.[2] For instance, to evaluate d(2), we can proceed as follows.

1. First, we express how 2 is generated: 2=s(s(0)).
2. Next, we use formulation (d) to evaluate d(s(s(0))) as follows

$$\underline{\mathrm{d}\big(s(s(0))\big)} \overset{(s)}{=} s\Big(s\underline{\big(\mathrm{d}(s(0))\big)}\Big) \overset{(s)}{=} s\Big(s\Big(s\underline{\big(\mathrm{d}(0)\big)}\Big)\Big) \overset{(0)}{=} s\Big(s\big(s(0)\big)\Big)$$

Finally, we note that s(s(s(0))) denotes 4, by evaluating s, as follows:

$$s\Big(s\underline{\big(s(0)\big)}\Big) = s\Big(s\underline{\big(s(1)\big)}\Big) = s\underline{\big(s(2)\big)} = \underline{s(3)} = 4$$

The entire evaluation involves three steps as follows:

(r) first, we represent the argument 2 = s(s(0));
(d) next, we use formulation (d) to eliminate d;

---

1 A programmer would write (using predecessor *p*) d(v) = ***if*** v = 0 ***then*** 0 ***else*** s s (d(p(v))).

2 Note that the non-zero case (s) should be understood as d(s(n))=s(s(d(n))), for every natural n∈ℕ. So, formulation (d) abbreviates d(0) = 0,d(s(0)) = s(s(d(0))),...

(s) finally, we evaluate s, obtaining the natural 4 = s(s(s(s(0)))).
These three steps can be summarized as follows

$$\begin{array}{ccc} \mathrm{d}(2) & & 4 \\ \textit{represent} \Downarrow & & \Uparrow \textit{evaluate } s \\ \mathrm{d}\left(s(s(0))\right) & \xRightarrow{\textit{eliminate } \mathrm{d}} & s\left(s\left(s(s(0))\right)\right) \end{array}$$

Two features of formulation (d) are apparent from this operational view:

- it eliminates the (new) symbol d: d(s(s(0))) evaluates to s(s(s(s(0)))));
- the reduced name s(s(s(s(0)))) denotes the correct double of 2.

One can also establish some properties of function in this manner. For instance:

(0) 0 is its own double, as follows: $\mathrm{d}(0) \overset{(0)}{=} 0$;
(+) a successor is not its own double, as follows: $\mathrm{d}(\mathrm{s}(\mathrm{n})) \overset{(s)}{=} \mathrm{s}(\mathrm{s}(\mathrm{d}(\mathrm{n}))) \neq 0$.

To clarify the matter, let us examine the language involved. Our language for the naturals has the symbols: constant 0 (for zero) and unary function s (for successor).

Within such a language, we have numerals as names for the naturals, as usual. For instance, $\overline{3}$ =sss(0) denotes the natural 3, being a name for it. For each natural n, we introduce its *numeral* $\bar{\mathrm{n}} := \mathrm{s}^{\mathrm{n}}(0)$ (where s^{n} is a shorthand for $\underbrace{s \ldots s}_{n}$.[3]

We can see that the numeral $\bar{\mathrm{n}}$ is a name for the natural n: it is a variable-free term denoting it. So, we see that every natural n is (uniquely) denoted by its name $\bar{\mathrm{n}} = \mathrm{s}^{\mathrm{n}}(0)$.

$$\begin{array}{ccccccccc} 0 & \overset{s}{\rightarrow} & s(0) & \cdots & s^n(0) & \overset{s}{\rightarrow} & s^{n+1}(0) & \cdots \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & \\ 0 & \rightarrow & 1 & \cdots & n & \rightarrow & n+1 & \cdots \end{array}$$

Thus, we have another natural formulation for double, as follows

3 One can also introduce numerals by recursion (over $\mathbb{N}$): $\overline{0} = 0$ and $\overline{\mathrm{s}(\mathrm{n})} := \mathrm{s}(\overline{\mathrm{n}})$.

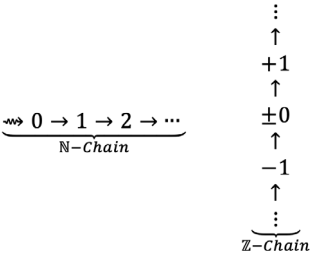
$$\begin{cases} \mathrm{dbl}[0] \doteq \overline{0} & case\ (0) \\ \mathrm{dbl}[s(\bar{n})] = s\big(s(\mathrm{dbl}[\bar{n}])\big) & case\ (s) \end{cases} \qquad (\mathrm{dbl})$$

This formulation (dbl) is just like the previous formulation (d).[4] We then have the following properties.

(=) f for n = 0: $\mathrm{dbl}[n] \doteq \overline{n}$	$\overline{0}$ does equal its own double
(≠) f for n ≠ 0: $\neg \mathrm{dbl}[n] \doteq \overline{n}$	a non-zero numeral is not its own double

So, we have $\mathrm{dbl}[\bar{n}] \doteq \bar{n} \rightarrow \bar{n} \doteq \overline{0}$, for every natural n.[5] Have we actually established the universal property $\forall v\,(\mathrm{dbl}[v] \doteq v \rightarrow v \doteq 0)$? If all we know about is the above formulation (dbl), the answer is no!

To see this claim, consider a non-standard model of the naturals with zero and successor cf. (Herbert, 1972, pp. 178-183 §3), consisting of two chains: a ℕ-chain (the standard part) and a ℤ-chain (the non-standard part) as follows:



In this structure, successor is indicated by the arrows. Now, let us define on it as expected:

- on the standard part: dbl(n)=2·n;
- on the non-standard part: dbl(z)=2· z.

This expanded structure satisfies both clauses of the recursive formulation (dbl).[6] The non-standard point ±0 is a counterexample for the above uni-

4 Again, case (s) is be understood as $\mathrm{dbl}[s(n)] \doteq s(s(\mathrm{dbl}[\bar{n}]))$, for every natural .

5 Notice that only evaluation is required to establish each specific version.

6 In fact, it satisfies the universal sentence $\forall v\ \mathrm{dbl}[s(v)] \doteq ss(\mathrm{dbl}[v])$.

versal formula: $\pm 0 \neq 0$ but $\mathrm{dbl}(\pm 0) = \pm 0$, i.e., ± 0 is a non-zero point that is its own double.

Now, let us take a closer look at what is happening with respect to induction. Let formula $\varphi(v)$ *be* $\mathrm{dbl}[v] \doteq v \rightarrow v \doteq 0$. Notice that we do have:

(0) *the basic instance* $\varphi(0)$	trivially, as $0 \doteq \overline{0}$;
(s) *each step instance* $\varphi(n) \rightarrow \varphi(s(\bar{n}))$	since $\neg s(n^{-}) \doteq \bar{n}$. ;

as well as $\phi(n)$, for every natural n. But we have failed to establish the universal formula $\forall v\, \varphi(v)$. Why is this so?

Induction is a tool to establish universal properties. What is involved here might be called induction on names. It captures part of the intuition we have about the naturals; its shortcoming is that it covers only the standard part, and not all "possible" naturals.

In order to establish the universal property $\forall v\, \varphi(v)$, one should use the induction schema

$$\left[\varphi(0) \wedge \forall v \left(\varphi(v) \rightarrow \varphi\big(s(v)\big)\right)\right] \rightarrow \forall v\, \varphi(v)$$

or the induction rule

$$\frac{\varphi(0) \quad \forall v(\varphi(v) \rightarrow \varphi\big(s(v)\big)}{\forall v\, \varphi(v)} \quad (\mathrm{IR})$$

In both cases, we have to establish:

(0) the basis $\varphi(0)$	as before;
(s) the induction step $\forall v(\varphi(v) \rightarrow \varphi(s(v)))$	a universal formula.

In this case, we do have the basic instance $\phi(0)$, but can we establish the inductive step $\forall v(\varphi(v) \rightarrow \varphi(s(v)))$? To establish such a universal inductive step, we need a universal formulation for double, such as

$$\begin{cases} db[0] \doteq 0 & case\ (\perp) \\ \forall v\, db[s(v)] \doteq ss(db[v]) & case\ (\triangleright) \end{cases} \quad (\mathrm{db})$$

With this formulation (db), the universal inductive step follows from ∀v ¬ s(v) ≐ 0 Therefore, we can infer the universal property ∀v ϕ(v).

3.2. Addition of naturals

We now return to addition of naturals. Much as in the case of d, we can use formulation (+) to evaluate specific cases of addition. For instance, consider 3+2 and 2+3.

1. We can evaluate 3+2 as follows

$$\overbrace{3 + s(s(0)) \overset{(s)}{=} s(3 + s(0)) \overset{(s)}{=} s(s(3 + 0)) \overset{(0)}{=} s\underbrace{(s(3)) = s(4) = 5}_{s}}^{(+)}$$

2. Similarly, we can evaluate 2 + 3 as follows

$$\overbrace{2 + s\left(s(s(0))\right) \overset{(s)}{=} s\left(2 + s(s(0))\right) \overset{(s)}{=} s\left(s(2 + s(0))\right) \overset{(s)}{=} s\left(s(s(2 + 0))\right) \overset{(0)}{=} s\underbrace{\left(s(s(2))\right) = s(s(3)) = s(4) = 5}_{s}}^{(+)}$$

In this manner, we see the specific commutativity 3+2=2+3.

As in the case of double, we have another natural formulation for addition, namely:

$$\begin{cases} \mathrm{add}[\overline{m}, 0] \doteq \overline{m} & case\ (0) \\ \mathrm{add}[\overline{m}, s(\overline{n})] \doteq s(\mathrm{add}[\overline{m}, \overline{n}]) & case\ (s) \end{cases} \qquad (\mathrm{add})$$

This formulation (add) is just like (+). So, for each given pair m and n of naturals, we can establish the specific commutativity add[$\overline{m}$, $\overline{n}$] ≐ add[$\overline{n}$,$\overline{m}$] by evaluation and comparison.

We can proceed similarly for each given pair m and n of naturals.

1. Evaluate the left-hand side add[$\overline{m}$, $\overline{n}$].[7]

2. Evaluate the right-hand side add[n, m].[8]

---

7 We have $\mathrm{add}[\overline{m}, s^n(0)] \overset{(\mathrm{add})}{\doteq} \underbrace{s \dots s}_{n}(\overline{m}) \overset{(\overline{m})}{=} \underbrace{s \dots s}_{n}\underbrace{s \dots s}_{m}(0)$

8 We have $\mathrm{add}[\overline{n}, s^m(0)] \overset{(\mathrm{add})}{\doteq} \underbrace{s \dots s}_{m}(\overline{n}) \overset{(\overline{n})}{=} \underbrace{s \dots s}_{m}\underbrace{s \dots s}_{n}(0)$

3. Notice that both sides evaluate to the same natural.[9]

Notice that we do not need induction to establish each given case add[$\overline{m}$, $\overline{n}$] ≐ add[$\overline{n}$,$\overline{m}$].

Thus, we have the specific commutativity of addition:

($K+_{\downarrow}$) add[$\overline{m}$, $\overline{n}$] ≐ add[$\overline{n}$,$\overline{m}$], for each pair m and n of naturals

We cannot, however, establish the universal property ∀u∀v add[u,v] = add[v,u]![10]

To establish such a universal property, we need a universal formulation for addition, such as the following one:

$$\begin{cases} \forall u\ ad[u,0] \doteq u & case\ (\bot) \\ \forall u \forall v\ ad[u,s(v)] \doteq s(ad[u,v]) & case\ (\rhd) \end{cases} \qquad (ad)$$

To see the difference between this universal formulation (ad) and formulation (add), notice that we can now have more general evaluations. For instance, we can now evaluate ad[u,ss(0)] to ss(u) as follows:

$$ad[u,ss(0)] \overset{(\rhd)}{\doteq} s(ad[u,s(0)]) \overset{(\rhd)}{\doteq} ss(ad[u,0]) \overset{(\bot)}{=} ss(u)$$

With this formulation , one can establish the following universal properties of ad.

($0+_{\forall}$) *Left zero*: ∀v ad[0,v] ≐ v	by induction on v.
($s+_{\forall}$) *Left successor:* ∀u∀v *ad*[s(u),v] ≐ s(ad[u,v])	by induction on v.
($K+_{\forall}$) ∀u∀v *ad*[u,v] ≐ *ad*[v,u] *using*(0+_∀)*and*($s+_{\forall}$)	by induction on u.

Note that the above auxiliary properties ($0+_{\forall}$) and ($s+_{\forall}$) are special cases of commutative property ($K+_{\forall}$). We also have specific instances of these auxiliary properties ($0+_{\forall}$) and ($s+_{\forall}$).

($0+_{\downarrow}$) add[0,$\overline{n}$] ≐ $\overline{n}$, for every natural n∈ℕ (of ($0+_{\forall}$))
($s+_{\downarrow}$) add[s($\overline{m}$),$\overline{n}$] ≐ s(add[$\overline{m}$,$\overline{n}$]), for all naturals

---

9 Both $\underbrace{s\dots s}_{n}\underbrace{s\dots s}_{m}(0)$ and $\underbrace{s\dots s}_{m}\underbrace{s\dots s}_{n}(0)$ have the same number of s's

10 We can argue much as in the case of dbl: we expand the non-standard structure by add[a,n]=a+n and add[a,b]=b+1; then we satisfy both clauses of the formulation (add), but add[±0,0] = ±0 whereas add[0,±0] = ±0+1=+1.

These specific versions, being special cases of the specific commutative property ($k+_{\downarrow}$), can also be established directly by evaluation and comparison.

Some remarks about these ideas are in order.

1. We mentioned that we do not need induction to establish each specific version of commutativity of addition. So, induction appears to play no role in such cases. Actually, induction (on numerals) might come in handy to reuse cases previously established.[11] For, all specific versions form an infinite set, so we have countably many induction-free proofs. Induction on $\mathbb{N}$ may help condensing this countable set of proofs to finitely many proofs. This seems to be the usual case.[12]

2. Similar methods, based on formulation (ad), yield associativity of addition.
- For the version $\forall u \forall v \forall w\ ad(u,ad[v,w]) \doteq ad(ad[u,v],w)$, we use (IR)[13].
- For the version $\forall u \forall v\ ad(u,ad[v,\overline{k}]) \doteq ad(ad[u,v],\overline{k})$, we can resort to evaluation and comparison.[14]

3. The reason why these methods work so nicely for addition has to do with the rather simple nature of its recursive formulation (see Section 4). One can also establish properties of other operations, such as multiplication, by similar methods, but the details tend to be more involved.

We can now return to some distinctions universal versus particular properties and direct and reductive approaches. These examples help clarifying the distinction between universal and particular properties. The distinction between the direct, reductive and algebraic approaches will become clearer later on. Meanwhile, our examples can perhaps provide some clarification about the distinction between the direct and reductive approaches. Consider, for instance associativity of addition. On the one hand, the direct approach will establish its universal version directly by the induction rule (IR). On the other hand, the reductive approach will establish its particular version by evaluating both sides, thereby reducing them to their normal forms, and comparing the results. The case of commutativity of addition is similar. The

---

11 Note that $add[\overline{m},\overline{n}] \doteq add[\overline{n},\overline{m}]$ yields immediately $add[\overline{m},s(\overline{n})] \doteq add[\overline{n},s(\overline{m})]$, by (add.s).

12 We will examine such issues in Section 7 (see 7.2).

13 We can employ induction on w (see 6.1 in Section 6).

14 Both sides reduce to $s^k(ad[u,v])$ (see 7.1 in Section 7).

so-called algebraic approach establishes a (universal) property by instantiating general conditions.

The following table may be of help in clarifying these distinctions.

Approach	Property	Method	Models	Range	Sections
Direct	Universal	Induction (IR)	all	wide	6
Reductive	Specific	Induction on $\mathbb{R}$	standard	narrow	7
Algebraic	Universal	General conditions	all	wide	8, 9

In the sequel, we shall take a closer look at some aspects of these questions. In the next section, we will examine universal recursive formulations.

4. Universal Recursive Formulations

We will now examine some examples of universal recursive formulations. We shall illustrate recursive formulations (in 4.1) and classify them (in 4.2).

4.1. Examples of recursive formulations

We will now illustrate recursive formulations for some functions.

We will consider universal formulations. To simplify the notation, however, we will leave implicit the universal quantifiers. So, we write the formulation , in Section 3, simply as

$$\begin{cases} ad[\mathrm{u},0] \doteq \mathrm{u} & case\ (\bot) \\ ad[\mathrm{u},s(\mathrm{v})] \doteq s(ab[\mathrm{u},\mathrm{v}]) & case\ (\rhd) \end{cases} \qquad \text{(ad)}$$

As mentioned in Section 3, by relying on this formulation (ad), one can establish, by induction (IR), some properties of addition (see Section 6).

$(\mathrm{K}+_{\forall})$	$\forall \mathrm{u} \forall \mathrm{v}\ \mathrm{ad}[\mathrm{u},\mathrm{v}] \doteq \mathrm{ad}[\mathrm{v},\mathrm{u}]$	commutativity of +
$(++_{\forall})$	$\forall \mathrm{u} \forall \mathrm{v} \forall \mathrm{w}\ \mathrm{ad}(\mathrm{u},\mathrm{ad}[\mathrm{v},\mathrm{w}]) \doteq \mathrm{ad}(\mathrm{ad}[\mathrm{u},\mathrm{v}],\mathrm{w})$	associativity of +

Also, we write the formulation , in Section 3, simply as

$$\begin{cases} db[0] \doteq 0 & case\ (\bot) \\ db[s(\mathrm{v})] \doteq ss(db[\mathrm{v}]) & case\ (\triangleright) \end{cases} \quad \text{(db)}$$

By relying on formulations (db) and (ad), one can establish, by induction (IR), the universal property ∀v db[v] ≐ ad[v,v]..[15]

Other examples of recursive formulations are as follows.

(pd) Function predecessor pd has the following recursive formulation:

$$\begin{cases} pd[0] \doteq 0 & case\ (\bot) \\ pd[s(\mathrm{v})] \doteq \mathrm{v} & case\ (\triangleright) \end{cases} \quad \text{(pd)}$$

Formulation (pd) gives the property ∀v pd[s(v)]= v.[16]

(mn) For function minus, with a∸b = a-b (for a ≥ b) and a∸b = 0 (for a < b), we have the following recursive formulation (using function pd)

$$\begin{cases} mn[\mathrm{u}, 0] \doteq \mathrm{u} & case\ (\bot) \\ mn[\mathrm{u}, s(\mathrm{v})] \doteq pd(mn[\mathrm{u}, \mathrm{v}]) & case\ (\triangleright) \end{cases} \quad \text{(mn)}$$

Formulations (mn) and (pd) give the property ∀v mn[0,v] ≐ 0.

(mt) A recursive formulation for multiplication · is as follows:

$$\begin{cases} mt[\mathrm{u}, 0] \doteq 0 & case\ (\bot) \\ mt[\mathrm{u}, s(\mathrm{v})] \doteq ad(\mathrm{u}, mt[\mathrm{u}, \mathrm{v}]) & case\ (\triangleright) \end{cases} \quad \text{(mt)}$$

Formulation (mt) gives property (K·): ∀u∀v mt[u,v] ≐ mt[v,u].

15 The formulations (dbl) and (add), in Section 3, would give the particular properties dbl[$\overline{\mathrm{n}}$] = add[$\overline{\mathrm{n}}$, $\overline{\mathrm{n}}$], for every natural n ∈ ℕ.

16 Function successor has a recursive formulation: sc[0]≐s(0); sc[s(v)]≐s(sc[v]).

(sq) A recursive formulation for square is as follows:

$$\begin{cases} sq[0] \doteq 0 & case\ (\bot) \\ sq[s(\mathrm{v})] \doteq s\left(ad\left(sq[\mathrm{v}], db(\mathrm{v})\right)\right) & case\ (\triangleright) \end{cases} \quad \text{(sq)}$$

Formulations (sq) and (mt) give the property ∀v sq[v] ≐ mt[v,v].

(pw) A recursive formulation for exponentiation (with $0^0 = 1$) is as follows:

$$\begin{cases} pw[\mathrm{u}, 0] \doteq \mathrm{s}(0) & case\ (\bot) \\ pw[\mathrm{u}, s(\mathrm{v})] \doteq mt(\mathrm{u}, pw[\mathrm{u}, \mathrm{v}]) & case\ (\triangleright) \end{cases} \quad \text{(pw)}$$

Formulation (pw) gives the property ∀u pw[u,s(0)] ≐ u.

4.2. Classes of recursive formulations

We will now classify our recursive formulations.

In each one of our recursive formulations in 4.1, the value of the function at the next input is computed from the present value: cf. case (⊳). There are, however, some distinctions.

1. In formulations (ad) and (mn), the value at the next input depends only on the present value. We may call them value recursive.
A *value* recursive formulation for a function f uses a unary step function h as follows

$$\begin{cases} f\left[\begin{matrix}\vec{\mathrm{u}} \\ 0\end{matrix}\right] \doteq \mathrm{g}(\vec{\mathrm{u}}) & case\ (\bot) \\ f\left[\begin{matrix}\vec{\mathrm{u}} \\ s(\mathrm{v})\end{matrix}\right] \doteq h\left(f\left[\begin{matrix}\vec{\mathrm{u}} \\ \mathrm{v}\end{matrix}\right]\right) & case\ (\triangleright) \end{cases} \quad \text{(f)}$$

The case of (db) also fits into this pattern: it has no parameter.

2. Formulations (mt) and (pw) are different: by clause (mt. ⊳), mt[u,s(v)] depends on both u and mt[u,v].. Indeed, we evaluate mt[u,ss(0)] as follows:

$$mt\begin{bmatrix}\mathrm{u}\\ ss(0)\end{bmatrix} \overset{(\triangleright)}{\doteq} ad\left(mt\begin{bmatrix}\mathrm{u}\\ s(0)\end{bmatrix}\right) \overset{(\triangleright)}{\doteq} ad\left(ad\left(mt\begin{bmatrix}\mathrm{u}\\ 0\end{bmatrix}\right)\right) \overset{(\perp)}{\doteq} ad\left(ad\begin{pmatrix}\mathrm{u}\\ 0\end{pmatrix}\right)$$

In these formulations (mt) and (pw), the value at the next input depends on the parameter as well as on the present value. We may call them simple recursive.

A *simple* recursive formulation for a function f´ uses a step function h´ as follows

$$\begin{cases} \acute{f}\begin{bmatrix}\vec{\mathrm{u}}\\ 0\end{bmatrix} \doteq \mathrm{g}(\vec{\mathrm{u}}) & case\ (\perp) \\ \acute{f}\begin{bmatrix}\vec{\mathrm{u}}\\ s(\mathrm{v})\end{bmatrix} \doteq \acute{h}\left(\begin{matrix}\vec{\mathrm{u}}\\ \acute{f}\begin{bmatrix}\vec{\mathrm{u}}\\ \mathrm{v}\end{bmatrix}\end{matrix}\right) & case\ (\triangleright) \end{cases} \qquad (\acute{f})$$

Formulations (pd) and (sq) illustrate yet another difference: clause (pd.⊳) shows that pd[s(v)] depends on v, rather than on the present value pd[v]; its step function is the left projection $\gamma\binom{\mathrm{v}}{\mathrm{w}} = \mathrm{v}$. In these formulations (pd) and (sq), the value at the next input depends on the input.

In general, a *primitive recursive* formulation for a function f' uses a step function h´ as follows

$$\begin{cases} \hat{f}\begin{bmatrix}\vec{\mathrm{u}}\\ 0\end{bmatrix} \doteq \mathrm{g}(\vec{\mathrm{u}}) & case\ (\perp) \\ \hat{f}\begin{bmatrix}\vec{\mathrm{u}}\\ s(\mathrm{v})\end{bmatrix} \doteq \hat{h}\left(\begin{matrix}\vec{\mathrm{u}}\\ \mathrm{v}\\ \hat{f}\begin{bmatrix}\vec{\mathrm{u}}\\ \mathrm{v}\end{bmatrix}\end{matrix}\right) & case\ (\triangleright) \end{cases} \qquad (\hat{f})$$

Next, in Section 5, we will examine parameterization. In Section 6, we will examine how one can establish some universal properties of addition: associativity and commutativity.

5. Parameterization

We will now examine parameterization. Here, the motivation is the reduction to compositions of step functions. In Sections 3 and 4, we have examined some features of recursive formulations. This has clearly led to compositions of the step function, in the case of addition, but not quite so in the case of multiplication. Indeed, we have the following evaluations:

$(ad)\quad ad\left[\begin{matrix}u\\ \overline{2}\end{matrix}\right] \overset{(ad)}{\doteq} s^2(u)$ composite successors (cf. 3.2)

$(mt)\quad mt\left[\begin{matrix}u\\ \overline{2}\end{matrix}\right] \overset{(mt)}{\doteq} ad\left(\begin{matrix}u\\ ad\left[\begin{matrix}u\\ \overline{2}\end{matrix}\right]\end{matrix}\right)$ nested additions (cf. 4.2)

One way of approaching the goal of compositions of step functions is parameterization. We introduce the basic ideas with the simple case of addition (cf. formulation (ad) in 4.1). For each natural m, we define the parametrized addition $_{\overline{m}}ad[v] = ad\left[\begin{matrix}\overline{m}\\ v\end{matrix}\right]$. So, we have the following connection between the two versions of addition:

$$\forall v\ {}_{\overline{m}}ad[v] \doteq ad\left[\begin{matrix}\overline{m}\\ v\end{matrix}\right]$$

Thus, this unary function $_{\overline{m}}$ad has the following (value) recursive formulation:

$$\begin{cases} {}_{\overline{m}}ad[0] \doteq \overline{m} & case\ (\bot) \\ {}_{\overline{m}}ad[s(v)] \doteq s({}_{\overline{m}}ad[v]) & case\ (\triangleright) \end{cases} \qquad ({}_{\overline{m}}ad)$$

This formulation ($_{\overline{m}}$ad) has basis value $\overline{m}$ (cf.($\bot$)) and step function s (cf. ($\triangleright$)). We can compare these two versions of addition as follows:

	Function	arity	Basis	arity	Step	arity
Original	$ad\left[\begin{matrix}u\\ v\end{matrix}\right]$	2	$\iota(u)=u$	1	s(w)	1
Parametrized	$_{\overline{m}}$ad[v]	1	$\overline{m}$	0	s(w)	1

We can now examine the ideas underlying parameterization: parameterizing a function by fixing some of its arguments.

We parameterize a function $F:\mathbb{N}^n \times V \to W$ as follows: for each tuple $\vec{m} \in \mathbb{N}^n$ introduce its *parameterized version* $_{\vec{m}}F:V \to W$ by:

$$\overrightarrow{m}F[v] = F\left(\begin{matrix}\overrightarrow{m}\\ v\end{matrix}\right)$$

Parameterization constructs several new functions form a given one. We can compare these two versions of a function F as follows:

Original function	Parameterized function	Definition
$F : \mathbb{N}^n \times V \to W$	$\vec{m}F : V \to W$	$\vec{m}F[v] = F\binom{\vec{m}}{v}$

We shall examine parameterization for classes of recursive formulations (cf. 4.2): value recursion (in 5.1), simple recursion (in 5.2) and primitive recursion (in 5.3).

5.1. Parameterization: value recursion

We will now examine parameterization for value recursive formulations (cf. 4.2).

The case of monus (cf. 4.1) is entirely similar to addition. For each natural m, we define parametrized monus $\vec{m}$mn by the following connection:

$$\forall v \ \ \vec{m}mn[v] \doteq ad\begin{bmatrix}\overline{m}\\ v\end{bmatrix}$$

Thus, this unary function $\vec{m}$mn has a value recursive formulation much as $\vec{m}$ ad), with basis value $\overline{m}$ and step function predecessor pd.

It is not difficult to extend these ideas to value recursive functions (cf. 4.2). Consider a value recursive function f satisfying formulation (f). We introduce the unary parameterized version $\vec{m}f$ by the following connection:

$$\forall v \ \ \vec{m}f[v] \doteq f\begin{bmatrix}\vec{m}\\ v\end{bmatrix}$$

We now consider the unary parameterized version $\vec{m}$ g $=\vec{m}$ and noting that $\vec{m}$ g $\in \mathbb{N}$, we consider its numeral $\vec{m}\bar{g} = \overline{g(\vec{m})}$. Then, the parameterized version $\vec{m}f$ has the following recursive formulation:

$$\begin{cases}\vec{m}f[0] \doteq \vec{m}\bar{g} & case\ (\perp)\\ \vec{m}f[s(v)] \doteq h(\vec{m}f[v]) & case\ (\triangleright)\end{cases} \qquad (\vec{m}f)$$

This formulation $\vec{m}f$ has basis value $\vec{m}\,\overline{g}$ and step function h.

We can compare these two versions of a value recursive function as follows:

	Function	arity	Basis	arity	Step	arity
Original	$f\begin{bmatrix}\vec{u}\\ v\end{bmatrix}$	n+1	$g(\vec{u})$	n	h(w)	1
Parametrized	$\vec{m}\, f[v]$	1	$\vec{m}\,\overline{g}$	0	h(w)	1

5.2. Parameterization: simple recursion

We now examine parameterization for simple recursive formulations (cf. 4.2). We will begin with parametrized multiplication $\overline{m}mt[v] = mt\begin{bmatrix}\overline{m}\\ v\end{bmatrix}$ for each natural m, by the following connection:

$$\forall v\ \overline{m}mt[v] \doteq mt\begin{bmatrix}\overline{m}\\ v\end{bmatrix}$$

Thus, this unary function has the following recursive formulation:

$$\begin{cases}\overline{m}mt[0] \doteq \overline{0} & case\ (\perp)\\ \overline{m}mt[s(v)] \doteq \overline{m}ad(\overline{m}mt[v]) & case\ (\triangleright)\end{cases} \qquad (\overline{m}mt)$$

This formulation ($\overline{m}$mt) has basis value: $\overline{0}$ (cf.($\perp$)) and step function: $\overline{m}$ad (cf. ($\triangleright$)). We can compare these two versions of multiplication as follows:

	Function	arity	Basis	arity	Step	arity
Original	$mt\begin{bmatrix}u\\ v\end{bmatrix}$	2	$\zeta(u)=0$	1	$ad\begin{pmatrix}u\\ w\end{pmatrix}$	2
Parametrized	$\overline{m}$mt[v]	1	$\overline{0}$	0	$\vec{\overline{m}}$ ad(w)	1

The case of exponentiation (cf. 4.1) entirely similar to multiplication. For each natural m, we define parametrized exponentiation $\vec{\overline{m}}$ pw, by the following connection:

$$\forall v\ \overline{m}pw[v] \doteq pw\begin{bmatrix}\overline{m}\\ v\end{bmatrix}$$

Thus, this unary function $_{pwm}$mn has a value recursive formulation much as ($\overline{m}$mt) , with basis value and step function parametrized multiplication $\overline{m}$mt.

Much as before, we can extend these considerations to a simple recursive function (cf. formulation ($\acute{f}$) in 4.2). Consider a simple recursive function $\acute{f}$ satisfying formulation ($\acute{f}$). We introduce the unary parameterized version $\vec{\mathrm{m}}\,\acute{f}$ by the following connection:

$$\forall \mathrm{v}\ {}_{\vec{m}}\acute{f}[\mathrm{v}] \doteq \acute{f}\begin{bmatrix}\vec{m}\\ \mathrm{v}\end{bmatrix}$$

Introducing the unary parameterized version ${}_{\vec{m}}\acute{h}[\mathrm{w}] \doteq \acute{h}\begin{bmatrix}\vec{m}\\ \mathrm{w}\end{bmatrix}$, we consider the numeral. Then, the parameterized version $\vec{\mathrm{m}}\,\acute{f}$ has the following recursive formulation:

$$\begin{cases} {}_{\vec{m}}f[0] \doteq {}_{\vec{m}}\bar{g} & case\ (\perp) \\ {}_{\vec{m}}f[s(\mathrm{v})] \doteq {}_{\vec{m}}\acute{h}\left({}_{\vec{m}}f[\mathrm{v}]\right) & case\ (\rhd) \end{cases} \qquad \left({}_{\vec{m}}f\right)$$

This formulation $\left({}_{\vec{\mathrm{m}}}\acute{f}\right)$ has basis value $\vec{\mathrm{m}}\,\bar{\mathrm{g}}$ and step function $\vec{\mathrm{m}}\,\acute{\mathrm{h}}$.

5.3 Parameterization: primitive recursion

We will now examine the general case of parameterization, namely functions with primitive recursive formulations (cf. 4.2); one can also parametrize them. Consider a primitive recursive function $\grave{f}$ (cf. formulation ($\grave{f}$) in 4.2). As before, we introduce the unary parameterized version $\vec{\mathrm{m}}\grave{f}$ by the following connection:

$$\forall \mathrm{v}\ {}_{\vec{m}}\grave{f}[\mathrm{v}] \doteq \grave{f}\begin{bmatrix}\vec{m}\\ \mathrm{v}\end{bmatrix}$$

Now, we consider the numeral ${}_{\vec{m}}\bar{g} = \overline{g(\vec{m})}$ and introduce the parameterized version ${}_{\vec{m}}\grave{h}\binom{\mathrm{v}}{\mathrm{w}} = \grave{h}\begin{pmatrix}\vec{m}\\ \mathrm{v}\\ \mathrm{w}\end{pmatrix}$. Then, the parameterized version $\vec{\mathrm{m}}\grave{f}$ has the following recursive formulation:

$$\begin{cases} {}_{\vec{m}}\grave{f}[0] \doteq {}_{\vec{m}}\bar{g} & case\ (\perp) \\ {}_{\vec{m}}\grave{f}[s(\mathrm{v})] \doteq {}_{\vec{m}}\grave{h}\left(\begin{matrix}\mathrm{v}\\ {}_{\vec{m}}\grave{f}[\mathrm{v}]\end{matrix}\right) & case\ (\rhd) \end{cases} \qquad \left({}_{\vec{m}}\grave{f}\right)$$

Notice that the parametrized step function ${}_{\vec{m}}\grave{h}\binom{\mathrm{v}}{\mathrm{w}} = \grave{h}\begin{pmatrix}\vec{m}\\ \mathrm{v}\\ \mathrm{w}\end{pmatrix}$ is no longer unary.

Notice that we take as parameters the inputs other than the recursive variable.

Now, consider functions without parameters, such as the unary functions db, sq, and pd (in 4.1). One could in principle apply the same idea to the recursive variable, but this does not seem to be of much help. Consider the case of unary predecessor. This function has no parameters. If we define $_{\overline{m}}\mathrm{pd} = \mathrm{pd}[\overline{m}]$, then will have distinct versions of predecessor, depending on the input $\overline{m} : {}_{\overline{0}}\mathrm{pd} = \overline{0}$ and ${}_{s(\overline{n})}\mathrm{pd} = \overline{n}$. In the next section, we will examine iteration.

6. Addition: universal properties

We will now examine how one can establish some universal properties of addition. We will establish associativity (in 6.1) and commutativity (in 6.2), using the universal formulation (ad) (cf. 3.2 in Section 3).

6.1. Universal associativity of addition

Consider the following universal formulation of associativity of addition:

$$(Aad_{\forall})\ \forall u \forall v \forall w\ ad\left(\begin{matrix} u \\ ad\begin{bmatrix} v \\ w \end{bmatrix}\end{matrix}\right) \doteq ad\left(\begin{matrix} ad\begin{bmatrix} u \\ v \end{bmatrix} \\ w \end{matrix}\right) \qquad a + (b + c) = (a + b) + c$$

We establish $\mathrm{Aad}_{\forall}$ by (IR): induction on w, using formulation in 3.2:

(0) Using (ad.⊥) twice

$$ad\left(\begin{matrix} u \\ ad\begin{bmatrix} v \\ 0 \end{bmatrix}\end{matrix}\right) \overset{(ad.\perp)}{\doteq} ad\begin{bmatrix} u \\ v \end{bmatrix} \overset{(ad.\perp)}{\doteq} ad\left(\begin{matrix} ad\begin{bmatrix} u \\ v \end{bmatrix} \\ 0 \end{matrix}\right)$$

(s) Using (ad.⊳) twice, (HI) and (ad.⊳)

$$\begin{array}{l} ad\left(\begin{matrix} u \\ ad\begin{bmatrix} v \\ s(w) \end{bmatrix}\end{matrix}\right) \\ \quad \cdot\|_{(ad.\triangleright)} \\ ad\left(\begin{matrix} u \\ s\left(ad\begin{bmatrix} v \\ w \end{bmatrix}\right)\end{matrix}\right) \\ \quad \cdot\|_{(ad.\triangleright)} \\ s\left(ad\left(\begin{matrix} u \\ ad\begin{bmatrix} v \\ w \end{bmatrix}\end{matrix}\right)\right) \overset{(HI)}{\doteq} s\left(ad\left(\begin{matrix} ad\begin{bmatrix} u \\ v \end{bmatrix} \\ w \end{matrix}\right)\right) \\ \qquad\qquad\qquad\qquad\quad \cdot\|_{(ad.\triangleright)} \\ \qquad\qquad\qquad\qquad\quad ad\left(\begin{matrix} ad\begin{bmatrix} u \\ v \end{bmatrix} \\ s(w) \end{matrix}\right) \end{array}$$

6.2. Universal commutativity of addition

We will now examine how one can establish the universal commutativity of addition. Consider the following universal formulation of commutativity of addition:

$$(Kad_{\forall}) \quad \forall u \forall v \;\; ad\begin{bmatrix} u \\ v \end{bmatrix} \doteq ad\begin{bmatrix} v \\ u \end{bmatrix} \qquad\qquad a + b = b + a$$

We will establish ($Kad_{\forall}$) by using its two special cases (cf. Section 3):

$$(0ad_{\forall}) \; \forall v \;\; ad\begin{bmatrix} 0 \\ v \end{bmatrix} \doteq v \qquad\qquad Left\ zero\colon 0 + b = b$$

$$(sad_{\forall}) \;\; \forall u \forall v \;\; ad\begin{bmatrix} s(u) \\ v \end{bmatrix} \doteq s\left(ad\begin{bmatrix} v \\ u \end{bmatrix}\right) \;\; Left\ succ\colon a + s(b) = s(a \; + \; b)$$

We first note that the, in the presence of formulation (ad), two properties ($0ad_{\forall}$): *left zero* and ($sad_{\forall}$): *left successor* are special instances of ($Kad_{\forall}$) *commutativity*.

$$[(Kad_{\forall}) \vdash^{ad} (0ad_{\forall})](ad) \underbrace{: \forall u \forall v \;\; ad\begin{bmatrix} u \\ v \end{bmatrix} \doteq ad\begin{bmatrix} v \\ u \end{bmatrix}}_{(Kad_{\forall})} \vdash \underbrace{\forall v \;\; ad\begin{bmatrix} 0 \\ v \end{bmatrix} \doteq v}_{(0ad_{\forall})} \qquad with\ u = 0^{17}$$

$$[(Kad_{\forall}) \vdash^{ad} (sad_{\forall})](ad) \underbrace{: \forall u \forall v \;\; ad\begin{bmatrix} u \\ v \end{bmatrix} \doteq ad\begin{bmatrix} v \\ u \end{bmatrix}}_{(Kad_{\forall})} \vdash \underbrace{: \forall u \forall v ad\begin{bmatrix} s(u) \\ v \end{bmatrix} \doteq s\left(ad\begin{bmatrix} v \\ u \end{bmatrix}\right)}_{(Kad_{\forall})} \quad with\ u = s(u)^{18}$$

We now establish these three properties by (IR), using formulation (ad) in 3.2.

$$(0ad_{\forall}) \; \forall v \;\; ad\begin{bmatrix} 0 \\ v \end{bmatrix} \doteq v \qquad (left\ zero) \qquad by\ (IR)\colon on\ v, \text{with(ad)}$$

(0) Direct by (ad.⊥)

$$ad\begin{bmatrix} 0 \\ 0 \end{bmatrix} \overset{(\text{ad.}\perp)}{\doteq} 0$$

17 Indeed $ad\begin{bmatrix} 0 \\ v \end{bmatrix} \overset{(Kad_{\forall})}{\doteq} ad[v, 0] \overset{(ad.\perp)}{\doteq} v$.

18 Indeed $ad\begin{bmatrix} s(u) \\ v \end{bmatrix} \overset{(Kad_{\forall})}{\doteq} ad\begin{bmatrix} v \\ s(u) \end{bmatrix} \overset{(ad.\triangleright)}{\doteq} s\left(ad\begin{bmatrix} v \\ u \end{bmatrix}\right)$

(s) Using (ad.⊳) and (HI)

$$\begin{array}{l} ad\begin{bmatrix}0\\ s(v)\end{bmatrix} \\ \quad \cdot\|_{(ad.\triangleright)} \\ s\left(ad\begin{bmatrix}0\\ v\end{bmatrix}\right) \overset{(HI)}{\doteq} s(v) \end{array}$$

$(sad_\forall)\ \forall u \forall v \quad ad\begin{bmatrix}s(u)\\ v\end{bmatrix} \doteq s\left(ad\begin{bmatrix}v\\ u\end{bmatrix}\right) \qquad (left\ successor) \qquad by\ (IR)\colon on\ v, with(ad)$

(0) Using (ad.⊥) twice

$$ad\begin{bmatrix}s(u)\\ 0\end{bmatrix} \overset{(ad.\perp)}{\doteq} s(u) \overset{(ad.\perp)}{\doteq} s\left(ad\begin{bmatrix}u\\ 0\end{bmatrix}\right)$$

(s) Using (ad.⊳), (HI) and (ad.⊳)

$$\begin{array}{l} ad\begin{bmatrix}s(u)\\ s(v)\end{bmatrix} \\ \quad \cdot\|_{(ad.\triangleright)} \\ s\left(ad\begin{bmatrix}s(u)\\ v\end{bmatrix}\right) \overset{(HI)}{\doteq} s\left(s\left(ad\begin{bmatrix}u\\ v\end{bmatrix}\right)\right) \\ \qquad\qquad \cdot\|_{(ad.\triangleright)} \\ \qquad\qquad s\left(ad\begin{bmatrix}u\\ s(v)\end{bmatrix}\right) \end{array}$$

$(Kad_\forall)\ \forall u \forall v \quad ad\begin{bmatrix}u\\ v\end{bmatrix} \doteq ad\begin{bmatrix}v\\ u\end{bmatrix} \qquad (commut) \qquad by\ (IR)\colon on\ u, with(ad), (0ad_\forall), (sad_\forall)$

(0) Using (0ad$_\forall$) and (ad.⊥)

$$ad\begin{bmatrix}0\\ v\end{bmatrix} \overset{(0ad_r)}{\doteq} v \overset{(ad.\perp)}{\doteq} ad\begin{bmatrix}v\\ 0\end{bmatrix}$$

(s) Using , (HI) and

$$\begin{array}{l} ad\begin{bmatrix}s(u)\\ v\end{bmatrix} \\ \quad \cdot\|_{(sad_r)} \\ s\left(ad\begin{bmatrix}u\\ v\end{bmatrix}\right) \overset{(HI)}{\doteq} s\left(ad\begin{bmatrix}u\\ v\end{bmatrix}\right) \\ \qquad\qquad \cdot\|_{(ad.\triangleright)} \\ \qquad\qquad ad\begin{bmatrix}u\\ s(v)\end{bmatrix} \end{array}$$

Table 1 compares the proof structures for these three universal commutativity-related properties of addition.

Property	Formulation	IR on	Structure
Left zero	$(0ad_{\forall}) \; ad \begin{bmatrix} \forall v \\ 0 \\ v \end{bmatrix} \doteq v$	v	(0): (ad.⊳) (s): (ad.⊳);(HI);(ad.⊳)
Left successor	$(sad_{\forall}) \; ad \begin{bmatrix} \forall u \forall v \\ s(u) \\ v \end{bmatrix} \doteq s\left(ad \begin{bmatrix} u \\ v \end{bmatrix}\right)$	v	(0): (ad.⊳) (s): (ad.⊳);(HI);(ad.⊳)
Commutativity	$(K + _{\forall}) \; ad \begin{bmatrix} \forall u \forall v \\ u \\ v \end{bmatrix} \doteq ad \begin{bmatrix} v \\ u \end{bmatrix}$	u	(0):(0ad$_{\forall}$);(ad.⊳) (s):(sad$_{\forall}$);(HI);(ad.⊳)

7. Addition: specific properties

We will now examine how one can establish some specific properties of addition. We will establish associativity (in 7.1) and commutativity (in 7.2), using the universal formulation (cf. 3.2 in Section 3). For such specific properties, one can avoid the induction rule, by resorting to auxiliary properties (established by induction on). For addition, the idea is (partially) reducing it to composition of successors.

We introduce the composites of a unary function , as expected. Given a unary function t: W→W, we define its k-*composite* t^k :W→W, for each k ∈ ℕ, by the following recursion (over ℕ)

$$t^0[w]=w, \qquad t^{s(k)}[w] = t(t^k[w])$$

Thus, $t^k[w] = \underbrace{t \ldots t}_{k}[w]$.[19]

The following commutativities of composites are clear.[20]

19 One could also introduce the iteration t* [v,w] (with a new variable v giving the number of iterations) by the recursion t* [0,w]≐w and t* [s(v),w] ≐ t(t* [v,w])

20 The instance (s^n s^m) of property ($t^n t^m$) will be used for establishing commutativity of addition.

$$(tt^n)\ \textit{Function and composite}\colon \forall w\ t(t^n[w]) \doteq t^n(t[w]) \qquad t\overbrace{t \ldots t}^{n}(a) = \underbrace{t \ldots t}_{n}\, t(a)$$

$$(t^n t^m)\ \textit{Composite}\colon \forall w\ t^n(t^m[w]) \doteq t^m(t^n[w]) \qquad \overbrace{t \ldots t}^{n}\overbrace{t \ldots t}^{m}(a) = \underbrace{t \ldots t}_{m}\underbrace{t \ldots t}_{n}(a)$$

Equation (tt^n) follows by induction on $n \in \mathbb{N}$.[21] Now, equation ($t^n t^m$) follows also by induction on $n \in \mathbb{N}$.[22]

Also, consider the generalizations of (ad.⊳) and (ad.⊳) to composite successors as follows.

(ad s^k) Addition of composite successors $a + \overbrace{s \ldots s}^{k}(b) = \underbrace{s \ldots s}_{k}(a + b)$

$$\forall u \forall v \left[s^k \begin{matrix} u \\ (v) \end{matrix}\right] \doteq s^k \left(ad \begin{bmatrix} u \\ v \end{bmatrix}\right), for\ every\ k \in \mathbb{N}$$

(ad $\overline{k}$) Addition of numeral $a + \overbrace{s \ldots s}^{k}(0) = \underbrace{s \ldots s}_{k}(a)$

$$\forall u\ ad \begin{bmatrix} u \\ \overline{k} \end{bmatrix} \doteq s^k(u), for\ every\ k \in \mathbb{N}$$

The proof of (ad s^k) is by induction on $k \in \mathbb{N}$. Now, (ad s^k) and (ad.⊥) entail (ad $\overline{k}$).

7.1. Specific associativity of addition

We will now examine how one can establish a specific version of associativity of addition. Consider the following specific version of associativity of addition (a+(b+k) = (a+b)+k):

$$(Aad\_)\forall u \forall v\ ad \left(\begin{matrix} u \\ ad \begin{bmatrix} v \\ \overline{k} \end{bmatrix} \end{matrix}\right) \doteq ad \left(\begin{matrix} ad \begin{bmatrix} u \\ v \end{bmatrix} \\ \overline{k} \end{matrix}\right), for\ every\ natural\ k \in \mathbb{N}$$

With equations (ad $\overline{k}$) and (ad s^k), we can establish directly this associative property (Aad_) as follows:

---

21 $(0)\colon t(t^0[w]) = t[w] = t^0(t[w])$; $(s)\colon t\big(t^{s(n)}[w]\big) = t\big(t(t^n[w])\big) \overset{(HI)}{\doteq} t\big(t^n(t[w])\big) = t^{s(n)}(t[w])$.

22 $(0)\colon t^0(t^m[w]) = t^m[w] = t^m(t^0[w])$; $(s)\colon t^{s(n)}(t^m[w]) = t(t^n(t^m[w])) \overset{(HI)}{\doteq} t(t^m(t^n[w])) \overset{(tt^n)}{\doteq} t^m\big(t(t^n[w])\big) = t^m(t^{s(n)}[w])$.

$$ad\left(\begin{matrix}\mathrm{u}\\ ad\left[\begin{matrix}\mathrm{v}\\ \bar{k}\end{matrix}\right]\end{matrix}\right) \overset{(ad\ \bar{k})}{\doteq} ad\left[\begin{matrix}\mathrm{u}\\ s^k(\mathrm{v})\end{matrix}\right] \overset{(ad\ \bar{k})}{\doteq} s^k\left(ad\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\end{matrix}\right]\right) \overset{(ad\ \bar{k})}{\doteq} ad\left(\begin{matrix}ad\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\end{matrix}\right]\\ \bar{k}\end{matrix}\right)$$

7.2. Specific commutativity of addition

We will now examine how one can establish particular versions of commutativity of addition. Consider the following quantifier-free version of commutativity of addition:

$$(Kad_{\downarrow}) \qquad ad\left[\begin{matrix}\bar{\mathrm{m}}\\ \bar{n}\end{matrix}\right] \doteq ad\left[\begin{matrix}\bar{\mathrm{n}}\\ \bar{\mathrm{m}}\end{matrix}\right], for\ every\ natural\ m,n \in \mathbb{N}$$

With , (ad $\bar{\mathrm{n}}$) instance ($\mathrm{s^n\,s^m}$) (of ($\mathrm{t^n\,t^m}$) ($\mathrm{s^n\,s^m}$)) and (ad $\bar{\mathrm{m}}$), one can show directly this commutative property ($\mathrm{K+_{\downarrow}}$) as follows:

$$ad\left[\begin{matrix}\bar{\mathrm{m}}\\ \bar{\mathrm{n}}\end{matrix}\right] \overset{(ad\ \bar{n})}{\doteq} s^n(\bar{m}) = s^n\left(s^m(0)\right) \overset{(s^n s^m)}{\doteq} s^m\left(s^n(0)\right) = s^m(\bar{n}) \overset{(ad\ \bar{k})}{\doteq} ad\left[\begin{matrix}\bar{\mathrm{n}}\\ \bar{\mathrm{m}}\end{matrix}\right]$$

8. General Conditions: value recursion

We will now examine some general conditions for two properties of a value recursive binary operation (cf. 4.2 in Section 4). Consider a binary function q with the following universal value recursive formulation:

$$\begin{cases}\forall \mathrm{u}\ q\left[\begin{matrix}\mathrm{u}\\ 0\end{matrix}\right] \doteq e(\mathrm{u}) & case\ (\perp)\\ \forall \mathrm{u}\forall \mathrm{v}\ q\left[\begin{matrix}\mathrm{u}\\ s(\mathrm{v})\end{matrix}\right] \doteq r\left(q\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\end{matrix}\right]\right) & case\ (\triangleright)\end{cases} \qquad (\mathrm{q})$$

This formulation (q) has unary basis function e(u) and unary step function r(w).

We wish to establish conditions for some properties of such a binary operation q. We will examine conditions for commutativity (in 8.1) and for associativity (in 8.2).

8.1 Conditions for commutativity: value recursion

We will now characterize the commutative value recursive binary operations. Consider the following universal formulation of commutativity of q:

$$(K\ q_\forall)\ \ \forall x \forall y\ q\begin{bmatrix} x \\ y \end{bmatrix} \doteq q\begin{bmatrix} y \\ x \end{bmatrix} \qquad\qquad aqb = bqa$$

Analogy with addition suggests the following two universal properties.

$$(0q_\forall)\ \ \forall v\ q\begin{bmatrix} 0 \\ v \end{bmatrix} \doteq e(v) \qquad\qquad Value\ at\ left\ zero\colon 0\ q\ b\ =\ e(b)$$

$$(sq_\forall)\ \ \forall u \forall v\ q\begin{bmatrix} s(u) \\ v \end{bmatrix} \doteq r\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right) \qquad Value\ at\ left\ successor\colon s(a)\ q\ b\ =\ r(a\ q\ b)$$

Now, consider $\overline{3}$ q 2 and 2 q 3. Let us evaluate them and compare the results. The above formulation (q) evaluates them as follows.

($\overline{3}$q$\overline{2}$) It evaluates $q\left[\begin{smallmatrix} \overline{3} \\ \overline{2} \end{smallmatrix}\right]$ *to* $(r^2\,e\,s^3)(0) = (r\,r\,e\,s\,s\,s)(0)$ as follows:

$$q\begin{bmatrix} \overline{3} \\ ss(0) \end{bmatrix} \overset{(\triangleright)}{\doteq} r\left(q\begin{bmatrix} \overline{3} \\ s(0) \end{bmatrix}\right) \overset{(\triangleright)}{\doteq} r\left(r\left(q\begin{bmatrix} \overline{3} \\ 0 \end{bmatrix}\right)\right) \overset{(\perp)}{\doteq} r\left(r\left(e(\overline{3})\right)\right) = r\left(r\left(e(sss(0))\right)\right)$$

($\overline{2}$q$\overline{3}$) It evaluates $q\left[\begin{smallmatrix} \overline{2} \\ \overline{3} \end{smallmatrix}\right]$ *to* $(r^3\,e\,s^2)(0) \coloneqq (r\,r\,r\,e\,s\,s)(0)$ as follows:

$$q\begin{bmatrix} \overline{2} \\ sss(0) \end{bmatrix} \overset{(\triangleright)}{\doteq} r\left(q\begin{bmatrix} \overline{2} \\ ss(0) \end{bmatrix}\right) \overset{(\triangleright)}{\doteq} r\left(r\left(q\begin{bmatrix} \overline{2} \\ s(0) \end{bmatrix}\right)\right) \overset{(\triangleright)}{\doteq} r\left(r\left(r\left(q\begin{bmatrix} \overline{2} \\ 0 \end{bmatrix}\right)\right)\right) \overset{(\perp)}{\doteq} r\left(r\left(r(e(\overline{2}))\right)\right)$$

If we had e s = r e, then both results $(r^2\,e\,s^3)(0)$ and $(r^3\,e\,s^2)(0)$ would be equal.[23] This example suggests yet another universal condition, as follows.

($es_\forall$) $\forall w\ e[s(w)] \doteq r(e[w])$ Basis at successor is step at basis: [e s](b)=[r e](b)

$$\begin{array}{ccc} W & \overset{s}{\rightarrow} & W \\ e\downarrow & & \downarrow e \\ W & \overset{r}{\rightarrow} & W \end{array}$$

23 Indeed, both $r^2\,es^3$ and $r^3\,es^2$ would be equal to r^5 e: $r^2\,es^3 = r^2\,\underline{e\,s}\ s\,s = r^2\,r\,\underline{e\,s}\ s = r^2\,r\,r\,\underline{e\,s} = r^2\ r\,r\,r\,e$, and $r^3\,e\,s^2 = r^3\,\underline{e\,s}\,s = r^3\,r\,\underline{e\,s} = r^3\,r\,r\,e$.

We can now see that we have two alternative characterizations for commutativity. For a binary operation q satisfying the value recursive formulation (q), the following conditions are equivalent.

$$(Kq_\forall)\ \ \forall x \forall y\ q\begin{bmatrix} x \\ y \end{bmatrix} \doteq q\begin{bmatrix} y \\ x \end{bmatrix} \qquad \textbf{commutativity of q}$$

$$(Vq_\forall)\ \textbf{Values}\ (0q_\forall)\ \ \forall \mathrm{v}\ q\begin{bmatrix} 0 \\ \mathrm{v} \end{bmatrix} \doteq e(\mathrm{v})\ \&\ (sq_\forall)\colon \forall \mathrm{u} \forall \mathrm{v}\ q\begin{bmatrix} s(\mathrm{u}) \\ \mathrm{v} \end{bmatrix} \doteq r\left(q\begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix}\right)$$

$$(es_\forall)\ \ \forall \mathrm{w}\ e[s(\mathrm{w})] \doteq r[e(\mathrm{w})] \qquad \textit{basis at successor is step at basis}$$

Proof: We will show $(\mathrm{Kq}_\forall) \Rightarrow (\mathrm{es}_\forall) \Rightarrow (\mathrm{Vq}_\forall) \Rightarrow (\mathrm{Kq}_\forall)$.

$(\mathrm{Kq}_\forall) \Rightarrow (\mathrm{es}_\forall)$ Commutativity of q yields, by (q), $(\mathrm{es}_\forall)$ as follows:

$$e[s(\mathrm{w})] \overset{(q.\perp)}{\doteq} q\begin{bmatrix} s(\mathrm{w}) \\ 0 \end{bmatrix} \overset{(Kq)}{\doteq} q\begin{bmatrix} 0 \\ s(\mathrm{w}) \end{bmatrix} \overset{(q.\rhd)}{\doteq} r\left(q\begin{bmatrix} 0 \\ \mathrm{w} \end{bmatrix}\right) \overset{(Kq)}{\doteq} r\left(q\begin{bmatrix} \mathrm{w} \\ 0 \end{bmatrix}\right) \overset{(q.\perp)}{\doteq} r[e(\mathrm{w})]$$

$(\mathrm{es}_\forall) \Rightarrow (\mathrm{Vq}_\forall)$ We show $(\mathrm{es}_\forall) \Rightarrow (0\mathrm{q}_\forall)$ and $(\mathrm{es}_\forall) \Rightarrow (\mathrm{sq}_\forall)$, both by induction on v, using (q).

$(\mathrm{es}_\forall) \Rightarrow (0\mathrm{q}_\forall)$ By induction on v, using (q).

$$(0)\colon q\begin{bmatrix} 0 \\ 0 \end{bmatrix} \overset{(q.\perp)}{\doteq} e(0); \quad (s)\ q\begin{bmatrix} 0 \\ s(\mathrm{v}) \end{bmatrix} \overset{(q.\rhd)}{\doteq} r\left(q\begin{bmatrix} 0 \\ \mathrm{v} \end{bmatrix}\right) \overset{(HI)}{\doteq} r[e(\mathrm{v})] \overset{(es)}{\doteq} e[s(\mathrm{v})]$$

$(\mathrm{es}_\forall) \Rightarrow (\mathrm{sq}_\forall)$ By induction on v, using (q).

$$(0)\ q\begin{bmatrix} s(\mathrm{u}) \\ 0 \end{bmatrix} \overset{(q.\perp)}{\doteq} e[s(\mathrm{u})] \overset{(es)}{\doteq} r[e(\mathrm{u})] \overset{(q.\perp)}{\doteq} r\left(q\begin{bmatrix} \mathrm{u} \\ 0 \end{bmatrix}\right);$$

$$(s)\ q\begin{bmatrix} s(\mathrm{u}) \\ s(\mathrm{v}) \end{bmatrix} \overset{(q.\rhd)}{\doteq} r\left(q\begin{bmatrix} s(\mathrm{u}) \\ \mathrm{v} \end{bmatrix}\right) \overset{(HI)}{\doteq} r\left(r\left(\begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix}\right)\right) \overset{(q.\perp)}{\doteq} r\left(q\begin{bmatrix} \mathrm{u} \\ s(\mathrm{v}) \end{bmatrix}\right).$$

$(\mathrm{Vq}_\forall) \Rightarrow (\mathrm{Kq}_\forall)(0\mathrm{q}_\forall)$ & $(\mathrm{sq}_\forall)$ yield, by induction on x, using (q), commutativity of q as follows:

$$(0)\ q\begin{bmatrix}0\\y\end{bmatrix} \overset{(0q)}{\doteq} e(y) \overset{(q.\perp)}{\doteq} q\begin{bmatrix}y\\0\end{bmatrix};$$

$$(s)\ q\begin{bmatrix}s(x)\\y\end{bmatrix} \overset{(sq)}{\doteq} r\left(q\begin{bmatrix}x\\y\end{bmatrix}\right) \overset{(HI)}{\doteq} r\left(q\begin{bmatrix}y\\x\end{bmatrix}\right) \overset{(q.\triangleright)}{\doteq} q\begin{bmatrix}y\\s(x)\end{bmatrix}.$$

Now, let us apply the criterion ($es_\forall$) to functions addition ad and monus mn (in Section 4), which have (similar) value recursive formulations (cf. 4.2). In each case, we will proceed in two steps:

1. first, we instantiate the criterion ($es_\forall$);
2. next, we check whether the instantiated criterion holds.

(ad) In the case of addition ad (cf. formulation (ad) in 4.1), we have

basis function: e(u)= ι(u);
step function: r(w) = s(w).

1. So, the above criterion ($es_\forall$) becomes

$$(\iota s_\forall)\colon \forall w\ \iota[s(w)] \doteq s(\iota[w]), i.e. \forall w\ s(w) \doteq s(w).$$

2. Now, ∀w s(w) ≐ s(w) is identically satisfied.

Hence, addition ad is commutative: $\forall x \forall y\ ad\begin{bmatrix}x\\y\end{bmatrix} \doteq ad\begin{bmatrix}y\\x\end{bmatrix}$

(mn) In the case of monus mn (cf. formulation (mn) in 4.1), we have

basis function: e(u)=ι(u);
step function: r(w)=pd(w).

1. So, the above criterion ($es_\forall$) becomes

$$(\iota pd_\forall)\colon \forall w\ \iota[s(w)] \doteq pd(\iota[w]), i.e. \forall w\ s(w) \doteq pd(w).$$

2. Now, $\forall w\, s(w) \doteq pd(w)$ is not satisfied at $w = 0$.[24]

This fact provides a counterexample for the commutativity of monus mn, namely

$$\neg mn \begin{bmatrix} s(0) \\ 0 \end{bmatrix} \doteq mn \begin{bmatrix} 0 \\ s(0) \end{bmatrix}.$$ [25]

8.2. Conditions for associativity: value recursion

We will now characterize the associative value recursive binary operations.

Consider the following universal formulation of associativity of q:

$$(Aq_{\forall})\ \forall x \forall y \forall z\ q\left(\begin{matrix} x \\ q\begin{bmatrix} y \\ z \end{bmatrix}\end{matrix}\right) \doteq q\left(\begin{matrix} q\begin{bmatrix} x \\ y \end{bmatrix} \\ z \end{matrix}\right) \qquad\qquad a\ q\ (b\ q\ c) = (a\ q\ b)\ q\ c$$

Analogy with the case of commutativity suggests the following two conditions.

($qe_{\forall}$) Value at basis

$$\forall u \forall v\ q\begin{bmatrix} u \\ e(v) \end{bmatrix} \doteq e\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right) \qquad\qquad a\ q\ e(b)\ =\ e(a\ q\ b)$$

This condition bridges the gap between the evaluations of $q\left(\begin{matrix} u \\ q\begin{bmatrix} v \\ 0 \end{bmatrix}\end{matrix}\right)$ and $q\left(\begin{matrix} q\begin{bmatrix} u \\ v \end{bmatrix} \\ 0 \end{matrix}\right)$.[26]

($q_{\forall}$) **Value at step** $\qquad ar(q\ (b\ q\ c)) = r((a\ q\ b)\ q\ c)$

$$\forall u \forall v \forall w\ \ q\left(\begin{matrix} u \\ r\left(q\begin{bmatrix} v \\ w \end{bmatrix}\right)\end{matrix}\right) \doteq r\left(q\left(\begin{matrix} u \\ q\begin{bmatrix} v \\ w \end{bmatrix}\end{matrix}\right)\right)$$

This condition serves to reduce the gap between the evaluations of $q\left(\begin{matrix} u \\ q\begin{bmatrix} v \\ s(w) \end{bmatrix}\end{matrix}\right)$ and $q\left(\begin{matrix} q\begin{bmatrix} u \\ v \end{bmatrix} \\ s(w) \end{matrix}\right)$.[27]

---

24 Indeed, $pd[0] \overset{(pd.\perp)}{\doteq} 0$, and $\neg\, s(0) \doteq 0$.

25 Indeed $mn\begin{bmatrix} s(0) \\ 0 \end{bmatrix} \overset{(mn)}{\doteq} s(0)$, whereas $mn\begin{bmatrix} 0 \\ s(0) \end{bmatrix} \overset{(mn)}{\doteq} 0$

26 Indeed $q\left(\begin{matrix} u \\ q\begin{bmatrix} v \\ 0 \end{bmatrix}\end{matrix}\right) \overset{(q.\perp)}{\doteq} q\begin{pmatrix} u \\ e(v) \end{pmatrix}$ and $q\left(\begin{matrix} q\begin{bmatrix} u \\ v \end{bmatrix} \\ 0 \end{matrix}\right) \overset{(q.\perp)}{\doteq} e\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right)$

27 Indeed $q\left(\begin{matrix} u \\ q\begin{bmatrix} v \\ s(w) \end{bmatrix}\end{matrix}\right) \overset{(q.\triangleright)}{\doteq} q\left(\begin{matrix} u \\ r\left(q\begin{bmatrix} v \\ w \end{bmatrix}\right)\end{matrix}\right)$ and $q\left(\begin{matrix} q\begin{bmatrix} u \\ v \end{bmatrix} \\ s(w) \end{matrix}\right) \overset{(q.\triangleright)}{\doteq} r\left(q\left(\begin{matrix} q\begin{bmatrix} u \\ v \end{bmatrix} \\ w \end{matrix}\right)\right)$

For instance, these two conditions lead to $q\left(q\begin{bmatrix}x\\y\\s(0)\end{bmatrix}\right) \doteq q\left(q\begin{pmatrix}q\begin{bmatrix}x\\y\end{bmatrix}\\s(0)\end{pmatrix}\right)$ as follows:

$$q\left(q\begin{bmatrix}x\\y\\s(0)\end{bmatrix}\right) \overset{(q.\triangleright)}{\doteq} q\left(r\left(q\begin{bmatrix}x\\y\\0\end{bmatrix}\right)\right) \overset{(qr)}{\doteq} r\left(q\left(q\begin{bmatrix}x\\y\\0\end{bmatrix}\right)\right) \overset{(q.\perp)}{\doteq} r\left(q\left(e\begin{bmatrix}x\\y\end{bmatrix}\right)\right) \overset{(qe)}{\doteq}$$

$$r\left(e\left(q\begin{bmatrix}x\\y\end{bmatrix}\right)\right) \overset{(q.\perp)}{\doteq} r\left(q\left(q\begin{bmatrix}x\\y\\0\end{bmatrix}\right)\right) \overset{(q.\triangleright)}{\doteq} q\left(q\begin{bmatrix}x\\y\\s(0)\end{bmatrix}\right)$$

We can now see that these two conditions jointly characterize associativity.

For a binary operation q satisfying the value recursive formulation (q), the following conditions are equivalent.

$$(Aq_\forall)\forall x\forall y\forall z\; q\left(q\begin{bmatrix}x\\y\\z\end{bmatrix}\right) \doteq q\left(q\begin{bmatrix}x\\y\\z\end{bmatrix}\right) \qquad \textit{associativity of } q$$

($qer_\forall$) Values at basis and at step

$$(q_\forall)\ \textit{Value at basis}\colon \forall u\forall v\; q\begin{bmatrix}u\\e(v)\end{bmatrix} \doteq e\left(\dot{q}\begin{bmatrix}u\\v\end{bmatrix}\right)$$

$$(qr_\forall)\ \textbf{Value at step}\colon \forall u\forall v\forall w\; q\left(r\left(q\begin{bmatrix}u\\v\\w\end{bmatrix}\right)\right) \doteq r\left(q\left(q\begin{bmatrix}u\\v\\w\end{bmatrix}\right)\right)$$

Proof. We will show $(qer_\forall) \Rightarrow (Aq_\forall)$ & $(Aq_\forall) \Rightarrow (qer_\forall)$.

($\Uparrow$) We show $(qer_\forall) \Rightarrow (Aq_\forall)$ by induction on z.

$$(0)\quad q\left(q\begin{bmatrix}x\\y\\0\end{bmatrix}\right) \overset{(q.\perp)}{\doteq} q\begin{bmatrix}x\\e(y)\end{bmatrix} \overset{(qe)}{\doteq} e\left(q\begin{bmatrix}x\\y\end{bmatrix}\right) \overset{(q.\perp)}{\doteq} q\left(q\begin{bmatrix}x\\y\\0\end{bmatrix}\right);$$

$$(s)\quad q\left(q\begin{bmatrix}x\\y\\s(z)\end{bmatrix}\right) \overset{(q.\triangleright)}{\doteq} q\left(r\left(q\begin{bmatrix}x\\y\\z\end{bmatrix}\right)\right) \overset{(qr)}{\doteq} r\left(q\left(q\begin{bmatrix}x\\y\\z\end{bmatrix}\right)\right) \overset{(HI)}{\doteq}$$

$$r\left(q\left(q\begin{bmatrix}x\\y\\z\end{bmatrix}\right)\right) \overset{(q.\triangleright)}{\doteq} q\left(q\begin{bmatrix}x\\y\\s(z)\end{bmatrix}\right);$$

(⇓) We show $(\mathrm{Aq}_\forall) \Rightarrow (\mathrm{qe}_\forall)$ & $(\mathrm{Aq}_\forall) \Rightarrow (\mathrm{qr}_\forall)$.

$$(Aq_\forall) \Rightarrow (qe_\forall): \; q\begin{bmatrix} u \\ e(v) \end{bmatrix} \overset{(q.\perp)}{\doteq} q\left(q\begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right) \overset{(Aq)}{\doteq} q\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right) \overset{(q.\perp)}{\doteq} e\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right)$$

$$(Aq_\forall) \Rightarrow (qr_\forall)\; q\left(r\left(q\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right) \overset{(q.\rhd)}{\doteq} q\left(q\begin{bmatrix} u \\ v \\ s(w) \end{bmatrix}\right) \overset{(Aq)}{\doteq} q\left(q\begin{bmatrix} u \\ v \end{bmatrix}\right) \overset{(q.\rhd)}{\doteq} r\left(q\left(q\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right)$$

Now, let us apply these criteria to the functions addition ad and monus mn as in 8.1. Much as before, in each case, we will proceed in two steps:

1. first, we instantiate the criteria $(\mathrm{qe}_\forall)$ and $(\mathrm{qr}_\forall)$;
2. next, we check whether the instantiated criteria hold.

(ad) For addition ad (cf. formulation (ad) in 4.1), we have (cf. 8.1):

basis function: e(u) =ι(u) ;
step function: r(w)= s(w).

So, the above conditions $(\mathrm{qe}_\forall)$ and $(\mathrm{qr}_\forall)$ become as follows.

$$(ad\iota_\forall): \; \forall u \forall v \; ad\begin{bmatrix} u \\ \iota(v) \end{bmatrix} \doteq \iota\left(ad\begin{bmatrix} u \\ v \end{bmatrix}\right), i.e., \forall u \forall v \; ad\begin{bmatrix} u \\ v \end{bmatrix} \doteq ad\begin{bmatrix} v \\ u \end{bmatrix}$$

which is identically satisfied.

$$(ads_\forall): \; \forall u \forall v \forall w \; ad\left(s\left(ad\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right) \doteq s\left(ad\left(ad\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right)$$

which follows from (ad.⊳).

Thus, addition ad is associative: $\forall x \forall y \forall z \; ad\left(ad\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) \doteq ad\left(ad\begin{bmatrix} x \\ y \end{bmatrix} \\ z\right)$

(mn) For monus mn (cf. formulation (mn) in 4.1), we have (cf. 8.1):

basis function: e(u)=ι(u);
step function: r(w)=pd(w).

So, the above conditions ($qe_\forall$) and ($qr_\forall$) become as follows.

$$(mn\iota_\forall): \ \forall u \forall v \ mn\begin{bmatrix} u \\ \iota(v) \end{bmatrix} \doteq \iota\left(mn\begin{bmatrix} u \\ v \end{bmatrix}\right), i.e. \forall u \forall v \ mn\begin{bmatrix} u \\ v \end{bmatrix} \doteq mn\begin{bmatrix} v \\ u \end{bmatrix}$$

which is identically satisfied.

$$(mnpd_\forall): \ \forall u \forall v \forall w \ mn\left(pd\left(mn\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right) \doteq pd\left(mn\left(mn\begin{bmatrix} u \\ v \\ w \end{bmatrix}\right)\right).$$

(w= 0) At w= 0, this condition becomes

$$\forall u \forall v \ mn\left(pd\left(mn\begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right)\right) \doteq pd\left(mn\left(mn\begin{bmatrix} u \\ v \\ 0 \end{bmatrix}\right)\right), \text{ i. e.,}$$

$$by\ (mn.\bot): \ \forall u \forall v \ mn\begin{pmatrix} u \\ pd[v] \end{pmatrix} \doteq pd\left(mn\begin{pmatrix} u \\ v \end{pmatrix}\right)$$

(w=0,v=0) At v=0, this last condition becomes

$$\forall u \ mn\begin{pmatrix} u \\ pd[0] \end{pmatrix} \doteq pd\left(mn\begin{bmatrix} u \\ 0 \end{bmatrix}\right), \text{ i. e.,}$$

by (pd.⊥) and (mn.⊥) : ∀u u≐ pd[u]; which is not satisfied at u=s(0).[28]

This fact provides a counterexample for the associativity of monus mn,namely $\neg\, mn\left(\begin{matrix} s(0) \\ mn\begin{bmatrix} 0 \\ s(0) \end{bmatrix} \end{matrix}\right) \doteq mn\left(mn\begin{bmatrix} s(0) \\ 0 \end{bmatrix}\right)$.[29]

We can now summarize our discussion so far about commutativity and associativity of value recursive binary operations. First, we will summarize our criteria and the analyses of addition ad and monus mn. Next, we will examine some further examples of value recursive binary operations.

The characterizations for a value recursive binary operation presented in this section serve, either to establish or to refute its commutativity, or

28 Indeed, $pd[s(0)] \overset{(pd)}{\doteq} 0$, and $\neg\, s(0) \doteq 0$.

29 Indeed, $mn\left(\begin{matrix} s(0) \\ mn\begin{bmatrix} 0 \\ s(0) \end{bmatrix} \end{matrix}\right) \overset{(mn)}{\doteq} mn\begin{pmatrix} s(0) \\ 0 \end{pmatrix} \overset{(mn.\bot)}{\doteq} s(0)$, whereas $mn\left(mn\begin{bmatrix} s(0) \\ 0 \end{bmatrix}\right) \overset{(mn.\bot)}{\doteq} mn\begin{bmatrix} s(0) \\ s(0) \end{bmatrix} \overset{(mn)}{\doteq} 0$.

associativity. They rely only on the basis and step functions of the value recursive formulation. As such, they can be employed to analyze commutativity or associativity of other binary operations satisfying value recursive formulations.

Some examples of such binary operations, with their analyses, are as follows.

(adb) Addition of double, with $\binom{a}{b} \mapsto a + 2 \cdot b$:

$$\begin{cases} adb[\mathrm{u}, 0] \doteq \mathrm{u} & case\ (\bot) \\ adb[\mathrm{u}, s(\mathrm{v})] \doteq ss(adb[\mathrm{u}, \mathrm{v}]) & case\ (\rhd) \end{cases} \qquad (\mathrm{adb})$$

This formulation (adb) has

basis function: ι(u) = u;
step function: ss(w).

So, conditions ($\mathrm{es}_\forall$),($\mathrm{qe}_\forall$) and ($\mathrm{qr}_\forall$) become as follows

($\mathrm{is}_\forall$)	($\mathrm{adb}\iota_\forall$)	($\mathrm{qr}_\forall$)
$\forall \mathrm{w}$	$\forall \mathrm{u} \forall \mathrm{v}$	$\forall \mathrm{u} \forall \mathrm{v} \forall \mathrm{w}$
$\iota[s(\mathrm{w})]$	$adb\left[{}^{\mathrm{u}}_{\iota(\mathrm{v})}\right]$	$adb\left(ss\left(adb\left[{}^{\mathrm{u}}_{\mathrm{w}}\right]\right)\right)$
$\cdot \Vdash$	$\cdot \Vdash$	$\cdot \Vdash$
$ss[\iota(\mathrm{w})]$	$\iota\left(adb\left[{}^{\mathrm{u}}_{\mathrm{v}}\right]\right)$	$ss\left(adb\left(adb\left[{}^{\mathrm{u}}_{\mathrm{w}}\right]\right)\right)$

Now, conditions ($\mathrm{is}_\forall$) and ($\mathrm{qr}_\forall$) fail.
Hence, addition of double adb is non-commutative and non-associative.

(dba) Double of addition, with $\binom{a}{b} \mapsto 2 \cdot (a + b)$:

$$\begin{cases} dba[\mathrm{u}, 0] \doteq db(\mathrm{u}) & case\ (\bot) \\ dba[\mathrm{u}, s(\mathrm{v})] \doteq ss(bda[\mathrm{u}, \mathrm{v}]) & case\ (\rhd) \end{cases} \qquad (\mathrm{adb})$$

This formulation (dba) has

basis function: db(u);
step function: ss(w)

So, conditions ($es_{\forall}$), ($qe_{\forall}$) and ($qr_{\forall}$) become as follows

($dbs_{\forall}$)	($dba\ db_{\forall}$)	($dba\ ss_{\forall}$)
$\forall w$ $db[s(w)]$	$\forall u \forall v$ $dba\begin{bmatrix} u \\ db(v) \end{bmatrix}$	$\forall u \forall v \forall w$ $dba\left(\begin{matrix} u \\ ss\left(dba\begin{bmatrix} v \\ w \end{bmatrix}\right) \end{matrix}\right)$
$\cdot\Vdash$	$\cdot\Vdash$	$\cdot\Vdash$
$ss[db(w)]$	$db\left(dba\begin{bmatrix} u \\ v \end{bmatrix}\right)$	$ss\left(dba\left(\begin{matrix} u \\ dba\begin{bmatrix} v \\ w \end{bmatrix} \end{matrix}\right)\right)$

Now, condition ($dbs_{\forall}$) holds, whereas conditions ($dba\ db_{\forall}$) and ($dba\ ss_{\forall}$) fail. Hence, double of addition dba is commutative and non-associative.

(γ) Left projection, with $\binom{a}{b} \mapsto a$:

$$\begin{cases} \gamma[u,0] \doteq u & case\ (\bot) \\ \gamma[u,s(v)] \doteq u & case\ (\rhd) \end{cases} \qquad (\gamma)$$

This formulation (γ) has

basis function: ι(u)= u;
step function: ι(w) = w.

So, conditions ($es_{\forall}$), ($qe_{\forall}$) and ($qr_{\forall}$) become as follows

($\iota s_{\forall}$)	($\gamma\ \iota_{\forall}$)	$\gamma\ \iota_{\forall}$
$\forall w$ $\iota[s(w)]$	$\forall u \forall v$ $\gamma\begin{bmatrix} u \\ \iota(v) \end{bmatrix}$	$\forall u \forall v \forall w$ $\gamma\left(\begin{matrix} u \\ \iota\left(\gamma\begin{bmatrix} v \\ w \end{bmatrix}\right) \end{matrix}\right)$
$\cdot\Vdash$	$\cdot\Vdash$	$\cdot\Vdash$
$\iota[\iota(w)]$	$\iota\left(\gamma\begin{bmatrix} u \\ v \end{bmatrix}\right)$	$\iota\left(\gamma\left(\begin{matrix} u \\ \gamma\begin{bmatrix} v \\ w \end{bmatrix} \end{matrix}\right)\right)$

Now, condition ($\iota s_{\forall}$) fails, whereas conditions ($\gamma\ \iota_{\forall}$) and ($\gamma\ \iota_{\forall}$) both hold. Hence, left projection γ is non-commutative and associative.

(δ) Right projection, with $\binom{a}{b} \mapsto b$:

$$\begin{cases} \delta[\mathrm{u}, 0] \doteq 0 & case\ (\bot) \\ \delta[\mathrm{u}, s(\mathrm{v})] \doteq s(\delta[\mathrm{u}, \mathrm{v}]) & case\ (\rhd) \end{cases} \qquad (\delta)$$

This formulation (δ) has

basis function: ζ(u)= 0;
step function: s(w).

So, conditions ($es_\forall$), ($qe_\forall$) and ($qr_\forall$) become as follows

($\zeta s_\forall$)	($\delta\ \zeta_\forall$)	($\zeta s_\forall$)
$\forall \mathrm{w}$ $\zeta[s(\mathrm{w})]$	$\forall \mathrm{u} \forall \mathrm{v}$ $\delta\left[\begin{matrix}\mathrm{u}\\ \zeta(\mathrm{v})\end{matrix}\right]$	$\forall \mathrm{u} \forall \mathrm{v} \forall \mathrm{w}$ $\delta\left(s\left(\delta\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\\ \mathrm{w}\end{matrix}\right]\right)\right)$
$\cdot\Vdash$	$\cdot\Vdash$	$\cdot\Vdash$
$s[\zeta(\mathrm{w})]$	$\zeta\left(\delta\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\end{matrix}\right]\right)$	$s\left(\delta\left(\delta\left[\begin{matrix}\mathrm{u}\\ \mathrm{v}\\ \mathrm{w}\end{matrix}\right]\right)\right)$

Now, condition ($\zeta s_\forall$) fails, whereas conditions ($\delta\ \zeta_\forall$) and ($\zeta s_\forall$) both hold. Hence, right projection δ is non-commutative and associative.

In the next section, we shall characterize commutativity and associativity of a binary operation satisfying a simple recursive formulation.

9. General Conditions: simple recursion

We will now examine some general conditions for two properties of a simple recursive binary operation (cf. 4.2 in Section 4).

Consider a binary function with the following universal simple recursive formulation:

$$\begin{cases} \forall u\ \acute{m}\begin{bmatrix} u \\ 0 \end{bmatrix} \doteq \acute{p}(u) & case\ (\bot) \\ \forall u \forall v\ \acute{m}\begin{bmatrix} u \\ s(v) \end{bmatrix} \doteq \acute{t}\left(\begin{matrix} u \\ \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \end{matrix}\right) & case\ (\rhd) \end{cases} \qquad (\acute{m})$$

This formulation (ḿ) has unary basis function ṕ(u) and binary step function $\acute{t}\binom{u}{w}$.

We wish to establish conditions for some properties of such a binary operation ḿ. We will examine conditions for commutativity (in 9.1) and for associativity (in 9.2).

9.1. Conditions for commutativity: simple recursion

We will now characterize the commutative simple recursive binary operations.

Consider the following universal formulation of commutativity of :

$$(K\acute{m}_{\forall})\ \forall x \forall y\ \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \doteq \acute{m}\begin{bmatrix} y \\ x \end{bmatrix} \qquad a\ \acute{m}\ b = b\ \acute{m}\ a$$

Analogy with the case of value recursion suggests the following three properties.

(V$\acute{m}_{\forall}$) Values of ḿ at left zero and at left successor

$$(0\acute{m}_{\forall})\ \ \forall v\ \acute{m}\begin{bmatrix} 0 \\ v \end{bmatrix} \doteq \acute{p}[v] \qquad \acute{m}\ at\ left\ 0\colon 0\ \acute{m}\ b\ =\ \acute{p}(b)$$

$$(s\acute{m}_{\forall})\ \ \forall u \forall v\ \acute{m}\begin{bmatrix} s(u) \\ v \end{bmatrix} \doteq \acute{t}\left(\begin{matrix} v \\ \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \end{matrix}\right) \qquad \acute{m}\ at\ left\ s\colon s(a)\ \acute{m}b\ =\ b\ \acute{t}\ (a\ \acute{m}b)$$

$$(\acute{p}s_{\forall})\ \ \forall w\ \acute{p}[s(w)] \doteq \acute{t}\begin{pmatrix} 0 \\ \acute{p}[w] \end{pmatrix} \qquad Basis\ \acute{p}\ at\ successor\colon \acute{p}\big(s(b)\big) =\ 0\ \acute{t}\acute{p}(b)$$

Let us see some examples of how these conditions are used to establish commutativity.

- Conditions (V$\acute{m}_{\forall}$) lead to $\acute{m}\begin{bmatrix} s(0) \\ y \end{bmatrix} \doteq \acute{m}\begin{bmatrix} y \\ s(0) \end{bmatrix}$ as follows:

$$\acute{m}\begin{bmatrix} s(0) \\ y \end{bmatrix} \overset{(s\acute{m})}{\doteq} \acute{t}\left(\begin{matrix} y \\ \acute{m}\begin{bmatrix} 0 \\ y \end{bmatrix} \end{matrix}\right) \overset{(0\acute{m})}{\doteq} \acute{t}\begin{pmatrix} y \\ \acute{p}(y) \end{pmatrix} \overset{(\acute{m}.\bot)}{\doteq} \acute{t}\left(\begin{matrix} y \\ \acute{m}\begin{bmatrix} 0 \\ y \end{bmatrix} \end{matrix}\right) \overset{(\acute{m}.\rhd)}{\doteq} \acute{m}\begin{bmatrix} y \\ s(0) \end{bmatrix}.$$

- Conditions (p′s$_\forall$) lead to $\acute{m}\begin{bmatrix}s(0)\\0\end{bmatrix} \doteq \acute{m}\begin{bmatrix}0\\s(0)\end{bmatrix}$ as follows:

$$\acute{m}\begin{bmatrix}s(0)\\0\end{bmatrix} \overset{(\acute{m}.)}{\doteq} \acute{p}(s(0)) \overset{(\acute{p}s_\forall)}{\doteq} \acute{t}\begin{pmatrix}0\\\acute{p}[0]\end{pmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{t}\left(\begin{matrix}0\\\acute{m}\begin{bmatrix}0\\0\end{bmatrix}\end{matrix}\right) \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{bmatrix}0\\s(0)\end{bmatrix}.$$

We can now see that we have two alternative characterizations for commutativity.

For a binary operation q satisfying the value recursive formulation , the fol- lowing conditions are equivalent.

$(K\acute{m}_\forall)\ \forall x \forall y\ \acute{m}\begin{bmatrix}x\\y\end{bmatrix} \doteq \acute{m}\begin{bmatrix}y\\x\end{bmatrix}$ *commutativity of ḿ*

$(V\acute{m}_\forall)\ Values\ (0\acute{m}_\forall)\colon \forall v\ \acute{m}\begin{bmatrix}0\\v\end{bmatrix} \doteq \acute{p}(v)\ \ \&\ \ (s\acute{m}_\forall)\colon \forall u \forall v\ \acute{m}\begin{bmatrix}s(u)\\v\end{bmatrix} \doteq \acute{t}\left(\begin{matrix}v\\\acute{m}\begin{bmatrix}u\\v\end{bmatrix}\end{matrix}\right)$

$(\acute{p}s_\forall)\ \forall w\ \acute{p}[s(w)] \doteq \acute{t}\begin{pmatrix}0\\\acute{p}[w]\end{pmatrix}$ *basis at successor*

Proof: We will show (Kḿ$_\forall$)⇒(p′s$_\forall$)⇒(Vḿ$_\forall$)⇒(Kḿ$_\forall$)

(Kḿ$_\forall$) ⇒ (p′s$_\forall$) Commutativity of ḿ yields, by (ḿ) (p′s)as follows:

$$\acute{p}[s(w)] \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{bmatrix}s(w)\\0\end{bmatrix} \overset{(K\acute{m}_\forall)}{\doteq} \acute{m}\begin{bmatrix}0\\s(w)\end{bmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{t}\left(\begin{matrix}0\\\acute{m}\begin{bmatrix}0\\v\end{bmatrix}\end{matrix}\right) \overset{(\acute{m}.\perp)}{\doteq} \acute{t}\begin{pmatrix}0\\\acute{p}[w]\end{pmatrix}$$

(p′s) ⇒ (Vḿ$_\forall$) We show (p′s) ⇒ (0ḿ$_\forall$) and (p′s) ⇒ (sḿ$_\forall$), both by induction on v, using (ḿ).

(p′s) ⇒ (0ḿ$_\forall$) By induction on v, using (ḿ).

$$(0)\ \acute{m}\begin{bmatrix}0\\0\end{bmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{p}(0);$$

$$(s)\ \acute{m}\begin{bmatrix}0\\s(v)\end{bmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{t}\left(\begin{matrix}0\\\acute{m}\begin{bmatrix}0\\v\end{bmatrix}\end{matrix}\right) \overset{(HI)}{\doteq} \acute{t}\begin{pmatrix}0\\\acute{p}[v]\end{pmatrix} \overset{(\acute{p}s)}{\doteq} \acute{p}[s(v)];$$

(p′s) ⇒ (sḿ$_\forall$)By induction on v, using (ḿ) .

$$(0)\ \acute{m}\begin{bmatrix}s(u)\\0\end{bmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{p}(s(u)) \overset{(\acute{p}s)}{\doteq} \acute{t}\begin{pmatrix}0\\\acute{p}[u]\end{pmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{t}\left(\begin{matrix}0\\\acute{m}\begin{bmatrix}u\\0\end{bmatrix}\end{matrix}\right);$$

$$(s)\ \acute{m}\begin{bmatrix}s(u)\\s(v)\end{bmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{t}\left(\begin{matrix}s[u]\\\acute{m}\begin{bmatrix}s[u]\\v\end{bmatrix}\end{matrix}\right) \overset{(HI)}{\doteq} ; \acute{t}\left(\begin{matrix}s[u]\\v\\\acute{t}\left(\begin{matrix}v\\\acute{m}\begin{bmatrix}u\\v\end{bmatrix}\end{matrix}\right)\end{matrix}\right) \overset{(\acute{m}.\triangleright)}{\doteq} \acute{t}\left(\begin{matrix}v\\\acute{m}\begin{bmatrix}s[u]\\v\end{bmatrix}\end{matrix}\right)$$

$(\forall\acute{m}_{\forall})\Rightarrow(K\acute{m}_{\forall})\ (0\acute{m}_{\forall})$ & $(s\acute{m}_{\forall})$ yield, by induction on x, using $(\acute{m})$, commutativity of $(\acute{m})$ as follows:

$$(0)\ \acute{m}\begin{bmatrix}0\\y\end{bmatrix} \overset{(0\acute{m})}{\doteq} \acute{p}(y) \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{bmatrix}y\\0\end{bmatrix};$$

$$(s)\ \acute{m}\begin{bmatrix}s(x)\\y\end{bmatrix} \overset{(s\acute{m})}{\doteq} \acute{t}\left(\begin{matrix}y\\ \acute{m}\begin{bmatrix}x\\y\end{bmatrix}\end{matrix}\right) \overset{(HI)}{\doteq} \acute{t}\left(\begin{matrix}y\\ \acute{m}\begin{bmatrix}y\\x\end{bmatrix}\end{matrix}\right) \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{bmatrix}y\\s(x)\end{bmatrix}.$$

Now, let us apply the criterion $(\acute{p}s_{\forall})$ to functions multiplication mt and exponentiation pw (in Section 4), which have (similar) value recursive formulations. As before, in each case, we will proceed in two steps as follows:

1. first, we instantiate the criterion $(\acute{p}s_{\forall})$;
2. next, we check whether the instantiated criterion holds.

(mt) In the case of multiplication mt (cf. formulation (mt) in 4.1), we have:

$$\textit{basis function: } \acute{p}(u) = \zeta(u); \qquad \textit{step function: } \acute{t}\begin{pmatrix}u\\w\end{pmatrix} = ad\begin{pmatrix}u\\w\end{pmatrix}$$

So, the above criterion $(\acute{p}s_{\forall})$ becomes

$$(\zeta s)\colon \forall w\ \ \zeta[s(w)] \doteq ad\begin{pmatrix}0\\ \zeta[w]\end{pmatrix}, i.e.\ 0 \doteq ad\begin{pmatrix}0\\0\end{pmatrix}.$$

Now, $0 \doteq ad\begin{pmatrix}0\\0\end{pmatrix}$ follows from (ad.⊥)

Hence, multiplication mt is commutative: $\forall x \forall y\ mt\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}y\\x\end{bmatrix}$.

(pw) In the case of exponentiation pw (cf. formulation (pw) in 4.1), we have:

$$\textit{basis function: } \acute{p}(u) = \upsilon(u); \qquad \textit{stepfunction: } \acute{t}\begin{pmatrix}u\\w\end{pmatrix} = mt\begin{pmatrix}u\\w\end{pmatrix}$$

1. So, the above criterion $(\acute{p}s_{\forall})$ becomes

$$(\upsilon s)\colon \forall w\ \upsilon[s(w)] \doteq mt\begin{pmatrix}0\\ \upsilon[w]\end{pmatrix}, i.e.\ \ s(0) \doteq mt\begin{pmatrix}0\\ s(0)\end{pmatrix}.$$

2. Now, $s(0) \doteq mt\begin{pmatrix} 0 \\ s(0) \end{pmatrix}$ is *not* satisfied.[30]

This fact provides a counterexample for the commutativity exponentiation pw, namely

$$\neg pw\begin{bmatrix} s(0) \\ 0 \end{bmatrix} \doteq pw\begin{bmatrix} 0 \\ s(0) \end{bmatrix}.$$[31]

9.2. Conditions for associativity: simple recursion

We will now characterize the associative value recursive binary operations.

Consider the following universal formulation of associativity of ḿ:

$$(A\acute{m}_{\forall})\ \forall x\forall y\forall z\ \acute{m}\begin{pmatrix} x \\ \acute{m}\begin{bmatrix} y \\ z \end{bmatrix} \end{pmatrix} \doteq \acute{m}\left(\begin{matrix} \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \\ z \end{matrix}\right) \qquad a\ \acute{m}(b\ \acute{m}\ c) = (a\ \acute{m}\ b)\ \acute{m}\ c$$

Analogy with the case of value recursion suggests the following two properties.

$$(\acute{m}\acute{p}_{\forall})\ \text{Value at basis} \qquad a\ \acute{m}\ \acute{p}(b) = \acute{p}\ (a\ \acute{m}\ b)$$

$$\forall u\forall v\ \acute{m}\begin{bmatrix} y \\ \acute{p}(v) \end{bmatrix} \doteq \acute{p}\left(\acute{m}\begin{bmatrix} y \\ v \end{bmatrix}\right)$$

This condition bridges the gap between the evaluations of $\acute{m}\begin{pmatrix} u \\ \acute{m}\begin{bmatrix} v \\ 0 \end{bmatrix} \end{pmatrix}$ and $\acute{m}\left(\acute{m}\begin{bmatrix} u \\ v \end{bmatrix}\right)$.[32]

$$(\acute{m}\acute{t}_{\forall})\ \text{Value at step} \qquad a\ \acute{m}[b\acute{t}(b\ \acute{m}c)] = (a\ \acute{m}b)\acute{t}[a\ \acute{m}(b\ \acute{m}c)]$$

$$\forall x\forall y\forall z\ \ \acute{m}\left(\begin{matrix} u \\ v \\ \acute{t}\left(\acute{m}\begin{bmatrix} v \\ w \end{bmatrix}\right) \end{matrix}\right) \doteq \acute{t}\left(\begin{matrix} \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \\ \acute{m}\left(\begin{matrix} u \\ \acute{m}\begin{bmatrix} v \\ w \end{bmatrix} \end{matrix}\right) \end{matrix}\right)$$

---

30 Indeed, $mt\begin{pmatrix} 0 \\ s(0) \end{pmatrix} \overset{(mt)}{\doteq} 0$, and $\neg s(0) \doteq 0$.

31 Indeed, $pw\begin{bmatrix} s(0) \\ 0 \end{bmatrix} \overset{(pw.\perp)}{\doteq} s(0)$, whereas $pw\begin{bmatrix} 0 \\ s(0) \end{bmatrix} \overset{(pw)}{\doteq} 0$.

32 Indeed, $\acute{m}\begin{pmatrix} u \\ \acute{m}\begin{bmatrix} v \\ 0 \end{bmatrix} \end{pmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{pmatrix} u \\ \acute{p}(v) \end{pmatrix}$ and $\acute{m}\left(\acute{m}\begin{bmatrix} u \\ v \end{bmatrix}\right) \overset{(\acute{m}.\perp)}{\doteq} \acute{p}\left(\acute{m}\begin{bmatrix} u \\ v \end{bmatrix}\right)$.

This condition serves to reduce the gap between the evaluations of $\acute{m}\begin{pmatrix}u\\ \acute{m}\begin{bmatrix}v\\ s(w)\end{bmatrix}\end{pmatrix}$ and $\acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ s(w)\end{pmatrix}$.[33]

For instance, these two conditions lead to $\acute{m}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ s(0)\end{bmatrix}\end{pmatrix} \doteq \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ s(0)\end{pmatrix}$ as follows:

$$\acute{m}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ s(0)\end{bmatrix}\end{pmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} q\left(\acute{t}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ 0\end{bmatrix}\end{pmatrix}\right) \overset{(\acute{m}\acute{t})}{\doteq} \acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ \acute{m}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ 0\end{bmatrix}\end{pmatrix}\end{matrix}\right) \overset{(\acute{m}.\perp)}{\doteq}$$

$$\acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ \acute{m}\begin{pmatrix}x\\ \acute{p}(y)\end{pmatrix}\end{matrix}\right) \overset{(\acute{m}\acute{p})}{\doteq} \acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ \acute{p}\left(\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\right)\end{matrix}\right) \overset{(\acute{m}.\perp)}{\doteq} \acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ 0\end{pmatrix}\end{matrix}\right) \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ s(0)\end{pmatrix}.$$

We can now see that these two conditions jointly characterize associativity.

For a binary operation satisfying the simple recursive formulation ḿ, the following conditions are equivalent.

$$(A\acute{m}_{\forall})\ \forall x\forall y\forall z\ \acute{m}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ z\end{bmatrix}\end{pmatrix} \doteq \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ z\end{pmatrix} \qquad \textit{associativity of } \acute{m}$$

(ḿ ṕ ť$_{\forall}$) Values at basis and at step

$$(\acute{m}\acute{p}\acute{t}_{\forall})\ \textbf{Values at basis:}\ \forall u\forall v\ \acute{m}\begin{bmatrix}u\\ \acute{p}(v)\end{bmatrix} \doteq \acute{p}\left(\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\right)$$

$$(\acute{m}\acute{t}_{\forall})\ \textbf{Values at step:}\ \forall u\forall v\forall w\ \acute{m}\begin{pmatrix}u\\ v\\ \acute{t}\left(\acute{m}\begin{bmatrix}v\\ w\end{bmatrix}\right)\end{pmatrix} \doteq \acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ \acute{m}\begin{pmatrix}u\\ \acute{m}\begin{bmatrix}v\\ w\end{bmatrix}\end{pmatrix}\end{matrix}\right)$$

Proof. We will show (ḿ ṕ ť$_{\forall}$) ⇒ (Aḿ$_{\forall}$) & (Aḿ$_{\forall}$) ⇒ (ḿ ṕ ť$_{\forall}$)

(⇑) We show (ḿ ṕ ť$_{\forall}$) ⇒ (Aḿ$_{\forall}$) by induction on z.

$$(0)\ \acute{m}\begin{pmatrix}x\\ \acute{m}\begin{bmatrix}y\\ 0\end{bmatrix}\end{pmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{pmatrix}x\\ \acute{p}(y)\end{pmatrix} \overset{(\acute{m}\acute{p})}{\doteq} \acute{p}\left(\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\right) \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}x\\ y\end{bmatrix}\\ 0\end{pmatrix};$$

33 Indeed, $\acute{m}\begin{pmatrix}u\\ \acute{m}\begin{bmatrix}v\\ s(w)\end{bmatrix}\end{pmatrix} \overset{(q.\triangleright)}{\doteq} \acute{m}\left(\acute{t}\begin{pmatrix}u\\ v\\ \acute{m}\begin{bmatrix}w\\ v\end{bmatrix}\end{pmatrix}\right)$ and $\acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ s(w)\end{pmatrix} \overset{(q.\triangleright)}{\doteq} \acute{t}\left(\begin{matrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ \acute{m}\begin{pmatrix}\acute{m}\begin{bmatrix}u\\ v\end{bmatrix}\\ w\end{pmatrix}\end{matrix}\right)$

$$(s)\ \acute{m}\begin{pmatrix} x \\ \acute{m}\begin{bmatrix} y \\ s(z) \end{bmatrix} \end{pmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{pmatrix} x \\ \acute{t}\begin{pmatrix} y \\ \acute{m}\begin{bmatrix} y \\ z \end{bmatrix} \end{pmatrix} \end{pmatrix} \overset{(\acute{m}\acute{t})}{\doteq} \acute{t}\begin{pmatrix} \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \\ x \\ \acute{m}\begin{pmatrix} \acute{m}\begin{bmatrix} y \\ z \end{bmatrix} \end{pmatrix} \end{pmatrix} \overset{(HI)}{\doteq}$$

$$\acute{t}\begin{pmatrix} \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \\ \acute{m}\begin{pmatrix} \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \\ z \end{pmatrix} \end{pmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{pmatrix} \acute{m}\begin{bmatrix} x \\ y \end{bmatrix} \\ s(z) \end{pmatrix}.$$

(⇓) We show $(A\acute{m}_\forall) \Rightarrow (\acute{m}\,\acute{p}_\forall)$ & $(A\acute{m}_\forall) \Rightarrow (\acute{m}\,\acute{t}_\forall)$

$$(A\acute{m}_\forall) \Rightarrow (\acute{m}\acute{p}_\forall)\colon \acute{m}\begin{bmatrix} u \\ \acute{p}(v) \end{bmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{m}\begin{pmatrix} u \\ \acute{m}\begin{bmatrix} v \\ 0 \end{bmatrix} \end{pmatrix} \overset{(A\acute{m})}{\doteq} \acute{m}\begin{pmatrix} \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \\ 0 \end{pmatrix} \overset{(\acute{m}.\perp)}{\doteq} \acute{p}\left(\acute{m}\begin{bmatrix} u \\ v \end{bmatrix}\right)$$

$$(A\acute{m}_\forall) \Rightarrow (\acute{m}\acute{t}_\forall)\colon \acute{m}\begin{pmatrix} u \\ v \\ \acute{t}\begin{pmatrix} v \\ \acute{m}\begin{bmatrix} v \\ w \end{bmatrix} \end{pmatrix} \end{pmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{m}\begin{pmatrix} u \\ v \\ \acute{m}\begin{bmatrix} v \\ s(w) \end{bmatrix} \end{pmatrix} \overset{(A\acute{m})}{\doteq}$$

$$\acute{m}\begin{pmatrix} \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \\ s(w) \end{pmatrix} \overset{(\acute{m}.\triangleright)}{\doteq} \acute{t}\begin{pmatrix} \acute{m}\begin{bmatrix} u \\ v \end{bmatrix} \\ u \\ \acute{m}\begin{pmatrix} v \\ \acute{m}\begin{bmatrix} v \\ w \end{bmatrix} \end{pmatrix} \end{pmatrix}$$

Now, let us apply these criteria to the functions multiplication mt and exponentiation pw used as examples in 9.1. Much as before, in each case, we will proceed in two steps:

1. first, we instantiate the criteria $(\acute{m}\,\acute{p}_\forall)$ and $(\acute{m}\,\acute{t}_\forall)$;
2. next, we check whether the instantiated criteria hold.

(mt) For multiplication mt (cf. formulation (mt) in 4.1), we have (cf. 9.1):

$$\textit{basis function}\colon \acute{p}(u) = \zeta(u); \textit{step function}\colon \acute{t}\begin{pmatrix} u \\ w \end{pmatrix} = ad\begin{pmatrix} u \\ w \end{pmatrix}$$

So, the above conditions $(\acute{m}\,\acute{p}_\forall)$ and $(\acute{m}\,\acute{t}_\forall)$ become as follows.

$$(mt\zeta_\forall)\colon \forall u \forall v\ \ mt\begin{bmatrix} u \\ \zeta(v) \end{bmatrix} \doteq \zeta\left(mt\begin{bmatrix} u \\ v \end{bmatrix}\right), i.e. \forall u\ mt\begin{bmatrix} u \\ 0 \end{bmatrix} \doteq 0;$$

which follows from (mt.⊥).

$$(mtad_{\forall})\colon \forall u \forall v \forall w\ mt\left(\begin{matrix} u \\ ad\left(\begin{matrix} v \\ mt\begin{bmatrix} v \\ w \end{bmatrix} \end{matrix}\right) \end{matrix}\right) = ad\left(\begin{matrix} mt\begin{bmatrix} u \\ v \end{bmatrix} \\ u \\ mt\left(mt\begin{bmatrix} v \\ w \end{bmatrix}\right) \end{matrix}\right);$$

which follows from distributivity.

(pw) For exponentiation pw (cf. formulation (pw) in 4.1), we have (cf. 9.1):

$$\textit{basis function}\colon \acute{p}(u) = v(u); \qquad \textit{step function}\colon \acute{t}\begin{pmatrix} u \\ w \end{pmatrix} = mt\begin{pmatrix} u \\ w \end{pmatrix}$$

So, the above conditions $(\acute{m}\acute{p}_{\forall})$ and $(\acute{m}\acute{t}_{\forall})$ become as follows.

$$(pwv_{\forall})\colon \forall u \forall v\ pw\begin{bmatrix} u \\ v(v) \end{bmatrix} \doteq v\left(pw\begin{bmatrix} u \\ v \end{bmatrix}\right), \text{ i.e., } \forall u\ pw\begin{bmatrix} u \\ s(0) \end{bmatrix} \doteq s(0);$$

which is *not* satisfied at $u = 0$.[34]

This fact provides a counter-example for the associativity of exponentiation pw, namely $\neg pw\left(\begin{matrix} 0 \\ pw\begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{matrix}\right) \doteq pw\left(pw\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$.[35]

$$(pwmt_{\forall})\colon \forall u \forall v \forall w\ pw\left(\begin{matrix} u \\ mt\left(\begin{matrix} v \\ pw\begin{bmatrix} v \\ w \end{bmatrix} \end{matrix}\right) \end{matrix}\right) \doteq mt\left(\begin{matrix} pw\begin{bmatrix} u \\ v \end{bmatrix} \\ u \\ pw\left(pw\begin{bmatrix} v \\ w \end{bmatrix}\right) \end{matrix}\right)$$

which is *not* satisfied at $u = ss(0), v = s(0), w = 0$.[36]

---

34 Indeed $pw\begin{bmatrix} 0 \\ s(0) \end{bmatrix} \overset{(pw)}{\doteq} 0.$

35 Indeed $pw\left(\begin{matrix} 0 \\ pw\begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{matrix}\right) \overset{(pw.\bot)}{\doteq} pw\begin{pmatrix} 0 \\ s(0) \end{pmatrix} \overset{(pw)}{\doteq} 0$ whereas $pw\left(pw\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) \overset{(pw.\bot)}{\doteq} pw\begin{pmatrix} s(0) \\ 0 \end{pmatrix} \overset{(\acute{m}.\bot)}{\doteq} s(0).$

36 Indeed $pw\left(\begin{matrix} s^2(0) \\ mt\left(\begin{matrix} s(0) \\ pw\begin{bmatrix} s(0) \\ 0 \end{bmatrix} \end{matrix}\right) \end{matrix}\right) \overset{(pw.\bot)}{\doteq} pw\left(\begin{matrix} s^2(0) \\ mt\begin{pmatrix} s(0) \\ s(0) \end{pmatrix} \end{matrix}\right) \overset{(mt)}{\doteq} pw\begin{pmatrix} s^2(0) \\ s(0) \end{pmatrix} \overset{(pw)}{\doteq} s^2(0),$ whereas

$mt\left(\begin{matrix} pw\begin{pmatrix} s^2(0) \\ s(0) \end{pmatrix} \\ s^2(0) \\ pw\begin{pmatrix} s(0) \\ 0 \end{pmatrix} \end{matrix}\right) \overset{(pw)}{\doteq} mt\left(pw\left(\begin{matrix} s^2(0) \\ pw\begin{pmatrix} s(0) \\ 0 \end{pmatrix} \end{matrix}\right)\right) \overset{(pw)}{\doteq} mt\begin{pmatrix} s^2(0) \\ s^2(0) \end{pmatrix} \overset{(mt)}{\doteq} s^4(0).$

This fact provides yet another counterexample for the associativity of exponentiation pw, namely

$$\neg pw\left(\begin{matrix}\mathrm{ss}(0)\\ pw\left[\begin{matrix}\mathrm{s}(0)\\ \mathrm{ss}(0)\end{matrix}\right]\end{matrix}\right) = pw\left(\begin{matrix}pw\left[\begin{matrix}\mathrm{ss}(0)\\ \mathrm{s}(0)\end{matrix}\right]\\ \mathrm{ss}(0)\end{matrix}\right)$$

References

ENDERTON, Herbert B. - *A Mathematical Introduction to Logic*. New York: Academic Press, 1972.

POLYA, G. - *How to Solve it: a new aspect of the mathematical method.* Princeton: Princeton Univ. Press, 1945 (2nd edition 1956, repr. 1971).

PORTO, Andr´e S. - Wittgenstein sobre as provas indutivas, 2009.

SHOENFIELD, Joseph R. - *Mathematical Logic*. Reading: Addison-Wesley, 1967.