On the Ambiguation of Polish Notation

Albert Visser

Department of Philosophy, Utrecht University

Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

email: `Albert.Visser@phil.uu.nl`

July 1, 2001

Abstract

In this paper we study an alternative way of treating Polish notation. Instead of proving unique reading for a well-behaved set of terms, we interpret any sting of the alphabet in a monoid that extends, in a specifiable sense, the given algebra of functions.

Key words: Polish Notation, unique reading, monoid, algebra, pushout

MSC2000 codes: 03B99

Contents

Bedeutungsvoll ist die scheinbar unwichtige Tatsache daß die logische Scheinbeziehungen wie $\vee$ und $\supset$, der Klammern bedürfen —im Gegensatz zu den wirklichen Beziehungen.

Though it seems unimportant, it is in fact significant that the pseudo-relations of logic, such as $\vee$ or $\supset$ need brackets —unlike real relations.

Wittgenstein, Tractatus 5461

Would Wittgenstein have taken this line if he had known Polish notation?

1 Introduction

Polish notation was invented by Łukasiewicz in the twenties. It is a bracket-free functional notation of great simplicity. Originally, the designation 'Polish Notation' was just used for Łukasiewicz' specific notation system for *propositional logic*. This includes e.g. writing '$\mathtt{C}$' instead of '$\rightarrow$' for implication. Later the usage was extended to include any bracket-free prefix notation system designed à la Łukasiewicz.

Polish notation has found all kinds of application. It, or rather its reverse variant RPN, is employed in the Hewlett-Packard calculators. RPN was also used in the programming language Forth (see [STT84]). Finally, it is used in Postscript.

The usual way to treat Polish Notation mathematically is as follows. Consider an alphabet A. The strings over A, can be considered as a monoid $\mathbb{A}_A$ with the concatenation operation $*$ as monoidal operation and the empty string ε as unit. The monoid $\mathbb{A}_A$ is the free monoid over A. We now interpret the letters of A as function symbols by introducing an arity function ar, which assigns to each letter a natural number. The class of terms is then defined inductively as the minimal class closed under the following clause.

- ▷ If the arity of $\mathtt{f}$ is n, and if $t_0,\ldots,t_{n-1}$ are terms, then $\mathtt{f}t_0\ldots t_{n-1}$ is also a term.

As usual, we suppress writing $*$. We can view $\Sigma := \langle A, \mathsf{ar}\rangle$ as a signature for algebras of one-sorted functions. The *unique reading theorem* tells us that the domain of terms enriched by the functions $[\mathtt{f}] : \langle t_0,\ldots,t_{n-1}\rangle \mapsto \mathtt{f}t_0\ldots t_{n-1}$ is the free algebra of signature Σ. The unique mapping associated to freedom gives us precisely recursion, allowing us to define interpetations of terms in algebras of signature Σ.

Unique reading theorems like the one described above, show how structure can be recovered from the 'linear' string format, by restricting the set of strings to the set of terms.

The present paper studies an alternative interpretation strategy. We do not *restrict* the set of strings to the terms in order to obtain an appropriate free algebra, rather we *extend* the semantical algebra to a monoid, so that we can use the fact that $\mathbb{A}_A$ is a free monoid to provide the interpretation.

Example 1.1 Consider e.g. the alphabet $\mathtt{0,S,A}$. Let $\mathsf{ar}(\mathtt{0}) = 0$, $\mathsf{ar}(\mathtt{S}) = 1$, $\mathsf{ar}(\mathtt{A}) = 2$. We interpret $\mathtt{0}$ as the natural number 0, $\mathtt{S}$ as *successor* over ω and $\mathtt{A}$ as addition over ω.

Here is a sample term: $\mathtt{AS0SS0}$. In the standard notation using brackets, we can view this as $\mathtt{A}(\mathtt{S}(\mathtt{0}), \mathtt{S}(\mathtt{S}(\mathtt{0})))$.[1] The term is interpreted as the function $+$ applied to 1 and 2. One way of looking at the interpretation of the term at hand —not supported by the semantics— is to say that $\mathtt{A}$ stands for $+$, $\mathtt{S0SS0}$ stands for the pair $\langle 1, 2\rangle$ and that the whole term stands for $+$ applied to $\langle 1, 2\rangle$. However, in the standard reconstruction we do not even consider such a string as $\mathtt{S0SS0}$ as syntactically well-formed. *If* we would allow the suggested reading, why not also the following one? $\mathtt{AS}$ stands for $\lambda xy \cdot S(x) + y$ and $\mathtt{0SS0}$ stands for $\langle 0, 2\rangle$.

The values of such extended readings will constitute the elements of the monoid 'extending' the standard algebra. ∎

We will present our approach in a somewhat abstract form. This has the disadvantage of making a perfectly simple idea complicated. On the other hand, further applications of the idea, such as the extension to the many-sorted case and the treatment of the proper way of introducing variables in the Polish context become a matter of selecting the right strictly monoidal category.

The work in the present paper springs from my interest in certain variants of dynamic semantics. In this kind of semantics the division of labour between syntax and semantics is changed. Syncategorematical bits of language, like brackets, which serve to make syntactic structure explicit, become meaningful objects which stand for instructions regulating the flow of information on a more semantical level. The study of Polish Notation serves as the study of a pure case of this kind of phenomenon. Seen in one way, Polish notation has a clearly structured syntax, connecting it to a free algebra. Seen in another way, it is a syntax-free notation system instantiating a free monoid. All the work is now done by the semantics.

Prerequisites

The paper presupposes some knowledge of basic category theory. The notions of category, functor, natural transformation and pushout ought to be familiar. We will make use of the notion of *strictly monoidal category*. However, everything one needs to know about this notion will be explained in the paper.

Acknowledgements

I thank Marc Bezem, Jan van Eijck, Marcus Kracht, Vincent van Oostrom, Doaitse Swierstra and Freek Wiedijk for helpful comments.

[1] The standard bracket notation is not entirely uniform in style. Of course, it should be $\mathtt{A}(\mathtt{S}(\mathtt{0}()), \mathtt{S}(\mathtt{S}(\mathtt{0}())))$.

2 Two Proofs of the Unique Reading Theorem

In this section we provide two proofs of the unique reading theorem. Here is a statement of the unique reading theorem.

Let a signature $\Sigma = \langle A, \mathsf{ar}\rangle$. Let be given. Let $\mathcal{T} = \langle \mathsf{Term}, [\cdot]\rangle$ be the term algebra with $[\mathtt{f}](t_0, \cdots, t_{n-1}) := \mathtt{f}\, t_0 \cdots t_{n-1}$. Then $\mathcal{T}$ is a free algebra over Σ.

Let $\mathcal{F}$ be any free algebra of signature Σ. We write $\overline{\mathtt{f}}$ for te interpretation of $\mathtt{f}$ in $\mathcal{F}$. Let Φ be the unique map from $\mathcal{F}$ to $\mathcal{T}$. By an easy induction on the terms we see that Φ is surjective. We only need to check that Φ is injective, since, in this case, Φ will be an isomorphism between $\mathcal{F}$ and $\mathcal{T}$.

We show, by induction on the length of w, that if the term t is an initial substring of the term w, $t = \Phi\tau$ and $w = \Phi\nu$, then $\tau = \nu$ and, hence, $t = w$. Suppose we have the desired fact for all terms shorter than w. Suppose t is an initial substring of w and that $w = \Phi\nu$, where $\nu = \overline{\mathtt{g}}(\nu_0, \cdots, \nu_{m-1})$, and, hence, $w = \mathtt{g}\Phi(\nu_0) \cdots \Phi(\nu_{m-1})$. Suppose further that $t = \Phi\tau$, where $\tau = \overline{\mathtt{f}}(\tau_0 \cdots \tau_{n-1})$ and, hence, $t = \mathtt{f}(\Phi(\tau_0) \cdots \Phi(\tau_{n-1}))$. Clearly, $\mathtt{g}$ must be equal to $\mathtt{f}$ and, thus, m must be n. Either $\Phi(\nu_0)$ is an initial segment of $\Phi(\tau_0)$ or vice versa. In both cases, by the Induction Hypothesis, we have $\nu_0 = \tau_0$. We proceed with this reasoning to show that $\tau_i = \nu_i$, for $i = 0, \ldots, n-1$. It follows that $\tau = \nu$.

Here is the second proof. Consider any algebra $\mathcal{A} = \langle D, I\rangle$ of signature Σ. Consider as language the set of all strings of the alphabet $A \oplus D$. Here $\oplus$ is disjoint union. To simplify our notations, we will assume that A and D are already disjoint and we will replace disjoint union by ordinary union. We build a rewrite system on this language by giving as rules:

- ▹ $\mathtt{f} d_0 \cdots d_{n-1} \to d$, if $I(\mathtt{f})(d_0, \cdots, d_{n-1}) = d$.

It is easy to see that this system is orthogonal and strongly normalizing. Hence every term reduces to a unique normal form. We define a relation S between terms and values as the smallest relation closed under the following rule.

- ▹ Suppose $\mathsf{ar}(\mathtt{f}) = n$ and $t_0\ S\ d_0, \ldots, t_{n-1}\ S\ d_{n-1}$, then $\mathtt{f} t_0 \cdots t_{n-1}\ S\ I(\mathtt{f})(d_0, \cdots, d_{n-1})$.

We now show by induction on terms that S is total and that if $t\ S\ d$, then $t \to d$. Hence every term has a unique value. We get the unique reading theorem by taking $\mathcal{A}$ to be the free algebra, since, by a simple induction, S considered as a mapping is a morphism, which is inverse to Φ.

3 Algebras and Monoids

There is the well known connection, exploited in functional programming languages, between the free monoid over an alphabet A and the free algebra of signature $\langle A \cup \{c\}, \mathsf{ar}\rangle$, where $\mathsf{ar}(a) = 1$, for $a \in A$, and $\mathsf{ar}(c) = 0$. We will not

pursue this connection here. Rather we will study a connection suggested by the proof of the unique reading theorem.

Consider a signature for algebras $\Sigma = \langle A, \mathsf{ar}\rangle$. A Σ-monoid is a pair $\langle \mathcal{M}, \phi\rangle$, where $\mathcal{M} = \langle M, \cdot, \mathsf{id}\rangle$ is a monoid, and $\phi : A \to M$. A morphism $f : \mathcal{S} \to \mathcal{S}'$ of Σ-monoids is a morphism of monoids such that $\phi' = f \circ \phi$.

To each element m of a monoid, we assign a function $[m]_n : M^n \to M$ given by $[m](m_0, \cdots, m_{n-1}) := mm_0 \cdots m_{n-1}$. (As usual, we suppress '$\cdot$'.)

The prefix-functor PREF from Σ-monoids to Σ-algebras is defined as follows. To the Σ-monoid $\mathcal{S}$ we assign the algebra $\langle M, I\rangle$, where M is the domain of $\mathcal{S}$ and $I(\mathtt{f}) = [\phi\mathtt{f}]_{\mathsf{ar}(\mathtt{f})}$. Each morphism of Σ-monoids is assigned the morphism of Σ-algebras with the same underlying function on the domain. Suppose that $n := \mathsf{ar}(\mathtt{f})$. We will sometimes write m^f for fm. Note that, if $f : \mathcal{S} \to \mathcal{S}'$, we have:

$$\begin{aligned}
\mathtt{f}^{I'}(m_0^f, \cdots, m_{n-1}^f) &= [\mathtt{f}^{\phi'}]_n(m_0^f, \cdots, m_{n-1}^f) \\
&= \mathtt{f}^{\phi'} m_0^f \cdots m_{n-1}^f \\
&= \mathtt{f}^{\phi f} m_0^f \cdots m_{n-1}^f \\
&= (\mathtt{f}^{\phi} m_0 \cdots m_{n-1})^f \\
&= ([\mathtt{f}^{\phi}]_n(m_0, \cdots, m_{n-1}))^f \\
&= (\mathtt{f}^{I}(m_0, \cdots, m_{n-1}))^f
\end{aligned}$$

So $\mathsf{PREF}(f)$ is indeed a morphism. Trivially PREF is a functor.

Let a Σ-algebra $\mathcal{A} = \langle D, I\rangle$ be given. We construct a rewrite system on the strings over $A \oplus D$ as in section 2. The Σ-monoid $\mathsf{M}_0(\mathcal{A})$ is constructed as follows. The domain consists of the normal forms of the rewrite system. $x \cdot y$ is the normal form of xy. The unit is the empty string. $\phi\mathtt{f} := \mathtt{f}$ for $\mathtt{f}$ of non-zero arity. $\phi\mathtt{c} := I(c)$ for constant $\mathtt{c}$. (In short: $\phi\mathtt{a}$ is the normal form of $\mathtt{a}$.) It is easily seen that we did indeed define a Σ-monoid.

We define $\tilde{\mathcal{A}} := \mathsf{PREF}(\mathsf{M}_0(\mathcal{A}))$. Let $\eta_{\mathcal{A}}$ be the morphism from $\mathcal{A}$ to $\tilde{\mathcal{A}}$ mapping d to d. Note that e.g. for $\mathtt{f}$ of arity $n > 0$:

$$\begin{aligned}
[\mathtt{f}^{\phi}]_n(d_0, \cdots, d_{n-1}) &= \mathsf{nf}(\mathtt{f} d_0 \cdots d_{n-1}) \\
&= I(f)(d_0, \cdots, d_{n-1})
\end{aligned}$$

We show that $\eta_{\mathcal{A}}$ is a universal arrow from $\mathcal{A}$ to PREF.

$$\begin{tikzcd}
\mathsf{M}_0(\mathcal{A}) \arrow[d, dotted, "g"'] & \mathcal{A} \arrow[r, "\eta_{\mathcal{A}}"] \arrow[dr, "f"'] & \tilde{\mathcal{A}} \arrow[d, dotted, "\mathsf{PREF}(g)"] \\
\mathcal{S} & & \mathsf{PREF}(\mathcal{S})
\end{tikzcd}$$

The diagram to the right shows that gd must be fd. The diagram to the left shows that, for $\mathtt{f}$ with $\mathsf{ar}(\mathtt{f}) > 0$, $g\mathtt{f}$ must be $\psi\mathtt{f}$, where $\mathcal{S} = \langle \mathcal{M}, \psi\rangle$. Since the

d's and the $\mathtt{f}$'s generate the domain of $\mathsf{M}_0(\mathcal{A})$, we see that *if* g exists, it must be unique.

To see that g exists, we define first g^+ on the strings over $A \oplus D$, by specifying that $g^+ d := fd$ and $g^+ \mathtt{f} := \psi \mathtt{f}$, where $\mathtt{f}$ has non-zero arity, and that $g^+(uv) = g^+(u) \cdot g^+(v)$. We show that if $x \to y$, then $g^+ x = g^+ y$. It is clearly sufficient to prove that $\mathtt{f}^{g^+} d_0^{g^+} \cdots d_{n-1}^{g^+} = (\mathtt{f}^I(d_0, \cdots, d_{n-1}))^{g^+}$. We have:

$$\begin{aligned}
\mathtt{f}^{g^+} d_0^{g^+} \cdots d_{n-1}^{g^+} &= \mathtt{f}^{\psi} d_0^f \cdots d_{n-1}^f \\
&= [\mathtt{f}^{\psi}]_n (d_0^f, \cdots, d_{n-1}^f) \\
&= (\mathtt{f}^I(d_0, \cdots, d_{n-1}))^f \\
&= (\mathtt{f}^I(d_0, \cdots, d_{n-1}))^{g^+}
\end{aligned}$$

By Theorem 2(ii), p 81, of [Mac71], we find that M_0 can be extended to a left adjoint, say M, of PREF. So we have proved:

Theorem 3.1 PREF *has a left adjoint* M. *Moreover, the universal arrows* $\eta_{\mathcal{A}}$ *are monomorphic embeddings.*

We can see that theorem 3.1 generalizes unique reading as follows. Let $\mathcal{F}$ be the free Σ-algebra. Since left adjoints preserve colimits, we find that $\mathsf{M}_0(\mathcal{F})$ is a free monoid on generators A. Ergo $\eta_{\mathcal{F}}$ is a monomorphic embedding of $\mathcal{F}$ into the prefix algebra for Σ of the free monoid over A. The image of $\eta_{\mathcal{F}}$ is the algebra of terms, which is, by isomorphism, a free algebra.

In the remaining sections we will pursue a different idea of constructing a monoid from an algebra. The basic difference is as follows. The functor M constructs monoids that are 'intensional', i.e. different 'terms' ($A \oplus D$-strings) over the algebra representing the same function may give rise to different elements of the monoid. We will study a construction that produces a monoid that is 'extensional'. The new construction will not give rise to a functor, since extension of te domain of an algebra makes it possible to add different values for previously co-extensional 'terms'.

4 Polish Composition

In this section we introduce *polish composition* and prove that this notion satisfies the desired properties. The basic tool is a construction to transform a category into a monoid by adjusting composed arrows to fit each other.

We need an auxiliary concept —based on ideas of Vincent van Oostrom— the concept of of *r-category*.

4.1 r-Categories

An r-category is a pair $\langle \mathcal{C}, (\cdot \backslash \cdot) \rangle$, where $\mathcal{C}$ is a category and where $(\cdot \backslash \cdot)$ is a partial binary operation on the arrows of $\mathcal{C}$, which satisfies the following constraints.

1. If f and g are morphisms with the same domain and if the pushout of f and g exists, then $f \setminus g$ and $g \setminus f$ exist, making

$$\begin{tikzcd} x \arrow[r, "f"] \arrow[d, "g"'] & y \arrow[d, "g \setminus f"] \\ z \arrow[r, "f \setminus g"'] & u \end{tikzcd}$$

into a pushout. Conversely, if $f \setminus g$ and $g \setminus f$ exist, then the above diagram is a pushout diagram.

2. We have:

   - ▷ $(f \setminus g) \setminus h \equiv f \setminus (h \circ g)$,
   - ▷ $(h \setminus (f \setminus g)) \circ (g \setminus f) \equiv (h \circ g) \setminus f$.

   Here '$\equiv$' means that the left-hand-side is defined iff the right-hand-side is defined and, when defined, the left-hand-side and the right-hand-side are equal. The diagram below illustrates these identities.

$$\begin{tikzcd} x \arrow[r, leftrightarrow, "\mathsf{id}_x"] \arrow[dd, "h \circ g"'] & x \arrow[r, "f"] \arrow[d, "g"'] & y \arrow[r, leftrightarrow, "\mathsf{id}_y"] \arrow[d, "g \setminus f"] & y \arrow[dd, "(h \circ g) \setminus f"] \\ & z \arrow[r, "f \setminus g"] \arrow[d, "h"'] & u \arrow[d, "h \setminus (f \setminus g)"] & \\ v \arrow[r, leftrightarrow, "\mathsf{id}_v"'] & v \arrow[r, "(f \setminus g) \setminus h", "f \setminus (h \circ g)"'] & w \arrow[r, leftrightarrow, "\mathsf{id}_w"'] & w \end{tikzcd}$$

3. We have, for $f : x \to y$,

   - ▷ $f \setminus \mathsf{id}_x = f$,
   - ▷ $\mathsf{id}_x \setminus f = \mathsf{id}_y$.

Summarizing, our notion $(\cdot \setminus \cdot)$ satisfies the following equations. For $f : x \to y$, we have:

R1 $(f \setminus g) \circ g \equiv (g \setminus f) \circ f$,

R2 $(f \setminus g) \setminus h \equiv f \setminus (h \circ g)$,

R3 $(h \setminus (f \setminus g)) \circ (g \setminus f) \equiv (h \circ g) \setminus f$,

R4 $f \setminus \mathsf{id}_x \equiv f$,

R5 $\mathsf{id}_x \setminus f \equiv \mathsf{id}_y$.

A possible further equation that an r-category could fulfill is the following one.

R6 $(f \setminus f) = \mathsf{id}_y$.

We call an r-category that satisfies R6 an r^+-category. All our examples will be r^+-categories.

We present a convenient sufficient condition for a category to be (extendable to) an r^+-category. A category is *skeletal* if all isomorphisms are identities. A theory *has bounded pushouts* if for any $f : x \to y$, $g : x \to z$, $h : y \to u$ and $i : z \to u$, there is a pushout for the pair f, g.

Theorem 4.1 *Suppose $\mathcal{C}$ is skeletal and has bounded pushouts. Then, $\mathcal{C}$ has a unique extension to an r^+-category.*

Proof

Suppose $\mathcal{C}$ is skeletal and has bounded pushouts. Since $\mathcal{C}$ is skeletal the arrows finishing the pushouts for f, g are unique. We take these to be $g \setminus f$ and $f \setminus g$. To get the desired equidefined identities, one uses the well known pasting lemmas for pushouts (see appendix A) in combination with te existence of bounded pushouts. ❑

Open Question 4.2 Is there an example of a category that has bounded pushouts, but *cannot be extended* to an r-category? ◘

4.2 Definition of Polish Composition

Let $\mathcal{A}$ be a category. We want to extend the composition $g \circ f$ of the category to cases where $\mathsf{cod}(f) \neq \mathsf{dom}(g)$. Let's call our extended composition '$\star$'. The strategy that we will follow is, first, to 'extend' $\mathsf{cod}(f)$ and $\mathsf{dom}(g)$ in a mimimal way to a common extension z. Subsequently we 'lift' f and g to morphisms f' and g' with $\mathsf{cod}(f') = z = \mathsf{dom}(g')$. Finally, we take $g \star f := g' \circ f'$.

To implement the notion of extension, we stipulate that there is a bifunctor $\circledast$ on $\mathcal{A}$ that is strictly monoidal. This means the following.

- ▷ $(x \circledast y) \circledast z = x \circledast (y \circledast z)$, $(f \circledast g) \circledast h = f \circledast (g \circledast h)$,
- ▷ $e \circledast x = x \circledast e = x$, $\mathsf{id}_e \circledast f = f \circledast \mathsf{id}_e = f$, for some designated element e.

The functoriality of $\circledast$ gives us the following principles.

- ▷ $(g_1 \circ f_1) \circledast (g_0 \circ f_0) \succeq (g_1 \circledast g_0) \circ (f_1 \circledast f_0)$,

  Here '$\succeq$' means: if the left-hand-side is defined, then so is the right-hand-side. If both sides are defined, their values are equal. The above formula is called *the interchange law.*

▷ $\mathsf{id}_x \circledast \mathsf{id}_y = \mathsf{id}_{x \circledast y}$.

The triple $\langle \mathcal{A}, \circledast, e \rangle$ is a strictly monoidal category. We could also view this triple as a 2-category, where $\circ$ is the horizontal composition and where $\circledast$ is the vertical composition. (See [Mac71], p. 44. For a more extensive treatment, see [Bor94], chapter 7.)

The monoid $\mathcal{B}$, given by $\circledast$, on the objects of $\mathcal{A}$ can be considered as a category. We demand that this category $\mathcal{B}$ has an extension to an r-category. We consider a fixed such extension.

Summarizing, the data needed for our definition, are that we have a strictly monoidal category $\langle \mathcal{A}, \circledast, e \rangle$, enriched with an operation $(\cdot \setminus \cdot)$ that makes the monoid given by $\circledast$ and e on the objects of $\mathcal{A}$ into an r-category.

Finally, we present the definition itself. Consider, for $i = 0, 1$, the morphisms $f_i : x_i \to y_i$ in $\mathcal{A}$. Suppose that, in $\mathcal{B}$, the pushout of y_0 and x_1 exists. So we have $(x_1 \setminus y_0) \circledast y_0 = (y_0 \setminus x_1) \circledast x_1$. We lift f_0 to $f_0' := \mathsf{id}_{x_1 \setminus y_0} \circledast f_0$ and f_1 to $f_1' := \mathsf{id}_{y_0 \setminus x_1} \circledast f_1$. We take:

$$f_1 \star f_0 := f_1' \circ f_0' = (\mathsf{id}_{y_0 \setminus x_1} \circledast f_1) \circ (\mathsf{id}_{x_1 \setminus y_0} \circledast f_0).$$

4.3 Verification of the Properties of Polish Composition

We show that id_e is the identity element for $\star$. We have, with id_e in the role of f_1 and $f : x \to y$ in the role of f_0:

$$\begin{aligned} \mathsf{id}_e \star f &\equiv (\mathsf{id}_{y \setminus e} \circledast \mathsf{id}_e) \circ (\mathsf{id}_{e \setminus y} \circledast f) \\ &\equiv (\mathsf{id}_y \circledast \mathsf{id}_e) \circ (\mathsf{id}_e \circledast f) \\ &\equiv \mathsf{id}_y \circ f \\ &\equiv f \end{aligned}$$

Note that in the second step we use R4 and R5. Similarly, we find that $g \star \mathsf{id}_e = g$. We show that $\star$ is associative. Suppose that, for $i = 0, 1, 2$, $f_i : x_i \to y_i$ in $\mathcal{A}$. Consider $(f_2 \star f_1) \star f_0$. Writing this out we obtain:

A) $(\mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)} \circledast ((\mathsf{id}_{y_1 \setminus x_2} \circledast f_2) \circ (\mathsf{id}_{x_2 \setminus y_1} \circledast f_1))) \circ (\mathsf{id}_{((x_2 \setminus y_1) \circledast x_1) \setminus y_0} \circledast f_0)$

Note that (A) is defined iff the subscripts are defined. The term $f_2 \star (f_1 \star f_0)$ takes the form:

B) $(\mathsf{id}_{((y_0 \setminus x_1) \circledast y_1) \setminus x_2} \circledast f_2) \circ (\mathsf{id}_{x_2 \setminus ((y_0 \setminus x_1) \circledast y_1)} \circledast ((\mathsf{id}_{y_0 \setminus x_1} \circledast f_1) \circ (\mathsf{id}_{x_1 \setminus y_0} \circledast f_0)))$

We replace $\mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)}$ in (A) by $\mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)} \circ \mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)}$ and apply the interchange law. We obtain the following term.

C) $(\mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)} \circledast \mathsf{id}_{y_1 \setminus x_2} \circledast f_2)$
$\circ\ (\mathsf{id}_{y_0 \setminus ((x_2 \setminus y_1) \circledast x_1)} \circledast \mathsf{id}_{x_2 \setminus y_1} \circledast f_1)$
$\circ\ (\mathsf{id}_{((x_2 \setminus y_1) \circledast x_1) \setminus y_0} \circledast f_0)$

Applying the functoriality to (C), we obtain:

D) $(\mathsf{id}_{y_0\backslash((x_2\backslash y_1)\circledast x_1)\circledast(y_1\backslash x_2)} \circledast f_2)$
$\circ\ (\mathsf{id}_{y_0\backslash((x_2\backslash y_1)\circledast x_1)\circledast(x_2\backslash y_1)} \circledast f_1)$
$\circ\ (\mathsf{id}_{((x_2\backslash y_1)\circledast x_1)\backslash y_0} \circledast f_0)$

Note that (D) is defined iff the subscripts are defined iff (A) is defined. We rewrite (B) in the same way, obtaining the following term.

E) $(\mathsf{id}_{((y_0\backslash x_1)\circledast y_1)\backslash x_2} \circledast f_2)$
$\circ\ (\mathsf{id}_{x_2\backslash((y_0\backslash x_1)\circledast y_1)\circledast(y_0\backslash x_1)} \circledast f_1)$
$\circ\ (\mathsf{id}_{x_2\backslash((y_0\backslash x_1)\circledast y_1)\circledast(x_1\backslash y_0)} \circledast f_0)$

It is clear that in order to obtain equality of (D) and (E) it is sufficient to show the following three equalities.

1. $y_0 \backslash ((x_2 \backslash y_1) \circledast x_1) \circledast (y_1 \backslash x_2) \equiv ((y_0 \backslash x_1) \circledast y_1) \backslash x_2$,
2. $y_0 \backslash ((x_2 \backslash y_1) \circledast x_1) \circledast (x_2 \backslash y_1) \equiv x_2 \backslash ((y_0 \backslash x_1) \circledast y_1) \circledast (y_0 \backslash x_1)$,
3. $((x_2 \backslash y_1) \circledast x_1) \backslash y_0 \equiv x_2 \backslash ((y_0 \backslash x_1) \circledast y_1) \circledast (x_1 \backslash y_0)$

Noting that (1) and (3) are alphabetic variants, we see that the following two computations give us the desired result.

$$\begin{aligned} y_0 \backslash ((x_2 \backslash y_1) \circledast x_1) \circledast (y_1 \backslash x_2) &\overset{R2}{\equiv} ((y_0 \backslash x_1) \backslash (x_2 \backslash y_1)) \circledast (y_1 \backslash x_2) \\ &\overset{R3}{\equiv} ((y_0 \backslash x_1) \circ y_1) \backslash x_2 \end{aligned}$$

$$\begin{aligned} y_0 \backslash ((x_2 \backslash y_1) \circledast x_1) \circledast (x_2 \backslash y_1) &\overset{R2}{\equiv} ((y_0 \backslash x_1) \backslash (x_2 \backslash y_1)) \circledast (x_2 \backslash y_1) \\ &\overset{R1}{\equiv} ((x_2 \backslash y_1) \backslash (y_0 \backslash x_1)) \circledast (y_0 \backslash x_1) \\ &\overset{R2}{\equiv} x_2 \backslash ((y_0 \backslash x_1) \circledast y_1) \circledast (y_0 \backslash x_1) \end{aligned}$$

Thus we have derived $(f_2 \star f_1) \star f_0 \equiv f_2 \star (f_1 \star f_0)$.[2] If we add an element 'undefined' to our morphisms, we can consider the result of our construction as a genuine monoid. A category is one form of a partial monoid. The partial monoid of polish composition, however, need not be a category. We will see an example later.

Inspection of the above reasoning, shows that the only source of possible partiality is the lack of pushouts. So if our monoid-as-category $\mathcal{B}$ has pushouts, then the result of our construction is a monoid.

A final property that we will prove uses the stronger assumption that $\mathcal{B}$ is an r^+-category. We show that, if $f_1 \circ f_0$ is defined, then $f_1 \star f_0 = f_1 \circ f_0$. We have:

$$\begin{aligned} f_1 \star f_0 &\equiv (\mathsf{id}_{y_0\backslash y_0} \circledast f_1) \circ (\mathsf{id}_{y_0\backslash y_0} \circledast f_0) \\ &\overset{R6}{\equiv} (\mathsf{id}_e \circledast f_1) \circ (\mathsf{id}_e \circledast f_0) \\ &\equiv f_1 \circ f_0 \end{aligned}$$

[2] These computations are in a sense the heart of this section. One could view the framework developed in this section as an attempt to make the meaning of these computations clear.

5 Stack Numbers

Consider again Polish notation, e.g. the terms `SOSSO` and `AS`. The desired interpretations of these terms are respectively $\langle 1, 2\rangle$ and $\lambda xy \cdot Sx + y$. The machinery of the previous section will allow us to obtain these interpretations. However, we will postpone this a bit, to do something simpler first: we will develop the calculus of the *types* of these terms. It is clear that `SOSSO` should have type $0 \to 2$, the type of a constant of arity 2, and `AS` should have type $2 \to 1$, the type of 2-place function. Generally, the types are $m \to n$, signifying that we are looking at a notation for a function from m-tuples to n-tuples.

To obtain a calculus of types, we consider the following category $\mathcal{A}$. The objects are the natural numbers, ω. The arrows are all pairs of natural numbers, ω^2. We will write the pair $\langle m, n\rangle$ as $m \to n$ or $n \leftarrow m$. We define: $\mathbf{0} := (0 \leftarrow 0)$. The functions dom, cod, id and composition are the obvious ones.

We take as the bifunctor $\circledast$, addition $+$ on the objects and pairwise addition on the pairs. We wil also write $+$ for the addition on pairs: $(n \to m) + (n' \to m') = ((n+n') \to (m+m'))$. It is easy to see that $+$ is indeed a bifunctor. The category $\mathcal{B}$ corresponding to the monoid $\langle \omega, +, 0\rangle$ is skeletal and has pushouts. We find $n \setminus m = n \dot{-} m$, where $\dot{-}$ is *monus* or *cut-off substraction*: $n \dot{-} m := \max(n - m, 0)$. Note that $(n \dot{-} m) + m = \max(n, m) = (m \dot{-} n) + n$. We get:

$$\begin{aligned}(n_1 \leftarrow m_1) \star (n_0 \leftarrow m_0) &= (\mathsf{id}_{n_0 \dot{-} m_1} + (n_1 \leftarrow m_1)) \circ (\mathsf{id}_{m_1 \dot{-} n_0} + (n_0 \leftarrow m_0)) \\ &= (((n_0 \dot{-} m_1) + n_1) \leftarrow ((m_1 \dot{-} n_0) + m_0))\end{aligned}$$

Thus we obtain the monoid $\mathbb{S} := \langle \omega^2, \star, \mathbf{0}\rangle$.

Example 5.1 Consider an alphabet $\{\mathtt{0}, \mathtt{S}, \mathtt{A}\}$. We assign the function symbols `0`, `S` and `A`, respectively, the types $(1 \leftarrow 0)$, $(1 \leftarrow 1)$ and $(1 \leftarrow 2)$. We consider as language the free monoid on `0`, `S` and `A`. To each term we assign the value in the monoid $\mathbb{S}$ of stack numbers, given by the unique map on monoids extending the type assignment to `0`, `S` and `A`. We can now compute the type $(2 \leftarrow 0)$ for `SOSSO` and $(1 \leftarrow 2)$ for `AS`. ∎

We may consider the pairs $(n \leftarrow m)$ of the monoid as a kind of numbers: stack numbers.[3] $\alpha, \beta, \ldots$ will range over stack numbers.

We give an alternative way of viewing stack numbers. It is pleasant, for our explanation, to have our polish composition move in the other direction. Thus, we define: $\alpha \cdot \beta := \beta \star \alpha$.

Consider the monoid on two generators pop and push generated by the equation $\mathsf{push} \cdot \mathsf{pop} = \mathsf{id}$. We can associate to this monoid a rewriting system (in the language without brackets) given by:

- $\triangleright$ $x \cdot \mathsf{id} \geq x$,
- $\triangleright$ $\mathsf{id} \cdot x \geq x$,

[3] Stack numbers are the *simple stacking cells* of [Vis94] and [VV96].

▷ $\mathsf{push} \cdot \mathsf{pop} \geq \mathsf{id}$.

This rewriting system is strongly confluent. Two terms are equal in our monoid iff they have the same normal form. The normal forms are id, pop^{m+1}, push^{m+1} and $\mathsf{pop}^{m+1} \cdot \mathsf{push}^{n+1}$. The mapping $(m \to n) \mapsto \mathsf{pop}^m \cdot \mathsf{push}^n$ is an isomorphism of monoids.

Writing '(' for: push, and ')' for: pop, and suppressing the $\cdot$ we can see that the stack numbers are the numbers of the bracket test. For example, (()))(will reduce to))(, the measure of '(un)balancedness'.

Finally, note that we have:

$$(((n_0 \dot{-} m_1) + n_1) - ((m_1 \dot{-} n_0) + m_0)) = (n_0 + n_1) - (m_0 + m_1).$$

So the mapping $(n \leftarrow m) \mapsto n - m$ is a morphism of monoids from $\mathbb{S}$ to the monoid $\langle Z, +, 0 \rangle$ of the integers with $+$ and 0.

6 Monoidal Interpretation of Polish Notation

We give the promised interpretation of Polish Notation. Let a non-empty domain D be given. We take, as our category $\mathcal{A}$, the category with as objects the natural numbers. We associate to each number n, the set D^n of sequences of elements of D of length n. $*$ is concatenation of sequences; ε is the empty sequence. The arrows from m to n are the functions from D^m to D^n. We take $m' \circledast m := m' + m$ and, for $f : D^m \to D^n$, $f' : D^{m'} \to D^{n'}$ and $\tau = \sigma * \sigma'$, where σ has length m and σ' has length m', we take: $(f' \circledast f)(\tau) := f(\sigma) * f'(\sigma')$. The functor $\circledast$, as defined here, is a strictly monoidal cartesian product for our category. Note that the monoid $\mathcal{B}$ is precisely the one of the previous section. Thus we can use our results do define the polish composition of functions. Note that this composition will be total. The mapping $\mathsf{TYPE} : (f : D^m \to D^n) \mapsto (m \to n)$ is obviously a morphism of monoids from the monoid of functions to $\mathbb{S}$. Thus, the calculus of stack numbers does indeed function as the calculus of types.

Consider any alphabet A. Suppose we have a mapping type assigning to each letter of A a stack number and a mapping $[\![\cdot]\!]$ assigning to each letter $\mathtt{f}$ of A a function of type $\mathsf{type}(\mathtt{f})$. The monoid, $\mathbb{A}_A = \langle A^*, *, \varepsilon \rangle$, of the strings over A, instantiates the free monoid generated by A. So we can uniquely extend type and $[\![\cdot]\!]$ to all strings over A. We will call the extensions, par abus de langage, again type and $[\![\cdot]\!]$. Note that $\mathsf{TYPE} \circ [\![\cdot]\!]$ is a morphism of monoids from the monoid of strings to the types. Moreover, $\mathsf{TYPE} \circ [\![\cdot]\!]$ coincides with type on the letters. Hence, by uniqueness, we find $\mathsf{type} = \mathsf{TYPE} \circ [\![\cdot]\!]$.

Example 6.1 We take:

▷ $A := \{\mathtt{0}, \mathtt{S}, \mathtt{A}\}$,

▷ $\mathsf{type}(\mathtt{0}) := (1 \leftarrow 0)$, $\mathsf{type}(\mathtt{S}) := (1 \leftarrow 1)$, $\mathsf{type}(\mathtt{A}) := (1 \leftarrow 2)$,

▷ $D := \omega$, $[\![\mathtt{0}]\!] := \lambda \cdot \langle \mathtt{0} \rangle$, the 0-ary function with output $\langle 0 \rangle$, $[\![\mathtt{S}]\!] = \lambda \mathtt{x} \cdot \langle \mathtt{Sx} \rangle$, $[\![\mathtt{A}]\!] := \lambda \mathtt{xy} \cdot \langle \mathtt{x} + \mathtt{y} \rangle$.

Here is a sample calculation. We analyse `ASOSSO` as the result of applying `AS` to `OSSO`. To get the value of `AS` we have to apply polish composition to $\lambda xy \cdot \langle x+y \rangle$ and $\lambda x \cdot \langle Sx \rangle$. To make the functions composable we have to boost $\lambda x \cdot \langle Sx \rangle$ to $\lambda xy \cdot \langle Sx, y \rangle$. the result of the composition is $\lambda xy \cdot \langle Sx + y \rangle$. To get the value of `OSSO`, we analyse it for example as the concatenation of `O` and `SSO`. Given this choice, we have to apply polish composition to $\lambda \cdot \langle 0 \rangle$ and $\lambda \cdot \langle 2 \rangle$. To make the composition work, we pump up $\lambda \cdot \langle 0 \rangle$ to $\lambda x \cdot \langle 0, x \rangle$. Composition gives us as value of `OSSO`: $\lambda \cdot \langle 0, 2 \rangle$. Composition of $\lambda xy \cdot \langle Sx + y \rangle$ and $\lambda \cdot \langle 0, 2 \rangle$ yields $\lambda \cdot \langle 3 \rangle$. ◻

How does our present construction compare to the discussion in section 3? Consider a signature $\Sigma = \langle A, \mathsf{ar} \rangle$ and consider a Σ-algebra $\mathcal{B} = \langle D, I \rangle$. In section 3, we showed how to construct the free Σ-monoid $\mathsf{M}(\mathcal{B})$ over $\mathcal{B}$. The construction of the present section provides for the domain D a monoid, say $\mathsf{N}_0(D)$, of functions $D^m \to D^n$ for any m, n. We can extend $\mathsf{N}_0(D)$, to a Σ-monoid $\mathsf{N}(\mathcal{B}) = \langle \mathsf{N}_0(D), \psi \rangle$, by taking $\psi(\mathsf{a}) := I(\mathsf{a})$. We can show that emb_0 with $\mathsf{emb}_0(d) := \{\langle \varepsilon, d \rangle\}$ uniquely extends to an embedding $\mathsf{emb} : \mathsf{M}(\mathcal{B}) \to \mathsf{N}(\mathcal{B})$. Note that in $\mathsf{M}(\mathcal{B})$ different elements a and b of A will correspond to different elements $\phi\mathsf{a}$ and $\phi\mathsf{b}$ of $\mathsf{M}(\mathcal{B})$. However, if $I(\mathsf{a}) = I(\mathsf{b})$, we will have $\psi\mathsf{a} = \psi\mathsf{b}$ in $\mathsf{N}(\mathcal{B})$. So emb might identify elements.

Note that N fails to be a functor from Σ-algebras to Σ-monoids. E.g. we could have a morphism $f : \mathcal{B} \to \mathcal{C}$, where $I_{\mathcal{B}}(\mathsf{a}) = I_{\mathcal{B}}(\mathsf{b})$, but $I_{\mathcal{C}}(\mathsf{a}) \neq I_{\mathcal{C}}(\mathsf{b})$. Hence, $\psi_{\mathcal{B}}(\mathsf{a}) = \psi_{\mathcal{B}}(\mathsf{b})$ and $\psi_{\mathcal{C}}(\mathsf{a}) \neq \psi_{\mathcal{C}}(\mathsf{b})$. So there is no possible way of supplying a f^{N}, since we must have $f^{\mathsf{N}}(\psi_{\mathcal{B}}\mathsf{a}) = \psi_{\mathcal{C}}\mathsf{a}$ and $f^{\mathsf{N}}(\psi_{\mathcal{B}}\mathsf{b}) = \psi_{\mathcal{C}}\mathsf{b}$.

Open Question 6.2 Is there a good alternative choice of our categories to make N into a functor? ◻

7 Variables and Sorts

In this section, we combine two further steps. We give one way to combine Polish notation with variables and we introduce Polish notation for a sorted calculus. We combine these two steps, just to make the paper not too long.

Introducing variables can be done in a trivial way: we can always vary the interpretations of the given language. Suppose we designate some letters of type $(0 \to 1)$ as *variables*. We can, then, define *a model* as any set M of interpretations of which all elements assign fixed values to letters that are not variables and of which the elements assign all possible values of the right type independently to the variables. We will not follow this easy solution, for two reasons. First, it is more elegant to treat variables locally: we only want to specify values for the variables actually occurring in a given term. Secondly, there is an analogy between variables an argument places that becomes better visible in the treatment we give below.

The addition of sorts, on the other hand, seems to go precisely as one would expect.

Suppose the following are given:

- ▷ a set of variables VAR,
- ▷ a set of sorts SORT,
- ▷ a function $\mathsf{sort} : \mathsf{VAR} \to \mathsf{SORT}$,
- ▷ for each sort $\mathfrak{s}$ a non-empty domain $D_{\mathfrak{s}}$.

The objects of our category $\mathcal{A}$ will pairs $\langle \sigma, V \rangle$, where σ is a sequence of sorts, i.o.w. $\sigma \in \mathsf{SORT}^*$, and where V a finite set of variables. We will often call our objects: *contexts*. We can view σ as a function from $n = \mathsf{length}(\sigma)$, considered as the set of numbers $\{0, \cdots, n-1\}$, to SORT. The set of immanent objects associated to $a = \langle \sigma, V \rangle$ is the set $\mathsf{imm}_a := n \cup V$. (We will assume that $\omega \cap \mathsf{VAR} = \emptyset$.) We define $\mathsf{sort}_a := \sigma \cup (\mathsf{sort} \restriction V)$. So sort_a gives us the sorting of the immanent objects of a. Note that the pair $\langle \mathsf{imm}_a, \mathsf{sort}_a \rangle$ provides an alternative representation of a —given the disjointness of ω and VAR.

Next we define:

- ▷ $\mathsf{ass}_a := \{\nu \mid \ \nu \text{ is a function on } \mathsf{imm}_a \text{ and } \forall r \in \mathsf{imm}_a \ \nu(r) \in D_{\mathsf{sort}_a(r)}\}$.

We could also write this as: $\mathsf{ass}_a := \prod_{r \in \mathsf{imm}_a} D_{\mathsf{sort}_a(r)}$. The morphisms of $\mathcal{A}$ between $a = \langle \sigma, V \rangle$ and $b = \langle \tau, W \rangle$ are the functions $f : \mathsf{ass}_a \to \mathsf{ass}_b$, which satisfy the following constraints: (i) $W \subseteq V$, (ii) $f(\nu) \restriction W = \nu \restriction W$.

Example 7.1 We produce some examples of meanings an their compositions. In this case we compose only meanings that 'fit'.

- ▷ The meaning $[\mathtt{P}]$ of a unary predicate $\mathtt{P}$ of objects of sort $\mathfrak{e}$ can be taken to be a morphism from $\langle \langle \mathfrak{e} \rangle, \emptyset \rangle$ to $\langle \langle \mathfrak{t} \rangle, \emptyset \rangle$.
- ▷ The meaning $[\mathtt{c}]$ of a constant $\mathtt{c}$ is a morphism between $\langle \varepsilon, \emptyset \rangle$ and $\langle \langle \mathfrak{e} \rangle, \emptyset \rangle$.
- ▷ The meaning $[\mathtt{f}]$ of a unary function symbol $\mathtt{f}$ is a morphism between $\langle \langle \mathfrak{e} \rangle, \emptyset \rangle$ and $\langle \langle \mathfrak{e} \rangle, \emptyset \rangle$.
- ▷ The meaning $[\mathtt{v}]$ of a variable $\mathtt{v}$ of sort $\mathfrak{e}$ is the morphism from $\langle \varepsilon, \{\mathtt{v}\} \rangle$ to $\langle \langle \mathfrak{e} \rangle, \emptyset \rangle$ given by $[\mathtt{v}]\{\langle \mathtt{v}, d \rangle\} := \{\langle 0, d \rangle\}$.

We can take $[\mathtt{Pfc}] := [\mathtt{P}] \circ [\mathtt{f}] \circ [\mathtt{c}]$, a morphism from $\langle \varepsilon, \emptyset \rangle$ to $\langle \langle \mathfrak{t} \rangle, \emptyset \rangle$. Similarly, we can take $[\mathtt{Pfv}] := [\mathtt{P}] \circ [\mathtt{f}] \circ [\mathtt{v}]$, a morphism from $\langle \varepsilon, \{\mathtt{v}\} \rangle$ to $\langle \langle \mathfrak{t} \rangle, \emptyset \rangle$. ∎

We proceed to introduce the operation $\circledast$ on the contexts by:

- ▷ $(\tau, W) \circledast (\sigma, V) := (\sigma * \tau, V \cup W)$.

The reversal of order is due to the fact that we treat $\circledast$ as composition in a category, so the order of application moves from right to left. On the other hand, our sequences grow from left to right.

Suppose the length of σ is n and that the length of τ is m. Let $a := \langle \sigma, V \rangle$ and $b := \langle \tau, W \rangle$. We have:

- $\triangleright$ $\mathsf{imm}_{b \circledast a} := (n+m) \cup V \cup W$,
- $\triangleright$ $\mathsf{sort}_{b \circledast a}(r) := \begin{cases} \mathsf{sort}_a(r) & \text{if } r \in \mathsf{imm}_a \\ \mathsf{sort}_b(r-n) & \text{if } n \leq r < n+m \\ \mathsf{sort}_b(r) & \text{if } r \in W \end{cases}$.

Note that the first and third case of this definition may show some overlap. This overlap is harmless, since *on variables* sort_a and sort_b coincide. It is useful to introduce two auxiliary functions $\mathsf{in}_0 : \mathsf{imm}_a \to \mathsf{imm}_{b \circledast a}$ and $\mathsf{in}_1 : \mathsf{imm}_b \to \mathsf{imm}_{b \circledast a}$. We take:

- $\triangleright$ $\mathsf{in}_0(r) := r$,
- $\triangleright$ $\mathsf{in}_1(r) := \begin{cases} n+r & \text{if } r \in m \\ r & \text{if } r \in W \end{cases}$.

We will write $\mathsf{in}_i^{a,b}$, if we need to make expicit which $b \circledast a$ we are considering. We can now, describe $\mathsf{sort}_{b \circledast a}$ by:

- $\triangleright$ $\mathsf{sort}_{b \circledast a} := (\mathsf{sort}_b \circ \mathsf{in}_1^{-1}) \cup (\mathsf{sort}_a \circ \mathsf{in}_0^{-1})$.

We define $\circledast$ on morphisms as follows. Let $f_i : a_i \to b_i$.

- $\triangleright$ $(f_1 \circledast f_0)(\nu) := (f_1(\nu \circ \mathsf{in}_1^{a_0,a_1}) \circ (\mathsf{in}_1^{b_0,b_1})^{-1}) \cup (f_0(\nu \circ \mathsf{in}_0^{a_0,a_1}) \circ (\mathsf{in}_0^{b_0,b_1})^{-1})$.

It is not difficult to see that $f_1 \circledast f_0$ is well defined. We can verify the fact that $\circledast$ is a strictly monoidal bifunctor using the observation that

$$\mathsf{in}_1^{a_1 \circledast a_0, a_2} = \mathsf{in}_1^{a_0, a_2 \circledast a_1} \circ \mathsf{in}_1^{a_1, a_2}.$$

One can show that $a_0 \times a_1 := a_1 \circledast a_0$, is a cartesian product for our category. We can define the projection functions as follows.

- $\triangleright$ $\pi_i(\nu) = \nu \circ \mathsf{in}_i$.

Since the in_i preserve variables and sorts, the π_i are indeed morphisms. Now consider $f_i : b \to a_i$. Define $(f_0, f_1) : b \to a_0 \times a_1$ as follows.

- $\triangleright$ $(f_0, f_1)(\nu) := (f_0(\nu) \circ \mathsf{in}_0^{-1}) \cup (f_1(\nu) \circ \mathsf{in}_1^{-1})$.

We can easily see that (f_0, f_1) is the unique arrow promised by the definition of product. Note that, as expected:

$$f_0 \times f_1 = (f_0 \circ \pi_0^{a_0,a_1}, f_1 \circ \pi_1^{a_0,a_1}) = f_1 \circledast f_0.$$

$(\cdot \setminus \cdot)$ is defined as follows.

$$\triangleright\ \sigma \setminus \tau = \begin{cases} \rho & \text{if } \tau * \rho = \sigma \\ \varepsilon & \text{if } \sigma \text{ is an initial segment of } \tau \\ \text{undefined} & \text{otherwise} \end{cases},$$

$$\triangleright\ (\sigma, V) \setminus (\tau, W) := \begin{cases} (\sigma \setminus \tau, V \setminus W) & \text{if } \sigma \setminus \tau \text{ is defined} \\ \text{undefined} & \text{otherwise} \end{cases}.$$

We easily verify that in this way we have defined an r^+-category. We now can apply our previous results to obtain a Polish system for a sorted calculus with variables. Note that the monoid of Polish composition will sometimes give the value undefined in case an argument of the wrong sort is provided.

Example 7.2 Here are some sample meanings.

- $\triangleright$ $[\mathtt{P}]$ is a morphism from $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$ to $\langle\langle \mathfrak{t}\rangle, \emptyset\rangle$.
- $\triangleright$ $[\mathtt{Q}]$ is a morphism from $\langle\langle \mathfrak{e}, \mathfrak{e}\rangle, \emptyset\rangle$ to $\langle\langle \mathfrak{t}\rangle, \emptyset\rangle$.
- $\triangleright$ $[\mathtt{c}]$ is a morphism between $\langle \varepsilon, \emptyset\rangle$ and $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$.
- $\triangleright$ $[\mathtt{f}]$ is a morphism between $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$ and $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$.
- $\triangleright$ $[\mathtt{g}]$ is a morphism between $\langle\langle \mathfrak{e}, \mathfrak{e}\rangle, \emptyset\rangle$ and $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$.
- $\triangleright$ $[\wedge]$ is a morphism between $\langle\langle \mathfrak{t}, \mathfrak{t}\rangle, \emptyset\rangle$ and $\langle\langle \mathfrak{t}\rangle, \emptyset\rangle$.
  We take $[\wedge](\{\langle 0, i\rangle, \langle 1, j\rangle\}) := \{\langle 0, \min(i, j)\rangle\}$.
- $\triangleright$ $[\mathtt{v}]$ is a morphism from $\langle \varepsilon, \{\mathtt{v}\}\rangle$ to $\langle\langle \mathfrak{e}\rangle, \emptyset\rangle$.
  We take $[\mathtt{v}]\{\langle \mathtt{v}, d\rangle\} := \{\langle 0, d\rangle\}$.

We compute $[\mathtt{Pg}] := [\mathtt{P}] \star [\mathtt{g}]$ in some detail, just to see that the underlying machinery indeed does what we expect it to do.

$$\begin{aligned} [\mathtt{Qg}] &= (\mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle \setminus \langle\langle \mathfrak{e},\mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{Q}]) \circ (\mathsf{id}_{\langle\langle \mathfrak{e},\mathfrak{e}\rangle,\emptyset\rangle \setminus \langle\langle \mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{g}]) \\ &= (\mathsf{id}_{\langle \varepsilon,\emptyset\rangle} \circledast [\mathtt{Q}]) \circ (\mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{g}]) \\ &= [\mathtt{Q}] \circ (\mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{g}]) \end{aligned}$$

We have $\mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{g}] : \langle\langle \mathfrak{e}, \mathfrak{e}, \mathfrak{e}\rangle, \emptyset\rangle \to \langle\langle \mathfrak{e}, \mathfrak{e}\rangle, \emptyset\rangle$ and

$$\begin{aligned} (\mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle} \circledast [\mathtt{g}])\{\langle 0, d_0\rangle, \langle 1, d_1\rangle, \langle 2, d_2\rangle\} &= \\ \mathsf{id}_{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle}(\{\langle 0, d_0\rangle, \langle 1, d_1\rangle, \langle 2, d_2\rangle\} \circ \mathsf{in}_1^{\langle\langle \mathfrak{e},\mathfrak{e}\rangle,\emptyset\rangle,\langle\langle \mathfrak{e}\rangle,\emptyset\rangle}) \circ (\mathsf{in}_1^{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle,\langle\langle \mathfrak{e}\rangle,\emptyset\rangle})^{-1} \cup & \\ [\mathtt{g}](\{\langle 0, d_0\rangle, \langle 1, d_1\rangle, \langle 2, d_2\rangle\} \circ \mathsf{in}_0^{\langle\langle \mathfrak{e},\mathfrak{e}\rangle,\emptyset\rangle,\langle\langle \mathfrak{e}\rangle,\emptyset\rangle}) \circ (\mathsf{in}_0^{\langle\langle \mathfrak{e}\rangle,\emptyset\rangle,\langle\langle \mathfrak{e}\rangle,\emptyset\rangle})^{-1} &= \\ \{\langle 1, d_2\rangle\} \cup \{\langle 0, [\mathtt{g}](\{\langle 0, d_0\rangle, \langle 1, d_1\rangle\})\rangle\} &= \\ \{\langle 0, [\mathtt{g}](\{\langle 0, d_0\rangle, \langle 1, d_1\rangle\})\rangle, \langle 1, d_2\rangle\} & \end{aligned}$$

If we rewrite our functions on numbers as sequences, we obtain the expected $\langle [\mathtt{g}](d_0, d_1), d_2\rangle$. So, finally, $[\mathtt{Qg}](d_0, d_1, d_2) = [\mathtt{Q}]([\mathtt{g}](d_0, d_1), d_2)$.

Here are two final examples. The formula $\wedge\mathtt{Qfvgfcv}\mathtt{Pfgcc}$ has as meaning a function from assignments on $\mathtt{v}$ to truthvalues. The formula $\mathtt{fP}$ is undefined. ∎

Open Question 7.3 Is there a nice way to add quantifiers to our semantics that is faithful to the monoidal style? ∎

References

[Bor94] F. Borceux. *Handbook of Categorical Algebra 1, Basic Category Theory.* Encyclopedia of mathematics and its applications. Cambridge University Press, Cambridge, 1994.

[Mac71] S. MacLane. *Categories for the Working Mathematician.* Number 5 in Graduate Texts in Mathematics. Springer, New York, 1971.

[STT84] W.P. Salman, O. Tisserand, and B. Toulout. *Forth.* Macmillan, throughout the world, 1984.

[Vis94] A. Visser. Actions under presuppositions. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 196–233. MIT Press, Cambridge, Mass., 1994.

[VV96] A. Visser and C. Vermeulen. Dynamic bracketing and discourse representation. *Notre Dame Journal of Formal Logic*, 37:321–365, 1996.

A Appendix: Pushouts

Suppose A is a pushout diagram. We will write $[p, q]_A$ for the unique arrow promised by the pushout property for the arrows p and q. See the diagram below, where $m =: [p, q]_A$.

$$\begin{tikzcd}
x \arrow[r, "f"] \arrow[d, "g"'] \arrow[dr, phantom, "A"] & y \arrow[d, "g'"'] \arrow[ddr, "q"] & \\
z \arrow[r, "f'"] \arrow[drr, "p"'] & u \arrow[dr, dotted, "m"] & \\
 & & v
\end{tikzcd}$$

Theorem A.1 *Consider the diagrams*

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "h"'] \arrow[dr, phantom, "A"] & b \arrow[r, "g"] \arrow[d, "h'"] \arrow[dr, phantom, "B"] & c \arrow[d, "h''"] \\
d \arrow[r, "f'"'] & e \arrow[r, "g'"'] & k
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "i := g \circ f"] \arrow[d, "h"'] \arrow[dr, phantom, "C"] & c \arrow[d, "h''"] \\
d \arrow[r, "i' := g' \circ f'"'] & k
\end{tikzcd}$$

Suppose A and B are pushouts. Then, so is C.

Proof

Suppose A and B are pushouts. We prove the pushout property for C. Let p and q be as given in the diagram below.

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "h"'] & b \arrow[r, "g"] \arrow[d, "h'"] & c \arrow[d, "h''"'] \arrow[ddr, "q"] & \\
d \arrow[r, "f'"] \arrow[drrr, "p"'] & e \arrow[r, "g'"] \arrow[drr, dotted, "n"] & k \arrow[dr, dotted, "m"] & \\
& & & \ell
\end{tikzcd}$$

Let $n := [p, q \circ g]_A$ be the unique arrow promised by the pushout property for A and let $m := [n, q]_B$ be the unique arrow promised by the pushout property for B. It is easy to see that m satisfies the existence claim of the pushout property for C. Suppose $m' : k \to \ell$ also satsifies this claim. By the uniqueness of n we find that $n = m' \circ g'$. By the uniqueness of m w.r.t. B we now find $m = m'$. ❑

Theorem A.2 *Consider the diagrams*

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "h"'] \arrow[dr, phantom, "A"] & b \arrow[r, "g"] \arrow[d, "h'"] \arrow[dr, phantom, "B"] & c \arrow[d, "h''"] \\
d \arrow[r, "f'"'] & e \arrow[r, dotted, "g'"'] & k
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[r, "i := g \circ f"] \arrow[d, "h"'] \arrow[dr, phantom, "C"] & c \arrow[d, "h''"] \\
d \arrow[r, "i'"'] & k
\end{tikzcd}$$

Suppose A and C are pushouts. Then there is a unique g' that makes B commute and $i' = g' \circ f'$. Moreover, B is a pushout.

Proof

Suppose A and C are pushouts. We take $g' := [i', h'' \circ g]_A$. It is easy to see that g' has the promised uniqueness property. We prove the pushout property for B. Let $n : e \to \ell$ and $q : c \to \ell$ be arrows satisfying the premiss of the pushout property for B.

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[d, "h"'] & b \arrow[r, "g"] \arrow[d, "h'"] & c \arrow[d, "h''"'] \arrow[ddr, "q"] & \\
d \arrow[r, "f'"] \arrow[drrr, "p"'] & e \arrow[r, "g'"] \arrow[drr, "n"] & k \arrow[dr, dotted, "m"] & \\
& & & \ell
\end{tikzcd}$$

Let $p := n \circ f'$. The pushout property for C yields the unique arrow $m := [p, q]_C$ with the property $m \circ h'' = q$ and $m \circ i' = p$. We show that m is the unique arrow $[n, q]_B$. The uniqueness is easy. For existence, we have to verify that $m \circ g' = n$. Consider A. We have $n \circ f' = p$ and $n \circ h' = q \circ g$. So we have $n = [p, q \circ g]_A$. But we also have:

1. $m \circ g' \circ f' = m \circ i' = p$.
2. $m \circ g' \circ h' = m \circ h'' \circ g = q \circ g$.

Ergo, by the uniqueness property for A, we find $n = [p, q \circ g]_A = m \circ g'$. ❑