Pheno Technology

Carl Pollard

June 25, 2012

Beyond Strings

- We can't keep pretending that all there is to pheno is strings and functions over strings.
- Often we need to ask: strings of what? Syllables? Phonological words? Intonation phrases?
- And it's not enough just to stick things together; often we need to know 'how tightly' or by 'what flavor of glue' things are stuck together.
- For example, there is a difference between putting two phonological words (a type we'll now call p) next to each other and attaching a clitic (which we'll call type c) to a phonological word.
- Also there is the issue of *non-determinism*: sometimes there is some freedom of variation in how things are ordered which does not affect the meaning.
- We need to develop some technology for talking about such things within the higher-order pheno theory.

The String Type Constructor

- Instead of just having a type s of strings, we assume that for each phenotype A there is a type Str_A of A-strings.
- That is, Str is not a type, but rather a unary type constructor.
- In terms of the Curry-Howard correspondence, Str can be thought of as similar to a modal operator.
- $\vdash \mathbf{e}_A : \mathrm{Str}_A$ (the **null** A-string)
- $\vdash \cdot_A : \mathrm{Str}_A \to \mathrm{Str}_A \to \mathrm{Str}_A$ (**concatenation**, written infix)
- $\vdash \mathbf{toS}_A : A \to \mathrm{Str}_A$ maps each A to an A-string. Intuitively, this can be thought of as a string of length one.
- We usually drop the subscript 'A' when it can be inferred from the context.

Axiom Schemas for Strings

Our previous string axioms now must be schematized over the type metavariable A (here the variables are of type Str_A):

$$\vdash \forall_{xyz}.(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

$$\vdash \forall_x.x \cdot \mathbf{e}_A = x$$

$$\vdash \forall_x.\mathbf{e}_A \cdot x = x$$

Notation for Phenotypes

- We revive the notation s as an *abbreviation* for Str_p, i.e. strings of phonological words.
- For any phenotype A, $\mathrm{Str}_A \to \mathrm{t}$ is the type of A-**languages**, i.e. sets of A-strings.
- We write S as an abbreviation for $\mathrm{s} \to \mathrm{t}$, the type of p-languages, i.e. sets of strings of phonological words.
- We write z as an abbreviation for Str_S, i.e. strings of p-languages.
- We write Z as an abbreviation for $\mathrm{z} \to \mathrm{t}$, the type of S-languages, i.e. sets of strings of p-languages!

Conventions for Pheno Variables

- We use c as a variable of type c.
- We use p and q as variables of type p.
- We use s, t, and u as variables of type s.
- We use P, Q, and R as variables of type S.
- We use w, x, y, and z as variables of type z.
- We use W, X, Y, and Z as variables of type Z.

Representing the Natural Numbers

- Often it's useful to be able to identify a numerical position in a string or to know the length of a string.
- We can represent the natural numbers as the type Str_T, which we abbreviate as n.
- We represent 0 as $\mathbf{e}_T$.
- We define the **successor** function $\mathbf{suc} : \mathrm{n} \to \mathrm{n}$ by

$$\textbf{suc} =_{\text{def}} \lambda_n.(\textbf{toS}_\text{n}\ *) \cdot n$$

- Then we write 0, 1, 2, 3, etc. as *abbreviations* for $\mathbf{e}_T$, $\textbf{toS}_\text{n}\ *$, $**$, $***$, etc.
- If necessary we can define the usual arithmetic functions (addition, multiplication, exponential) by mimicking in HOL the way they are recursively defined in set theory.

Abbreviations for Pheno Terms

- $\mathbf{e}_\text{p}$, the null p-string, is abbreviated to $\mathbf{e}$.
- $\cdot_\text{p}$, concatenation of p-strings, is abbreviated to $\cdot$.
- $\cdot_\text{S}$, concatenation of S-strings, is abbreviated to $\circ$.
- $\textbf{toS}_\text{p} : \text{p} \to \text{s}$ is abbreviated to $\textbf{toS}$.
- $\textbf{toS}_\text{S} : \text{S} \to \text{z}$ is abbreviated to $\textbf{toZ}$.
- For a phonological word foo:
  - $\textbf{toS}$ foo is abbreviated to foo_s
  - the singleton p-language $\lambda_s.s = \text{foo}_\text{s}$ is abbreviated to FOO
  - $\textbf{toZ}$ FOO is abbreviated FOO_z
- $\vdash \textbf{toS} : \text{p} \to \text{s}$ (abbreviates $\textbf{toS}_\text{p}$)
- $\vdash \textbf{toZ} : \text{S} \to \text{z}$ (abbreviates $\textbf{toS}_\text{S}$)
- If $a_0, \ldots, a_n$ are terms of type A ($n > 0$), then $a_0 \ldots a_n$ abbreviates the term $(\textbf{toS}\ a_0) \cdot \ldots \cdot (\textbf{toS}\ a_n)$ of type Str_A.

Operations on p-Languages

$\vdash 0_\text{p} : \text{S}$ (the empty p-language)

$\vdash 1_\text{p} : \text{S}$ (the singleton language $\lambda_s.s = \mathbf{e}$)

$\vdash \bullet_\text{p} : \text{S} \to \text{S} \to \text{S}$ (**language fusion**)

$$\bullet_\text{p} =_{\text{def}} \lambda_{PQs}.\exists_{tu}.(P\ t) \wedge (Q\ u) \wedge (s = t \cdot u)$$

$\vdash \cup_\text{p} : \text{S} \to \text{S} \to \text{S}$ (**language union**)

$$\cup_\text{p} =_{\text{def}} \lambda_{PQs}.(P\ s) \vee (Q\ s)$$

$\vdash \textbf{per}_\text{p} : \text{s} \to \text{S}$

For any p-string s, ($\textbf{per}\ s$) is the set of **permutations** of s.

All these have counterparts when p is replaced by any other pheno type (most often, S).

Standard String Functions

The following are all schematized over a phenotype A.

cns : $A \rightarrow \mathrm{Str}_A \rightarrow \mathrm{Str}_A$: sticks an A onto the left edge of an A-string

fst : $\mathrm{Str}_A \rightarrow A$: returns the first A of a (non-null) A-string

rst : $\mathrm{Str}_A \rightarrow \mathrm{Str}_A$ returns all but the first A of a (non-null) A-string, in the same order

snc : $A \rightarrow \mathrm{Str}_A \rightarrow \mathrm{Str}_A$: sticks an A onto the right edge of an A-string

lst : $\mathrm{Str}_A \rightarrow A$: returns the last A of a (non-null) A-string

tsr : $\mathrm{Str}_A \rightarrow \mathrm{Str}_A$ returns all but the last A of a (non-null) A-string, in the same order

Some Relationships between String Functions

$\forall_{ps}.(\mathbf{cns}\ p\ s) = (\mathbf{toS}\ p) \cdot s$

$\forall_{ps}.(\mathbf{snc}\ p\ s) = s \cdot (\mathbf{toS}\ p)$

$\forall_{p}.(\mathbf{toS}\ p) = (\mathbf{cns}\ \mathrm{p}\ \mathbf{e})$

$\forall_{s}.s = (\mathbf{cns}\ (\mathbf{fst}\ s)\ (\mathbf{rst}\ s))$

$\forall_{s}.s = (\mathbf{snc}\ (\mathbf{lst}\ s)\ (\mathbf{tsr}\ s))$

Note: the last two are not quite correct, because they have to be restricted to the case where s is non-null.

This calls for a slightly more sophisticated approach in which each string type is decomposed into a *coproduct* (i.e. disjoint union) of a null string type and a non-null string type.

Linguification

- $\vdash \mathbf{L} : \mathrm{z} \rightarrow \mathrm{S}$
- This fuses a string of p-languages into a single language:

  $\vdash (\mathbf{L}\ \mathbf{e}_{\mathrm{S}}) = 1_{\mathrm{S}}$

  $\vdash \forall_{Pz}.(\mathbf{L}\ (\mathbf{cns}\ P\ z)) = P \bullet (\mathbf{L}\ z)$
- So for any p-language P:

$$(\mathbf{L}\ (\mathbf{toZ}\ P)) = P$$

- And for any string of p-languages $P_0 \ldots P_n$ $(n > 0)$,

$$(\mathbf{L}\ P_0 \ldots P_n) = P_0 \bullet \ldots \bullet P_n$$

Compaction

- $\vdash \mathbf{k} : \mathrm{Z} \to \mathrm{S}$

- Compaction fuses an S-language (i.e. a *set* of strings of p languages) into a single *p*language by unioning together the linguifications of all the strings in the set:

  $\vdash (\mathbf{k}\ 0_{\mathrm{Z}}) = 0_{\mathrm{S}}$

  Here 0_{Z} is the empty set of strings of languages.

  $\vdash \forall_{Zw}.(\mathbf{k}\ (Z \cup (\lambda_z.z = w))) = (\mathbf{k}\ Z) \cup (\mathbf{L}\ w)$

The Length of a String
We can define the **length** function $\mathbf{len}_A : \mathrm{Str}_A \to \mathrm{n}$ by the axioms:

$\vdash (\mathbf{len}\ \mathbf{e}) = 0$

$\vdash \forall_{xs}.(\mathbf{len}\ (\mathbf{cns}\ x\ s)) = (\mathbf{suc}\ (\mathbf{len}\ s))$

Cliticization

- Pro- and en-cliticization to a phonological word are distinguished contextually, not typographically:

  $\vdash \# : \mathrm{c} \to \mathrm{p} \to \mathrm{p}$ (**procliticization**, written infix)
  $\vdash \# : \mathrm{p} \to \mathrm{c} \to \mathrm{p}$ (**encliticization**, written infix)

- Likewise for pro- and en-cliticization to a p-string:

  $\vdash + : \mathrm{c} \to \mathrm{s} \to \mathrm{s}$ (**procliticization**, written infix)
  $\vdash + : \mathrm{s} \to \mathrm{c} \to \mathrm{s}$ (**encliticization**, written infix)

  which are defined, respectively, as follows:

  $+ =_{\mathrm{def}} \lambda_{cs}.\mathbf{cns}\ c\#(\mathbf{fst}\ s)\ (\mathbf{rst}\ s)$
  $+ =_{\mathrm{def}} \lambda_{cs}.\mathbf{snc}\ (\mathbf{lst}\ s)\#c\ (\mathbf{tsr}\ s)$