# PHONOLOGICAL STRUCTURE AND ABSTRACT SPECIFICATION

**Ewan Klein & Steven Bird**

**Centre for Cognitive Science, University of Edinburgh, Scotland.**

## ABSTRACT

Much recent work in theoretical phonology has revolved around issues of *representation*. As the structures grow in size and complexity, it becomes increasingly difficult to represent them and reason about them. In this article we shall explore techniques from computer science for abstract specification in order to provide a solution to these representation and reasoning problems. Our starting point is the assumption that phonological representations are simply rather special kinds of *data types*. Once this connection has been noted, techniques for specifying and accessing data structures can be carried over to phonology, with some interesting results. Example applications to metrical structure and feature geometry will be provided[1].

## 1 ABSTRACT DATA TYPES

In this section we illustrate how a particular theory of phonological representation may be recast as a definition of a certain class of data structures, i.e. as a specification of an *abstract data type*[2].

Specifications are written in a conventional format consisting of a declaration of *sorts*, operation symbols (*opns*), and equations (*eqns*). Preceding the equations we list all the variables (*vars*) which figure in them. As an illustration, we give below a specification of the data type BINARY TREE, where the leaves are labelled $\sigma$.

BINARY TREE =

| | |
|---|---|
| *sorts:* | leaf, netree < tree |
| *opns:* | $\sigma$ : → leaf |
| | $\langle \_,\_ \rangle$ : tree tree → netree |
| | *left* _ : netree → tree |
| | *right* _ : netree → tree |
| *vars:* | $T_1, T_2$ : tree |
| *eqns:* | *left* $\langle T_1, T_2 \rangle = T_1$ |
| | *right* $\langle T_1, T_2 \rangle = T_2$ |

The *sorts* line lists the three sorts **leaf**, **netree** and **tree**. The < sign indicates that **leaf** and **netree** are subsorts of **tree**. Anything of sort **leaf** or sort **netree** is also of sort **tree**, and anything of sort **tree** also has the sort **leaf** or **netree**, but not both. The operation symbol $\langle \_,\_ \rangle$ (where '_' marks the position of the operator's arguments) is a called a *constructor*: it builds trees out of trees (and indeed out of leaves and non-empty trees, since operators are defined for all subsorts of their domain sort)[3]. *left* _ and *right* _ are known as *selectors*: they pull trees into their component parts. The equations specify the behaviour of the two selectors.

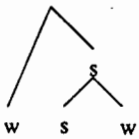
Suppose we now wish to modify the definition of binary trees to obtain metrical trees. These are binary trees where branches are ordered according to whether they are labelled 's' (strong) or 'w' (weak). We encode this

[1] We are grateful to Michael Newton and Jonathan Calder for their comments on this paper. Our work is supported by ESPRIT Basic Research Action 3175 (DYANA).

[2] For a more thorough definition of these terms, see [7]. There are various systems for the computational implementation of abstract specifications, e.g. [8,9].

[3] In general, let $\sigma$, $\sigma'$, and $\tau$ be sorts such that $\sigma' < \sigma$, let $f$ be an operator of rank $\sigma \rightarrow \tau$, and let $t$ be a term of sort $\sigma'$. Then $f(t)$ is defined, and is a term of sort $\tau$. From a semantic point of view, we are saying that if a function assigns values to members of particular set $X$, then it will also assign values to members of any subset $X'$ of $X$. See [5,11] for discussion of this approach to inheritance.

information by augmenting the left or right angle bracket of our $\langle \_, \_ \rangle$ constructor with 's' according to whether the left or right branch is considered strong.




All trees have a distinguished leaf node called the 'highest terminal element', which is connected to the root of the tree by a path of 's' nodes. The specification is as follows:

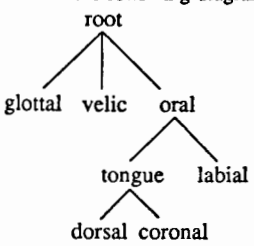
METRICAL TREE

| | |
|---|---|
| *sorts:* | **leaf, netree < tree** |
| *opns:* | $\sigma$ : → **leaf** |
| | $\langle \_, \_ \rangle$ : **tree tree** → **netree** |
| | $\langle \_, \_ \rangle_s$ : **tree tree** → **netree** |
| *vars:* | $L$ : **leaf**, $T_1, T_2$ : **tree** |
| *eqns:* | *hte* $L = L$ |
| | *hte* $\langle s T_1, T_2 \rangle$ = *hte* $T_1$ |
| | *hte* $\langle T_1, T_2 \rangle_s$ = *hte* $T_2$ |

The equations state that the highest terminal element (*hte*) of a tree is the highest terminal element of its strong subtree. Another way of stating this is that the information about the highest terminal element of a subtree $T$ is *percolated* up to its parent node, just in case $T$ is the 's' branch of that node.

## 2 FEATURE GEOMETRY

The particular feature geometry we shall specify here is based on the articulatory structure defined in [4]. The five active articulators are grouped into a hierarchical structure involving a tongue node and an oral node, as shown in the following diagram.




This structure is specified below. The nine sorts and the first three operations describe the desired tree structure, using an approach which should be familiar by now. However, in contrast with our previous specifications, this specification permits ternary branching: the third constructor takes something of sort **glottal** and something of sort velic and combines them with something of sort **oral** to build an object of sort **root**.

FEATURE GEOMETRY =

| | |
|---|---|
| *sorts:* | **glottal, velic, dorsal, coronal** |
| | **labial, tongue, oral, root < gesture** |
| *opns:* | $\langle \_, \_ \rangle$ : **coronal dorsal** → **tongue** |
| | $\langle \_, \_ \rangle$ : **tongue labial** → **oral** |
| | $\langle \_, \_, \_ \rangle$ : **glottal velic oral** → **root** |
| | $\_$ *coronal* : **tongue** → **coronal** |
| | $\_$ *dorsal* : **tongue** → **dorsal** |
| | $\_$ *tongue* : **oral** → **tongue** |
| | $\_$ *labial* : **oral** → **labial** |
| | $\_$ *glottal* : **root** → **glottal** |
| | $\_$ *velic* : **root** → **velic** |
| | $\_$ *oral* : **root** → **oral** |
| *vars:* | $C$ : **coronal**, $D$ : **dorsal**, $T$ : **tongue**, |
| | $L$ : **labial**, $G$ : **glottal**, $V$ : **velic**, $O$ : **oral** |
| *eqns:* | $\langle C, D \rangle$ *coronal* $= C$ |
| | $\langle C, D \rangle$ *dorsal* $= D$ |
| | $\langle T, L \rangle$ *tongue* $= T$ |
| | $\langle T, L \rangle$ *labial* $= L$ |
| | $\langle G, V, O \rangle$ *velic* $= V$ |
| | $\langle G, V, O \rangle$ *oral* $= O$ |
| | $\langle G, V, O \rangle$ *glottal* $= G$ |

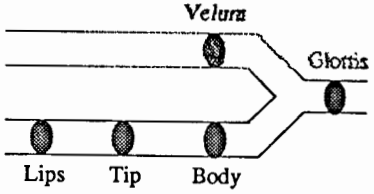
The selectors (e.g. *coronal*) occupy most of the above specification. Notice how each selector mentioned in the *opns* section appears again in the *eqns* section. Consider the *coronal* selector. Its *opns* specification states that it is a function defined on objects of sort tongue which returns something of sort coronal. The corresponding equation states that $\langle C, D \rangle$ *coronal* $= C$. Now $C$ has the sort coronal and $D$ has the sort dorsal. By the definition of the first constructor, $\langle C, D \rangle$ has the sort tongue. Furthermore, by the definition of the *coronal* selector, $\langle C, D \rangle$ *coronal* has the sort coronal. So the equation $\langle C, D \rangle$ *coronal* $= C$ respects the sort definitions.

Selectors can be used to implement structure-sharing (or re-entrancy). Suppose that two segments $S_1$ and $S_2$ share a voicing specification. We can write this as follows: $S_1$ *glottal* $=$ $S_2$ *glottal* . This structure sharing is consis-

tent with one of the main motivating factors behind autosegmental phonology, namely, the undesirability of rules such as [$\alpha$ voice] → [$\alpha$ nasal]. The equation *S glottal = S velic* is illsorted.

Now we can illustrate the function of selectors in phonological rules. Consider the case of English regular plural formation (**-s**), where the voicing of the suffix segment agrees with that of the immediately preceding segment, unless it is a coronal fricative (in which case there must be an intervening vowel). Suppose we introduce the variables $S_1, S_2$ : **root**, where $S_1$ is the stem-final segment and $S_2$ is the suffix. The rule must be able to access the coronal node of $S_1$. Making use of the selectors, this is simply $S_1$*oral tongue coronal* (a notation reminiscent of paths in feature logic, [10]). The rule must test whether this coronal node contains a fricative specification. This necessitates an extension to our specification, which will now be described.

Browman & Goldstein [4:234ff] define 'constriction degree percolation', based on what they call 'tube geometry'. The vocal tract can be viewed as an interconnected set of tubes, and the articulators correspond to valves which have a number of settings ranging from fully open to fully closed. These settings will be called *constriction degrees* (CDs), where fully closed is the maximal constriction and fully open is the minimal constriction.




The net constriction degree of the oral cavity may be expressed as the maximum of the constriction degrees of the lips, tongue tip and tongue body. The net constriction degree of the oral and nasal cavities together is simply the minimum of the two component constriction degrees. To recast this in the present framework we employ our notion of percolation again. In order to simplify exposition, the definition of *max* and *min* are omitted. Moreover, we will also assume, without giving details, that a 'constriction degree' (CD) value is specified for every gesture and is selected by the operator *cd* .

CD = FEATURE GEOMETRY +

*sorts:* **obs, open < cd**
**clo, crit < obs**
**narrow, mid, wide < open**

*opns:* $\_\ cd$ : **gesture → cd**
$max, min$ : **cd cd → cd**

*vars:* $C$ : **coronal**, $D$ : **dorsal**, $T$ : **tongue**, $L$ : **labial**, $G$ : **glottal**, $V$ : **velic**, $O$ : **oral**

*eqns:* $\langle G,V,O\rangle\ cd\ =$
$max(G\ cd\ , min(V\ cd\ , O\ cd\ ))$
$\langle T,L\rangle\ cd\ = max(T\ cd\ , L\ cd\ )$
$\langle C,D\rangle\ cd\ = max(C\ cd\ , D\ cd\ )$

There are five basic constriction degrees (**clo**, **crit**, **narrow**, **mid**, and **wide**), and these are grouped into two sorts **obs** and **open**.

Using the above extension, the condition on the English voicing assimilation rule could be expressed as follows[4], where *Crit* : **crit**:

$S_1$ *oral tongue coronal cd* ≠ **crit**

If this condition is met, the effect of the rule would be:

$S_1$ *glottal cd* = $S_2$ *glottal cd*

This is how we say that $S_1$ and $S_2$ have the same voicing.

Now the manner features can be expressed as follows (omitting **strident** and **lateral**).

MANNER FEATURES = CD

*opns:* $+\_$ : → **bool**
$-\_$ : → **bool**
*son* $\_$ : **root** → **bool**
*cont* $\_$ : **root** → **bool**
*cons* $\_$ : **root** → **bool**
*nas* $\_$ : **root** → **bool**

*vars:* $R$ : **root**, $G$ : **glottal**, $V$ : **velic**, $O$ : **oral**
*Open* : **open**, *Obs* : **obs**, *Clo* : **clo**

*eqns:* *son* $R = +$ *iff* $R\ cd$ = *Open*
*cont* $\langle G,V,O\rangle = -$ *iff* $O\ cd$ = *Clo*
*cons* $\langle G,V,O\rangle = -$ *iff* $O\ cd$ = *Obs*
*nas* $\langle G,V,O\rangle = -$
*iff* $V\ cd$ = *Open and* $O\ cd$ = *Obs*

[4] A proviso is necessary here. Just because there is a critical CD at the tongue tip does not mean that a fricative is being produced. For example, the lips might be closed. We can get around this problem with the use of CD percolation (as already defined) and the equation $S_1$*oral* = **crit**. Further discussion of this option may be found in [2].



It follows directly from the above definitions that the collection of noncontinuants is a subset of the set of consonants (since **clo** < **obs**). Similarly, the collection of nasals is a subset of the set of consonants. Note also that these definitions permit manner specification independently of place specification, which is often important in phonological description.

## 3 CONCLUSIONS

We began this article by pointing out the difficulty of defining and using complex phonological structures. In addressing this problem we have used a strategy from computer science known as abstract specification. We believe this brings us a step further towards our goal of developing a computational phonology.

This approach contrasts with the finite state approach to computational phonology [1,6]. Finite state grammars have employed a rigid format for expressing phonological information, and have not hitherto been able to represent the complex hierarchical structures that phonologists are interested in. Our approach has been to view phonological structures as abstract data types, and to obtain a rich variety of methods for structuring those objects and for expressing constraints on their behaviour.

We have briefly examined the idea that data can be structured *in terms of* sorts and operations on items of specific sorts. We also explored the organization of data into a hierarchy of classes and subclasses, where data at one level in the hierarchy inherits all the attributes of data higher up in the hierarchy. Inheritance hierarchies provide a succinct and attractive method for expressing a wide variety of linguistic generalizations. A useful extension would be to incorporate *default inheritance* into this system.

Further exploration of these proposals, we believe, will ultimately enable the mechanical testing of predictions made by phonological systems and the incorporation of phonological components into existing computational grammars.

## 4 REFERENCES


[1] ANTWORTH, E. L. (1990). *PC-KIMMO: A Two-level Processor for Morphological Analysis.* Dallas: Summer Institute of Linguistics.

[2] BIRD, S. (1990). *Constraint-Based Phonology.* Ph.D. Thesis. University of Edinburgh.

[3] BIRD, S. & E. KLEIN (1990). Phonological events. *Journal of Linguistics, 26,* 33–56.

[4] BROWMAN, C. & L. GOLDSTEIN (1989). Articulatory gestures as phonological units. *Phonology, 6,* 201–251.

[5] CARDELLI, L. (1988). A semantics of multiple inheritance. *Information and Computation, 76,* 138–164.

[6] DALRYMPLE, M., R. KAPLAN, L. KARTTUNEN, K. KOSKENNIEMI, S. SHAIO & M. WESCOAT (1987). *Tools for Morphological Analysis.* CSLI–87–108. CSLI, Stanford.

[7] EHRIG, H. & B. MAHR (1985). *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics.* Berlin: Springer Verlag.

[8] GOGUEN, J.A., J.W. THATCHER & E.G. WAGNER (1976). An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. Yeh (ed.) *Current Trends in Programming Methodology IV: Data Structuring.* 80–144. Englewood Cliffs, NJ : Prentice Hall.

[9] GOGUEN, J.A., & T. WINKLER (1988). *Introducing OBJ3.* Technical Report SRI-CSL-88-9, SRI International, Computer Science Laboratory, Menlo Park, CA.

[10] KASPER, R. & W. ROUNDS (1990). The logic of unification in grammar. *Linguistics and Philosophy, 13,* 35–58.

[11] SMOLKA, G. & H. AIT-KACI (1989) Inheritance hierarchies: semantics and unification. *Journal of Symbolic Computation, 7,* 343–370.