

Phonological Interpretation into Preordered Algebras

Yusuke Kubota and Carl Pollard
The Ohio State University

{kubota,pollard}@ling.ohio-state.edu

1 Introduction

Standard denotational semantics of functional programming languages (FPLs) follows Scott (1993 [1969]) in interpreting programs and their parts into a Henkin frame of (complete) posets whose orders correspond to *degree of definedness*. Maximal members of the posets that interpret the base types are the interpretations of *values*, the irreducible terms returned by terminating programs. One reason about meanings of program constructs in an *inequational* logic that extends the familiar equational logic for reasoning about meanings of lambda-terms.

In a separate development initiated by Solias (1992) and Solias and Morrill (1993), building on Oehrle (1988), multimodal categorical grammars (MMCGs) are *phonologically* interpreted by labelling syntactic derivations not just with meaning terms, but also with terms which take their denotations in an algebra whose operations model modes of phonological combination. (Following Oehrle, we call these ϕ -terms, and write $\mathbf{a}; m; A$ for syntactic type A labelled with ϕ -term \mathbf{a} and meaning term m .)

Combining these two ideas, we propose ϕ -term labelling for MMCGs with interpretation into preordered algebras whose ‘modes of phonological combination’ are binary operations monotonic in both arguments. Unlike the denotational semantics of FPLs, though, deducibility of $\mathbf{a} \leq \mathbf{a}'$ means not that \mathbf{a}' is at least as *defined* as \mathbf{a} , but rather that \mathbf{a}' is at least as *pronounceable* as \mathbf{a} , in the sense that any syntactic derivation that can be phonologically interpreted as \mathbf{a} can also be phonologically interpreted as \mathbf{a}' . And the maximal elements of the algebra interpreting the base type *phon*—the interpretations of the phonological *values*, are the ones that can be ‘returned’ in the sense of being actual pronunciations. Besides conceptual clarity, the main advantage of this approach is the wholesale elimination from the syntactic type logic of structural axioms or rules that govern semantically inert word order variation; all can be replaced by a single *interface schema* asserting that if $\mathbf{a} \leq \mathbf{a}'$ is provable in the inequational theory associated with the ϕ -calculus and $\mathbf{a}; m; A$ is derivable in the syntactic calculus, then so is $\mathbf{a}'; m; A$.

The paper is organized as follows. Section 2 recalls basic definitions about preordered algebras. Section 3 reviews relevant aspects of the semantics of the programming language PCF. Section 4 sketches our system of ϕ -labelling in the

context of an MMCG for a Japanese fragment. And section 5 uses this grammar to analyze some complex facts about word order in Japanese.

2 Preordered Algebras

A **preorder** is a reflexive transitive relation, and an **order** is an antisymmetric preorder. A **preordered set** (resp. **poset**) is a set P together with a preorder (resp. order), but usually we just call a preordered set a preorder. Any preorder \sqsubseteq induces an equivalence relation defined by $p \equiv q$ iff $p \sqsubseteq q$ and $q \sqsubseteq p$. A function from one preorder to another is **monotonic** (resp. **antitonic**) if it preserves (resp. reverses) the preorder, and **tonic** if it is either monotonic or antitonic.

Any collection of preorders generates a Henkin frame by closure under exponentiation, where the exponential $P \rightarrow Q$ of two preorders is taken to be the set of monotonic functions from P to Q with the pointwise preorder.

A **preordered** (resp. **monotonic**) **algebra** is a preorder together with a collection of tonic (resp. monotonic) operations. A simple example of a monotonic algebra is a **presemigroup**, with one binary operation \frown which is associative up to equivalence (u.t.e.), i.e. $(p \frown q) \frown r \equiv p \frown (q \frown r)$. A less familiar example is a monotonic algebra with one operation $\frown_{<}$ which is only left-associative, i.e. $(p \frown_{<} q) \frown_{<} r \sqsubseteq p \frown_{<} (q \frown_{<} r)$.

A **cpo** is a poset with a bottom (\perp) element in which every chain has a least upper bound (lub). A function from one cpo to another is **continuous** if it preserves lubs of chains. Continuous functions can be shown to be monotonic. The cpo's form a Henkin frame with the sets of continuous functions as exponentials. Every continuous function f from a cpo to itself has a least fixed point, equal to $\bigsqcup_{n \geq 0} f^n(\perp)$. A **flat** cpo is one where $p \sqsubseteq q$ implies $p = q$ or $p = \perp$.

3 PCF

As an alternative to early untyped FPLs, Scott (1993 [1969]) proposed a typed lambda calculus, later dubbed LCF (logic of computable functions), with base types *nat* and *bool*. LCF includes constants for truth values, natural numbers, successor, predecessor, and zero-test, as well as McCarthy conditionals and recursion operators. Plotkin (1977) designed a simple LCF-based FPL, PCF, in which programs are modelled as closed terms of a base type, and computation as a form of call-by-name (CBN) **evaluation**. The irreducible terms returned by terminating programs are called **values**. Evaluation is confluent, but because of the recursion operators, program termination is not guaranteed.

LCF/PCF is equipped with an interpretation I (denotational semantics) into a Henkin frame whose domains are cpo's, with functional terms interpreted as continuous functions. Base types are interpreted as flat domains, with the bottoms being the meanings of nonterminating programs. For two functional terms s and t of the same type, $I(s) \sqsubseteq I(t)$ means that, thought of as partial functions, $I(t)$ is at least as defined as $I(s)$.

One reason about the meanings of (parts of) PCF programs using an **inequational** logic whose formulas are inequalities $s \leq t$ between terms of the

same type. Predictably, the axioms and rules of the inequational logic include β -conversion, Reflexivity and Transitivity (sound because the domains are cpo's, hence preorders), as well as a form of Monotonicity (of function application only with respect to the first (the function) argument,¹ sound because the order on functions is defined pointwise).

4 MMCG with Preordered Phonology

Following Morrill and Solias 1993, we assign phonological and semantic labels to types in syntactic derivations, e.g. $\mathbf{a}; m; A$; such triples are called *signs*. Derivations themselves are written in the familiar natural-deduction style with hypotheses introduced at the top. Besides the usual logical rules (Introduction and Elimination rules for the various flavors of $/$ and \backslash), the syntactic calculus includes an *Interface* rule, which asserts that if $\mathbf{a} \leq \mathbf{a}'$ is deducible in the inequational theory associated with the calculus of phonological terms (hereafter, ϕ -calculus, to be described below), then any sign with ϕ -component \mathbf{a} has a counterpart with the same meaning and the same syntactic category but with ϕ -component \mathbf{a}' . The Interface rule, together with the inequational ϕ -theory that governs the various modes of phonological combination and their interactions with each other, obviate the need for any structural rules in the syntax. In short: the ϕ -calculus governs semantically inert surface-oriented word order variation; the logical rules of the syntax govern the semantically relevant syntactic combinatorics; and the Interface rule is the channel of communication between the syntax and the phonology that enables signs to become pronounceable.

The rules of the syntactic calculus are as follows:

$$(1) \quad \begin{array}{ll} \text{a. } \mathbf{Forward\ Slash\ Elimination} & \text{b. } \mathbf{Backward\ Slash\ Elimination} \\ \frac{\mathbf{a}; m; A /_i B \quad \mathbf{b}; n; B}{\mathbf{a} \circ_i \mathbf{b}; m(n); A} /_i E & \frac{\mathbf{b}; n; B \quad \mathbf{a}; m; B \backslash_i A}{\mathbf{b} \circ_i \mathbf{a}; m(n); A} \backslash_i E \end{array}$$

$$(2) \quad \begin{array}{ll} \text{a. } \mathbf{Forward\ Slash\ Introduction} & \text{b. } \mathbf{Backward\ Slash\ Introduction} \\ \frac{\begin{array}{c} \vdots \quad \vdots \quad [p; x; A]^n \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array}}{\mathbf{b} \circ_i p; m; B} /_i I^n & \frac{\begin{array}{c} \vdots \quad \vdots \quad [p; x; A]^n \quad \vdots \quad \vdots \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array}}{p \circ_i \mathbf{b}; m; B} \backslash_i I^n \\ \frac{\mathbf{b} \circ_i p; m; B}{\mathbf{b}; \lambda x. m; B /_i A} /_i I^n & \frac{p \circ_i \mathbf{b}; m; B}{\mathbf{b}; \lambda x. m; A \backslash_i B} \backslash_i I^n \end{array}$$

(3) **Interface rule**

$$\frac{\mathbf{a}; m; A}{\mathbf{a}'; m; A} \text{PI}$$

(where $\mathbf{a} \leq \mathbf{a}'$ is a theorem in the inequational theory of ϕ -terms)

¹Corresponding to CBN evaluation.

Although most of the generalizations regarding word order are taken care of in the ϕ -calculus, the left- vs. right-slash distinction is retained for syntactic types. The semantic and phonological annotations on the rules are mostly as one would expect. The semantics is just function application and lambda abstraction for Elimination and Introduction rules, respectively. For Forward and Backward Slash Elimination, the phonology of the derived expression is obtained by combining the phonologies of the functor and the argument in the right order and in the right mode. And the Forward (respectively, Backward) Slash Introduction rule says roughly that a linguistic expression whose phonology is \mathbf{b} alone is of category B/A (resp. $A \setminus B$), given the hypothetical proof that \mathbf{b} concatenated with p (whose category is A) to its right (resp. left) is of category B .

The phonology for the derived expression in the Introduction rules might look somewhat non-standard, in that, unlike the semantics, it does not seem to involve lambda abstraction. Instead, the phonology of the hypothetically assumed expression (which is a variable of type *phon*) is simply stripped off from the phonology of the whole expression. To motivate this, we can think of the derived expression in the Introduction rule as actually having the following phonology:

$$(4) \quad \lambda p. [\mathbf{b} \circ_i p](\epsilon)$$

where ϵ is the identity element (interpreted as a null string). But in the inequational ϕ -theory (see below), this is provably equal to \mathbf{b} (by β -conversion followed by one of the inequalities for the null phonology). That is, unlike in semantics, when the variable is bound, the resulting abstract is immediately applied to the null-phonology term, so that nothing can be ‘reconstructed’ to a gap in the surface string. The Introduction rules as stated can be thought of as implicitly compiling in these reduction steps together with an application of the Interface rule to produce a conclusion with the simplified phonology \mathbf{b} .

This is as it should be, given the typical way in which hypothetical reasoning is used in linguistic analyses in categorial grammar. That is, hypothetical reasoning is used in the analyses of phenomena that involve some kind of displaced constituency (such as long-distance dependencies and verb raising), and the phonologies of the hypothesized expressions should appear in the displaced position on the surface string rather than reconstructed in the original position. For this reason, the original ‘gap’ of the displaced element is filled by a null element as soon as the hypothetical reasoning step is taken. (We will illustrate how this works in an actual linguistic analysis with the Japanese fragment in the next section.)

As stated above, the Interface rule is the channel between syntax and phonology, and the actual relation of relative pronounceability among phonologies of linguistic expressions is defined by the preorder imposed on the domain that interprets the phonological base type in the Henkin frame that models the inequational ϕ -theory. Thus this theory is analogous to the inequational theory used to reason about relative definedness of PCF programs, and closely resembles it in form.

Specifically, the ϕ -calculus is a typed lambda calculus with one base type *phon*, constants of type *phon* (lexical phonologies and null phonology), and constants of type *phon* \multimap *phon* \multimap *phon* (modes of ϕ -composition). As with

PCF, the formulas of the inequational theory are of the form $\mathbf{a} \leq \mathbf{b}$, but now the inequated terms denote not program constructs with varying degrees of definedness, but rather phonological entities with varying degrees of pronounceability. Also as with PCF, the axioms (or reduction rules) include β -conversion, Reflexivity (5a), and Transitivity (5b), and a form of Monotonicity (for all the ϕ -modes, in both arguments) (6). Of course PCF also has parochial axioms (reduction rules for the built-in arithmetic operators, McCarthy conditionals, and fixed-point operators); and the ϕ -calculus does too, namely word-order variation rules for the specific modes ((8) with i and j being the same mode), rules of interaction between modes ((8) with i and j being different modes), and a rule that allows any of the ‘flavored’, unpronounceable modes to be replaced by the ‘plain vanilla’ fully pronounceable mode (concatenation) (9). In the model, *phon* is interpreted as a monotonic algebra whose maximal members (values) are the pronounceable phonologies (strings). ϕ -term reduction always terminates, but is non-confluent, corresponding to semantically inert word order variation.

(5) Structured phonologies form a preorder.

$$\text{a. } \frac{}{\mathbf{a} \leq \mathbf{a}} \text{REFL} \qquad \text{b. } \frac{\mathbf{a} \leq \mathbf{b} \quad \mathbf{b} \leq \mathbf{c}}{\mathbf{a} \leq \mathbf{c}} \text{TRANS}$$

(6) The \circ_i are monotonic in both arguments.

$$\frac{\mathbf{a} \leq \mathbf{a}' \quad \mathbf{b} \leq \mathbf{b}'}{\mathbf{a} \circ_i \mathbf{b} \leq \mathbf{a}' \circ_i \mathbf{b}'} \text{MON} \qquad (\circ_i \in \{\circ_*, \circ_>, \circ_<, \circ_\times, \circ.\})$$

(7) ϵ is a two-sided identity for all the \circ_i .

$$\text{a. } \frac{}{\epsilon \circ_i \mathbf{a} \leq \mathbf{a}} \text{ID}_l \qquad \text{b. } \frac{}{\mathbf{a} \circ_i \epsilon \leq \mathbf{a}} \text{ID}_r$$

$$(\circ_i \in \{\circ_*, \circ_>, \circ_<, \circ_\times, \circ.\}) \qquad (\circ_i \in \{\circ_*, \circ_>, \circ_<, \circ_\times, \circ.\})$$

(8) Mode-specific rules:

$$\text{a. (LA; left associativity)} \qquad \frac{}{(\mathbf{a} \circ_i \mathbf{b}) \circ_j \mathbf{c} \leq \mathbf{a} \circ_i (\mathbf{b} \circ_j \mathbf{c})} \text{LA} \qquad (\circ_i, \circ_j \in \{\circ_<, \circ.\})^2$$

$$\text{b. (RA; right associativity)} \qquad \frac{}{\mathbf{a} \circ_i (\mathbf{b} \circ_j \mathbf{c}) \leq (\mathbf{a} \circ_i \mathbf{b}) \circ_j \mathbf{c}} \text{RA} \qquad (\circ_i, \circ_j \in \{\circ_>, \circ.\})$$

$$\text{c. (Perm; permutation)} \qquad \frac{}{\mathbf{a} \circ_i (\mathbf{b} \circ_j \mathbf{c}) \leq \mathbf{b} \circ_i (\mathbf{a} \circ_j \mathbf{c})} \text{PERM} \qquad (\circ_i, \circ_j \in \{\circ_\times, \circ.\})$$

(9) Concatenation mode is most pronounceable (phonological values are strings).

$$\frac{}{\mathbf{a} \circ_i \mathbf{b} \leq \mathbf{a} \circ \mathbf{b}} \text{CONCAT} \qquad (\circ_i \in \{\circ_*, \circ_>, \circ_<, \circ_\times, \circ.\})$$

²Note about notation: the modes i and j can (but need not) be identical.

5 A Japanese Fragment

We illustrate the system developed in the previous section with an analysis of word-order variation found in the *-te* form complex predicate construction in Japanese. A fuller set of facts and an analysis embodying the same basic idea but formulated within the framework of Multi-Modal Combinatory Categorical Grammar (Baldrige, 2002) can be found in Kubota (2008). We here focus on two facts, the basic scrambling pattern and the patterns of argument cluster coordination in the *-te* form complex predicate. In both of these cases, the embedded verb (V1) and the embedding verb (V2) cluster together, suggesting that they are combined in a mode that is tighter than the mode in which ordinary nominal arguments are combined with the verbs that subcategorize for them. V1 and V2 are independent words syntactically, however, given that this construction systematically differs from the more typical cases of lexical complex predicate constructions in Japanese in terms of the wordhood of the sequence of the V1 and V2.³

Japanese does not freely allow long-distance scrambling. That is, normally, elements of an embedded clause cannot be scrambled to the domain of the higher clause. However, in the *-te* form complex predicate construction, such scrambling patterns are perfectly acceptable, suggesting that V1 and V2 form a single clausal domain in this construction. (10b) is a case in which the accusative object *piano-o* of V1 is scrambled over a matrix dative argument *John-ni*:

- (10) a. Mary-ga John-ni piano-o **hii-te morat-ta.**
 Mary-NOM John-DAT piano-ACC play-TE BENEF-PAST
 ‘Mary had John play the piano for her.’
 b. Mary-ga *piano-o* John-ni **hii-te morat-ta.**

Argument cluster coordination patterns also suggest that, in this construction, (semantic) arguments of V1 and V2 are clausemates. As in (11), argument cluster coordination involving arguments of both V1 and V2 is possible, as long as the cluster of V1 and V2 is not split apart.

- (11) Mary-ga [John-ni piano-o], [Bill-ni gitaa-o] **hii-te**
 Mary-NOM John-DAT piano-ACC Bill-DAT guitar-ACC play-TE
morat-ta.
 BENEF-PAST
 ‘Mary had John play the piano and Bill play the guitar for her.’

Under the account we propose, these word-order variation facts receive a straightforward account in terms of ϕ -modalities. The gist of the analysis is that V1 and V2 combine in a mode tighter than the mode in which ordinary arguments combine with the verbs that subcategorize for them, but looser than the mode in which components of lexical complex predicates combine. Specifically, we specify in the lexicon that V1 and V2 in this construction combine with one another in the left-associative mode $\circ_{<}$, which is distinct from the default scrambling mode \circ . used for putting together nominal arguments with their verbal heads, which is both left- and right-associative and permutative. A sample lexicon is given in (12):

³For the relevant set of facts, see Kubota (2008).

$$(12) \quad \begin{array}{ll} \text{mary-ga; } \mathbf{m}; \text{NP}_n & \text{bill-ni; } \mathbf{b}; \text{NP}_d \\ \text{piano-o; } \mathbf{p}; \text{NP}_a & \text{hii-te; } \mathbf{play}; \text{NP}_a \backslash . \text{NP}_n \backslash . \text{S} \\ \text{gitaa-o; } \mathbf{g}; \text{NP}_a & \text{morat-ta; } \lambda P \lambda y \lambda x. \mathbf{benef}(x, P(y)); \\ \text{john-ni; } \mathbf{j}; \text{NP}_d & (\text{NP}_n \backslash . \text{S}) \backslash < (\text{NP}_d \backslash . \text{NP}_n \backslash . \text{S}) \end{array}$$

The derivation for (10) is given in (13):

$$(13) \quad \frac{\frac{\frac{\text{piano-o; } \mathbf{p}; \text{NP}_a \quad \text{hii-te; } \mathbf{play}; \text{NP}_a \backslash . \text{NP}_n \backslash . \text{S}}{\text{piano-o} \circ \text{hii-te; } \mathbf{play}(\mathbf{p}); \text{NP}_n \backslash . \text{S}} \backslash . \text{E} \quad \frac{\text{morat-ta; } \lambda P \lambda y \lambda x. \mathbf{benef}(x, P(y)); (\text{NP}_n \backslash . \text{S}) \backslash < (\text{NP}_d \backslash . \text{NP}_n \backslash . \text{S})}{\text{piano-o} \circ \text{hii-te} \circ < \text{morat-ta; } \lambda y \lambda x. \mathbf{benef}(x, \mathbf{play}(\mathbf{p})(y)); \text{NP}_d \backslash . \text{NP}_n \backslash . \text{S}} \backslash . \text{E}}{\text{john-ni; } \mathbf{j}; \text{NP}_d \quad \frac{\text{piano-o} \circ \text{hii-te} \circ < \text{morat-ta; } \lambda y \lambda x. \mathbf{benef}(x, \mathbf{play}(\mathbf{p})(y)); \text{NP}_d \backslash . \text{NP}_n \backslash . \text{S}}{\text{john-ni} \circ ((\text{piano-o} \circ \text{hii-te}) \circ < \text{morat-ta}); \lambda x. \mathbf{benef}(x, \mathbf{play}(\mathbf{p})(\mathbf{j})); \text{NP}_n \backslash . \text{S}} \backslash . \text{E}} \backslash . \text{E}}{\text{piano-o} \circ (\text{john-ni} \circ (\text{hii-te} \circ \text{morat-ta})); \lambda x. \mathbf{benef}(x, \mathbf{play}(\mathbf{p})(\mathbf{j})); \text{NP}_n \backslash . \text{S}} \text{PI}$$

The key step in this derivation is the last one. Two things are happening here: (i) the direct object *piano-o* of the embedded verb scrambles over the dative argument of the higher verb, resulting in the surface order in which the former linearly precedes the latter and (ii) the modes by which the ϕ -terms of lexical words are combined are all converted to the concatenation mode \circ , so that we get a pronounceable ϕ -term for the expression derived. Technically, this last step is an application of the Interface rule, whose validity is supported by the following lemma in the inequational logic for ϕ -terms:

$$(14) \quad \textbf{Lemma: } a \circ ((b \circ c) \circ < d) \leq b \circ (a \circ (c \circ d))$$

Proof:

$$\frac{\frac{\frac{a \leq a}{\text{RFF}} \quad \frac{(b \circ c) \circ < d \leq b \circ (c \circ d)}{\text{LA}}}{a \circ ((b \circ c) \circ < d) \leq a \circ (b \circ (c \circ d))} \text{MON} \quad \frac{a \circ (b \circ (c \circ d)) \leq b \circ (a \circ (c \circ d))}{\text{PERM}}}{a \circ ((b \circ c) \circ < d) \leq b \circ (a \circ (c \circ d))} \text{TRANS}$$

$$\frac{\frac{\vdots}{a \circ ((b \circ c) \circ < d) \leq b \circ (a \circ (c \circ d))} \quad \frac{b \circ (a \circ (c \circ d)) \leq b \circ (a \circ (c \circ d))}{\text{CONCAT}}}{a \circ ((b \circ c) \circ < d) \leq b \circ (a \circ (c \circ d))} \text{TRANS}$$

Intuitively, what is going on here is that, due to the fact that the mode employed in combining V1 and V2 is left associative, V1 and V2 can be analyzed as forming a verb cluster by left associativity (8a). Once this verb cluster is formed, the object of the embedded verb has the same status as arguments of the higher verb and thus can scramble over the matrix dative argument by permutation (8c). In short, the less pronounceable modes (ones other than \circ) govern the way abstract ‘structured’ phonologies⁴ are mapped to other ‘structured’ phonologies until they are ultimately mapped to the actually pronounceable phonologies involving only the \circ mode.

The derivation for the more complex, argument cluster coordination case (11) goes as follows (semantics is omitted for the sake of simplicity of presentation):⁵

⁴Of course, it is actually the ϕ -terms themselves that are ‘structured’, not their denotations, which are merely algebra elements.

⁵The step marked by ‘&’ in (15b) is licensed by a non-logical coordination rule. This is needed because the type of coordination in Japanese under consideration does not involve an overt conjunction. Another possibility would be to posit a phonologically empty conjunction. The choice between these two options does not have any significance to the overall analysis we propose here, but we opt for the treatment in the text in view of avoiding phonologically empty elements whenever possible.

$$(15) \quad \text{a.} \quad \frac{\frac{\frac{[p; ; NP_a]^I \quad hii-te; ; NP_a \setminus VP}{p \circ hii-te; ; VP} \setminus_E \quad morat-ta ; ; VP \setminus NP_d \setminus VP}{(p \circ hii-te) \circ_{<} morat-ta; ; NP_d \setminus VP} \setminus_{<E}}{\frac{p \circ (hii-te \circ_{<} morat-ta); ; NP_d \setminus VP}{hii-te \circ_{<} morat-ta; ; NP_a \setminus NP_d \setminus VP} \setminus_{I^1}} \text{PI}$$

b.

$$\frac{\frac{\frac{\frac{john-ni; ; NP_d \quad \frac{piano-o; ; NP_a \quad [q; ; NP_a \setminus NP_d \setminus VP]^2}{piano-o \circ q; ; NP_d \setminus VP} \setminus_E}{john-ni \circ (piano-o \circ q); ; VP} \setminus_E}{(john-ni \circ piano-o) \circ q; ; VP} \text{PI}}{\frac{john-ni \circ piano-o; ; VP / (NP_a \setminus NP_d \setminus VP)}{(john-ni \circ piano-o) \circ_* (bill-ni \circ gitaa-o); ; VP / (NP_a \setminus NP_d \setminus VP)} \setminus_{I^2} \quad \begin{array}{c} \vdots \\ \vdots \end{array} \quad \text{bill-ni} \circ \text{gitaa-o}; ; VP / (NP_a \setminus NP_d \setminus VP)} \&$$

c.

$$\frac{\frac{john-ni \circ piano-o) \circ_* (bill-ni \circ gitaa-o); ; VP / (NP_a \setminus NP_d \setminus VP) \quad hii-te \circ_{<} morat-ta; ; NP_a \setminus NP_d \setminus VP}{((john-ni \circ piano-o) \circ_* (bill-ni \circ gitaa-o)) \circ (hii-te \circ_{<} morat-ta); ; VP} \setminus_E}{((john-ni \circ piano-o) \circ (bill-ni \circ gitaa-o)) \circ (hii-te \circ morat-ta); ; VP} \text{PI}$$

The key point in this derivation is that the sequence of V1 and V2 is analyzed as a derived ditransitive verb. This is shown in the first chunk of the derivation (15a). By hypothesizing a direct object for the embedded verb, we can form an embedded VP which can then be given as an argument to the matrix verb. Once the embedded verb combines with the matrix verb, the Interface rule is applicable to shift the phonology of the embedded object to the left edge. This feeds into the Slash Introduction rule at the next step, which assigns a ditransitive verb-like category to the string of words composed solely of V1 and V2. The rest of the derivation is straightforward. Two argument clusters are coordinated to form a larger expression of the same category (15b) and this coordinated argument cluster takes as its argument the derived ditransitive verb (i.e. the sequence of V1 and V2) as an argument to produce a VP (15c).

In this analysis, the whole derivation crucially depends on the possibility of assigning a derived ditransitive verb category to the sequence of the V1 and V2. This is made possible in the current fragment since instances of the Interface rule can be interleaved with instances of the logical rules in the course of syntactic derivations, licensing hypothetical reasoning that partly depends on (the abstract representations of) the surface phonological forms of linguistic expressions. Note also that the present treatment of the cluster of V1 and V2 as a ‘derived’ ditransitive verb depends on the way the Introduction rules are formulated: the phonology of the derived ditransitive verb at step (15a) is simply of type *phon*, rather than being of a functional type *phon* \multimap *phon*. This enables it to be given as an argument to the argument cluster in the application of the Elimination rule at the final step (15c).

To summarize, we have demonstrated above that the proposed system that interprets phonological terms in a preordered algebra enables an elegant and formally precise analysis of complex word order facts of the Japanese *-te* form complex predicate construction.

6 Conclusion

The proposed system of ϕ -labelling for MMCG most closely resembles that of Morrill (1994). In Morrill’s system, in addition to logical rules analogous to ours, the syntactic calculus includes several kinds of structural rules that manipulate

only the forms of ϕ -terms. That is, much of the syntactic logic is given over to modelling reasoning about semantically irrelevant surface-oriented phonological forms of linguistic expressions. Essentially the same idea is present in other variants of Type-Logical Grammar, such as Moortgat (1997) and Bernardi (2002); in the latter, for example, the relevant idea is implemented via the notion of ‘structured antecedents’ in the sequent-style natural deduction presentation. The main way our approach differs from these is that it more explicitly recognizes such reasoning as pertaining to a separate ϕ -component through which linguistic expressions are mapped to their surface phonological realizations, while the syntactic calculus (apart from the Interface rule) governs the abstract syntactic combinatorics that guide semantic composition. Moreover, our approach builds straightforwardly on decades-old technology—modelling the approximation of realizable values by interpreting typed lambda calculi into Henkin frames of preorders—in a way that illuminates phonological realization as a kind of computation.

References

- Baldrige, J. 2002. *Lexically Specified Derivational Control in Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- Bernardi, R. 2002. *Reasoning with Polarity in Categorical Type Logic*. Ph.D. thesis, University of Utrecht.
- Kubota, Y. 2008. Solving the morpho-syntactic puzzle of the Japanese *-te* form complex predicate: A Multi-Modal Combinatory Categorical Grammar analysis. In *Empirical Issues in Syntax and Semantics 7*, 283–306.
- Moortgat, M. 1997. Categorical Type Logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, 93–177. Amsterdam: Elsevier.
- Morrill, G. and T. Solias. 1993. Tuples, discontinuity, and gapping in categorical grammar. In *Proceedings of EACL 6*, 287–296. Morristown, NJ, USA: Association for Computational Linguistics.
- Morrill, G. V. 1994. *Type Logical Grammar: Categorical Logic of Signs*. Dordrecht: Kluwer.
- Oehrle, R. T. 1988. Multi-dimensional compositional functions as a basis for grammatical analysis. In R. T. Oehrle, et al., eds., *Categorical Grammars and Natural Language Structures*, 349–389. Dordrecht: Reidel.
- Plotkin, G. D. 1977. LCF considered as a programming language. *Theoretical Computer Science* 5:223–255.
- Scott, D. 1993. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science* 121:411–440. (Revision of unpublished 1969 manuscript.)
- Solias, M. T. 1992. *Gramáticas Catoriales, Coordinación Generalizada y Elisión*. Ph.D. thesis, Universidad Autónoma de Madrid.