Scope ambiguities, monads and strengths

Justyna Grudzińska[1] *and Marek Zawadowski*[2]
[1] Institute of Philosophy, University of Warsaw, Warsaw, Poland
[2] Institute of Mathematics, University of Warsaw, Warsaw, Poland

ABSTRACT

In this paper, we will discuss three semantically distinct scope assignment strategies: traditional movement strategy, polyadic approach, and continuation-based approach. Since generalized quantifiers on a set X are elements of $\mathscr{C}(X)$, which is the value of the continuation monad $\mathscr{C}$ on X, quantifier phrases are interpreted as $\mathscr{C}$-computations, in all three approaches. The main goal of this paper is to relate the three strategies to the computational machinery connected to the monad $\mathscr{C}$ (strength and derived operations). As will be shown, both the polyadic approach and the continuation-based approach make heavy use of monad constructs. In the traditional movement strategy, monad constructs are not used but we still need them to explain how the three strategies are related and what can be expected of them with regard to handling scopal ambiguities in simple sentences.

Keywords: scope ambiguity, continuation monad, strength

1 MULTI-QUANTIFIER SENTENCES AND THREE SCOPE-ASSIGNMENT STRATEGIES

Multi-quantifier sentences can be ambiguous, with different readings corresponding to how various quantifier phrases (QPs) are semantically related in the sentence. For example,

(1) Every girl likes a boy

admits of the subject wide-scope reading ($S > O$) where each girl likes a potentially different boy, and the object wide-scope reading ($O > S$) where there is one boy whom all the girls like. As the number of QPs in a sentence increases, the number of distinct readings also increases.

Thus a simple sentence with three QPs admits of six possible readings, and in general a simple sentence with n QPs will be (at least) $n!$ ways ambiguous (we only consider readings where QPs are linearly ordered – what we will call asymmetric readings). In this paper, we will discuss three semantically distinct scope-assignment strategies:

Strategy A: Traditional movement strategy (Cooper 1983; May 1978; Montague 1973).

Strategy B: Polyadic approach (Keenan 1992, 1987; May 1985; Van Benthem 1989; Zawadowski 1989).

Strategy C: Continuation-based approach (Barker 2002; Barker and Shan 2014; Bekki and Asai 2009; De Groote 2001; Kiselyov and Shan 2014).

In all three strategies, QPs are interpreted as generalized quantifiers. A generalized quantifier on a set X is of type $\mathscr{C}(X) = (X \to \mathbf{t}) \to \mathbf{t}$ (with $\mathbf{t} = \{\mathbf{true}, \mathbf{false}\}$). The main difference between the three approaches lies in the semantic operations used to compute the truth-value of the relevant multi-quantifier sentences.

1.1 *Strategy A*

Strategy A has been implemented in various ways, using May's QR (1978), Montague's Quantifying In Rule (1973), or Cooper Storage (1983). In this strategy, the scope relations for multi-quantifier sentences like (1) are derived by applying quantifiers to the predicate (of type $\mathscr{P}(X \times Y) = (X \times Y) \to \mathbf{t}$) one by one – the later the quantifier is introduced, the wider its scope. In the terminology to be adopted in this paper, strategy A makes use of what we will call, after Mostowski, partial **mos**-operations:

$$\mathbf{mos}_Y : \mathscr{C}(Y) \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X)$$

defined by a lambda term as:

$$\mathbf{mos}_Y = \lambda Q_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y));$$

and total **mos**-operations:

$$\mathbf{mos}_X : \mathscr{C}(X) \times \mathscr{P}(X) \longrightarrow \mathbf{t}$$

defined by a lambda term as:

$$\mathbf{mos}_X = \lambda Q_{:\mathscr{C}(X)}.\lambda c_{:\mathscr{P}(X)}.Q(c).$$

Strategy A can be straightforwardly extended to account for sentences involving three or more QPs (by allowing permutations of QPs).

1.2 *Strategy B*

Strategy B involving polyadic quantification was introduced and developed in the works of May (1985), Keenan (1987, 1992), Zawadowski (1989) and Van Benthem (1989). In this strategy, the scope relations for multi-quantifier sentences like (1) can be derived by turning a sequence of quantifiers into a polyadic quantifier, using what we will call left and right **pile′up**-operations (also known as iterations):

$$\mathbf{pile'up}^l, \mathbf{pile'up}^r : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

defined, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(Y)$, by lambda terms as:

$$\mathbf{pile'up}^l(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.N(\lambda y_{:Y}c(x,y))$$

and

$$\mathbf{pile'up}^r(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.M(\lambda x_{:X}c(x,y)).$$

The polyadic quantifier thus formed is only then applied to the predicate. Again, strategy B can be straightforwardly extended to account for sentences involving three or more QPs (by allowing permutations of QPs).

1.3 *Strategy C*

Strategy C, the most recent, involves continuations and was first proposed in the works of Barker (2002) and De Groote (2001), and then further developed and modified in the works of Barker and Shan (2014), Kiselyov and Shan (2014) and Bekki and Asai (2009). Continuation-based strategies can be divided into two groups: those that locate the source of scope-ambiguity in the rules of semantic composition, and those that attribute it to the lexical entries for the quantifier words. In this paper, we consider only the first group: operation-based approaches (as in Barker 2002). In this strategy, a predicate gets lifted ('continuized'), i.e. a predicate of type $X \to \mathbf{t}$ will be lifted

to an expression of type $\mathscr{C}(X \to \mathbf{t}) = ((X \to \mathbf{t}) \to \mathbf{t}) \to \mathbf{t}$; etc. Scope relations for multi-quantifier sentences like (1) are derived by first combining the lifted predicate with the object QP, and then merging the result thus obtained with the subject QP, using the so-called **CPS** transforms:[1]

$$\mathbf{CPS}^{l}(ev), \mathbf{CPS}^{r}(ev) : \mathscr{C}(X) \times \mathscr{C}(X \to Y) \longrightarrow \mathscr{C}(Y)$$

given, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(X \to Y)$, by

$$\mathbf{CPS}^{l}(ev)(M,N) = \lambda c_{:\mathscr{P}(Y)}.M(\lambda x_{:X}.N(\lambda g_{:X\to Y}.c(g\,x)))$$

and

$$\mathbf{CPS}^{r}(ev)(M,N) = \lambda c_{:\mathscr{P}(Y)}.N(\lambda g_{:X\to Y}.M(\lambda x_{:X}.c(g\,x))).$$

Strategy C can be seen as a compelling alternative to the traditional movement strategy (Strategy A), and the polyadic approach (Strategy B), for a uniform non-movement (in situ) analysis of quantifiers. However, it cannot be straightforwardly extended to account for sentences involving three QPs.

As will be explained below, a generalized quantifier on a set X is an element of $\mathscr{C}(X)$, the value of the continuation monad $\mathscr{C}$ on X. In this paper, we will show that the continuation monad can be taken as a common basis for the three scope-assignment strategies just described. This will allow us to present these strategies against a uniform background and explicitly spell out the semantic operations used in each strategy. Two of the three strategies, B and C, use strength: the additional structure that exists on the continuation monad. This shows that the **pile′up** operations employed in the now widely accepted and well-understood strategy B, and the **CPS** operations employed in the less popular strategy C, considered more difficult, are in fact very close in spirit. We thus hope that our results will help to make the continuation-based strategy more popular.

The remaining part of this paper is organized as follows. We first introduce the notion of monad, starting with some informal remarks,

[1] In standard categorial grammar approaches, the scope relations for multi-quantifier sentences like (1) can be obtained via higher-order verb types (Hendriks 1993). For a comparison of standard type-shifting approaches and continuation-based strategies, see e.g. Barker and Shan (2014).

followed by a definition, and examples relevant to linguistics. We then introduce the continuation monad itself. Next, we define the notion of bi-strong monads, and show how the relevant algebraic operations (**pile′ups**, T-transforms and, in particular, **CPS**-transforms) are to be derived from strengths. Then we precisely state three specific implementations of the scope-assignment strategies: the traditional movement strategy (as implemented in May 1978), the polyadic approach (as in May 1985), and the continuation-based approach (as proposed in Barker 2002). With this background, we can explain how the three strategies are related, and what can be expected of them with regard to handling scopal ambiguities in simple sentences. An appendix contains the relevant proofs.

2 MONADS AND STRENGTHS

It is widely accepted that the notion of monad (also called 'triple') was first introduced in 1958 by Godement under the name of 'standard construction' (Godement 1958). It was soon realized that any pair of adjoint functors gives rise to a monad. Later, in 1965, it was discovered independently by Kleisli (1965) and by Eilenberg *et al.* (1965) that any monad is induced by an adjunction. For many years, the Eilenberg-Moore algebras were the most popular with mathematicians. It was Moggi who in 1989 used monads to build semantics for programming languages (Moggi 1991). Soon afterwards, Wadler employed monads to model side-effects in functional programming (Wadler 1990). In these new applications, the Kleisli algebras gained more importance. In both cases, monads are used to extend the notion of a function. After such a prelude, it did not take long for these ideas to be adopted in linguistics. The Kleisli construction can be thought of as an extension of a function/transformation f:

$$f : X \longrightarrow Y$$

between two sets, X and Y, which somehow reflects the fact that such a transformation is not considered as a mere mapping of arguments to values, but that there is also a particular computational process related to this association. This process, when applied to an element x of the domain set X, can indeed result in returning a value $f(x)$ of the codomain set Y. But it can also provide a more involved result

belonging to a set $T(Y)$, related in some way, but possibly bigger (or even much bigger) than Y itself. Thus we can think about such an extended[2] transformation between X and Y as a mere function:

$$f : X \longrightarrow T(Y)$$

from the set X to the extension $T(Y)$ of the set Y. This seems to be an intuitively clear and simple idea, but then we need to see whether we can still work with such 'extended functions' as we can with ordinary functions. The answer varies depending on how demanding we are. The minimum (which should be adequate for most purposes) that we should expect from these 'extended functions' is that they compose, that this composition should be associative, and that there should be 'extended functions' that act as if they were 'doing nothing' (i.e. like identities). Once we agree that these expectations are natural, we can try to specify the reasonable condition to guarantee this for construction T, i.e. that T should be a monad. Then the unit (return) $\eta_X : X \to T(X)$ acts as an identity on X, and the multiplication $\mu_Z : T^2(Z) \to T(Z)$, together with the fact that T is a functor, can be used to define the composition of the two 'extended functions', $f : X \longrightarrow T(Y)$ and $g : Y \to T(Z)$, as follows:

$$X \xrightarrow{\quad f \quad} T(Y) \xrightarrow{\quad T(g) \quad} T^2(Z) \xrightarrow{\quad \mu_Z \quad} T(Z)$$

The conditions imposed on T ensure that η_X is in fact the identity on X and that the composition thus defined is associative.

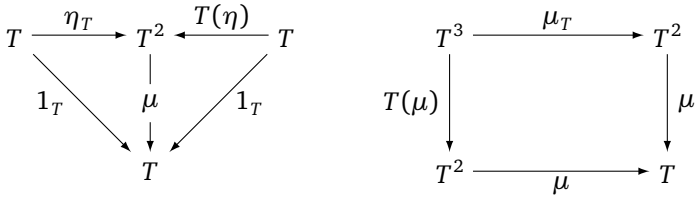
It is fair to say that the above is a short mathematician's introduction to monads, as used by computer scientists. In fact, the very notion of monad is usually formulated differently by computer scientists. This is, we think, due to the fact that the actual computation of the set $T(Y)$ even for the finite set Y can easily be infinite. This can be taken as a form of potential infinity. But then the second iteration $T(T(Y)) = T^2(Y)$, needed to express the multiplication μ, is even more challenging (since it requires applying the functor T to an already potentially complicated set $T(Y)$). The computer scientists' solution to this problem is to consider the combined operation **bind** : $T(X) \to (X \to T(Y)) \to T(Y)$ (instead of the multiplication μ),

[2] In some degenerate cases, the set $T(Y)$ might even be smaller than Y.

which never uses the second iteration of T. No matter how we define the monad T, the Kleisli category is the same. The potential gain from this extension for linguistics is that some processes which could not be described as compositional processes, applying (ordinary) functions to arguments, can become compositional after all, if we relax our notion of function to the 'extended function' we described above. The so-called continuation semantics for natural language, or 'strategy C' in this paper, is an illustration of such a phenomenon.

2.1 *Monads – definition and examples*

For unexplained notions related to category theory, we refer the reader to standard textbooks on the subject. We shall be exclusively working in the Cartesian closed category of sets *Set*. The category *Set* of sets has sets as objects. A morphism in *Set* from an object (set) X to an object (set) Y is a function[3] $f : X \longrightarrow Y$ from X to Y. A *monad* on *Set* is a triple (T, η, μ) where $T : Set \longrightarrow Set$ is an endofunctor (the underlying functor of the monad), $\eta : 1_{Set} \longrightarrow T$ and $\mu : T^2 \longrightarrow T$ are natural transformations (the first from the identity functor on *Set* to T, the second from the composition of T with itself to T) making the following diagrams commute:



These diagrams express the essence of the algebraic calculations. We shall explain their meaning while describing the list monad below. The symbols η and μ are often referred to as the *unit* and *multiplication* of the monad, respectively, while T is its functor part. When η and μ are clear from the context, it is customary to refer to the whole monad (T, η, μ) as T.

Before we focus on the continuation monad, the main notion of computation considered in this paper, we shall illustrate the concept

[3] We always consider functions with specified domains and codomains. For a pedantic reader, a function can be thought of as a triple $\langle X, Y, f \rangle$, such that X and Y are sets and f is a subset of the product $X \times Y$, which is total and univalued.

with some examples also relevant to linguistics (see e.g. Charlow 2014, and Shan 2002).

Examples of monads

1. The *Identity monad* is the simplest possible monad, but it is not very interesting. In this case, the functor T and the natural transformations η and μ are identities. For this monad, the notion of a T-computation in X is just an element of X, as the function $f : X \longrightarrow T(Y)$ is just $f : X \longrightarrow Y$.
2. The *Maybe monad* is the simplest non-trivial monad. The functor T sends every set X to the set $T(X) = X + \{\bot\}$ (the disjoint sum of X and singleton $\{\bot\}$), and every function $f : X \longrightarrow Y$ to a function $T(f) : T(X) \longrightarrow T(Y)$, such that, for $x \in T(X)$:

$$T(f)(x) = \begin{cases} x & \text{if } x \in X \\ \bot & \text{if } x = \bot \end{cases}$$

   So T adds to X an additional element $\bot$, called *bottom* or *nothing*. The component at X of the natural transformation η is a function $\eta_X : X \longrightarrow X + \{\bot\}$, such that $\eta_X(x) = x$, i.e. it sends x to the same x but in the set $X + \{\bot\}$. The component at X of the natural transformation μ is a function $\mu_X : X + \{\bot, \bot'\} \longrightarrow X + \{\bot\}$, such that, for $x \in X + \{\bot, \bot'\}$:

$$\mu_X(x) = \begin{cases} x & \text{if } x \in X \\ \bot & \text{if } x = \bot \text{ or } x = \bot' \end{cases}$$

   i.e. it sends x in X to the same x, and two bottoms $\bot$ and $\bot'$ in $T^2(X)$ to the only bottom $\bot$ in $T(X)$.
   For this monad, the notion of a T-computation in X consists of elements of X, and an additional computation $\bot$, which says that we do not get a value in X. The function $X \longrightarrow T(Y)$ carries the same information as a partial function $X \hookrightarrow Y$. So this monad allows partial computations to be treated as total computations.
3. The *Exception monad* is less trivial than the maybe monad. We are given a fixed set of exceptions E and, for a set X, the monad functor is $T(X) = X + E$, i.e. the disjoint union of X and E. If E is empty, it is the identity monad; if E is a singleton, then it is a maybe monad; otherwise is it like the maybe monad but with many options for nothingness.

4. The *List monad* or *monoid monad* is even more interesting than the previous monad, and we shall work it out in detail. It is not needed for the applications in the paper but it provides some insights before we move on to the continuation monad. To any set X, the list monad functor associates the set $T(X)$ of (finite) words over X (treated as an alphabet). This includes the empty word ε. To a function $f : X \longrightarrow Y$, the functor T associates the function $T(f) : T(X) \longrightarrow T(Y)$, sending the word $x_1x_2\ldots x_n$ over X to the word $f(x_1)f(x_2)\ldots f(x_n)$ over Y. The component at X of the natural transformation η is a function $\eta_X : X \longrightarrow T(X)$, such that $\eta_X(x) = x$, i.e. it sends the letter x to the one-letter word x in $T(X)$. The component at X of the natural transformation μ is a function $\mu_X : T^2(X) \longrightarrow T(X)$. Note that $T^2(X) = T(T(X))$ is the set of words whose letters are words over the alphabet X. Thus it can be thought of as a list of lists. Applying μ_X to such a list of lists flattens it to a single list. A three-letter word $t = (x_1x_2)(x_3x_4x_5)\varepsilon$ is a typical element of $T^2(X)$. The result of flattening T is the list $\mu_X(T) = x_1x_2x_3x_4x_5$ in $T(X)$. We can think of a word w as a term/word/computation $u = y_1y_2y_3$, in which we intend to substitute the term $v_1 = x_1x_2$ for variable y_1, the term $v_2 = x_3x_4x_5$ for variable y_2, and the term $v_3 = \varepsilon$ for variable y_3, i.e. $u[y_1\backslash v_1, y_2\backslash v_2, y_3\backslash v_3]$. Now the multiplication μ can be thought of as an actual substitution. With this interpretation, one can understand the intuitions behind the monad diagrams. In the left triangle, an element of $T(X)$, say $x_1x_2x_3$, is mapped through $\eta_{T(X)}$ to a single-letter word $(x_1x_2x_3)$ and μ_X flattens it back to $x_1x_2x_3$, as required for the triangle to commute. In other words, the substitution $y[y\backslash v]$ results in v. In the right triangle, the map $T(\eta_x)$ sends, say $x_1x_2x_3$, to the letter word $(x_1)(x_2)(x_2)$, with each letter being a single-letter word. Thus, again, flattening such a list gives $x_1x_2x_3$ back, as required. In other words, the substitution $y_1y_2y_3[y_1\backslash x_1, y_2\backslash x_2, y_3\backslash x_3]$ results in $x_1x_2x_3$. The commutation of the square diagram, in this case, expresses the fact that, if we have a list of lists of lists and we flatten it in two different ways, starting either with the upper two levels of lists, or with the lower two levels, and then we flatten the results again to get the ordinary lists over X in $T(X)$, these lists coincide. On a more conceptual level, this square expresses the fact that evaluation

commutes with substitution. In this sense, these diagrams capture the essence of all algebraic calculations.

For this monad, the notion of a T-computation in X consists of words over X to be evaluated/multiplied in a monoid when elements of X will be (interpreted) in a monoid. The function $f : X \longrightarrow T(Y)$ is just a function $f : X \longrightarrow T(Y)$ sending elements of X to words over Y. So this monad allows a list of values for a given input.

2.2 *Notation*

Before we explain the notion of computation that accompanies the continuation monad, we restate the monad in a more functional way. To do this, we need to introduce some form of notation. As *Set* is a Cartesian closed category, it is customary to denote functions between sets using λ notation. One can think of it as if we were to work in the internal language of *Set*, i.e. λ theory, where all functions have their names represented. For sets X and Y, we shall use $X \times Y$ to denote the binary product of X and Y, and $X \to Y$ to denote the set of functions from X to Y. As is customary, we associate $\to$ to the right, i.e. $X \to Y \to Z$ means $X \to (Y \to Z)$, and this set is naturally bijective with $(X \times Y) \to Z$. If we have a function:

$$f : X \times Y \longrightarrow Z,$$

then by:

$$\lambda y_{:Y}.f : X \longrightarrow Y \to Z$$

we denote its exponential adjunction, i.e. the function from X to the set of functions $Y \to Z$, such that, for an element $x \in X$, $\lambda y_{:Y}.f(x)$ is a function from Y to Z such that, for an element $y \in Y$, $(\lambda y_{:Y}.f)(x)(y)$ is by definition equal to $f(x, y)$. Note that, in the expression $(\lambda y_{:Y}.f)(x)(y)$, the first occurrence of y is an occurrence of a variable (as it is part of the name of a function), whereas the second occurrence of y in this expression denotes an element of the set Y.

Then π_i will denote the projection on i-component from the product. Any function $\sigma : \{1, \ldots m\} \longrightarrow \{1, \ldots, n\}$ induces a generalized projection denoted:

$$\pi_\sigma = \langle \pi_{\sigma(1)}, \ldots, \pi_{\sigma(m)} \rangle : X_1 \times \ldots \times X_n \longrightarrow X_{\sigma(1)} \times \ldots \times X_{\sigma(m)}.$$

We will use this notation mainly when σ is bijective, i.e. when π_σ is just a permutation of the component for the product.

We have a fixed set of truth values $\mathbf{t} = \{\mathbf{true}, \mathbf{false}\}$. We shall use the usual (possibly infinitary) operations on this set. For a set X, we write $\mathscr{P}(X) = X \to \mathbf{t}$, i.e. the (functional) powerset of X.

2.3 *Continuation monad*

The *Continuation monad*, the most important for us, is denoted $\mathscr{C}$. Its functor part (also denoted $\mathscr{C}$), at the level of objects, is just a twice-iterated power-set construction, i.e. for set X, $\mathscr{C}(X) = \mathscr{P}^2(X)$. At the level of morphisms, it is an inverse image of an inverse image, i.e., function $f : X \longrightarrow Y$ induces an inverse image function between powersets:

$$\mathscr{P}(f) = f^{-1} : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X)$$
$$h \mapsto h \circ f,$$

in λ-notation,

$$\mathscr{P}(f) = \lambda h_{:\mathscr{P}(Y)}.\lambda x_{:X}.h(f\ x).$$

Taking again an inverse image function, we have

$$\mathscr{C}(f) = \mathscr{P}(f^{-1}) : \mathscr{C}(X) \longrightarrow \mathscr{C}(Y)$$

$$Q \mapsto Q \circ f^{-1},$$

in λ-notation:

$$\mathscr{C}(f)(Q) = \lambda h_{:\mathscr{P}(Y)}.Q(\lambda x_{:X}.h(f\ x)),$$

for $Q \in \mathscr{C}(X)$.

The unit $\eta_X : X \longrightarrow \mathscr{C}(X)$ is given by:

$$\eta_X(x) = \lambda h_{:\mathscr{P}(X)}.h(x), \qquad \text{for } x \in X.$$

The multiplication $\mu_X : \mathscr{C}^2(X) \longrightarrow \mathscr{C}(X)$ can be explained in terms of η:

$$\mu_X = \mathscr{P}(\eta_{\mathscr{P}(X)}) : \mathscr{P}^4(X) \longrightarrow \mathscr{P}^2(X).$$

In other words, $\mu_X(\mathscr{F}) : \mathscr{P}(X) \longrightarrow \mathbf{t}$ is a function such that:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\eta_{\mathscr{P}(X)}(h))$$

for

$$\mathscr{F} : \mathscr{P}^3(X) \longrightarrow \mathbf{t} \quad \text{and} \quad h : X \longrightarrow \mathbf{t}.$$

In λ-notation, we write:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\lambda D_{:\mathscr{C}(X)}.D(h)).$$

Now we can look at the notion of computation related the continuation monad. Consider the function:

$$f : X \longrightarrow \mathscr{C}(Y).$$

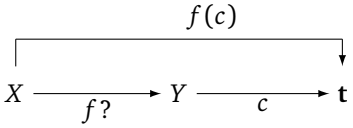
By exponential adjunction (uncurrying), it corresponds to a function:

$$f' : \mathscr{P}(Y) \times X \longrightarrow \mathbf{t}$$

and again, by exponential adjunction (currying), it corresponds to a function:

$$f'' : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X).$$

Thus a $\mathscr{C}$-computation from X to Y is a function that sends functions from $\mathscr{P}(Y) = Y \to \mathbf{t}$ to functions in $\mathscr{P}(X)$. So instead of having for a given element $x \in X$ a direct answer to the question what is the value of f at x, i.e. the element $f(x)$ in Y, we are given for every continuation function $c : Y \longrightarrow \mathbf{t}$ a value in the answer type $\mathbf{t}$ that could be thought of as $c(f(x))$ (if there were an element in Y that could be reasonably called $f(x)$). We can draw a picture illustrating the situation:



Instead of 'procedure' $f?$ computing y's from x's (that we do not have), we provide a continuation $f(c)$ for any continuation (of the computation) c. If $f?$ were indeed a genuine function $f? : X \longrightarrow Y$, then $f(c)$ would be the composition $c \circ f?$.

2.4 *Bi-strong monads*

As noted in Moggi (1991), a monad has to be strong, in order to have a well-behaved notion of computation.[4] Fortunately, all monads on

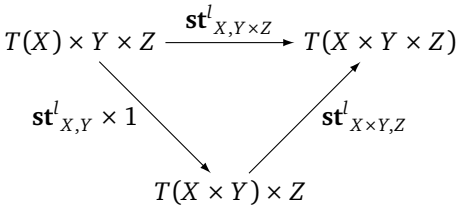
[4] As the notion of strength is new in this context, we shall briefly recall its history. There are three manifestations of strength on a functor. Historically, the

Set are strong. More precisely, all monads on *Set* can be canonically equipped with two strengths, left and right, and these strengths are compatible in a precise technical sense. This additional structure on the continuation monad will be essential when we analyze the meaning of multi-quantifier sentences.

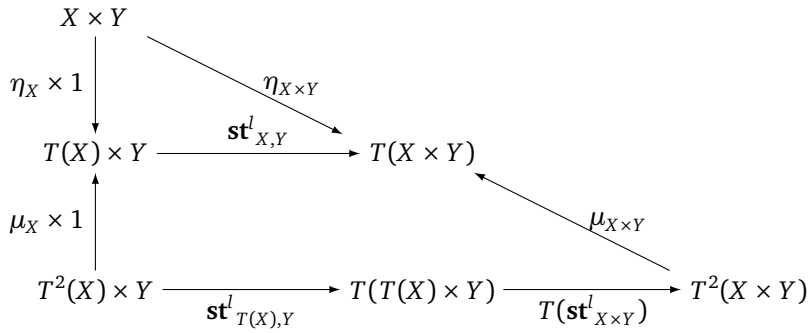
Let (T, η, μ) be a monad on *Set*. The *left strength* is a natural transformation with components:

$$\mathbf{st}^l{}_{X,Y} : T(X) \times Y \longrightarrow T(X \times Y)$$

for sets X and Y, making the following two diagrams commute:



and

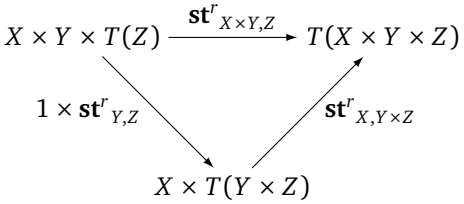


The *right strength* is a natural transformation with components:

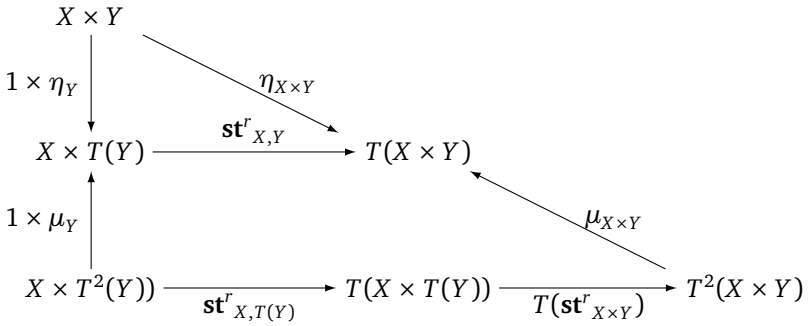
$$\mathbf{st}^r{}_{X,Y} : X \times T(Y) \longrightarrow T(X \times Y)$$

for sets X and Y, making the following two diagrams commute:

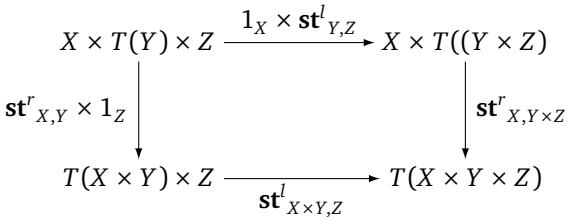
first one was the notion of enrichment of a functor (c.f. Eilenberg and Kelly 1966). Tensorial strength (i.e., natural transformation of type $X \otimes T(Y) \longrightarrow T(X \otimes Y)$ used in this paper) was introduced in Kock (1970) and further developed in Kock (1972). Cotensorial strength (i.e., natural transformation of type $T(X \to Y) \longrightarrow X \to T(Y)$) introduced in Kock (1971) has also proved useful in some contexts. In symmetric monoidal closed categories, these concepts are equivalent (c.f. Kock (1971)).



and



The monad (T, η, μ) on *Set* together with two natural transformations $\mathbf{st}^l$ and $\mathbf{st}^r$ of right and left strength is a *bi-strong monad* if, for any sets X, Y, Z, the following square commutes:



As we already mentioned, each monad (T, η, μ) on *Set* is bi-strong. We shall define the right and left strength. Fix sets X and Y. For $x \in X$ and $y \in Y$, we have functions:

$$l_y : X \longrightarrow X \times Y, \quad \text{and} \quad r_x : Y \longrightarrow X \times Y,$$

such that:

$$l_y(x) = \langle x, y \rangle, \quad \text{and} \quad r_x(y) = \langle x, y \rangle.$$

The left and right strength:

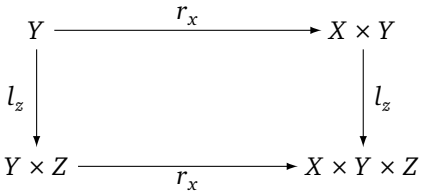
$$\mathbf{st}^l{}_{X,Y} : T(X) \times Y \longrightarrow T(X \times Y) \quad \text{and} \quad \mathbf{st}^r{}_{X,Y} : X \times T(Y) \longrightarrow T(X \times Y)$$

are given respectively for $x \in X$, $s \in T(X)$, $y \in Y$ and $t \in T(Y)$ by:

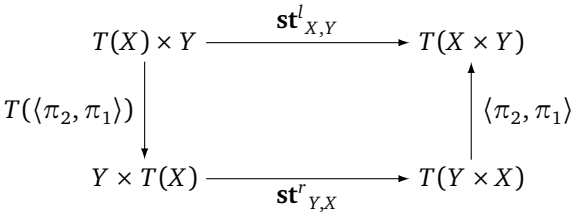
$$\mathbf{st}^l{}_{X,Y}(s, y) = T(l_y)(s) \quad \text{and} \quad \mathbf{st}^r{}_{X,Y}(x, t) = T(r_x)(t).$$

When it does not lead to confusion, we drop the indices $_{X,Y}$.

It is not difficult to verify that the above defines left ($\mathbf{st}^l$) and right ($\mathbf{st}^r$) strength on the monad T. Since for any $x \in X$ and $z \in Z$, the following square commutes:



they are compatible and make the monad T bi-strong. Note that these strengths are related by the following diagram:



Examples of strength on monads in Set

1. Maybe monad. The left strength $\mathbf{st}^l{}_{X,Y} : (X + \{\bot\}) \times Y \longrightarrow (X \times Y) + \{\bot\}$ is given by:

$$\mathbf{st}^l(x, y) = \begin{cases} \bot & \text{if } x = \bot \\ \langle x, y \rangle & \text{otherwise.} \end{cases}$$

   Right strength is similar.

2. List monad. The left strength $\mathbf{st}^l : T(X) \times Y \longrightarrow T(X \times Y)$ is given by:

$$\mathbf{st}^l(\vec{x}, y) = \begin{cases} \varepsilon & \text{if } \vec{x} = \varepsilon \\ \langle x_1, y \rangle, \ldots, \langle x_n, y \rangle & \text{if } \vec{x} = x_1, \ldots, x_n. \end{cases}$$

   Right strength is similar.

3. Continuation monad. We shall describe the strength morphisms by lambda terms. The left strength is:

$$\mathbf{st}^l = \lambda N_{:\mathscr{C}(X)}.\lambda y_{:Y}.\lambda c_{:\mathscr{P}(X\times Y)}.\\ N(\lambda x_{:X}.c(x,y)) : \mathscr{C}(X)\times Y \longrightarrow \mathscr{C}(X\times Y)$$

and the right strength is:

$$\mathbf{st}^r = \lambda x_{:X}.\lambda M_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\\ M(\lambda y_{:Y}.c(x,y)) : X\times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X\times Y).$$

2.5 *Combining computations in arbitrary monad T on Set*

Using both strengths, we can define two **pile′up** natural transformations, left and right. For any sets X and Y, the *left pile up* $\mathbf{pile'up}^l{}_{X,Y}$ is defined from the diagram:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{pile'up}^l{}_{X,Y}} & T(X\times Y) \\ \Big\downarrow{\scriptstyle \mathbf{st}^l{}_{X,T(Y)}} & & \Big\uparrow{\scriptstyle \mu_{X\times Y}} \\ T(X\times T(Y)) & \xrightarrow[T(\mathbf{st}^r{}_{X,Y})]{} & T^2(X\times Y) \end{array}$$

In the above diagram, the function $\mathbf{pile'up}^l{}_{X,Y}$ is defined as a composition of three operations: the first takes the T-computation on X 'outside' to be a computation on $X\times T(Y)$, the second takes the T-computation on Y 'outside' to be a T-computation on $X\times Y$. In this way, we have T-computations coming from X on T-computations coming from Y on $X\times Y$. Now the last morphism $\mu_{X\times Y}$ flattens these two levels to one, i.e. the T-computation on T-computations to T-computations.

The *right pile up* $\mathbf{pile'up}^r{}_{X,Y}$ is defined from the diagram:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{pile'up}^r{}_{X,Y}} & T(X\times Y) \\ \Big\downarrow{\scriptstyle \mathbf{st}^r{}_{T(X),Y}} & & \Big\uparrow{\scriptstyle \mu_{X\times Y}} \\ T(T(X)\times Y) & \xrightarrow[T(\mathbf{st}^l{}_{X,Y})]{} & T^2(X\times Y) \end{array}$$

This operation takes the T-computations in reverse order and so they pile up in the opposite way.

If these **pile′up** operations agree for all sets X and Y, the monad is called *commutative*. In our list of monads, both the identity and maybe monads are commutative. The exception, list and continuation monads are not commutative. Most monads, including the continuation monad $\mathscr{C}$, are not commutative. It should be noted that even if the monad T is not commutative, both lift morphisms agree for pairs in which at least one component comes from the actual value (not an arbitrary T-computation). In other words, the functions:

$$T(X_1) \times T(X_2) \underset{\mathbf{pile'up}^r{}_{X_1,X_2}}{\overset{\mathbf{pile'up}^l{}_{X_1,X_2}}{\rightrightarrows}} T(X_1 \times X_2)$$

are equalized by both the following morphisms:

$$X_1 \times T(X_2) \xrightarrow{\eta_{X_1} \times 1} T(X_1) \times T(X_2)$$

and

$$T(X_1) \times X_2 \xrightarrow{1 \times \eta_{X_2}} T(X_1) \times T(X_2)$$

Both $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ are associative. All this is shown in the Appendix.

Examples of **pile′up***-operations*

1. Maybe monad. The left and right **pile′up**s coincide in this case, as in any commutative monad. We have

   $$\mathbf{pile'up}^l{}_{X,Y} = \mathbf{pile'up}^r{}_{X,Y} : (X + \{\bot\}) \times (Y + \{\bot'\}) \longrightarrow (X \times Y) + \{\bot\}$$

   given by:

   $$\mathbf{pile'up}^l(x,y) = \mathbf{pile'up}^r(x,y) = \begin{cases} \bot & \text{if } \{x,y\} \cap \{\bot,\bot'\} \neq \emptyset \\ \langle x,y\rangle & \text{otherwise.} \end{cases}$$

2. List monad. The left pile'up $\mathbf{pile'up}^l : T(X) \times T(Y) \longrightarrow T(X \times Y)$ is given by:

   $$\mathbf{pile'up}^l(\langle x_1 \ldots x_n\rangle, \langle y_1 \ldots y_m\rangle) =$$
   $$= \langle x_1,y_1\rangle\langle x_1,y_2\rangle \ldots \langle x_1,y_m\rangle\langle x_2,y_1\rangle \ldots \langle x_n,y_{m-1}\rangle\langle x_n,y_m\rangle$$

   and the right pile'up $\mathbf{pile'up}^r : T(X) \times T(Y) \longrightarrow T(X \times Y)$ is given by:

   $$\mathbf{pile'up}^r(\langle x_1 \ldots x_n\rangle, \langle y_1 \ldots y_m\rangle) =$$
   $$= \langle x_1,y_1\rangle\langle x_2,y_1\rangle \ldots \langle x_n,y_1\rangle\langle x_1,y_2\rangle \ldots \langle x_{n-1},y_m\rangle\langle x_n,y_m\rangle.$$

3. Continuation monad. Both **pile′up** operations:

$$\mathbf{pile'up}^{l}, \mathbf{pile'up}^{r} : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

can be defined, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(Y)$, by lambda terms, such as:

$$\mathbf{pile'up}^{l}(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.N(\lambda y_{:Y}\, c(x,y))$$

and

$$\mathbf{pile'up}^{r}(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.M(\lambda x_{:X}\, c(x,y)).$$

The calculations for these operations are in the Appendix.
Thus in the case of the continuation monad, 'piling up' computations one on top of the other is nothing but putting (interpretations of) quantifiers (= computations in the continuation monad) in order, either the first before the second or the second before the first.

2.6 *T-transforms on arbitrary monad T on Set*

There are two (binary) T-transformations, right and left. For a function $f : X \times Y \longrightarrow T(Z)$, the left T-transform is defined as the composition:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{TR}^{l,T}{}_{X,Y}(f)} & T(Z) \\ \Big\downarrow{\scriptstyle \mathbf{pile'up}^{l}} & & \Big\uparrow{\scriptstyle \mu_Z} \\ T(X\times Y) & \xrightarrow[T(f)]{} & T^2(Z) \end{array}$$

and the right T-transform is defined as the composition:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{TR}^{r,T}{}_{X,Y}(f)} & T(Z) \\ \Big\downarrow{\scriptstyle \mathbf{pile'up}^{r}} & & \Big\uparrow{\scriptstyle \mu_Z} \\ T(X\times Y) & \xrightarrow[T(f)]{} & T^2(Z) \end{array}$$

The most popular T-transforms are for the evaluation morphism:

$$ev : X \times (X \to Y) \longrightarrow Y$$

but there are also other morphisms with useful transforms.

Examples of T-transforms and in particular **CPS***-transforms*

1. The evaluation map $ev : X \times (X \to Y) \longrightarrow Y$ gives rise to application transforms:

$$\mathbf{TR}^{l,T}(ev), \mathbf{TR}^{r,T}(ev) : T(X) \times T(X \to Y) \longrightarrow T(Y).$$

When T is the continuation monad $\mathscr{C}$, they are the usual **CPS**-transforms $\mathbf{CPS}^{l}(ev), \mathbf{CPS}^{r}(ev) : \mathscr{C}(X) \times \mathscr{C}(X \to Y) \longrightarrow \mathscr{C}(Y)$ given by:

$$\mathbf{CPS}^{l}(ev)(M,N) = \lambda h_{:\mathscr{P}(Y)}.M(\lambda x_{:X}.N(\lambda g_{:X\to Y}.h(g\,x)))$$

for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(X \to Y)$.The right transform is similar.

2. Various evaluation maps are typically defined as maps from a product. Thus they give rise to various T-transforms. We list some of them below, mainly to introduce notation that will be used later. The definitions are given by lambda terms.
(a) Left evaluation:

$$\mathbf{eps}^{l}{}_{X} = \lambda h_{:\mathscr{P}(X)}.\lambda x_{:X}.h(x) : \mathscr{P}(X) \times X \longrightarrow \mathbf{t};$$

(b) Right evaluation:

$$\mathbf{eps}^{r}{}_{X} = \lambda x_{:X}.\lambda h_{:\mathscr{P}(X)}.h(x) : X \times \mathscr{P}(X) \longrightarrow \mathbf{t};$$

(c) Left partial evaluation:

$$\mathbf{eps}^{l,X}_{Y}(\mathbf{eps}^{l}{}_{Y}) = \lambda c_{:\mathscr{P}(X\times Y)}.\lambda y_{:Y}.\lambda x_{:X}.c(x,y) : \mathscr{P}(X \times Y) \times Y \longrightarrow \mathscr{P}(X);$$

(d) Right partial evaluation:

$$\mathbf{eps}^{r,X}_{Y}(\mathbf{eps}^{r}{}_{Y}) = \lambda y_{:Y}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.c(x,y) : Y \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X).$$

3. What we call Mostowski maps are maps similar to **eps**es that are the algebraic counterpart of the interpretation of generalized quantifiers by Mostowski. Again, we give a definition for total and partial case.
(a) Left Mostowski:

$$\mathbf{mos}^{l}{}_{X} = \lambda Q_{:\mathscr{C}(X)}.\lambda c_{:\mathscr{P}(X)}.Q(c) : \mathscr{C}(X) \times \mathscr{P}(X) \longrightarrow \mathbf{t};$$

(b) Right Mostowski:

$$\mathbf{mos}^{r}{}_{X} = \lambda c_{:\mathscr{P}(X)}.\lambda Q_{:\mathscr{C}(X)}.Q(c) : \mathscr{P}(X) \times \mathscr{C}(X) \longrightarrow \mathbf{t};$$

(c) Left partial Mostowski:

$$\mathbf{mos}^{l}{}_{Y} = \lambda Q_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y)) :$$
$$\mathscr{C}(Y) \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X);$$

(d) Right partial Mostowski:

$$\mathbf{mos}^{r}{}_{Y} = \lambda c_{:\mathscr{P}(X\times Y)}.\lambda Q_{:\mathscr{C}(Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y)) :$$
$$\mathscr{P}(X \times Y) \times \mathscr{C}(Y) \longrightarrow \mathscr{P}(X).$$

3 SCOPE-ASSIGNMENT STRATEGIES

Using the notions connected to the continuation monad introduced above, we shall now precisely state and compare three strategies (A, B, and C) for determining the meaning of multi-quantifier sentences.

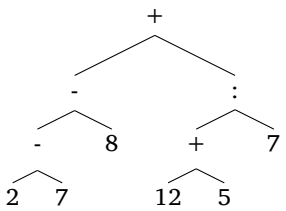
3.1 *General remarks*

In each strategy, the starting point is the surface structure tree of a sentence. This tree is rewritten so as to obtain formal structure trees that correspond to all the available meanings of the sentence. Finally, we relabel those trees to obtain computation trees[5] that provide the semantics for the sentence in each of its readings.

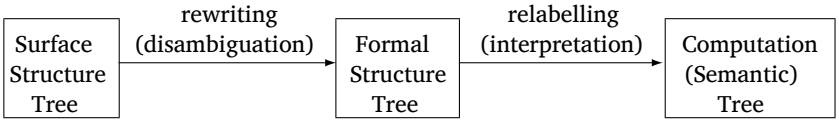
[5] We think of computation trees by analogy with mathematical expressions, e.g.

$$((2-7)-8)+((12+5):7)$$

that can be represented as:

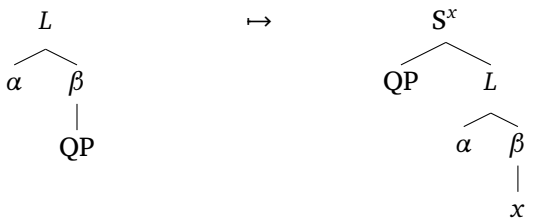


i.e. a labeled binary tree where the leaves are labeled with values and the internal nodes are labeled with operations that will be applied in the computation to the values obtained from the computations of the left and right subtrees.

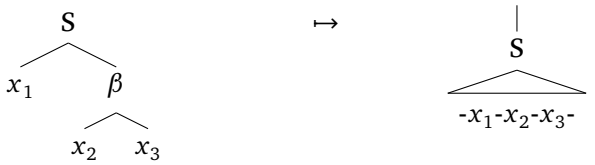


Rewriting. Scope-assignment strategies can be divided into two families: movement analyses (rewriting rules include QR, Predicate Collapsing, and possibly Rotation) and in situ analyses (no rewriting rules). Below we define three rewrite rules on trees: QR Rule, Predicate Collapsing, and Rotation.

- QR (Quantifier Raising) Rule
  - applies when we have a chosen QP in a leaf of a tree;
  - adjoins QP to S;
  - indexes S with the variable bound by the raised QP.



  (L-label, α-subtree, β-subtree.)
- Predicate Collapsing
  - applies when all the leaves under the node labeled S are labeled with variables (not QPs);
  - collapses the whole subtree with the root S to a single leaf labeled with the variables x_1, x_2, x_3 from the leaves under the S-node.



- Rotation
  - applies to a tree with two nodes labeled with S'es superscripted with some variables: the mother labeled $S^{\vec{x}}$ and its right daughter labeled $S^{\vec{y}}$;

- it rotates left the subtree with the root labeled $S^{\vec{x}}$;
- the root of this subtree is labeled $S^{\vec{x}\vec{y}}$ and the (new) left daughter is labeled *Polyadic*.

(α-subtree, β-subtree, γ-subtree.)

Relabelling. In each scope-assignment strategy, the leaves in the computation tree have the same labels: QPs are interpreted as $\mathscr{C}$-computations, and predicates are interpreted as usual or lifted. The main difference between the three approaches consists in the shape of the formal structure trees and the operations (**eps**es, **mos**es, **pile′up**s, **CPS**es) used as labels for the inner nodes of the computation trees.

3.2 *Strategy A*

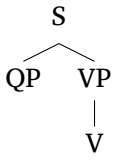
In the traditional movement strategy (as implemented in May 1978)

- Surface structure trees are rewritten (disambiguated) as formal structure trees (Logical Forms) via:
  - QR Rule;
  - Predicate Collapsing.
- Formal structure trees (LFs) are relabelled as computation trees as follows:
  - S^x (root of a subtree representing a formula) is interpreted as a suitably typed **mos**-operation (the only operation allowed);
  - S (leaf of a tree) is interpreted as a predicate;
  - QP (leaf of a tree) is interpreted as a generalized quantifier $\|Q\|$ quantifying over a set X (i.e. as a $\mathscr{C}$-computation on X).

We will illustrate each strategy with examples involving one, two and three QPs.

Sentence with one QP, e.g. *Every kid (most kids) entered.*

(A1) Surface structure tree:



(A1) Formal structure tree (LF) and the corresponding computation tree:

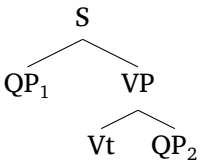
The computation tree in (A1) gives rise to the following general map:

$$\mathbf{strat}^1_A : \mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{\mathbf{mos}^l{}_X} \mathbf{t}$$

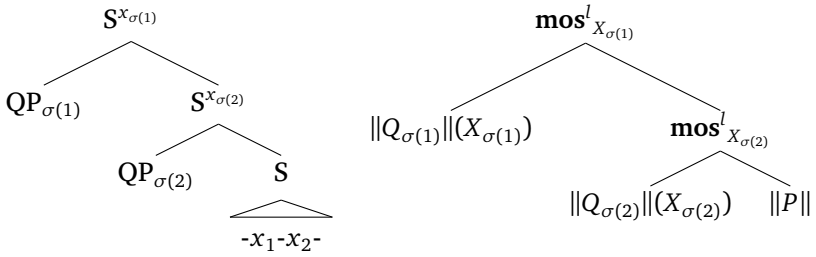
In this case, there is one such map, so strategy A yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy*.

(A2) Surface structure tree:



(A2) Formal structure tree (LF) and the corresponding computation tree:



The computation tree in (A2) gives rise to the following general map, with $\sigma \in S_2$ (where S_2 is the set of permutations of the set $\{1,2\}$):

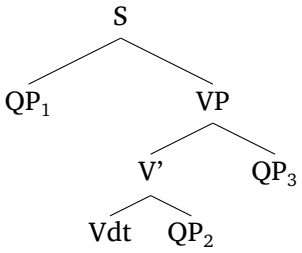
$$\mathbf{strat}_A^{2,\sigma}:\quad \begin{array}{c} \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2) \xrightarrow{\bar{\pi}_{\sigma(i)}} \mathscr{C}(X_{\sigma(i)}) \\ \downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \pi_2\rangle \\ \mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{P}(X_1\times X_2) \\ \downarrow 1\times \mathbf{mos}^l{}_{X_{\sigma(2)}} \\ \mathscr{C}(X_{\sigma(1)})\times\mathscr{P}(X_{\sigma(1)}) \\ \downarrow \mathbf{mos}^l{}_{X_{\sigma(1)}} \\ \mathbf{t} \end{array}$$

where $\bar{\pi}_{\sigma(i)}$ is the projection on the 1st factor if $\sigma(i)=1$, and on the 3rd factor if $\sigma(i)=2$, i.e. as it should be. This convention will be used in all similar diagrams without any further explanations.

There are two such maps corresponding to the two permutations σ of $\{1,2\}$. These maps are different in general. Thus strategy A yields two (both) asymmetric readings for a sentence with two QPs.

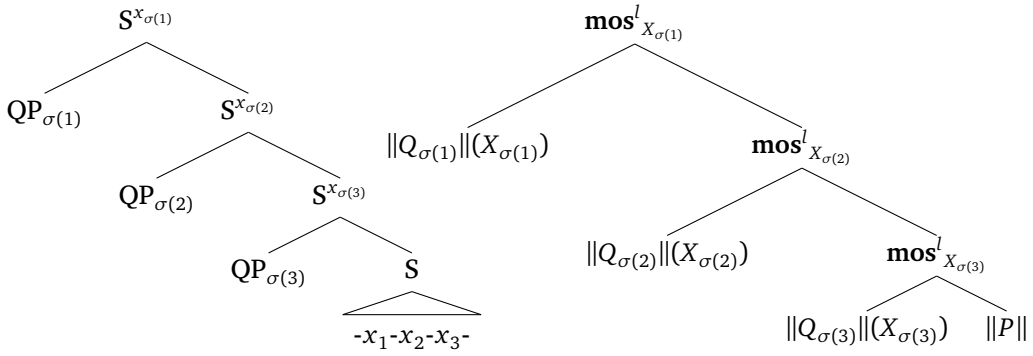
Sentence with three QPs, e.g. *Some teacher gave every student most books.*

(A3) Surface structure tree:[6]



[6] In this paper, we adopt the structure postulated by Chomsky (1993).

(A3) Formal structure tree (LF) and the corresponding computation tree:



The computation tree in (A3) gives rise to the following general map, with $\sigma \in S_3$ (where S_3 is the set of permutations of the set $\{1,2,3\}$):

$$\mathbf{strat}_A^{3,\sigma}:\quad \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$
$$\downarrow\ \langle\bar{\pi}_{\sigma(1)},\bar{\pi}_{\sigma(2)},\bar{\pi}_{\sigma(3)},\pi_2\rangle$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{C}(X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow\ 1\times 1\times\mathbf{mos}^{l}{}_{X_{\sigma(3)}}$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{P}(\ldots\times\widehat{X_{\sigma(3)}}\times\ldots)$$
$$\downarrow\ 1\times\mathbf{mos}^{l}{}_{X_{\sigma(2)}}$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{P}(X_{\sigma(1)})$$
$$\downarrow\ \mathbf{mos}^{l}{}_{X_{\sigma(1)}}$$
$$\mathbf{t}$$

There are six such maps corresponding to the six permutations σ of $\{1,2,3\}$. These maps are different in general. Thus strategy A yields 6 asymmetric readings for a sentence with three QPs.

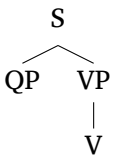
3.3 *Strategy B*

In the polyadic approach (as implemented in May 1985):

- Surface structure trees are rewritten (disambiguated) as formal structure trees (Polyadic Logical Forms) via:
  - QR Rule;
  - Predicate Collapsing;
  - Rotation.
- Formal structure trees (PLFs) are relabelled as computation trees as follows:
  - *Polyadic* (root of a subtree representing a polyadic quantifier) is interpreted as a suitably typed **pile′up**-operation (we can choose whether to use only $\mathbf{pile'up}^l$ or $\mathbf{pile'up}^r$ and then stick to that decision).
  - S^x, S, QP are interpreted as above.

Sentence with one QP, e.g. *Every kid (most kids) entered.*

(B1) Surface structure tree:



(B1) Formal structure tree (PLF) and the corresponding computation tree:

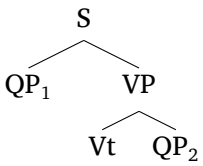
The computation tree in (B1) gives rise to the following general map:

$$\mathbf{strat}^1_B : \quad \mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{\mathbf{mos}^l{}_X} \mathbf{t}$$

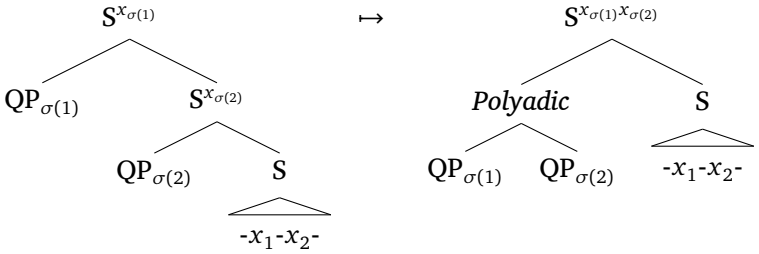
In this case, there is one such map, so strategy B yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy*.

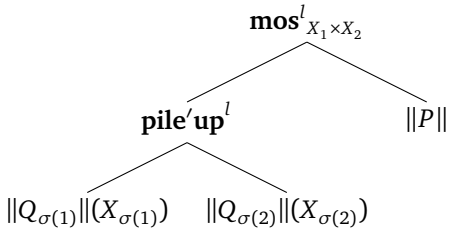
(B2) Surface structure tree:



(B2) Formal structure tree (PLF) obtained from LF in (A2) via rotation:



and the corresponding computation tree:



The computation tree in (B2) gives rise to the following general map, with $\sigma \in S_2$:

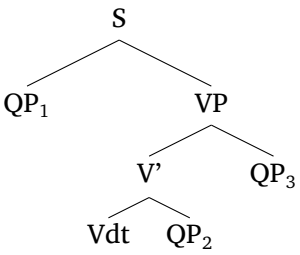
$$\mathbf{strat}_B^{2,\sigma}:$$

$$\begin{array}{c}
\mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2)\\
\downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \pi_2\rangle\\
\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{P}(X_1\times X_2)\\
\downarrow \mathbf{pile}'\mathbf{up}^l\times 1\\
\mathscr{C}(X_{\sigma(1)}\times X_{\sigma(2)})\times\mathscr{P}(X_1\times X_2)\\
\downarrow \mathscr{C}(\pi_{\sigma^{-1}})\times 1\\
\mathscr{C}(X_1\times X_2)\times\mathscr{P}(X_1\times X_2)\\
\downarrow \mathbf{mos}^l{}_{X_1\times X_2}\\
\mathbf{t}
\end{array}$$

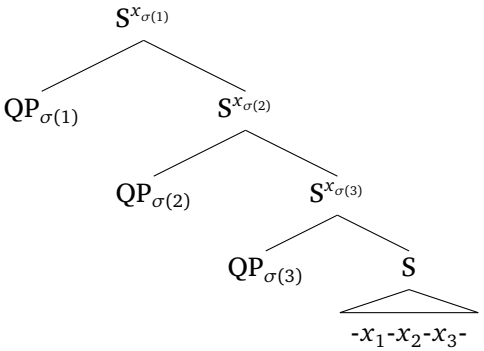
There are two such maps, corresponding to the two permutations σ of $\{1,2\}$ combined with a $\mathbf{pile}'\mathbf{up}^l$-operation (here, we can also choose to use both **pile′ups** instead and no permutations at all). These maps are different in general. Thus strategy B yields two (both) asymmetric readings for a sentence with two QPs.

Sentence with three QPs, e.g. *Some teacher gave every student most books.*

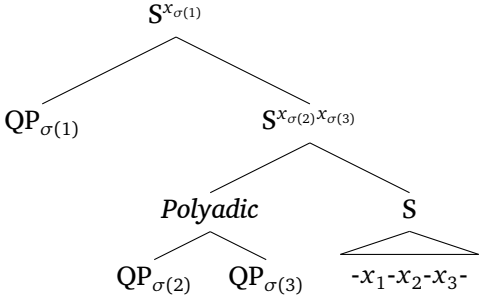
(B3) Surface structure tree:



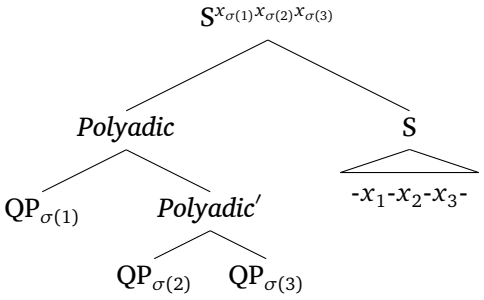
(B3) Formal structure tree (PLF) obtained from LF in (A3) via rotation:



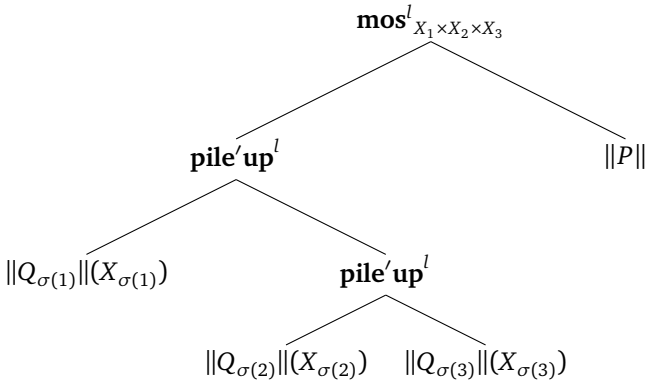
$\mapsto$



$\mapsto$



and the corresponding computation tree:



The computation tree in (B3) gives rise to the following general map, with $\sigma \in S_3$:

$$\mathbf{strat}_B^{3,\sigma}:$$

$$\mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$
$$\downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \bar{\pi}_{\sigma(3)}, \pi_2\rangle$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{C}(X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow 1\times \mathbf{pile'up}^{l}\times 1$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)}\times X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathbf{pile'up}^{l}\times 1$$
$$\mathscr{C}(X_{\sigma(1)}\times X_{\sigma(2)}\times X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathscr{C}(\pi_{\sigma^{-1}})\times 1$$
$$\mathscr{C}(X_1\times X_2\times X_3)\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathbf{mos}^{l}{}_{X_1\times X_2\times X_3}$$
$$\mathbf{t}$$

There are six such maps, corresponding to the six permutations σ of $\{1,2,3\}$ combined with a $\mathbf{pile}'\mathbf{up}^l$-operation (here, we can also choose to use $\mathbf{pile}'\mathbf{up}^r$ instead). These maps are different in general. Thus strategy B yields 6 asymmetric readings for a sentence with three QPs.

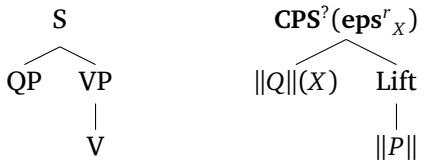
3.4 *Strategy C*

In the continuation-based strategy approach (as proposed in Barker 2002):

- A surface structure tree is rewritten as a formal structure tree via:
  - no rewriting rules (formal structure trees are just surface structure trees – this is what is understood by in situ).
- Relabelling formal structure trees (= surface structure trees) as computation trees follows this procedure:
  - S, VP, V' (roots of a (sub)tree with some (possibly all) arguments provided) are interpreted as suitably typed **CPS**-operations (left and right);
  - V, Vt, Vdt (leaves of a tree) are interpreted as 'continuized' predicates (1-, 2-, 3-ary, respectively).

Sentence with one QP, e.g. *Every kid (most kids) entered.*

(C1) Surface structure tree and the corresponding computation tree:



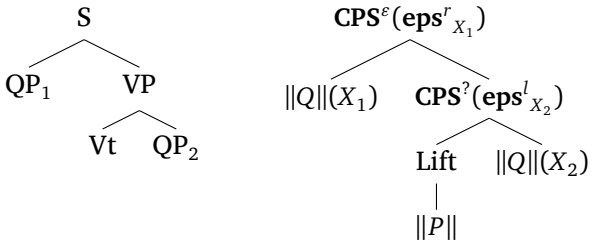
The computation tree in (C1) gives rise to the following general map:

$$\mathbf{strat}^1_c : \mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{1 \times \eta_{\mathscr{P}(X)}} \mathscr{C}(X) \times \mathscr{C}\mathscr{P}(X) \xrightarrow{\mathbf{CPS}^?(\mathbf{eps}^r{}_X)} \mathscr{C}(\mathbf{t}) \xrightarrow[ev_{id_{\mathbf{t}}}]{} \mathbf{t}$$

We use $\mathbf{CPS}^{?}$ when it does not matter whether we apply $\mathbf{CPS}^{l}$ or $\mathbf{CPS}^{r}$. This is the case when one of the arguments is a lifted element (like interpretations of predicates in this strategy). Strategy C yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy*.

(C2) Surface structure tree and the corresponding computation tree:



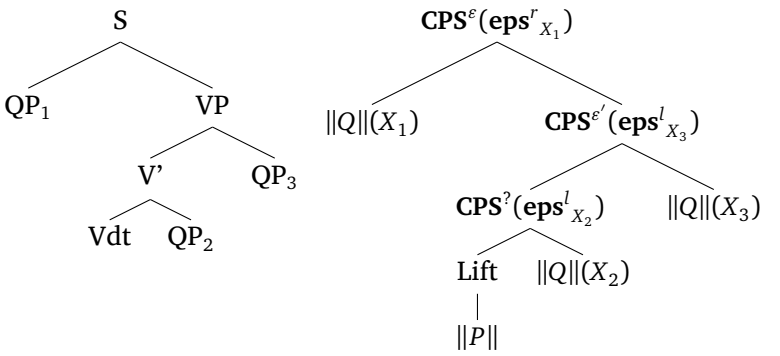
The computation tree in (C2) gives rise to the following general map:

$$\mathbf{strat}^{2,\varepsilon}_{C}:\quad \begin{array}{c} \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2) \\ \downarrow\ 1\times\eta_{\mathscr{P}(X_1\times X_2)}\times 1 \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2) \\ \downarrow\ 1\times\mathbf{CPS}^{?}(\mathbf{eps}^{l}_{X_2}) \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1) \\ \downarrow\ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}_{X_1}) \\ \mathscr{C}(\mathbf{t}) \xrightarrow[ev_{id_{\mathbf{t}}}]{} \mathbf{t} \end{array}$$

with $\varepsilon \in \{l, r\}$. Depending on whether we use $\mathbf{CPS}^{l}$ or $\mathbf{CPS}^{r}$, we get the relevant one of the two asymmetric readings for a sentence with two QPs. Strategy C yields two readings for a sentence with two QPs, corresponding to the two forms of **CPS**.

Sentence with three QPs, e.g. *Some teacher gave every student most books.*

(C3) Surface structure tree and the corresponding computation tree:



The computation tree in (C3) gives rise to the following general map:

$$\mathbf{strat}_C^{3,\varepsilon',\varepsilon}:\quad \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$

$$\downarrow\ 1\times\eta_{\mathscr{P}(X_1\times X_2\times X_3)}\times 1\times 1$$

$$\mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$

$$\downarrow\ 1\times\mathbf{CPS}^{?}(\mathbf{eps}^{l}{}_{X_2})\times 1$$

$$\mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_3)\times\mathscr{C}(X_3)$$

$$\downarrow\ 1\times\mathbf{CPS}^{\varepsilon'}(\mathbf{eps}^{l}{}_{X_3})$$

$$\mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1)$$

$$\downarrow\ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_1})$$

$$\mathscr{C}(\mathbf{t})\xrightarrow[ev_{id_{\mathbf{t}}}]{}\mathbf{t}$$

Strategy C provides four asymmetric readings for the sentence, such that QP in subject position can be placed either first or last only

(corresponding to the four possible combinations of the two forms of **CPS**). Thus it yields four out of the six readings accounted for by strategies A and B. Of course, it is a matter of empirical discovery which readings are available for such sentences, and the status of the two missing 'interleaved' interpretations (*every* > *some* > *most* and *most* > *some* > *every*) is still under discussion.

The tables below summarize the main features of the three approaches.

Passing from Surface Structure Trees to Formal Structure Trees

Strategy	***A***	***B***	***C***
Rewrite rules	*QR, Predicate Collapsing*	*QR, Predicate Collapsing, Rotation*	*No rewrite rules (in situ)*

Passing from Formal Structure Trees to Computation Trees

Strategy	***A***	***B***	***C***
Relabelling inner nodes	$S^x \mapsto$ **mos**	$S^{\vec{x}} \mapsto$ **mos** *Polyadic* $\mapsto$ **pile′up**	$S, VP, V' \mapsto$ **CPS**
Relabelling leaves	$S \mapsto$ *relation* $QP \mapsto$ $\mathscr{C}$*-comp.*	$S \mapsto$ *relation* $QP \mapsto$ $\mathscr{C}$*-comp.*	$V, Vt, Vdt \mapsto$ *continuized relation* $QP \mapsto$ $\mathscr{C}$*-comp.*

The semantics for sentences with intransitive or transitive verbs, as defined by strategies A, B, and C, will be equivalent. The semantics for sentences with ditransitive verbs, as defined by strategies A, and B, will be equivalent, providing six asymmetric readings for the sentence. The semantics for sentences with ditransitive verbs, as defined by strategy C, will provide four asymmetric readings for the sentence, such that QP in subject position can be placed either first or last only, corresponding to four out of the six readings accounted for by strategies A and B. The proofs are given in the Appendix.

4 CONCLUSIONS AND FUTURE WORK

We compared three scope-assignment strategies for simple multi-quantifier sentences: the traditional movement strategy, the polyadic approach, and the continuation-based approach. These strategies can be viewed as instances of the same general pattern: first transform the SS-tree of a sentence so as to obtain the shape of the computation tree, then relabel the leaves of that tree, with the interpretation of lexical items (predicate and QPs), and the inner nodes, using algebraic operations, and finally evaluate this computation in order to get the truth value of the whole sentence. We have shown that while the traditional movement strategy is very close to the original semantics for the logic with generalized quantifiers due to Mostowski, the polyadic approach and the continuation-based strategy are in fact cognate in spirit, as they can both be defined using operations derived from the strength of the continuation monad. As the polyadic strategy is well-understood among linguists, we hope that our results will help to make the continuation-based strategy more popular. With the continuation monad as a common basis for the three scope-assignment strategies discussed, it is also easy to identify their relative merits and weaknesses. Traditional and polyadic approaches cannot provide a non-movement (in situ) analysis of quantifiers. The continuation-based strategy is in situ but does not account for all the asymmetric readings possible for sentences involving three QPs. As discussed in Bekki and Asai (2009) and proved in this paper, it only provides four out of the six readings possible for such sentences. In the sequel to this paper, we show how to overcome this problem, keeping the resulting strategy in situ (Grudzinska and Zawadowski 2016). We take the results of this work to be the first step towards an in situ semantics that will be sufficient to account for the whole range of possible readings for multi-quantifier sentences.

5 APPENDIX

5.1 *The continuation monad*

In this subsection, we gather all the basic facts (sometimes repeated from the text) of the *continuation monad* $\mathscr{C}$ on *Set*. We have an adjunction:

$$Set \underset{\mathscr{P}^{op}}{\overset{\mathscr{P}}{\rightleftarrows}} Set^{op}$$

where both $\mathscr{P}$ and $\mathscr{P}^{op}$ are the contravariant powerset functors[7] with the domains and codomains as displayed. In particular, for $f : X \longrightarrow Y$, the function $\mathscr{P}(f) = f^{-1} : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X)$ is given by:

$$f^{-1}(h) = h \circ f$$

for $h : Y \longrightarrow \mathbf{t}$. Function $\eta_X : X \longrightarrow \mathscr{C}(X)$, the component at set X of the unit of this adjunction $\eta : 1_{Set} \longrightarrow \mathscr{P}\mathscr{P}^{op} = \mathscr{C}$, is given by:

$$\eta_X(x) = \lambda h_{:\mathscr{P}(X)}.h(x).$$

Function $\varepsilon_X : X \longrightarrow \mathscr{C}(X)$, the component at set X of the co-unit of this adjunction $\varepsilon : 1_{Set} \longrightarrow \mathscr{P}^{op}\mathscr{P}$, is given by (essentially the same formula):

$$\varepsilon_X(x) = \lambda h_{:\mathscr{P}^{op}(X)}.h(x)$$

for $x \in X$. The function $\mathscr{C}(f) : \mathscr{C}(X) \longrightarrow \mathscr{C}(Y)$, for $Q : \mathscr{P}(X) \longrightarrow \mathbf{t} \in \mathscr{C}(X)$, is a function $\mathscr{C}(f)(Q) : \mathscr{P}(Y) \longrightarrow \mathbf{t}$ given by:

$$\mathscr{C}(f)(Q)(h) = Q(h \circ f)$$

for $h : Y \longrightarrow \mathbf{t}$.

The monad induced by this adjunction is the continuation monad. Its multiplication is given by the co-unit of the above adjunction transported back to *Set*, i.e. $\mu = \mathscr{P}^{op}(\varepsilon_{\mathscr{P}})$. For X in *Set*, the function

$$\mu_X : \mathscr{C}^2(X) \longrightarrow \mathscr{C}(X)$$

is given by:

$$\mu_X(\mathscr{R}) = \mathscr{R} \circ \eta_{\mathscr{P}(X)} \qquad \text{for} \quad \mathscr{R} \in \mathscr{C}^2(X).$$

In λ-notation we write:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\lambda D_{:\mathscr{C}(X)}.D(h)).$$

[7] Note that this is in contrast with the functor $\mathscr{P}$, where $\mathscr{P}$ is the covariant power-set functor.

The left strength for the monad $\mathscr{C}$ is:

$$\mathbf{st}^l : \mathscr{C}(X) \times Y \longrightarrow \mathscr{C}(X \times Y)$$

for $M \in \mathscr{C}(X)$ and $y \in Y$, given by:

$$\mathbf{st}^l(M, y) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.c(x, y)) : \mathscr{P}(X \times Y) \longrightarrow \mathbf{t}$$

and the right strength, for $x \in X$ and $N \in \mathscr{C}(Y)$, is given by:

$$\mathbf{st}^r(x, N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.c(x, y)) : \mathscr{P}(X \times Y) \longrightarrow \mathbf{t}.$$

The left pile'up operation:

$$\mathbf{pile'up}^l : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

is the following composition:

$$\mathscr{C}(X) \times \mathscr{C}(Y) \xrightarrow{\mathbf{st}^l} \mathscr{C}(X \times \mathscr{C}(Y)) \xrightarrow{\mathscr{C}(\mathbf{st}^r)} \mathscr{C}^2(X \times Y) \xrightarrow{\mu_{X\times Y}} \mathscr{C}(X \times Y)$$

where, for $Q \in \mathscr{C}(X)$, $Q' \in \mathscr{C}(Y)$, $c \in \mathscr{P}(X \times \mathscr{C}(Y))$, we have:

$$\mathbf{st}^l(Q, Q')(c) = Q(\lambda x_{:X} c(x, Q))$$

and, for $d \in \mathscr{C}(X \times \mathscr{C}(Y))$, $\mathscr{U} \in \mathscr{P}\mathscr{C}(X \times Y)$, we have:

$$\mathscr{C}(\mathbf{st}^r)(d)(\mathscr{U}) = d(\mathscr{U} \circ \mathbf{st}^r).$$

Now, using the above formulas, we can calculate $\mathbf{pile'up}^l$ as the composition on $Q \in \mathscr{C}(X)$, $Q' \in \mathscr{C}(Y)$, and $c \in \mathscr{P}(X \times Y)$ as follows:

$$\begin{aligned}
\mathbf{pile'up}^l(Q, Q')(c) &= \mu_{X\times Y}(\mathscr{C}(\mathbf{st}^r)(\mathbf{st}^l(Q, Q')))(c) \\
&= \mathscr{C}(\mathbf{st}^r)(\mathbf{st}^l(Q, Q'))(\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \\
&= \mathbf{st}^l(Q, Q')((\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \circ \mathbf{st}^r) \\
&= Q(\lambda x_{:X}((\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \circ \mathbf{st}^r)(x, Q')) \\
&= Q(\lambda x_{:X}((\lambda D_{:\mathscr{C}(X\times Y)} D(c))(\mathbf{st}^r(x, Q')) \\
&= Q(\lambda x_{:X}\mathbf{st}^r(x, Q')(c)) \\
&= Q(\lambda x_{:X} Q'(\lambda y_{:Y} c(x, y)))
\end{aligned}$$

Similarly, we can show that:

$$\mathbf{pile'up}^r(Q,Q')(c) = Q'(\lambda y_{:Y} Q(\lambda x_{:X} c(x,y))).$$

One can easily verify that **pile′up**'s are related by:

$$\mathbf{pile'up}^r{}_{X,Y} = \mathscr{C}(\pi_{(2,1)}) \circ \mathbf{pile'up}^l{}_{Y,X} \circ \pi_{(2,1)}.$$

5.2 *Some properties of* **pile′up** *operations*

Lemma 5.1 (pile′up lemma) **pile′up***s on pairs where one element is continuized agree and are equal to the corresponding strength.*

Proof. We have to show that the functions:

$$T(X_1) \times T(X_2) \underset{\mathbf{pile'up}^r{}_{X_1,X_2}}{\overset{\mathbf{pile'up}^l{}_{X_1,X_2}}{\rightrightarrows}} T(X_1 \times X_2)$$

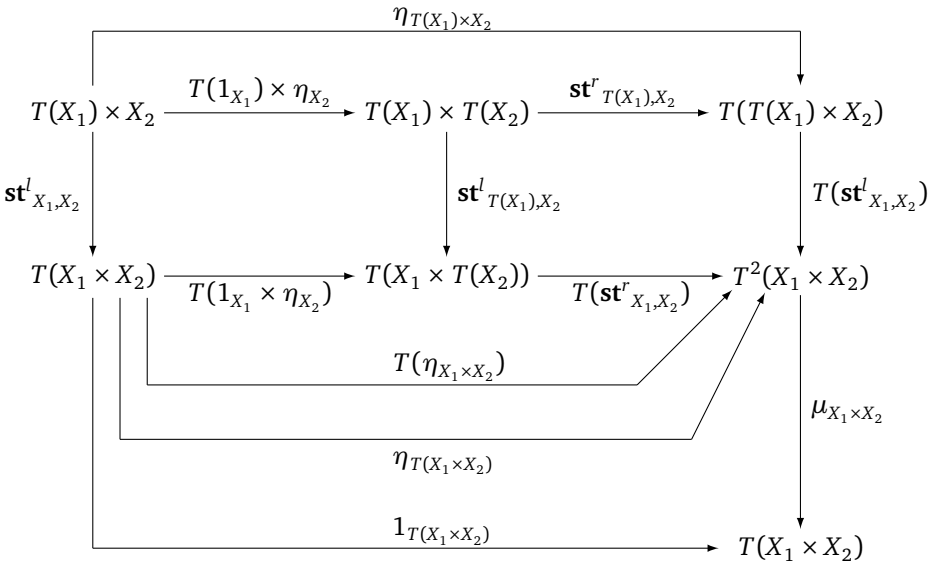
are equalized by both:

$$X_1 \times T(X_2) \xrightarrow{\eta_{X_1} \times T(1_{X_2})} T(X_1) \times T(X_2)$$

and

$$T(X_1) \times X_2 \xrightarrow{T(1_{X_1}) \times \eta_{X_2}} T(X_1) \times T(X_2)$$

and their composition with these functions is equal to strength morphisms. Using the diagram:



we shall show that:

$$\mathbf{pile'up}^{r}{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \mathbf{st}^{l}{}_{X_1,X_2} = \mathbf{pile'up}^{l}{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}).$$

The other cases are symmetric. We have:

$$\mathbf{pile'up}^{r}{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \quad (\text{def of } \mathbf{pile'up}^{r})$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^{l}{}_{X_1,X_2}) \circ \mathbf{st}^{r}{}_{T(X_1),X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) \quad (\eta \text{ strong w.r.t. } \mathbf{st}^{r})$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^{l}{}_{X_1,X_2}) \circ \eta_{T(X_1) \times X_2} \quad (\eta \text{ nat transf})$$

$$= \mu_{X_1 \times X_2} \circ \eta_{T(X_1 \times X_2)} \circ \mathbf{st}^{l}{}_{X_1,X_2} \quad (T \text{ monad})$$

$$= \mathbf{st}^{l}{}_{X_1,X_2}$$

To show the remaining equation, we can continue the penultimate formula above as follows:

$$\mathbf{pile'up}^{r}{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \ldots = \mu_{X_1 \times X_2} \circ \eta_{T(X_1 \times X_2)}) \circ \mathbf{st}^{l}{}_{X_1,X_2}$$

$$= \quad (T \text{ monad})$$

$$= \mu_{X_1 \times X_2} \circ T(\eta_{X_1 \times X_2}) \circ \mathbf{st}^{l}{}_{X_1,X_2} \quad (\eta \text{ strong w.r.t. } \mathbf{st}^{r})$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^{r}{}_{X_1,X_2}) \circ T(1_{X_1} \times \eta_{X_2}) \circ \mathbf{st}^{l}{}_{X_1,X_2} \quad (\mathbf{st}^{l} \text{ nat transf})$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^{r}{}_{X_1,X_2}) \circ \mathbf{st}^{l}{}_{X_1,X_2} \circ T(1_{X_1} \times \eta_{X_2}) \quad (\text{def of } \mathbf{pile'up}^{l})$$

$$= \mathbf{pile'up}^{l}{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) \qquad \diamondsuit$$

Corollary 5.2 *The left and right* **CPS***-operation on pairs where one element is continuized agree.*

Proof. The corollary states that, for any sets X, Y, Z and a function $f : X \times Y \longrightarrow Z$, both morphisms:

$$X \times T(Y) \xrightarrow{\eta_X \times 1} T(X) \times T(Y)$$

and

$$T(X) \times Y \xrightarrow{1 \times \eta_Y} T(X) \times T(Y)$$

equalize the pair of morphisms:

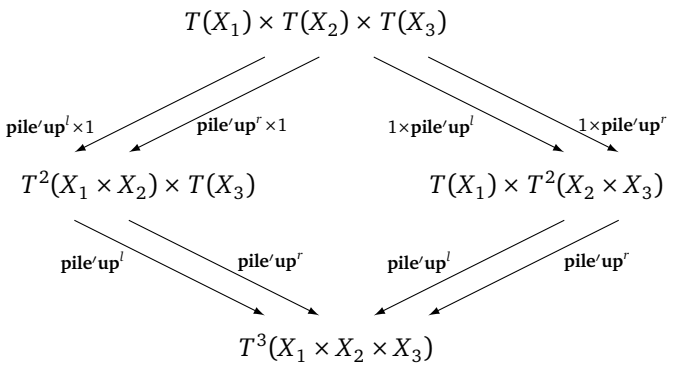
$$T(X) \times T(Y) \underset{\mathbf{CPS}^{r}(f)}{\overset{\mathbf{CPS}^{l}(f)}{\rightrightarrows}} Z$$

This immediately follows from the above lemma and the definition of **CPS**es. ♢

Using binary $\mathbf{pile'up}$ operations, we can define eight ternary $\mathbf{pile'up}$ operations:

$$T(X_1) \times T(X_2) \times T(X_3) \longrightarrow T(X_1 \times X_2 \times X_3)$$

out of the following diagram:



However, both $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ operations are associative (Proposition 5.3 below) and hence only six of them are different, in general.

Proposition 5.3 *Both* $\mathbf{pile'up}^l$ *and* $\mathbf{pile'up}^r$ *operations are associative on any monad on Set.*

Proof. In fact, $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ are associative on any bi-strong monad in the monoidal category. We shall show this fact for a monad T on *Set* with canonical strength.

We need to show that:

$$\mathbf{pile'up}^r \circ (\mathbf{pile'up}^r \times 1) = \mathbf{pile'up}^r \circ (1 \times \mathbf{pile'up}^r)$$

and

$$\mathbf{pile'up}^l \circ (\mathbf{pile'up}^l \times 1) = \mathbf{pile'up}^l \circ (1 \times \mathbf{pile'up}^l)$$

But as $\mathbf{pile'up}$s are mutually definable, either of these equalities readily implies the other. We shall show the second equality. For sets X_1, X_2, X_3, using all the assumptions, we have:

$\mathbf{pile'up}^l{}_{X_1 \times X_2, X_3} \circ (\mathbf{pile'up}^l{}_{X_1, X_2} \times 1_{T(X_3)}) =$

$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1\times X_2,X_3})\circ \underline{\mathbf{st}^l{}_{X_1\times X_2,T(X_3)}\circ(\mu_{X_1\times X_2}\times T(1_{X_3}))}$$
$$\circ(T(\mathbf{st}^r{}_{X_1,X_2})\times 1_{T(X_3)})\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ \underline{T(\mathbf{st}^r{}_{X_1\times X_2,X_3})\circ\mu_{X_1\times X_2\times T(X_3)}}\circ T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})$$
$$\circ \underline{\mathbf{st}^l{}_{T(X_1\times X_2),T(X_3)}\circ(T(\mathbf{st}^r{}_{X_1,X_2})\times T(1_{X_3}))}\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})$$
$$\circ(T(\mathbf{st}^r{}_{X_1,X_2}\times 1_{X_3}))\circ \underline{\mathbf{st}^l{}_{T(X_1\times X_2),T(X_3)}\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ \underline{T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})}$$
$$\underline{\circ(T(\mathbf{st}^r{}_{X_1,X_2}\times 1_{T(X_3)}))}\circ \mathbf{st}^l{}_{X_1,T(X_2)\times T(X_3)}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})$$
$$\circ \underline{T(1_{X_1}\times \mathbf{st}^l{}_{X_2,T(X_3)})\circ \mathbf{st}^l{}_{X_1,T(X_2)\times T(X_3)}}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ \underline{T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})}\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})$$
$$\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ \underline{T^2(1_{X_1}\times \mathbf{st}^r{}_{X_2,X_3})}$$
$$\underline{\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})}\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ T(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})$$
$$\circ \underline{T(1_{X_1}\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ \underline{\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ T(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})}$$
$$\circ \mathbf{st}^l{}_{X_1,T^2(X_2\times X_3)}\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1,X_2\times X_3})\circ \underline{T(1_{X_1}\times \mu_{X_2\times X_3})\circ \mathbf{st}^l{}_{X_1,T^2(X_2\times X_3)}}$$
$$\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1,X_2\times X_3})\circ \mathbf{st}^l{}_{X_1,T(X_2\times X_3)}\circ(T(1_{X_1})\times \mu_{X_2\times X_3})$$
$$\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mathbf{pile'up}^l{}_{X_1,X_2\times X_3}\circ(1_{T(X_3)}\times \mathbf{pile'up}^l{}_{X_2,X_3}) \qquad \diamond$$

5.3 *Arity one: intransitive verbs*

Proposition 5.4 *The semantics for sentences with intransitive verbs, as defined by strategies A, B, and C, will be equivalent.*

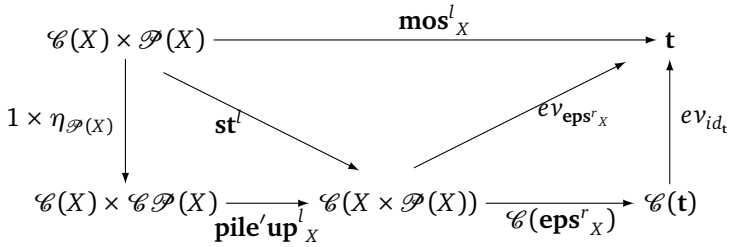
Proof. In case of a sentence with an intransitive verb, the semantics will be defined by the morphisms $\mathbf{strat}^1_A$, $\mathbf{strat}^1_B$, and $\mathbf{strat}^1_C$. We need to show that they are equal. We have:

$$\mathbf{strat}^1_A = \mathbf{mos}^l{}_X = \mathbf{strat}^1_B.$$

$\mathbf{strat}^1_C$ is the composition of the following morphisms:

$$\mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{1 \times \eta_{\mathscr{P}(X)}} \mathscr{C}(X) \times \mathscr{C}\mathscr{P}(X) \xrightarrow{\mathbf{CPS}^l(\mathbf{eps}^r{}_X)} \mathscr{C}(\mathbf{t}) \xrightarrow{ev_{id_\mathbf{t}}} \mathbf{t}$$

Thus we need to show that this composition is equal to $\mathbf{mos}^l{}_X$. Consider the following diagram:



The left triangle commutes, as a consequence of Lemma 5.1. To see that the central triangle commutes, we take $M \in \mathscr{C}(X)$ and $h \in \mathscr{P}(X)$, and calculate:

$$\begin{aligned} ev_{\mathbf{eps}^r{}_X} \circ \mathbf{st}^r(Q,h) &= ev_{\mathbf{eps}^r{}_X}(\lambda D_{:\mathscr{P}(X\times\mathscr{P}(X))} M(\lambda x_{:X} D(x,h))) \\ &= M(\lambda x_{:X} \mathbf{eps}^r{}_X(x,h)) \\ &= M(\lambda x_{:X} h(x)) \\ &= N(h) \\ &= \mathbf{mos}^l(N,h). \end{aligned}$$

Finally, to see that the right triangle commutes, we take $N \in \mathscr{C}(X \times \mathscr{P}(X))$ and calculate:

$$\begin{aligned} ev_{id_\mathbf{t}} \circ \mathscr{C}(\mathbf{eps}^r{}_X)(N) &= ev_{id_\mathbf{t}}(\lambda c_{:\mathscr{P}(\mathbf{t})} N(c \circ \mathbf{eps}^r{}_X)) \\ &= N(\mathbf{eps}^r{}_X) \\ &= ev_{\mathbf{eps}^r{}_X}(N). \end{aligned}$$

Thus the whole diagram commutes, and hence $\mathbf{strat}^1_C = \mathbf{mos}^l{}_X$, as required. $\diamondsuit$

The above proof also shows the following technical lemma.

Lemma 5.5 *For any set X, the following diagram commutes:*

$$\begin{array}{ccc} \mathscr{C}(X) \times \mathscr{P}(X) & \xrightarrow{\mathbf{mos}^l{}_X} & \mathbf{t} \\ \Big\downarrow{\scriptstyle \mathbf{st}^l} & & \Big\uparrow{\scriptstyle ev_{id_{\mathbf{t}}}} \\ \mathscr{C}(X \times \mathscr{P}(X)) & \xrightarrow[\mathscr{C}(\mathbf{eps}^r{}_X)]{} & \mathscr{C}(\mathbf{t}) \end{array}$$

5.4 *Arity two: transitive verbs*

Proposition 5.6 *The semantics for sentences with transitive verbs, as defined by strategies A, B, and C, will be equivalent, providing two asymmetric readings for the sentence.*

Proof. In the case of sentences with transitive verbs, the semantics will be defined by morphisms $\mathbf{strat}^{2,\sigma}_A$, $\mathbf{strat}^{2,\sigma}_B$, and $\mathbf{strat}^{2,\varepsilon}_C$, with $\sigma \in S_2 = \{id_2, \tau\}$ and $\varepsilon \in \{l, r\}$. We need to show the equalities:

$$\mathbf{strat}^{2,\sigma}_A = \mathbf{strat}^{2,\sigma}_B,$$

for $\sigma \in S_2$, and

$$\mathbf{strat}^{2,id_2}_B = \mathbf{strat}^{2,l}_C, \qquad \mathbf{strat}^{2,\tau}_B = \mathbf{strat}^{2,r}_C.$$

To show the first equality, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, and $P \in \mathscr{P}(X_1 \times X_2)$, we have:

$$\begin{aligned}
&\mathbf{strat}^{2,\sigma}_A(Q_1, Q_2, P) = \\
&= \mathbf{mos}^l{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \mathbf{mos}^l{}_{X_{\sigma(2)}}(Q_{\sigma(2)}, P)) \\
&= \mathbf{mos}^l{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(x_1, x_2))) \\
&= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(x_1, x_2))) \\
&= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(\pi_{\sigma^{-1}}(x_{\sigma(1)}, x_{\sigma(2)})))) \\
&= \mathbf{pile'up}^l(Q_{\sigma(1)}, Q_{\sigma(2)})(P \circ \pi^{-1}) \\
&= \mathscr{C}(\pi_{\sigma^{-1}})(\mathbf{pile'up}^l(Q_{\sigma(1)}, Q_{\sigma(2)}))(P) \\
&= \mathbf{strat}^{2,\sigma}_B(Q_1, Q_2, P)
\end{aligned}$$

To show the remaining two equalities, let us first note that if either $\sigma = id_2$ and $\varepsilon = l$ or $\sigma = \tau$ and $\varepsilon = r$, we have:

$$\mathbf{pile'up}^{\varepsilon} = \mathscr{C}(\pi_{\sigma^{-1}}) \circ \mathbf{pile'up}^{l} \circ \pi_{\sigma}.$$

Thus we shall assume the above equation relating σ with ε, and, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, and $P \in \mathscr{P}(X_1 \times X_2)$, we obtain:[8]

$\mathbf{strat}^{2,\varepsilon}_{C} =$

$= ev_{id_t} \circ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_1}) \circ (1 \times \mathbf{CPS}^{?}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times 1 \times \eta_{\mathscr{P}(X_1 \times X_2)})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathbf{pile'up}^{\varepsilon} \circ (\mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2}) \circ \mathbf{pile'up}^{\varepsilon} \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{\varepsilon} \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{?} \circ (\mathbf{pile'up}^{\varepsilon} \times 1) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{?} \circ (1 \times \eta) \circ (1 \times \mathbf{pile'up}^{\varepsilon})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{st}^{l} \circ (1 \times \mathbf{pile'up}^{\varepsilon})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{st}^{l} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times 1) \circ (\mathbf{pile'up}^{l} \times 1) \circ (\pi_{\sigma} \times 1)$

$= \mathbf{mos}^{l}{}_{X_1 \times X_2} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times 1) \circ (\mathbf{pile'up}^{l} \times 1) \circ (\pi_{\sigma} \times 1)$

$= \mathbf{strat}^{2,\sigma}_{B}$

In the above calculations, we used the definition of **CPS**es, the naturality of $\mathbf{pile'up}^{\varepsilon}$, the relations between **eps** morphisms, the associativity of $\mathbf{pile'up}^{\varepsilon}$ (Proposition 5.3), the properties of product morphisms, the **pile'up** lemma, and finally, Lemma 5.5.

Here and below, $\mathbf{CPS}^{?}$, $\mathbf{pile'up}^{?}$ stands for either $\mathbf{CPS}^{l}$, $\mathbf{pile'up}^{l}$ or $\mathbf{CPS}^{r}$, $\mathbf{pile'up}^{r}$,whichever is more convenient at the time, as it does not influence the end result. ♢

[8] The diagram illustrating these calculations would be too big to fit on a page but the reader is encouraged to draw one.

Proposition 5.7 *The semantics for sentences with ditransitive verbs, as defined by strategies A and B, will be equivalent, providing six asymmetric readings of the sentence.*

Proof. In the case of sentences with ditransitive verbs, the semantics will be defined by the morphisms $\mathbf{strat}_A^{3,\sigma}$, $\mathbf{strat}_B^{3,\sigma}$, and $\mathbf{strat}_C^{2,\varepsilon}$, with $\sigma \in S_3$ and $\varepsilon \in \{l, r\}$. We need to show the equalities:

$$\mathbf{strat}_A^{3,\sigma} = \mathbf{strat}_B^{3,\sigma},$$

for $\sigma \in S_3$.

The calculations are similar to those for transitive verbs. We present them for completeness. With $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, $Q_3 \in \mathscr{C}(X_3)$, and $P \in \mathscr{P}(X_1 \times X_2 \times X_3)$, we have:

$$\mathbf{strat}_A^{3,\sigma}(Q_1, Q_2, Q_3, P) =$$

$$= \mathbf{mos}^{l}{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \mathbf{mos}^{l}{}_{X_{\sigma(2)}}(Q_{\sigma(2)}, \mathbf{mos}^{l}{}_{X_{\sigma(3)}}(Q_{\sigma(3)}, P)))$$

$$= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.Q_{\sigma(3)}(\lambda x_{\sigma(3):X_{\sigma(3)}}.P(x_1, x_2, x_3))))$$

$$= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.Q_{\sigma(3)}($$

$$\lambda x_{\sigma(3):X_{\sigma(3)}}.P(\pi_{\sigma^{-1}}(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)})))))$$

$$= \mathbf{pile'up}^{l}(Q_{\sigma(1)}, \mathbf{pile'up}^{l}(Q_{\sigma(2)}, Q_{\sigma(3)}))(P \circ \pi_{\sigma^{-1}})$$

$$= \mathscr{C}(\pi_{\sigma^{-1}})(\mathbf{pile'up}^{l}(Q_{\sigma(1)}, \mathbf{pile'up}^{l}(Q_{\sigma(2)}, Q_{\sigma(3)})))(P)$$

$$= \mathbf{strat}_B^{2,\sigma}(Q_1, Q_2, Q_3, P)$$

as required. ♢

Proposition 5.8 *The semantics for sentences with ditransitive verbs, as defined by strategy C, provides four asymmetric readings of the sentence, such that QP in subject position can be placed either first or last only. Thus they correspond to four out of the six readings accounted for by strategies A and B.*

Proof. In the case of sentences with ditransitive verbs, the semantics, according to strategies B and C, are defined by the morphisms $\mathbf{strat}_B^{3,\sigma}$,

$\mathbf{strat}_C^{3,\varepsilon,\varepsilon'}$, respectively. As we shall show, these morphisms are equal whenever $\sigma \in S_3$ is related to the pair $\langle \varepsilon', \varepsilon \rangle \in \{l, r\}^2$ via the relation:

$$\mathbf{pile'up}^{\varepsilon'} \circ (1 \times \mathbf{pile'up}^{\varepsilon}) = \mathscr{C}(\pi_{\sigma^{-1}}) \circ \mathbf{pile'up}^{l} \circ (1 \times \mathbf{pile'up}^{l}) \circ \pi_{\sigma}$$

As $\mathbf{pile'up}^{l}$ leaves the order intact and $\mathbf{pile'up}^{r}$ swaps the order, we can see that we have the following correspondence:

σ	$\langle \varepsilon', \varepsilon \rangle$
$(1,2,3)$	$\langle l, l \rangle$
$(1,3,2)$	$\langle l, r \rangle$
$(2,3,1)$	$\langle r, l \rangle$
$(3,2,1)$	$\langle r, r \rangle$
$(2,1,3)$	–
$(3,1,2)$	–

Thus we shall assume that σ is related to the pair $\langle \varepsilon, \varepsilon' \rangle$, and, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, $Q_3 \in \mathscr{C}(X_3)$, and $P \in \mathscr{P}(X_1 \times X_2 \times X_3)$, we obtain:[9]

$\mathbf{strat}_C^{3,\varepsilon',\varepsilon} =$

$= ev_{id_t} \circ \mathbf{CPS}^{\varepsilon'}(\mathbf{eps}^{r}{}_{X_1}) \circ (1 \times \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times 1 \times \mathbf{CPS}^{?}(\mathbf{eps}^{r}{}_{X_3}))$
$\quad \circ (1 \times 1 \times 1 \times \eta)$
$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathbf{pile'up}^{\varepsilon'} \circ (\mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times \mathbf{pile'up}^{\varepsilon})$
$\quad \circ (\mathscr{C}(1) \times \mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_3})) \circ (1 \times 1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times 1 \times \eta)$
$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ (\mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2})) \circ \mathbf{pile'up}^{\varepsilon'} \circ (\mathscr{C}(1) \times \mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_3}))$
$\quad \circ (1 \times \mathbf{pile'up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times 1 \times \eta)$
$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ (\mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2})) \circ (\mathscr{C}(1 \times 1 \times \mathbf{eps}^{r}{}_{X_3})) \circ \mathbf{pile'up}^{\varepsilon'}$
$\quad \circ (1 \times \mathbf{pile'up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times 1 \times \eta)$
$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile'up}^{\varepsilon'}$
$\quad \circ (1 \times \mathbf{pile'up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times 1 \times \eta)$

[9] The diagram illustrating these calculations would again be too big to fit on a page, but the reader is encouraged to draw one.

$$
\begin{aligned}
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{\varepsilon'} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&\overset{*}{=} ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{?} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \circ (\pi_\sigma \times 1) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times \times \eta) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times 1 \times \eta) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (1 \times \eta) \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{st}^{l\,l} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^r{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{st}^{l\,l} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= \mathbf{mos}^{l}_{X_1 \times X_2 \times X_3} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_\sigma \times 1) \\
&= \mathbf{strat}^{3,\sigma}_{B}
\end{aligned}
$$

In the above calculations, we used the definition of **CPS**es, the naturality of **pile′up**s (four times in three non-consecutive steps!), the relations between **eps** morphisms, the associativity of **pile′up**s (Proposition 5.3), the relations between σ and $\langle \varepsilon', \varepsilon \rangle$ in the equation marked with $* \overset{*}{=}$, the properties of product morphisms (three consecutive steps), the **pile′up** lemma, the naturality of strength, and finally, Lemma 5.5. $\diamondsuit$

6 ACKNOWLEDGMENTS

This article is funded by the National Science Center on the basis of decision DEC-2016/23/B/HS1/00734. The authors would like to thank the anonymous reviewers for valuable comments.

REFERENCES

Chris BARKER (2002), Continuations and the nature of quantification, *Natural language semantics*, 10(3):211–242.

Chris BARKER and Chung-chieh SHAN (2014), *Continuations and natural language*, volume 53, Oxford Studies in Theoretical Linguistics.

Daisuke BEKKI and Kenichi ASAI (2009), Representing covert movements by delimited continuations, in *JSAI International Symposium on Artificial Intelligence*, pp. 161–180, Springer.

Simon CHARLOW (2014), *On the semantics of exceptional scope*, Ph.D. thesis, New York University.

Noam CHOMSKY (1993), *Lectures on government and binding: The Pisa lectures*, 9, Walter de Gruyter.

Robin COOPER (1983), *Quantification and semantic theory*, Dordrecht: Reidel.

Philippe DE GROOTE (2001), Type raising, continuations, and classical logic, in *Proceedings of the thirteenth Amsterdam Colloquium*, pp. 97–101.

Samuel EILENBERG and G Max KELLY (1966), Closed categories, in *Proceedings of the Conference on Categorical Algebra*, pp. 421–562, Springer.

Samuel EILENBERG, John C MOORE, *et al.* (1965), Adjoint functors and triples, *Illinois Journal of Mathematics*, 9(3):381–398.

Roger GODEMENT (1958), *Topologie algébrique et théorie des faisceaux*, volume 13, Hermann Paris.

Justyna GRUDZINSKA and Marek ZAWADOWSKI (2016), Continuation semantics for multi-quantifier sentences: operation-based approaches, *arXiv preprint arXiv:1608.00255*.

Herman HENDRIKS (1993), *Studied flexibility: Categories and types in syntax and semantics*, Institute for Logic, Language and Computation.

Edward L KEENAN (1987), Unreducible n-ary quantifiers in natural language, in *Generalized quantifiers*, pp. 109–150, Springer.

Edward L KEENAN (1992), Beyond the Frege boundary, *Linguistics and Philosophy*, 15(2):199–221.

Oleg KISELYOV and Chung-chieh SHAN (2014), Continuation hierarchy and quantifier scope, in *Formal Approaches to Semantics and Pragmatics*, pp. 105–134, Springer.

Heinrich KLEISLI (1965), Every standard construction is induced by a pair of adjoint functors, *Proceedings of the American Mathematical Society*, 16(3):544–546.

Anders KOCK (1970), Monads on symmetric monoidal closed categories, *Archiv der Mathematik*, 21(1):1–10.

Anders KOCK (1971), Closed categories generated by commutative monads, *Journal of the Australian Mathematical Society*, 12(04):405–424.

Anders KOCK (1972), Strong functors and monoidal monads, *Archiv der Mathematik*, 23(1):113–120.

Robert MAY (1978), *The grammar of quantification.*, Ph.D. thesis, Massachusetts Institute of Technology.

Robert MAY (1985), *Logical Form: Its structure and derivation*, volume 12, MIT press.

Eugenio MOGGI (1991), Notions of computation and monads, *Information and computation*, 93(1):55–92.

Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in *Approaches to natural language*, pp. 221–242, Springer.

Chung-chieh SHAN (2002), Monads for natural language semantics, *arXiv preprint cs/0205026*.

Johan VAN BENTHEM (1989), Polyadic quantifiers, *Linguistics and Philosophy*, 12(4):437–464.

Philip WADLER (1990), Comprehending monads, in *Proceedings of the 1990 ACM conference on LISP and functional programming*, pp. 61–78, ACM.

Marek ZAWADOWSKI (1989), Formalization of the feature system in terms of preorders, *Feature System for Quantification Structures in Natural Language [3]*, pp. 155–175.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.
http://creativecommons.org/licenses/by/3.0/