Unifying local and nonlocal modelling of respective and symmetrical predicates

Yusuke Kubota[1] and Robert Levine[2]

[1] Ohio State University, JSPS <kubota.7@osu.edu>
[2] Ohio State University <levine@ling.ohio-state.edu>

Abstract. We propose a unified analysis of 'respective' readings of plural and conjoined expressions and the internal readings of symmetrical predicates such as *same* and *different*. The two problems have both been recognized as significant challenges in the literature of syntax and semantics, but so far there is no analysis which captures their close parallel via some uniform mechanism. In fact, the representative compositional analyses of the two phenomena in the current literature (Gawron and Kehler (2004) (G&K) on 'respective' readings and Barker (2007) on symmetrical predicates) look superficially quite different from each other, where one (Barker) employs a movement-like nonlocal mechanism for mediating the dependency between the relevant terms whereas the other (G&K) achieves a similar effect via a chain of local composition operations.
In this paper, we first point out the parallels and interactions between the two phenomena that motivate a unified analysis. We then briefly review G&K's and Barker's analyses and show that the G&K-style analysis can be modelled by the Barker-style analysis once we formulate the relevant rules within an explicit syntax-semantics interface couched in a variant of Type-Logical Categorial Grammar called Hybrid TLCG. After clarifying the hitherto unnoticed formal relations between the Barker-style nonlocal modelling and the G&K-style local modelling by focusing on the analysis of 'respective' readings, we present our unified analysis of 'respective' readings and symmetrical predicates and show how their parallel behaviors and interactions can be systematically accounted for.

Keywords: 'respective' reading, symmetrical predicate, categorial grammar, Hybrid Type-Logical Categorial Grammar, coordination, parasitic scope

1 Introduction

The so-called 'respective' readings of plural and conjoined expressions and the internal readings of symmetrical predicates such as *same* and *different* as in (1) have posed difficult challenges to theories of the syntax-semantics interface.

(1) a. John and Bill married Mary and Sue, (respectively).
(= 'John married Mary and Bill married Sue.')
b. John and Bill bought the same book.
(= 'There is a single identical book which both John and Bill bought.')

These phenomena interact with coordination, including the 'noncanonical' types of coordination (both Right-Node Raising and Dependent Cluster Coordination):

(2) a. John read, and Bill reviewed, *Barriers* and *LGB*, (respectively).
    b. John introduced the same girl to Chris on Thursday and (to) Peter on Friday.

Moreover, these expressions can themselves be iterated and interact with one another to induce multiple dependencies:

(3) a. John and Bill introduced Mary and Sue to Chris and Pat (respectively).
    b. John and Bill gave the same book to the same man.
    c. John and Bill gave the same book to Mary and Sue (respectively).

Any adequate analysis of these phenomena needs to account for these empirical facts. In particular, the parallel between the phenomena in the multiple dependency cases in (3), especially, the interdependency between 'respective' and symmetrical predicates in (3c), raises the interesting possibility that the same (or a similar) mechanism is at the core of the semantics of these two phenomena.

The present paper has two inter-related goals, one empirical and the other theoretical. The empirical goal is to develop an explicit analysis of 'respective' and symmetrical predicates that systematically accounts for the empirical facts just reviewed above. In particular, we argue that the core mechanism underlying both 'respective' and symmetrical predicates is a pairwise predication that establishes a one-to-one correspondence between elements of two ordered sets of denotata each associated with plural, conjoined or symmetrical terms (i.e. expressions like *the same man*). Formally, we treat such 'ordered sets' by means of tuples, enriching the semantic ontology slightly by introducing product-type elements for semantic objects of any arbitrary type. This enables us to formulate a unified analysis of these phenomena that immediately accounts for the complex yet systematic empirical facts noted above.

The theoretical goal of the paper is to explicitly establish a (hidden) connection between two representative compositional analyses of these phenomena proposed by previous authors: Gawron and Kehler (2004) (G&K) on 'respective' readings and Barker (2007) on symmetrical predicates. G&K's analysis builds on the idea of recursively assigning a tuple-like object as the denotation of a phrase containing a plural or a conjoined term at each step of local semantic composition, so that the ordering inherent in the original conjoined or plural term is retained in the larger structure which undergoes pairwise predication. By contrast, Barker (2007) proposes to analyze the semantics of symmetrical predicates in terms of a nonlocal, movement-like process of 'parasitic scope' whereby the symmetrical term (*the same book*) and the plural NP (*John and Bill*) that is related to it are scoped out of their local positions and are treated essentially as an interdependent complex quantifier.

While the strictly local approach by G&K and the nonlocal approach by Barker look superficially quite different, the effects of the two types of operations (or series of operations) that they respectively invoke are rather similar: they both allow one to establish some correspondence between the internal structures of two terms that do not necessarily appear adjacent to each other in the surface form of the sentence. The main difference is *how* this correspondence is established: G&K opt for a series of local composition operations (somewhat reminiscent of the way long-distance dependencies are handled in lexicalist frameworks such as CCG and G/HPSG), whereas Barker does it by a single step of nonlocal mechanism (in a way analogous to a movement-based analysis of long-distance dependencies). But then, is it just an accident that G&K and Barker proposed their respective solutions for the phenomena they were dealing with, or do we need both types of approach, but for different phenomena, or can we unify the two approaches somehow?

We attempt to shed some light on these questions by simulating G&K's and Barker's approaches in Hybrid Type-Logical Categorial Grammar (Hybrid TLCG), a variant of categorial grammar that is notable for its flexible and systematic syntax-semantics interface (Kubota and Levine, 2012, 2013; Kubota, to appear). A comparison of the two approaches in this setting reveals that the G&K-style local modelling of 'respective' predication can be modelled by the Barker-style approach once we recognize one independently needed mechanism for dealing with (non-'respective') distributive predication. We take this result to be highly illuminating, as it once again shows that the logic-based setup of TLCG enables us to gain a deeper insight into the underlying connections between two related empirical phenomena and two apparently different but deeply related approaches to each, which, without such a perspective, may have gone unnoticed.

2 Modelling 'respectively' readings locally and nonlocally

We start by briefly reviewing the key components of G&K's and Barker's analyses. In order to facilitate the comparison (both to each other and to the unified analysis that we present below), we replace sums in their analyses that model complex structured objects with the notion of tuples (which has inherent ordering of elements), but nothing essential in their respective analyses are lost by this adjustment.

2.1 Local modelling of 'respective' readings by Gawron and Kehler (2004)

G&K propose to analyze 'respective' readings of sentences like the following via a recursive application of 'respective' and distributive operators:

(4) John and Bill married Mary and Sue, (respectively).

Since they assume a simple phrase structure grammar for syntax, we model it via a simple AB grammar, with the following two rules of /E and \E alone:

(5) a. **Forward Slash Elimination**

$$\frac{a;\ \mathcal{F};\ A/B \quad b;\ \mathcal{G};\ B}{a \circ b;\ \mathcal{F}(\mathcal{G});\ A}\,/\mathrm{E}$$

b. **Backward Slash Elimination**

$$\frac{b;\ \mathcal{G};\ B \quad a;\ \mathcal{F};\ B\backslash A}{b \circ a;\ \mathcal{F}(\mathcal{G});\ A}\,\backslash\mathrm{E}$$

As noted above, we replace their sum-based treatment with a tuple-based treatment, where the two NPs *John and Bill* and *Mary and Sue* both denote tuples (or pairs) of individuals $\langle \mathbf{j}, \mathbf{b} \rangle$ and $\langle \mathbf{m}, \mathbf{s} \rangle$.[3]

The core (empty) semantic operators that G&K exploit are the following dist(ributive) and resp(ective) operators:

(6) $\varepsilon;\ \lambda P \lambda g. \prod_i^n P(\pi_i(g));\ \mathrm{X/X}$

(7) $\varepsilon;\ \lambda F \lambda x. \prod_i^n \pi_i(F)(\pi_i(x));\ \mathrm{X/X}$

There is in addition the following 'boolean reduction' operator which takes a tuple of propositions and returns their boolean conjunction at the S level:

(8) $\varepsilon;\ \lambda p. \bigwedge_i \pi_i(p);\ \mathrm{S|S}$

We can analyze (4) as follows:

(9)

$$\dfrac{\begin{array}{l}\varepsilon;\\ \lambda p. \bigwedge_i \pi_i(p);\\ \mathrm{S|S}\end{array} \quad \begin{array}{c}\dfrac{\begin{array}{l}\mathsf{john} \circ \\ \mathsf{and} \circ \\ \mathsf{bill};\\ \langle \mathbf{j}, \mathbf{b}\rangle;\\ \mathrm{NP}\end{array} \quad \dfrac{\begin{array}{l}\varepsilon;\\ \lambda F \lambda x.\\ \prod_i^n \pi_i(F)(\pi_i(x));\\ \mathrm{X/X}\end{array} \quad \begin{array}{c}\dfrac{\dfrac{\begin{array}{l}\varepsilon;\\ \lambda P \lambda g.\\ \prod_i^n P(\pi_i(g));\\ \mathrm{X/X}\end{array} \quad \begin{array}{l}\mathsf{married};\\ \mathbf{marry};\\ (\mathrm{NP\backslash S})/\mathrm{NP}\end{array}}{\begin{array}{l}\mathsf{married};\\ \lambda g. \prod_i^n \mathbf{marry}(\pi_i(g));\\ (\mathrm{NP\backslash S})/\mathrm{NP}\end{array}} \quad \begin{array}{l}\mathsf{mary} \circ \\ \mathsf{and} \circ \\ \mathsf{sue};\\ \langle \mathbf{m}, \mathbf{s}\rangle;\\ \mathrm{NP}\end{array}}{\begin{array}{l}\mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \prod_i^n \mathbf{marry}(\pi_i(\langle \mathbf{m}, \mathbf{s}\rangle)); \mathrm{NP\backslash S}\end{array}} \\ \hdashline \begin{array}{l}\mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \langle \mathbf{marry}(\mathbf{m}), \mathbf{marry}(\mathbf{s})\rangle; \mathrm{NP\backslash S}\end{array}\end{array}}{\begin{array}{l}\mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \lambda x. \prod_i^n \pi_i(\langle \mathbf{marry}(\mathbf{m}), \mathbf{marry}(\mathbf{s})\rangle)(\pi_i(x)); \mathrm{NP\backslash S}\end{array}}}{\begin{array}{l}\mathsf{john} \circ \mathsf{and} \circ \mathsf{bill} \circ \mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \prod_i^n \pi_i(\langle \mathbf{marry}(\mathbf{m}), \mathbf{marry}(\mathbf{s})\rangle)(\pi_i(\langle \mathbf{j}, \mathbf{b}\rangle)); \mathrm{S}\end{array}} \\ \hdashline \begin{array}{l}\mathsf{john} \circ \mathsf{and} \circ \mathsf{bill} \circ \mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \langle \mathbf{marry}(\mathbf{m})(\mathbf{j}), \mathbf{marry}(\mathbf{s})(\mathbf{b})\rangle; \mathrm{S}\end{array}\end{array}}{\begin{array}{l}\mathsf{john} \circ \mathsf{and} \circ \mathsf{bill} \circ \mathsf{married} \circ \mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\\ \mathbf{marry}(\mathbf{m})(\mathbf{j}) \wedge \mathbf{marry}(\mathbf{s})(\mathbf{b}); \mathrm{S}\end{array}}$$

The derivation in (11) for (10) illustrates a more complex case involving a recursive application of the 'respective' operator (here, **dist** and **resp** abbreviate the semantic translations of the two operators in (6) and (7)).

[3] This also removes G&K's ontological commitment of taking propositions rather than worlds as primitives (which is necessary for them since sums of two extensionally identical properties in the Montagovian setup collapse to a single property). While such a position is not necessarily implausible, we do not think that the semantics of respective readings should be taken to form a basis for this ontological choice.

(10) John and Mary drove to Berkeley and Santa Cruz on Monday and Tuesday.

(11)

$$\dfrac{\varepsilon;\ \mathbf{dist};\ \mathrm{X/X} \qquad \mathsf{drove};\ \mathbf{drive};\ \mathrm{VP/PP}}{\mathsf{drove};\ \lambda g. \prod_i^n \mathbf{drive}(\pi_i(g));\ \mathrm{VP/PP}}$$

$$\dfrac{\mathsf{drove};\ \lambda g. \prod_i^n \mathbf{drive}(\pi_i(g));\ \mathrm{VP/PP} \qquad \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc};\ \langle \mathbf{b}, \mathbf{s} \rangle;\ \mathrm{PP}}{\mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc};\ \langle \mathbf{drive}(\mathbf{b}), \mathbf{drive}(\mathbf{s}) \rangle;\ \mathrm{VP}}$$

$$\dfrac{\varepsilon;\ \mathbf{dist};\ \mathrm{X/X} \qquad \mathsf{on};\ \mathbf{on};\ (\mathrm{VP}\backslash\mathrm{VP})/\mathrm{NP}}{\mathsf{on};\ \lambda g. \prod_i^n \mathbf{on}(\pi_i(g));\ (\mathrm{VP}\backslash\mathrm{VP})/\mathrm{NP}}$$

$$\dfrac{\mathsf{on};\ \lambda g. \prod_i^n \mathbf{on}(\pi_i(g));\ (\mathrm{VP}\backslash\mathrm{VP})/\mathrm{NP} \qquad \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \langle \mathbf{m}, \mathbf{t} \rangle;\ \mathrm{NP}}{\mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \langle \mathbf{on}(\mathbf{m}), \mathbf{on}(\mathbf{t}) \rangle;\ \mathrm{VP}\backslash\mathrm{VP}}$$

$$\dfrac{\varepsilon;\ \mathbf{resp};\ \mathrm{X/X} \qquad \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \langle \mathbf{on}(\mathbf{m}), \mathbf{on}(\mathbf{t}) \rangle;\ \mathrm{VP}\backslash\mathrm{VP}}{\mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \lambda g. \prod_i^n \pi_i(\langle \mathbf{on}(\mathbf{m}), \mathbf{on}(\mathbf{t}) \rangle)(\pi_i(g));\ \mathrm{VP}\backslash\mathrm{VP}}$$

$$\dfrac{\mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc};\ \langle \mathbf{drive}(\mathbf{b}), \mathbf{drive}(\mathbf{s}) \rangle;\ \mathrm{VP} \qquad \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \lambda g. \prod_i^n \pi_i(\langle \mathbf{on}(\mathbf{m}), \mathbf{on}(\mathbf{t}) \rangle)(\pi_i(g));\ \mathrm{VP}\backslash\mathrm{VP}}{\mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc} \circ \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \langle \mathbf{on}(\mathbf{m})(\mathbf{drive}(\mathbf{b})), \mathbf{on}(\mathbf{t})(\mathbf{drive}(\mathbf{s})) \rangle;\ \mathrm{VP}}$$

$$\dfrac{\varepsilon;\ \mathbf{resp};\ \mathrm{X/X} \qquad \mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc} \circ \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \langle \mathbf{on}(\mathbf{m})(\mathbf{drive}(\mathbf{b})), \mathbf{on}(\mathbf{t})(\mathbf{drive}(\mathbf{s})) \rangle;\ \mathrm{VP}}{\mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc} \circ \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \lambda x. \prod_i^n \pi_i(\langle \mathbf{on}(\mathbf{m})(\mathbf{drive}(\mathbf{b})), \mathbf{on}(\mathbf{t})(\mathbf{drive}(\mathbf{s})) \rangle)(\pi_i(x));\ \mathrm{VP}}$$

$$\dfrac{\mathsf{john} \circ \mathsf{and} \circ \mathsf{mary};\ \langle \mathbf{j}, \mathbf{m} \rangle;\ \mathrm{NP} \qquad \mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc} \circ \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \lambda x. \prod_i^n \pi_i(\langle \mathbf{on}(\mathbf{m})(\mathbf{drive}(\mathbf{b})), \mathbf{on}(\mathbf{t})(\mathbf{drive}(\mathbf{s})) \rangle)(\pi_i(x));\ \mathrm{VP}}{\mathsf{john} \circ \mathsf{and} \circ \mathsf{mary} \circ \mathsf{drove} \circ \mathsf{to} \circ \mathsf{bkl} \circ \mathsf{and} \circ \mathsf{sc} \circ \mathsf{on} \circ \mathsf{mon} \circ \mathsf{and} \circ \mathsf{tue};\ \prod_i^n \pi_i(\langle \mathbf{on}(\mathbf{m})(\mathbf{drive}(\mathbf{b})), \mathbf{on}(\mathbf{t})(\mathbf{drive}(\mathbf{s})) \rangle)(\pi_i(\langle \mathbf{j}, \mathbf{m} \rangle));\ \mathrm{S}}$$

Note that at each step where a functor takes a product-type term as an argument, the dist operator is first applied to the functor so that the functor is distributively applied to each member of the tuple and the result is 'summed up' as a tuple (rather than conjoined by a generalized conjunction operator as in the standard definition of the distributive operator in the plurality literature). Thus, the larger constituent inherits the ordering of elements in its subconstituent.

Another notable property of G&K's analysis is that after the application of the resp operator, the larger constituent still denotes a tuple (of two properties of type $e \to t$, in the case of (11)), rather than boolean conjunction. This is crucial for making the recursive application of the resp operator straightforward. Since the tuple structure is preserved after the application of the first resp operator, the result can simply be taken up by another resp operator which relates it to another tuple in a 'respective' manner.

Although G&K does not discuss this point explicitly, in order to generalize this analysis to cases like the following in which the tuple structure is percolated from the functor rather than the argument, one either needs to assume that type-raising is generally available in the grammar so that the functor-argument relation of any arbitrary pair of functor and argument types can be flipped, or else needs to introduce another version of the dist operator, call it dist', which distributes a single argument meaning to a tuple of functor meanings.[4]

(12) a. John and Bill read and reviewed the book, respectively.

[4] G&K speculate on a possibility of unifying their dist and resp operators toward the end of their paper; if this unification is successfully done, both the argument-distributing dist operator in (6) above and the functor-distributing dist' operator under discussion here might be thought of as special cases of a single unified 'predication' operator. But this part of their proposal remains somewhat obscure and not worked out in full detail.

b. John and Bill sent the bomb and the letter to the president yesterday, respectively.

Essentially, at the expense of applying either the dist or resp operator at each step of local composition, G&K does away with hypothetical reasoning entirely and their fragment can be modelled by a simple AB grammar.

2.2 Nonlocal modeling of 'respective' readings building on Barker (2007)

In contrast to G&K, Barker (2007) extensively relies on hypothetical reasoning for characterizing the semantics of symmetrical predicates. In order to facilitate a comparison with G&K's analysis, we first discuss an extension of Barker's approach to 'respective' readings (it should be noted that Barker himself confines his analysis to the case of symmetrical predicates, mostly focusing on the analysis of *same*), and come back to the case of symmetrical predicates in the next section.

The key idea behind Barker's proposal is that the interdependency between the relevant two complex terms (i.e. the two plural or conjoined terms in the case of 'respective' readings) can be straightforwardly mediated by abstracting over the positions in the sentence that such terms occupy and then directly giving the relevant terms (and the abstracted proposition) as arguments to the operator that mediates their interdependency.

For modelling this 'covert' movement treatment of 'respective'/symmetrical predicates, we introduce here a new connective |, called 'vertical slash', together with the Elimination and Introduction rules for it formulated in (21) (just like /, we write the argument to the right for this slash; thus, in $A|B$, B is the argument).[5]

(13) a. **Vertical Slash Introduction**

$$\dfrac{\dfrac{\begin{matrix}\vdots & \vdots & \dfrac{}{[\varphi;\, x;\, A]^n} & \vdots & \vdots \\ \vdots & \vdots & \vdots\;\;\vdots & \vdots & \vdots\end{matrix}}{\mathsf{b};\; \boldsymbol{\mathcal{F}};\; B}}{\lambda\varphi.\mathsf{b};\; \lambda x.\boldsymbol{\mathcal{F}};\; B|A}\;|\mathrm{I}^n$$

b. **Vertical Slash Elimination**

$$\dfrac{\mathsf{a};\; \boldsymbol{\mathcal{F}};\; A|B \quad \mathsf{b};\; \boldsymbol{\mathcal{G}};\; B}{\mathsf{a}(\mathsf{b});\; \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{G}});\; A}\;|\mathrm{E}$$

These rules are essentially the same as the rules for the linear implication connective ($\multimap$) posited in the family of 'Linear Categorial Grammars' (Oehrle, 1994; de Groote, 2001; Muskens, 2003; Mihaliček and Pollard, 2012).

[5] These rules introduce functional prosodic objects. One might wonder how the grammar (or the prosodic calculus that is part of it) is constrained in such a way that it does not admit of uninterpretable prosodic objects such as $\mathsf{john} \circ \lambda\varphi.\varphi$ (i.e., 'concatenation' of a string and a function from strings to strings). In fact, Hybrid TLCG does not admit of any such ill-formed prosodic objects. Such an expression would be obtained only by applying a functor that has a syntactic type of the form X/(Y|Z) to its argument Y|Z, but a syntactic type of the form X/(Y|Z), with the vertical slash 'under' a directional slash, are explicitly excluded from the grammar. For the details of the syntax-prosody mapping which ensures this, see Kubota and Levine (2014).

With this vertical slash, extending Barker's 'parasitic scope' analysis to 'respective' readings is in fact mostly straightforward, with one extra complication discussed below. Assuming (as above) that plural and conjoined terms denote tuples (of the relevant type of semantic objects), we just need the following three-place 'respective' operator which semantically takes a relation (denoted by the sentence containing the two 'gap' positions for the two product-type terms) and two tuples as arguments and returns a tuple as an output (this is so that, as above, multiple 'respective' readings can be handled by recursive application of this operator).

(14) $\lambda\sigma_0\lambda\varphi_1\lambda\varphi_2.\sigma_0(\varphi_1)(\varphi_2)$; **resp3**; $(Z|X|Y)|(Z|X|Y)$

As can be seen in (14), the resp operator is a (polymorphic) identity function both syntactically and prosodically. The semantics is unpacked in (15).

(15) $\mathbf{resp3} = \lambda\mathcal{R}\lambda\mathcal{T}_\times\lambda\mathcal{U}_\times.\prod_i \mathcal{R}(\pi_i(\mathcal{T}_\times))(\pi_i(\mathcal{U}_\times))$

Semantically, this operator relates the elements of the two tuples in a pairwise manner with respect to the relation in question. Note that this three-place **resp3** operator is distinct from the two place **resp** operator posited in G&K's approach though their semantic effects are similar. We come back to the relationship between these two operators immediately below (see (19)).

The analysis of the simple 'respective' sentence is then straightforward:

(16)

$$\mathsf{married};\ \mathbf{marry};\ (\mathrm{NP}\backslash\mathrm{S})/\mathrm{NP}$$

$$\vdots \quad \vdots$$

$$\lambda\varphi_3\lambda\varphi_4.\varphi_4 \circ \mathsf{married} \circ \varphi_3;\ \mathbf{marry};\ (\mathrm{S}|\mathrm{NP})|\mathrm{NP}$$

$$\lambda\sigma_0\lambda\varphi_1\lambda\varphi_2.\sigma_0(\varphi_1)(\varphi_2);\ \mathbf{resp3};\ (Z|X|Y)|(Z|X|Y)$$

$$\lambda\varphi_1\lambda\varphi_2.\varphi_2 \circ \mathsf{married} \circ \varphi_1;\ \mathbf{resp3}(\mathbf{marry});\ \mathrm{S}|\mathrm{NP}|\mathrm{NP}$$

$$\mathsf{john} \circ \mathsf{and} \circ \mathsf{bill};\ \langle \mathbf{j}, \mathbf{b}\rangle;\ \mathrm{NP}$$

$$\lambda\varphi_2.\varphi_2 \circ \mathsf{married} \circ \mathsf{john} \circ \mathsf{and} \circ \mathsf{bill};\ \mathbf{resp3}(\mathbf{marry})(\langle \mathbf{j}, \mathbf{b}\rangle);\ \mathrm{S}|\mathrm{NP}$$

$$\mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue};\ \langle \mathbf{m}, \mathbf{s}\rangle;\ \mathrm{NP}$$

$$\mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue} \circ \mathsf{married} \circ \mathsf{john} \circ \mathsf{and} \circ \mathsf{bill};\ \mathbf{resp3}(\mathbf{marry})(\langle \mathbf{j}, \mathbf{b}\rangle)(\langle \mathbf{m}, \mathbf{s}\rangle);\ \mathrm{S}$$

$$\mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue} \circ \mathsf{married} \circ \mathsf{john} \circ \mathsf{and} \circ \mathsf{bill};\ \langle \mathbf{marry}(\mathbf{j})(\mathbf{m}), \mathbf{marry}(\mathbf{b})(\mathbf{s})\rangle;\ \mathrm{S}$$

$$\lambda\varphi_1.\varphi_1;\ \lambda p.\bigwedge_i \pi_i(p);\ \mathrm{S}|\mathrm{S}$$

$$\mathsf{mary} \circ \mathsf{and} \circ \mathsf{sue} \circ \mathsf{married} \circ \mathsf{john} \circ \mathsf{and} \circ \mathsf{bill};\ \mathbf{marry}(\mathbf{j})(\mathbf{m}) \wedge \mathbf{marry}(\mathbf{b})(\mathbf{s});\ \mathrm{S}$$

Just as in G&K's analysis, multiple 'respective' readings in examples like the following are obtained via recursive application of the **resp** operator:

(17) Tolstoy and Dostoevsky sent Anna Karenina and the Idiot to Dickens and Thackeray (respectively).

The analysis is in fact straightforward. After two of the tuple-denoting terms are related to each other with respect to the predicate denoted by the verb, the resultant S|NP expression is a tuple of two properties.

(18)

$$
\dfrac{
\begin{array}{l}\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \langle\mathbf{di},\mathbf{th}\rangle;\mathrm{NP}\end{array}
\quad
\dfrac{
\begin{array}{l}\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id};\\ \langle\mathbf{ak},\mathbf{id}\rangle;\mathrm{NP}\end{array}
\quad
\dfrac{
\begin{array}{l}\lambda\sigma_0\lambda\varphi_1\lambda\varphi_2.\sigma_0(\varphi_1)(\varphi_2);\\ \mathbf{resp3};\\ (Z|X|Y)|(Z|X|Y)\end{array}
\quad
\begin{array}{l}\vdots\quad\vdots\\ \lambda\varphi_1\lambda\varphi_2\lambda\varphi_3.\\ \varphi_3\circ\mathsf{sent}\circ\varphi_1\circ\mathsf{to}\circ\varphi_2;\\ \mathbf{send};\mathrm{S|NP|NP|NP}\end{array}
}{
\begin{array}{l}\lambda\varphi_1\lambda\varphi_2\lambda\varphi_3.\varphi_3\circ\mathsf{sent}\circ\varphi_1\circ\mathsf{to}\circ\varphi_2;\\ \mathbf{resp3(send)};\mathrm{S|NP|NP|NP}\end{array}
}
}{
\begin{array}{l}\lambda\varphi_2\lambda\varphi_3.\varphi_3\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\varphi_2;\\ \mathbf{resp3(send)}(\langle\mathbf{ak},\mathbf{id}\rangle);\mathrm{S|NP|NP}\end{array}
}
}{
\begin{array}{l}\lambda\varphi_3.\varphi_3\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \mathbf{resp3(send)}(\langle\mathbf{ak},\mathbf{id}\rangle)(\langle\mathbf{di},\mathbf{th}\rangle);\mathrm{S|NP}\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \lambda\varphi_3.\varphi_3\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \langle\mathbf{send(ak)(di)},\mathbf{send(id)(th)}\rangle;\mathrm{S|NP}\end{array}
}
$$

And the remaining conjoined term $\langle\mathbf{to},\mathbf{do}\rangle$ is related to this product-type property in the following way:

(19)

$$
\dfrac{
\begin{array}{l}\lambda\rho\lambda\sigma\lambda\varphi.\rho(\sigma)(\varphi);\\ \lambda\mathcal{R}\lambda\mathcal{T}_\times\lambda\mathcal{U}_\times.\prod_i\mathcal{R}(\pi_i(\mathcal{T}_\times))(\pi_i(\mathcal{U}_\times));\\ (Z|X|Y)|(Z|X|Y)\end{array}
\quad
\dfrac{
\dfrac{
\dfrac{[\sigma;f;\mathrm{S|NP}]^1\quad[\varphi;x;\mathrm{NP}]^2}{\sigma(\varphi);\ f(x);\ \mathrm{S}}\;|\mathrm{E}
}{\lambda\varphi.\sigma(\varphi);\ \lambda x.f(x);\ \mathrm{S|NP}}\;|\mathrm{I}^2
}{\lambda\sigma\lambda\varphi.\sigma(\varphi);\ \lambda f\lambda x.f(x);\ (\mathrm{S|NP})|(\mathrm{S|NP})}\;|\mathrm{I}^1
}{
\lambda\sigma_1\lambda\varphi_1.\sigma_1(\varphi_1);\ \lambda P_\times\lambda X_\times.\prod_i\pi_i(P_\times)(\pi_i(X_\times));\ (\mathrm{S|NP})|(\mathrm{S|NP})
}\;|\mathrm{E}
$$

$$
\dfrac{
\begin{array}{l}\mathsf{To}\circ\mathsf{and}\circ\mathsf{Do};\\ \langle\mathbf{to},\mathbf{do}\rangle;\\ \mathrm{NP}\end{array}
\quad
\dfrac{
\begin{array}{l}\vdots\quad\vdots\\ \lambda\sigma_2\lambda\varphi_2.\sigma_2(\varphi_2);\\ \lambda P_\times\lambda X_\times.\\ \quad\prod_i\pi_i(P_\times)(\pi_i(X_\times));\\ (\mathrm{S|NP})|(\mathrm{S|NP})\end{array}
\quad
\begin{array}{l}\vdots\quad\vdots\\ \lambda\varphi_3.\varphi_3\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\\ \mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \langle\mathbf{send(ak)(di)},\mathbf{send(id)(th)}\rangle;\mathrm{S|NP}\end{array}
}{
\begin{array}{l}\lambda\varphi_2.\varphi_2\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \lambda Y_\times.\prod_i\pi_i(\langle\mathbf{send(ak)(di)},\mathbf{send(id)(th)}\rangle)(\pi_i(Y_\times));\mathrm{S|NP}\end{array}
}\;|\mathrm{E}
}{
\begin{array}{l}\mathsf{To}\circ\mathsf{and}\circ\mathsf{Do}\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \prod_i\pi_i(\langle\mathbf{send(ak)(di)},\mathbf{send(i)(th)}\rangle)(\pi_i(\langle\mathbf{to},\mathbf{do}\rangle));\mathrm{S}\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ \mathsf{To}\circ\mathsf{and}\circ\mathsf{Do}\circ\mathsf{sent}\circ\mathsf{AK}\circ\mathsf{and}\circ\mathsf{Id}\circ\mathsf{to}\circ\mathsf{Di}\circ\mathsf{and}\circ\mathsf{Th};\\ \langle\mathbf{send(ak)(di)(to)},\mathbf{send(id)(th)(do)}\rangle;\mathrm{S}\end{array}
}\;|\mathrm{E}
$$

The first chunk of derivation in (19) (where X is instantiated as NP, Y as S|NP, and Z as S), the point of which may not be immediately clear, can be thought of as a way of deriving the two-place **resp** operator (identical to the one that G&K posit) from the lexically specified three-place **resp3** operator introduced above. As in G&K's analysis discussed above, the two place **resp** operator directly relates the product-type property (of type S|NP) derived in (18) with the product-type NP occupying the subject position via function application of the corresponding elements.

3 Comparison of local modelling and nonlocal modelling

We now show that the G&K-style 'local' modelling of 'respective' predication can be simulated by the Barker-style 'nonlocal' approach. Consider first a case which contains only two product-type terms to be related in a 'respective' manner. The structure of the derivation for a sentence containing two product-type terms in the G&K-style analysis can be schematically shown in (20), where $i, j, n, m \geq 0, n \geq i, m \geq j$ and $l \geq 2$ and for any k, $\boldsymbol{\gamma}_k$ or $\boldsymbol{\delta}_k$ is some linguistic sign.[6] Note here that both Ψ and Φ, which are meanings of expressions that contain exactly one tuple-denoting (lexical) term inside themselves, denote tuples, and they are then related by the two-place **resp** operator with each other.

(20)

$$\dfrac{\dfrac{\varepsilon;\ \mathbf{resp};\ \mathrm{Z/Z} \qquad \dfrac{\begin{matrix}\boldsymbol{\gamma}_1 \ldots \boldsymbol{\gamma}_i \\ \vdots\ \vdots\ \vdots\end{matrix} \quad \begin{matrix}\dfrac{\mathsf{a};\ \langle a_1, \ldots, a_l\rangle;\ \mathrm{A}}{\vdots\ \vdots}\end{matrix} \quad \begin{matrix}\boldsymbol{\gamma}_{i+1} \ldots \boldsymbol{\gamma}_n \\ \vdots\ \vdots\ \vdots\end{matrix}}{\mathsf{c};\ \Psi;\ \mathrm{X/Y}}}{\mathsf{c};\ \mathbf{resp}(\Psi);\ \mathrm{X/Y}} \qquad \dfrac{\begin{matrix}\boldsymbol{\delta}_1 \ldots \boldsymbol{\delta}_j \\ \vdots\ \vdots\ \vdots\end{matrix} \quad \begin{matrix}\dfrac{\mathsf{b};\ \langle b_1, \ldots, b_l\rangle;\ \mathrm{B}}{\vdots\ \vdots}\end{matrix} \quad \begin{matrix}\boldsymbol{\delta}_{j+1} \ldots \boldsymbol{\delta}_m \\ \vdots\ \vdots\ \vdots\end{matrix}}{\mathsf{d};\ \Phi;\ \mathrm{Y}}}{\mathsf{c} \circ \mathsf{d};\ \mathbf{resp}(\Psi)(\Phi);\ \mathrm{X}}$$

We derive two auxiliary rules in G&K's system to facilitate the comparison to the Barker-style analysis.[7]

(21) a. **Rule 1**

$$\dfrac{\mathsf{a};\ f;\ \mathrm{A/B} \quad \mathsf{b};\ \langle a_1 \ldots a_l\rangle;\ \mathrm{B}}{\mathsf{a} \circ \mathsf{b};\ \langle f(a_1) \ldots f(a_l)\rangle;\ \mathrm{A}}$$

b. **Rule 2**

$$\dfrac{\mathsf{a};\ \langle f_1 \ldots f_n\rangle;\ \mathrm{A/B} \quad \mathsf{b};\ a;\ \mathrm{B}}{\mathsf{a} \circ \mathsf{b};\ \langle f_1(a) \ldots f_n(a)\rangle;\ \mathrm{A}}$$

Rule 1 is obtained by applying the dist operator (6) to the functor f and then applying it to its tuple argument. Rule 2 is obtained by applying the dist' operator discussed above (see the discussion pertaining to (12)) to the argument a and applying it to the tuple functor. (We remain agnostic about how dist' is obtained in G&K's setup.) These two rules are introduced here just for expository ease. We show below how they can be derived from the more general **resp3** operator in the present setup with the use of hypothetical reasoning by introducing one auxiliary rule converting an atomic object to an n-tuple of identical objects.

By assumption, among the signs $\boldsymbol{\gamma}_1 \ldots \boldsymbol{\gamma}_n$, $\boldsymbol{\delta}_1 \ldots \boldsymbol{\delta}_n$, and a and b constituting the leaves of (20), only a and b have product-type meanings. Thus, at each step of local composition inside c and d, either the functor or the argument (but not both) has a product-type meaning. From this it further follows that each local step of composition inside c and d instantiates either Rule 1 or Rule 2.

Now, consider a structure in which we replace the two product-type terms in (20) by the variables x and y, both fresh in Ψ and Φ.

[6] We assume here that the lefthand substructure is the functor. The same result obtains for a structure where the righthand substructure is the functor, by merely replacing the linear order between c and d in (20).

[7] Here we are inspired by Bekki's (2006) reformulation of G&K's analysis in terms of product-types.

(22)
$$\dfrac{\dfrac{\boldsymbol{\gamma}_1 \ldots \boldsymbol{\gamma}_i \quad \dfrac{\varphi_1;\ x;\ \mathrm{A}}{\vdots\ \vdots} \quad \boldsymbol{\gamma}_{i+1} \ldots \boldsymbol{\gamma}_n}{\begin{array}{l}\mathsf{c}';\\ \Gamma; \mathrm{X/Y}\end{array}} \qquad \dfrac{\boldsymbol{\delta}_1 \ldots \boldsymbol{\delta}_j \quad \dfrac{\varphi_2;\ y;\ \mathrm{B}}{\vdots\ \vdots} \quad \boldsymbol{\delta}_{j+1} \ldots \boldsymbol{\delta}_m}{\begin{array}{l}\mathsf{d}';\\ \Delta; \mathrm{Y}\end{array}}}{\mathsf{c}' \circ \mathsf{d}';\ \Gamma(\Delta);\ \mathrm{X}}$$

The relation between the internal structures of (22) and (20) is such that each step of function application in (22) is replaced by an application of either Rule 1 or Rule 2 in (20). Thus, by induction,[8]

(23) $\Psi = \langle \Gamma[x/a_1], \ldots, \Gamma[x/a_l] \rangle$

(where $\Gamma[x/a_k]$ is a term identical to Γ except that all occurrences of x in Γ are replaced by a_k). Similarly,

(24) $\Phi = \langle \Delta[y/b_1], \ldots, \Delta[y/b_l] \rangle$

Thus,

(25)
$$\begin{array}{l}\mathbf{resp}(\Phi)(\Psi)\\ = \mathbf{resp}(\langle \Gamma[x/a_1], ., \Gamma[x/a_l] \rangle)(\langle \Delta[y/b_1], \rangle ., \Delta[y/b_l] \rangle)\\ = \langle \Gamma[x/a_1](\Delta[y/b_1]), \ldots, \Gamma[x/a_l](\Delta[y/b_l]) \rangle\end{array}$$

This is exactly the same interpretation that we obtain in the following Barker-style analysis of the same string of words:

(26)
$$\dfrac{\dfrac{\dfrac{\dfrac{\boldsymbol{\gamma}_1 \ldots \boldsymbol{\gamma}_i \quad \dfrac{\begin{bmatrix}\varphi_1;\\ x;\\ \mathrm{A}\end{bmatrix}^1}{\vdots\ \vdots} \quad \boldsymbol{\gamma}_{i+1} \ldots \boldsymbol{\gamma}_n}{\begin{array}{l}\mathsf{c}';\\ \Gamma; \mathrm{X/Y}\end{array}} \qquad \dfrac{\boldsymbol{\delta}_1 \ldots \boldsymbol{\delta}_j \quad \dfrac{\begin{bmatrix}\varphi_2;\\ y;\\ \mathrm{B}\end{bmatrix}^2}{\vdots\ \vdots} \quad \boldsymbol{\delta}_{j+1} \ldots \boldsymbol{\delta}_m}{\begin{array}{l}\mathsf{d}';\\ \Delta; \mathrm{Y}\end{array}}}{\mathsf{c}' \circ \mathsf{d}';\ \Gamma(\Delta);\ \mathrm{X}}}{\lambda\varphi_2.\mathsf{c}' \circ \mathsf{d}';\ \lambda y.\Gamma(\Delta);\ \mathrm{X|B}}\,|\mathrm{I}^2}{\lambda\varphi_1\lambda\varphi_2.\mathsf{c}' \circ \mathsf{d}';\ \lambda x \lambda y.\Gamma(\Delta);\ \mathrm{X|B|A}}\,|\mathrm{I}^1$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\ \ \vdots\\ \lambda\varphi_1\lambda\varphi_2.\mathsf{c}' \circ \mathsf{d}';\ \lambda x \lambda y.\Gamma(\Delta);\ \mathrm{X|B|A}\end{array} \quad \begin{array}{l}\lambda\varphi.\varphi;\\ \mathbf{resp3};\\ \mathrm{(Z|X|Y)|(Z|X|Y)}\end{array}}{\lambda\varphi_1\lambda\varphi_2.\mathsf{c}' \circ \mathsf{d}';\ \mathbf{resp3}(\lambda x \lambda y.\Gamma(\Delta));\ \mathrm{X|B|A}} \quad \begin{array}{l}\mathsf{a};\\ \langle a_1, ..., a_l \rangle;\\ \mathrm{A}\end{array}}{\lambda\varphi_2.\mathsf{c} \circ \mathsf{d}';\ \mathbf{resp3}(\lambda x \lambda y.\Gamma(\Delta))(\langle a_1, ..., a_l \rangle);\ \mathrm{X|B}} \quad \begin{array}{l}\mathsf{b};\\ \langle b_1, ..., b_l \rangle;\\ \mathrm{B}\end{array}}{\mathsf{c} \circ \mathsf{d};\ \mathbf{resp3}(\lambda x \lambda y.\Gamma(\Delta))(\langle a_1, ..., a_l \rangle)(\langle b_1, ..., b_l \rangle);\ \mathrm{X}}$$

The final translation we obtain in this derivation is

(27) $\mathbf{resp3}(\lambda x \lambda y.\Gamma(\Delta))(\langle a_1, \ldots, a_l \rangle)(\langle b_1, \ldots, b_l \rangle)$

[8] See Appendix for a formal proof.

Since $|$ is linear, x is fresh in Δ and y in Γ. Thus, for any k, $\lambda x \lambda y.[\Gamma(\Delta)](a_k)(b_k) = \Gamma[x/a_k](\Delta[y/b_k])$. From this it follows that

(28) $\mathbf{resp3}(\lambda x \lambda y.\Gamma(\Delta))(\langle a_1, \ldots, a_l\rangle)(\langle b_1, \ldots, b_l\rangle)$
$\quad = \langle \Gamma[x/a_1](\Delta[y/b_1]), \ldots, \Gamma[x/a_l](\Delta[y/b_l])\rangle$

For cases containing more than two respective terms, the correspondence between the G&K-style analysis and the Barker style analysis can be established recursively by taking the whole structure (20)/(26) to instantiate either a or b and relating it to the next 'adjacent' product-type term one by one.

It now remains to show how Rule 1 and Rule 2 can be derived in the Barker-style setup. For this, we need a mechanism that derives the two dist operators in the G&K setup from the three-place resp3 operator posited in the Barker system in (15). Following Bekki (2006), we assume that the following 'product duplicator' is responsible for this operation, which takes some term x and returns an n-tuple consisting of x: $\langle x, \ldots, x\rangle$:

(29) $\lambda \varphi.\varphi;\ \lambda x.\prod_i^n x;\ \mathrm{X|X}$

With this operator and the three-place resp3 operator in (15), Rule 1 and Rule 2 can be derived as follows:

(30)

$$
\begin{array}{c}
\dfrac{
\dfrac{\begin{array}{l}\lambda\varphi.\varphi;\\ \lambda x.\prod_i^l x;\\ \mathrm{X|X}\end{array} \quad \begin{array}{l}\mathsf{a};\\ f;\mathrm{A/B}\end{array}}{\mathsf{a};\ \langle f,\ldots,f\rangle;\ \mathrm{A/B}}
\quad
\dfrac{\begin{array}{l}\mathsf{b};\\ \langle a_1,\ldots,a_l\rangle;\\ \mathrm{B}\end{array} \quad
\dfrac{\begin{array}{l}\lambda\sigma_0\lambda\varphi_1\lambda\varphi_2.\\ \sigma_0(\varphi_1)(\varphi_2);\\ \mathbf{resp3};\\ (Z|X|Y)|(Z|X|Y)\end{array} \quad
\dfrac{\dfrac{\dfrac{\begin{bmatrix}\varphi_1;\\ f;\mathrm{A/B}\end{bmatrix}^1 \quad \begin{bmatrix}\varphi_2;\\ x;\mathrm{B}\end{bmatrix}^2}{\varphi_1 \circ \varphi_2;\ f(x);\ \mathrm{A}}}{\begin{array}{c}\lambda\varphi_1.\varphi_1\circ\varphi_2;\\ \lambda f.f(x);\mathrm{A|(A/B)}\end{array}}\,|\mathrm{I}^1}{\begin{array}{c}\lambda\varphi_2\lambda\varphi_1.\varphi_1\circ\varphi_2;\\ \lambda x\lambda f.f(x);\mathrm{A|(A/B)|B}\end{array}}\,|\mathrm{I}^2}
{\lambda\varphi_1\lambda\varphi_2.\varphi_2\circ\varphi_1;\ \mathbf{resp3}(\lambda x\lambda f.f(x));\ \mathrm{A|(A/B)|B}}}
{\lambda\varphi_2.\varphi_2\circ\mathsf{b};\ \mathbf{resp3}(\lambda x\lambda f.f(x))(\langle a_1,\ldots,a_l\rangle);\ \mathrm{A|(A/B)}}}
{\mathsf{a}\circ\mathsf{b};\ \mathbf{resp3}(\lambda x\lambda f.f(x))(\langle a_1,\ldots,a_l\rangle)(\langle f,\ldots,f\rangle);\ \mathrm{A}} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\mathsf{a}\circ\mathsf{b};\ \langle f(a_1),\ldots,f(a_l)\rangle;\ \mathrm{A}
\end{array}
$$

(31)

$$
\begin{array}{c}
\dfrac{\begin{array}{l}\mathsf{a};\\ \langle f_1,\ldots,f_l\rangle;\\ \mathrm{A/B}\end{array} \quad
\dfrac{\dfrac{\begin{array}{l}\lambda\varphi.\varphi;\\ \lambda x.\prod_i^l x;\\ \mathrm{X|X}\end{array} \quad \begin{array}{l}\mathsf{b};\\ a;\mathrm{B}\end{array}}{\mathsf{b};\ \langle a,\ldots,a\rangle;\ \mathrm{B}} \quad
\begin{array}{c}\vdots \quad \vdots\\ \lambda\varphi_1\lambda\varphi_2.\varphi_2\circ\varphi_1;\\ \mathbf{resp3}(\lambda x\lambda f.f(x));\mathrm{A|(A/B)|B}\end{array}}
{\lambda\varphi_2.\varphi_2\circ\mathsf{b};\ \mathbf{resp3}(\lambda x\lambda f.f(x))(\langle a,\ldots,a\rangle);\ \mathrm{A|(A/B)}}}
{\mathsf{a}\circ\mathsf{b};\ \mathbf{resp3}(\lambda x\lambda f.f(x))(\langle a,\ldots,a\rangle)(\langle f_1,\ldots,f_l\rangle);\ \mathrm{A}} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\mathsf{a}\circ\mathsf{b};\ \langle f_1(a),\ldots,f_l(a)\rangle;\ \mathrm{A}
\end{array}
$$

4 Extension of the analysis

In this section, we extend the above analysis in two ways. We first show that, by enriching the calculus with rules for hypothetical reasoning for directional slashes / and \, the interaction between 'respective' readings and nonconstituent coordination exemplified by the data such as (2) straightforwardly falls out. We then extend the tuple-based analysis to symmetrical predicates and show that this analysis immediately extends to multiple dependencies among symmetrical and 'respective' predicates observed in (3).

For the analysis of NCC, we add the following Introduction rules for directional slashes / and \:

(32) a. **Forward Slash Introduction**

$$\dfrac{\dfrac{\vdots \quad \vdots \quad \dfrac{[\varphi;\ x;\ A]^n}{\vdots \quad \vdots} \quad \vdots \quad \vdots}{\mathsf{b} \circ \varphi;\ \boldsymbol{\mathcal{F}};\ B}}{\mathsf{b};\ \lambda x.\boldsymbol{\mathcal{F}};\ B/A}\ /\mathrm{I}^n$$

b. **Backward Slash Introduction**

$$\dfrac{\dfrac{\vdots \quad \vdots \quad \dfrac{[\varphi;\ x;\ A]^n}{\vdots \quad \vdots} \quad \vdots \quad \vdots}{\varphi \circ \mathsf{b};\ \boldsymbol{\mathcal{F}};\ B}}{\mathsf{b};\ \lambda x.\boldsymbol{\mathcal{F}};\ A\backslash B}\ \backslash\mathrm{I}^n$$

In TLCG, dependent cluster coordination is analyzed by directly analyzing the apparent nonconstituents that are coordinated in examples like (33) to be a (higher-order) derived constituent, via hypothetical reasoning.

(33) I lent *Syntactic Structures* and *Barriers* to Robin on Thursday and to Mary on Friday, respectively.

Specifically, by hypothesizing the verb and the direct object and withdrawing them after a whole VP is formed, the string *to Robin on Thursday* can be analyzed as a constituent of type NP\(VP/PP/NP)\VP:

(34)
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\varphi_1; P; \mathrm{VP/PP/NP}]^1 \quad [\varphi_2; x; \mathrm{NP}]^2}{\varphi_1 \circ \varphi_2;\ P(x);\ \mathrm{VP/PP}}/\mathrm{E} \quad \begin{array}{l}\mathsf{to} \circ \mathsf{robin};\\ \mathbf{r}; \mathrm{PP}\end{array}}{\varphi_1 \circ \varphi_2 \circ \mathsf{to} \circ \mathsf{robin};\ P(x)(\mathbf{r});\ \mathrm{VP}}/\mathrm{E} \quad \begin{array}{l}\mathsf{on} \circ \mathsf{thursday};\\ \mathbf{onTh}; \mathrm{VP\backslash VP}\end{array}}{\varphi_1 \circ \varphi_2 \circ \mathsf{to} \circ \mathsf{robin} \circ \mathsf{on} \circ \mathsf{thursday};\ \mathbf{onTh}(P(x)(\mathbf{r}));\ \mathrm{VP}}\backslash\mathrm{E}}{\varphi_2 \circ \mathsf{to} \circ \mathsf{robin} \circ \mathsf{on} \circ \mathsf{thursday};\ \lambda P.\mathbf{onTh}(P(x)(\mathbf{r}));\ \mathrm{(VP/PP/NP)\backslash VP}}\backslash\mathrm{I}^1}{\mathsf{to} \circ \mathsf{robin} \circ \mathsf{on} \circ \mathsf{thursday};\ \lambda x\lambda P.\mathbf{onTh}(P(x)(\mathbf{r}));\ \mathrm{NP\backslash(VP/PP/NP)\backslash VP}}\backslash\mathrm{I}^2}{}$$

We then derive a sentence containing gap positions corresponding to this derived constituent and the object NP, that is, an expression that has the syntactic type S|(NP\(VP/PP/NP)\VP)|NP, to be given as an argument to the three-place **resp3** operator (15) introduced above. Since the relevant steps are the same as in the previous examples, we omit the details and just reproduce the derived sign:

(35) $\lambda\varphi_1\lambda\varphi_2.\mathsf{I} \circ \mathsf{lent} \circ \varphi_1 \circ \varphi_2;$
$\lambda x\lambda f.f(x)(\mathbf{lend})(\mathbf{I}); \mathrm{S|(NP\backslash(VP/PP/NP)\backslash VP)|NP}$

The rest of the derivation just involves giving this relation and the two product-type arguments of types NP and NP\(VP/PP/NP)\VP respectively as arguments to the **resp3** operator. The final translation obtained:

(36) $\mathbf{onTh}(\mathbf{lend}(\mathbf{s})(\mathbf{r}))(\mathbf{I}) \wedge \mathbf{onFr}(\mathbf{lend}(\mathbf{b})(\mathbf{l}))(\mathbf{I})$

corresponds exactly to the relevant reading of the sentence.

We now turn to an extension of the analysis to symmetrical predicates. The key intuition behind our proposal here is that the NP containing *same*, *different*, etc. (we call such NPs 'symmetrical terms' below) in examples like (37) denotes a tuple (linked to the other tuple denoted by the plural *John and Bill* in the same way as in the 'respective' readings above) but that it imposes a special condition on each member of the tuple.

(37) John and Bill read the same book.

Specifically, to assign the right meaning to (37), John and Bill need to be each paired with an identical book, and in the case of *different*, they need to be paired with distinct books. To capture this additional constraint on the tuples denoted by symmetrical terms, we assign to them GQ-type meanings of type S|(S|NP), where the abstracted NP in their arguments are product-type expressions:[9]

(38) a. $\lambda\varphi_0\lambda\sigma_0.\sigma_0(\mathsf{the} \circ \mathsf{same} \circ \varphi_0)$;
$\lambda P\lambda Q.\exists X_\times \forall i\, P(\pi_i(X_\times)) \wedge \forall i \forall j[\pi_i(X_\times) = \pi_j(X_\times)] \wedge Q(X_\times)$;
S|(S|NP)|N

b. $\lambda\varphi_0\lambda\sigma_0.\sigma_0(\mathsf{different} \circ \varphi_0)$;
$\lambda P\lambda Q.\exists X_\times \forall i\, P(\pi_i(X_\times)) \wedge \forall i \forall j[i \neq j \rightarrow \pi_i(X_\times) \neq \pi_j(X_\times)] \wedge Q(X_\times)$;
S|(S|NP)|N

For both *the same N* and *different Ns*, the relevant tuple (which enters into the 'respective' relation with another tuple via the **resp3** operator) consists of

[9] So far as we can tell, the lexical meanings given in (38) capture the truth conditions for the internal readings of *same* and *different* correctly. A reviewer raises a concern that (38a) may be too weak as the meaning of *same* since the existentially bound $X_\times$ would merely be 'a common subset of the books read by John and by [Bill], while the actual sets of read books may still differ'. We do not agree with this reviewer. We believe that (37) is true and felicitous as long as one can identify (at least) one book commonly read by John and Bill. They may have read other books in addition, but that doesn't make (37) false or infelicitous. Such an implication, if felt to be present, is presumably a conversational implicature since it's clearly cancellable:

(i) John and Bill read the same book, although they both read several different books in addition.

Likewise, the same reviewer says that in (38b), as it stand, 'it suffices if $X_\times$ is taken, say, as a pair of different books read by both, all other books still being the same', on the basis of which s/he claims that the truth conditions need to be strengthed in such a way that $X_\times$ satisfies some maximality condition. Here again, we disagree. The following (ii) shows that the maximality implication excluding the existence of common books read by the two (if present at all) is not part of the entailment of the sentence.

(ii) John and Bill read different books, although they read the same books too.

objects that satisfy the description provided by the *N*. The difference is that in the case of *same*, the elements of the tuple are all constrained to be identical, whereas in the case of *different*, they are constrained to differ from one another.

The analysis for (37) now goes as follows:

(39)

$$
\frac{
\dfrac{\begin{array}{l}\lambda\varphi_0\lambda\sigma_0.\\ \quad\sigma_0(\mathsf{the}\circ\\ \quad\mathsf{same}\circ\varphi_0);\\ \mathbf{same};\\ \mathrm{S}|(\mathrm{S}|\mathrm{NP})|\mathrm{N}\end{array}
\quad
\begin{array}{l}\mathsf{book};\\ \mathbf{book};\\ \mathrm{N}\end{array}}
{\begin{array}{l}\lambda\sigma_0.\sigma_0(\mathsf{the}\circ\\ \quad\mathsf{same}\circ\mathsf{book});\\ \mathbf{same}(\mathbf{book});\\ \mathrm{S}|(\mathrm{S}|\mathrm{NP})\end{array}}
\quad
\dfrac{
\dfrac{\begin{array}{l}\lambda\varphi_1.\varphi_1;\\ \lambda p.\bigwedge_i\\ \pi_i(p);\\ \mathrm{S}|\mathrm{S}\end{array}
\quad
\dfrac{\begin{array}{l}\mathsf{john}\circ\\ \mathsf{and}\circ\\ \mathsf{bill};\\ \langle\mathbf{j},\mathbf{b}\rangle;\\ \mathrm{NP}\end{array}
\quad
\dfrac{\begin{bmatrix}\varphi;\\ X_\times;\\ \mathrm{NP}\end{bmatrix}^1
\quad
\dfrac{\begin{array}{l}\lambda\sigma_0\lambda\varphi_1\lambda\varphi_2.\\ \sigma_0(\varphi_1)(\varphi_2);\\ \mathbf{resp3};\\ (Z|X|Y)|(Z|X|Y)\end{array}
\quad
\begin{array}{l}\vdots\quad\vdots\\ \lambda\varphi_3\lambda\varphi_4.\\ \varphi_4\circ\mathsf{read}\circ\\ \varphi_3;\\ \mathbf{read};\\ \mathrm{S}|\mathrm{NP}|\mathrm{NP}\end{array}}
{\begin{array}{l}\lambda\varphi_1\lambda\varphi_2.\varphi_2\circ\mathsf{read}\circ\varphi_1;\\ \mathbf{resp3}(\mathbf{read});\\ \mathrm{S}|\mathrm{NP}|\mathrm{NP}\end{array}}}
{\begin{array}{l}\lambda\varphi_2.\varphi_2\circ\mathsf{read}\circ\varphi;\\ \mathbf{resp3}(\mathbf{read})(X_\times);\\ \mathrm{S}|\mathrm{NP}\end{array}}}
{\begin{array}{l}\mathsf{john}\circ\mathsf{and}\circ\mathsf{bill}\circ\mathsf{read}\circ\varphi;\\ \mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle);\mathrm{S}\end{array}}}
{\begin{array}{l}\mathsf{john}\circ\mathsf{and}\circ\mathsf{bill}\circ\mathsf{read}\circ\varphi;\\ \bigwedge_i\pi_i(\mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle));\mathrm{S}\end{array}}}
{\begin{array}{l}\lambda\varphi.\mathsf{john}\circ\mathsf{and}\circ\mathsf{bill}\circ\mathsf{read}\circ\varphi;\\ \lambda X_\times.\bigwedge_i\pi_i(\mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle));\mathrm{S}|\mathrm{NP}\end{array}}\,|\mathrm{I}^1}
{\begin{array}{l}\mathsf{john}\circ\mathsf{and}\circ\mathsf{bill}\circ\mathsf{read}\circ\mathsf{the}\circ\mathsf{same}\circ\mathsf{book};\\ \mathbf{same}(\mathbf{book})(\lambda X_\times.\bigwedge_i\pi_i(\mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle)));\mathrm{S}\end{array}}
$$

The derivation proceeds by first positing a product-type variable $X_\times$, which is related to the other product-type term denoted by *John and Bill* via the resp operator. Then, after the boolean reduction operator reduces the pair of propositions to their conjunction, the variable $X_\times$ is abstracted over to yield a property of product-type objects (of syntactic type S|NP). Since *the same book* is a GQ over product-type terms, it takes this property as an argument to return a proposition.

The final translation is unpacked in (40):

(40) $$\begin{array}{l}\mathbf{same}(\mathbf{book})(\lambda X_\times.\bigwedge_i\pi_i(\mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle)))\\ =\exists X_\times\forall i\,\mathbf{book}(\pi_i(X_\times))\wedge\forall i\forall j[\pi_i(X_\times)=\pi_j(X_\times)]\wedge\bigwedge_i\pi_i(\mathbf{resp3}(\mathbf{read})(X_\times)(\langle\mathbf{j},\mathbf{b}\rangle))\\ =\exists X_\times\forall i\,\mathbf{book}(\pi_i(X_\times))\wedge\forall i\forall j[\pi_i(X_\times)=\pi_j(X_\times)]\wedge\mathbf{read}(\pi_1(X_\times))(\mathbf{j})\wedge\\ \quad\mathbf{read}(\pi_2(X_\times))(\mathbf{b})\end{array}$$

Since, by definition, $\pi_1(X_\times)=\pi_2(X_\times)$, this correctly ensures that the book that John read and the one that Bill read are identical.

Importantly, since the same 'respective' operator is at the core of the analysis as in the case of 'respective' readings, this analysis immediately predicts that symmetrical predicates can enter into multiple dependencies both among themselves and with respect to 'respective' predication, as exemplified by the data in (3). Since the relevant derivations can be reconstructed easily by taking (18)–(19) as a model, we omit the details and reproduce here only the derived meanings for (3b) and (3c) in (41) and (42), respectively.

(41) $\mathbf{same}(\mathbf{book})(\lambda X_\times.\mathbf{give}(\mathbf{m})(\pi_1(X_\times))(\mathbf{j}) \wedge \mathbf{give}(\mathbf{s})(\pi_2(X_\times))(\mathbf{b}))$
$= \exists X_\times \forall i\, \mathbf{book}(\pi_i(X_\times)) \wedge \forall i \forall j[\pi_i(X_\times) = \pi_j(X_\times)] \wedge \mathbf{give}(\mathbf{m})(\pi_1(X_\times))(\mathbf{j}) \wedge \mathbf{give}(\mathbf{s})(\pi_2(X_\times))(\mathbf{b})$

(42) $\mathbf{same}(\mathbf{book})(\lambda X_\times.\mathbf{same}(\mathbf{man})(\lambda Y_\times.\mathbf{give}(\pi_1(Y_\times))(\pi_1(X_\times))(\mathbf{j}) \wedge \mathbf{give}(\pi_2(Y_\times))(\pi_2(X_\times))(\mathbf{b})))$
$= \exists X_\times \forall i\, \mathbf{book}(\pi_i(X_\times)) \wedge \forall i \forall j[\pi_i(X_\times) = \pi_j(X_\times)] \wedge \exists Y_\times \forall i\, \mathbf{man}(\pi_i(Y_\times)) \wedge \forall i \forall j[\pi_i(Y_\times) = \pi_j(Y_\times)] \wedge \mathbf{give}(\pi_1(Y_\times))(\pi_1(X_\times))(\mathbf{j}) \wedge \mathbf{give}(\pi_2(Y_\times))(\pi_2(X_\times))(\mathbf{b})$

5 Conclusion

In this paper, we have proposed a unified analysis of 'respective' readings and symmetrical predicates, building on the previous accounts of the two phenomena by Gawron and Kehler (2004) and Barker (2007). While these two previous proposals look apparently quite different from each other, in that one involves a nonlocal mechanism for obtaining the right meaning of the sentence whereas the other involves a chain of local operations, we showed that the underlying mechanisms that they rely on are not so different from each other, and that, by recasting the two analyses in a general calculus of the syntax-semantics interface, one (G&K) can essentially be seen as a 'lexicalized' version of the other (Barker), in the sense that it involves only local composition rules but these local composition rules themselves can be derived from the general rules for 'pairwise' predication posited in the latter. We argued that this enables us to unify the analyses of 'respective' readings and symmetrical predicates, and that such a unified analysis is empirically desirable; it immediately accounts for the close parallels and interactions between 'respective' and symmetrical predication via a single uniform mechanism of pairwise predication that is at the core of the semantics of both phenomena. We have demonstrated this point by working out an explicit analysis that captures these parallels and interactions between the two phenomena systematically.

We would like to comment on one technical (and conceptual) point (albeit briefly) before concluding the paper. As noted by two reviewers, the present system relies heavily on empty operators manipulating tuple-denoting objects to yield 'respective' readings and these operators do not affect the syntactic types of the expressions that they apply to. So, for example, a perfectly well-formed syntactic derivation may nonetheless yield an incongruent semantic translation because there is a type mismatch in the semantics. One could alternatively explicitly distinguish tuple-denoting expressions from expressions denoting non-tuple objects by enriching the syntactic typing system with product connectives (so that, for example, *John and Bill* denoting the tuple $\langle \mathbf{j}, \mathbf{b} \rangle$ has syntactic type NP$\times$NP rather than NP). This will enable us to retain the straightforward functional mapping from syntactic types to semantic types standard in the categorial grammar syntax-semantics interface.[10] Moreover, such an approach may enable

[10] But note that, though standard, it's not clear whether this assumption is empirically motivated. See for example Linear Grammar (Mihaliček and Pollard, 2012), which

us to do away with the empty operators that we posit as lexical assumptions in the current system by letting the deductive rules for the product types do the work that these operators undertake in the current fragment. Thus, this seems to be a promising possibility to explore, which may elucidate the 'logic' underlying 'respective' and symmetrical predication even more. We do not see any obstacle in principle for refining the analysis presented above along these lines, and would like to explore this possibility in a future study.

Acknowledgments. We thank reviewers for FG 2014 for their comments. The first author acknowledges the financial support from the Japan Society for the Promotion of Science (Postdoctoral Fellowship for Research Abroad).

References

Barker, Chris. 2007. Parasitic scope. *Linguistics and Philosophy* 30:407–444.

Bekki, Daisuke. 2006. Heikooteki-kaishaku-niokeru yoosokan-junjo-to bunmyaku-izonsei (The order of elements and context dependence in the 'respective' interpretation). In *Nihon Gengo-Gakkai Dai 132-kai Taikai Yokooshuu (Proceedings from the 132nd Meeting of the Linguistic Society of Japan)*, 47–52.

de Groote, Philippe. 2001. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter*, 148–155.

Gawron, Jean Mark and Andrew Kehler. 2004. The semantics of respective readings, conjunction, and filler-gap dependencies. *Linguistics and Philosophy* 27(2):169–207.

Kubota, Yusuke. to appear. Nonconstituent coordination in Japanese as constituent coordination: An analysis in Hybrid Type-Logical Categorial Grammar. *Linguistic Inquiry* 46(1).

Kubota, Yusuke and Robert Levine. 2012. Gapping as like-category coordination. In D. Béchet and A. Dikovsky, eds., *Logical Aspects of Computational Linguistics 2012*, 135–150. Heidelberg: Springer.

Kubota, Yusuke and Robert Levine. 2013. Determiner gapping as higher-order discontinuous constituency. In G. Morrill and M.-J. Nederhof, eds., *Proceedings of Formal Grammar 2012 and 2013*, 225–241. Heidelberg: Springer.

Kubota, Yusuke and Robert Levine. 2014. Pseudogapping as pseudo-VP ellipsis. In N. Asher and S. Soloviev, eds., *Logical Aspects of Computational Linguistics 2014*, 122–137. Heidelberg: Springer.

Mihaliček, Vedrana and Carl Pollard. 2012. Distinguishing phenogrammar from tectogrammar simplifies the analysis of interrogatives. In P. de Groote and M.-J. Nederhof, eds., *Formal Grammar 2010/2011*, 130–145. Heidelberg: Springer.

Muskens, Reinhard. 2003. Language, lambdas, and logic. In G.-J. Kruijff and R. Oehrle, eds., *Resource Sensitivity in Binding and Anaphora*, 23–54. Dordrecht: Kluwer.

Oehrle, Richard T. 1994. Term-labeled categorial type systems. *Linguistics and Philosophy* 17(6):633–678.

Worth, Chris. 2014. The phenogrammar of coordination. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, 28–36. Gothenburg, Sweden: Association for Computational Linguistics.

---

explicitly rejects the functional mapping from syntactic types to semantic types (see in particular Worth (2014, section 1.1) for an explicit statement of this point).

Appendix

Lemma: For any arbitrary complex structure S licensed by the G&K fragment with semantic translation Γ and which contains exactly one occurrence of a term t whose semantic translation is x, we obtain a structure S' by replacing t in S with a term whose translation is $\langle a_1, \ldots, a_l \rangle$. Then for the semantic translation of S' Ψ, the following holds:

$(\star)\quad \Psi = \langle \Gamma[x/a_1], \ldots, \Gamma[x/a_l] \rangle$

Proof: The proof is by induction.

Base case: Since $\Gamma = x$ and $\Psi = \langle a_1, \ldots, a_l \rangle$, it trivially follows that $(\star)$ holds.

Inductive step: We have two cases to consider: (i) S consists of a function f and a structure T (with translation Ω, which is an argument of f) that satisfies $(\star)$; (ii) S consists of a structure T (with translation Ω) that satisfies $(\star)$ and a term c that is an argument of Ω. We consider (i) first.

(i)

$$\dfrac{\varphi_2;\ f;\ \mathrm{X/Y} \qquad \dfrac{\vdots\ \vdots\ \vdots \quad \dfrac{\varphi_0;\ x;\ \mathrm{A}}{\vdots\ \vdots} \quad \vdots\ \vdots\ \vdots}{\varphi_1;\ \Omega;\ \mathrm{Y}}}{\varphi_2 \circ \varphi_1;\ f(\Omega);\ \mathrm{X}}$$

Since T satisfies $(\star)$, there is a structure T' in which x in T is replaced by $\langle a_1, ..., a_l \rangle$ such that the following holds between Ω and Ω', the translations of T and T': $\Omega' = \langle \Omega[x/a_1], ..., \Omega[x/a_n] \rangle$.

We are interested in the translation Γ' of a structure S', which can be obtained by replacing t with a term whose translation is $\langle a_1, ..., a_l \rangle$. By replacing T in S with T', we obtain just such a structure:

(i')

$$\dfrac{\varphi_2;\ f;\ \mathrm{X/Y} \qquad \dfrac{\vdots\ \vdots\ \vdots \quad \dfrac{\varphi_0';\ \langle a_1, ..., a_l \rangle;\ \mathrm{A}}{\vdots\ \vdots} \quad \vdots\ \vdots\ \vdots}{\varphi_1';\ \Omega';\ \mathrm{Y}}}{\varphi_2 \circ \varphi_1';\ \Gamma';\ \mathrm{X}}\ \text{Rule 1}$$

Thus,

$$\begin{aligned} \Gamma' &= \langle f(\Omega[x/a_1]), ..., f(\Omega[x/a_n]) \rangle && \text{(via Rule 1)} \\ &= \langle (f(\Omega))[x/a_1], ..., (f(\Omega))[x/a_n] \rangle && \text{(since } x \text{ is fresh in } f\text{)} \\ &= \langle \Gamma[x/a_1], ..., \Gamma[x/a_n] \rangle && \text{(since } \Gamma = f(\Omega)\text{)} \end{aligned}$$

Case (ii) can be proven similarly to case (i).