Non-canonical Coordination in the Transformational Approach

Oleg Kiselyov

Tohoku University, Japan
oleg@okmij.org

Abstract. Recently introduced Transformational Semantics (TS) formalizes, restraints and makes rigorous the transformational approach epitomized by QR and Transformational Grammars: deriving a meaning (in the form of a logical formula or a logical form) by a series of transformations from a suitably abstract (tecto-) form of a sentence. TS generalizes various 'monad' or 'continuation-based' computational approaches, abstracting away irrelevant details (such as monads, etc) while overcoming their rigidity and brittleness. Unlike QR, each transformation in TS is rigorously and precisely defined, typed, and deterministic. The restraints of TS and the sparsity of the choice points (in the order of applying the deterministic transformation steps) make it easier to derive negative predictions and control over-generation.

We apply TS to right-node raising (RNR), gapping and other instances of non-constituent coordination. Our analyses straightforwardly represent the intuition that coordinated phrases must in some sense be 'parallel', with a matching structure. Coordinated material is not necessarily constituent – even 'below the surface' – and we do not pretend it is. We answer the Kubota, Levine and Moot challenge (the KLM problem) of analyzing RNR and gapping without directional types, yet avoiding massive over-generation. We thus formalize the old idea of 'coordination reduction' and show how to make it work for generalized quantifiers.

1 Introduction

Non-canonical coordination – right-node raising (RNR) as in (1), argument-cluster coordination (2) and, in particular, gapping (3-7) – provides an unending stream of puzzles for the theory of semantics [8, 10]:

(1) John likes and Mary hates Bill.
(2) John gave a present to Robin on Thursday and to Leslie on Friday.
(3) Mary liked Chicago and Bill Detroit.
(4) One gave me a book and the other a CD.
(5) Terry can go with me and Pat with you.
(6) Mrs. J can't live in Boston and Mr. J in LA.
(7) Pete wasn't called by Vanessa but rather John by Jesse.

With gapping, it is not just a simple verb that can "go missing", as in (3). It can be a complex phrase of a verb with arguments and complements – or, as in (4), a verb and an auxiliary verb. Interactions of coordination with scope-taking are particularly challenging: a competent theory needs to handle both narrow- and wide-scope reading of "a present" in (2) and the narrow- and wide-scope coordination in (6). In (7), negation somehow scopes over the first "coordinated structure" but not over the second.

Recently in [8, 9], Kubota and Levin put forward new analyses of non-canonical coordination, applying hybrid categorial grammars they have been developing. In contrast, the analyses in [6] use plain old non-associative Lambek grammar. However, the main ideas of [6] are completely hidden behind thickets of complicated types and their interactions within a derivation. The intuition that coordinated structures must be parallel is thus lost in the details.

We present a new analysis of non-constituent coordination using the more intuitive and less round-about framework TS (formerly called AACG) [7], designed to take the 'hacking' out of tree-hacking. TS lets us talk about QR and other transformations towards some semantic form in a rigorous, formal, mostly deterministic way. We remind of TS in §2.

Our analyses re-expose ideas from the earlier approach of [6], but free them from the bondage of encoding. A notable feature of TS is the absence of directional types. We use it to answer the challenge posited by Kubota, Levin [10] and Moot (dubbed "the KLM problem" by Morrill): to analyze RNR within categorial-grammar–like formalisms without directional types, while avoiding massive overgeneration.

One may categorize the various approaches to non-canonical coordination based on what exactly is being coordinated. Take (1), repeated below

(1) John likes and Mary hates Bill.

which will be our running example for a while. Are the complete sentences being coordinated behind the scene, as in "John likes Bill" and "Mary hates Bill" with "Bill" being later elided? Or perhaps sentences with holes are being coordinated, as in "John likes hyp_{obj}"? (as done in [6, 8, 9].) Or perhaps we regard "John likes" and "Mary hates" as constituents and coordinate as such (as in CCG). In this paper we give another answer: we analyze (1) as the coordination of the complete clause "Mary hates Bill" with the cluster "John" and "likes". The types of the cluster components and their order guide the transformation that picks the needed material from the clause "Mary hates Bill" to make the cluster the complete clause. The 'picking transformation' can be naturally supported within the existing setup of TS, using the same mechanism used in [7] to analyze quantification and inverse linking. The intuition of 'picking' is made precise and formal in §3.

The structure of the paper is as follows. §2 reminds TS, in a different, clearer presentation. We then describe our approach to coordination: transforming non-canonical one to the ordinary coordination of clauses. §4 discusses the related work that forms the context of our approach. The rigorous nature of TS makes it easier to carry analyses mechanically, by a computer. In fact, the analyses in the paper have been so programmed and executed. The implementation, in the form of a domain-specific language embedded in Haskell – 'the semantic calculator' – is publicly available at `http://okmij.org/ftp/gengo/transformational-semantics/`.

2 TS Background

Traditional Categorial Grammar approaches draw parallels between proof systems and grammars: grammaticality is identified with the existence of a derivation. It is rather challenging however to prove the absence of a derivation, and to overview the space of possible derivations in general.

TS (formerly, AACG) [7] in contrast pursues the computational approach, harking back to Transformational Generative Grammars [2] of 1960s: Rather than trying to *deduce* a derivation, it tries to *induce* the meaning (the logical formula) by applying a sequence

of precisely and formally defined transformations to a suitably abstract form of a sentence. The latter abstracts away the case and the number agreement, declination, etc. The transformations are deterministic; the order of their applications is generally not. (There may still be dependencies between particular transformations imposing the order.) The transformations are partial: the failure is taken as ungrammaticality of the original sentence.

Formally, TS deals with term languages that represent typed finite trees. Each T-language is a set of well-typed terms built from typed constants (function symbols) c. Types are

$$\begin{array}{ll} \text{Base types} & \upsilon \\ \text{T-Types} & \sigma ::= \upsilon \mid \sigma \to \sigma \end{array}$$

The set terms d is then inductively defined as: (i) each constant c of the type σ is a term; (ii) if c has the type $\sigma_1 \to \sigma$ and d is a term of type σ_1, then $c\ d$ is a term of type σ; (iii) nothing else is a term. The set of constants and their types is a (multi-sorted) algebraic signature; A T-language is hence a term language over the signature, which defines the language.

Table 1 shows three sample languages. T_S has the single base type string and numerous constants `"John"`, `"greet"`, `"every"`, etc. of that type. It describes the surface, "phonetic", form of a sentence. The constant $\text{-}\cdot\text{-} : \mathsf{string} \to \mathsf{string} \to \mathsf{string}$ (usually written as the infix operation) signifies string concatenation. The language T_A whose types are familiar categories represents the abstract form. T_L is the language of formulas of predicate logic, which describe the meaning of sentences. The (infinite) sets of *constants* $\mathsf{var_x}, \mathsf{var_y}, \ldots$ and the corresponding $\mathsf{U_x}, \ldots$ and $\mathsf{E_x}, \ldots$ represent (to be) bound variables and their binders. Unlike the conventional (lambda-bound) variables, they are not subject to substitution, α-conversion or capture-avoidance. T_L likewise has constants $x, y, z, \ldots$ of the type e and the corresponding sets of *constants* $\forall_x, \forall_y, \ldots, \exists_x, \exists_y, \ldots$ intended as binders.

As a way to introduce TS we show the quantification analysis of "John greeted every participant". The sample sentence in the language T_A has the form

$$\mathsf{cl\ john\ (argp\ greet\ (every_x\ participant))}$$

	v	c
T_S	string	$\cdot$: string $\to$ string $\to$ string "John" , "greet" , "every" , . . . : string
T_A	S, NP, N, VP, PP, TV	John: NP participant: N greet: TV cl: $NP \to VP \to S$ argp: $TV \to NP \to VP$ ppadv: $VP \to PP \to VP$ $\mathsf{every_x}, \mathsf{every_y}, \mathsf{a_z}$: $N \to NP$ $\mathsf{var_x}, \mathsf{var_y}, \ldots$: NP $\mathsf{U_x}, \mathsf{U_y}, \ldots, \mathsf{E_x}, \mathsf{E_y}, \ldots$: $N \to S \to S$
T_L	e, t	conj , disj , . . . : $t \to t \to t$ john: e participant: $e \to t$ greet: $e \to t \to t$ $\forall_x, \exists_y$: $t \to t$ $x, y, z, \ldots$: e

Table 1. Signatures of various T-languages

to be referred to as *jgep*. The constant cl combines an NP and a VP into a clause. (Likewise, argp attaches an argument to a verb and ppadv attaches a prepositional phrase (PP) as a VP complement.) Quantifiers are uniquely labeled by x, y, z, etc. We assume it is the job of a parser to uniquely label the quantifiers in the abstract form.

Before taking on meaning we illustrate the recovering of the surface form of *jgep*, by applying the following 'phonetic' transformation $\mathcal{L}_{syn}$.

$$
\begin{array}{lcl}
\mathcal{L}_{syn}\ulcorner \mathsf{cl}\ d_1\ d_2 \urcorner & \mapsto & \mathcal{L}_{syn}\ulcorner d_1 \urcorner \cdot \mathcal{L}_{syn}\ulcorner d_2 \urcorner \\
\mathcal{L}_{syn}\ulcorner \mathsf{argp}\ d_1\ d_2 \urcorner & \mapsto & \mathcal{L}_{syn}\ulcorner d_1 \urcorner \cdot \mathcal{L}_{syn}\ulcorner d_2 \urcorner \\
\mathcal{L}_{syn}\ulcorner \mathsf{john} \urcorner & \mapsto & \texttt{"john"} \\
\mathcal{L}_{syn}\ulcorner \mathsf{every_x} \urcorner & \mapsto & \texttt{"every"} \\
\mathcal{L}_{syn}\ulcorner \mathsf{participant} \urcorner & \mapsto & \texttt{"participant"} \\
 & \ldots &
\end{array}
$$

The rules are written in the form reminiscent of top-down tree transducers. The result $\mathcal{L}\ulcorner d \urcorner$ of transforming a term d is obtained by trying to match d against the pattern in the left-hand-side of every rule. The right-hand-side of the matching rule gives the result. If

no matching rule is found, the transformation is not defined (i.e., 'fails'). The patterns may contain variables, which stand for the corresponding subterms. For example, in the first rule, d_1 and d_2 match the two children of a term whose head is cl. The occurrences of these variables in the right-hand side of the rule are replaced by the corresponding matching branches. Intuitively, $\mathcal{L}_{sem}$ looks like a context-free-grammar of the sample sentence, with $jgep$ being its derivation tree.

The meaning is derived by applying a sequence of transformations to a T_A term. The transformation $\mathcal{L}_{Ux}$ gets rid of $\mathsf{every_x}$, introducing $\mathsf{var_x}$ and $\mathsf{U_x}$ instead. This transformation is context-sensitive. Therefore, we first define context C – a term (tree) with a hole – as follows:

$$C = [] \mid \mathsf{cl}\ C\ d \mid \mathsf{cl}\ d\ C \mid \mathsf{argp}\ d\ C \mid \mathsf{ppadv}\ C\ d \mid \mathsf{ppadv}\ d\ C$$

where the meta-variable d stands for an arbitrary term. In words: a context is the bare hole [], or a clause (the cl term) that contains a hole in the subject or the predicate, or a VP made of a transitive verb whose argument has a hole, or a complemented VP with the hole in the head or the complement, etc. We write $C[d]$ for the term obtained by plugging d into the hole of C. We further distinguish two subsets of contexts C_{cl} and C_{ncl}:

$$C_{cl} = \mathsf{cl}\ C_{ncl}\ d \mid \mathsf{cl}\ \ d\ C_{ncl}$$
$$C_{ncl} = [] \mid \mathsf{argp}\ d\ C_{ncl} \mid \mathsf{ppadv}\ C_{ncl}\ d \mid \mathsf{ppadv}\ d\ C_{ncl}$$

Intuitively, C_{cl} is the smallest context that has a hole within a clause.

The transformation $\mathcal{L}_{Ux}$ is then stated as follows:

$$\mathcal{L}_{Ux}\ulcorner C_{cl}[\mathsf{every_x}\ \ d_r]\urcorner \ \mapsto\ \mathsf{U_x}\ (\mathcal{L}_{Ux}\ulcorner d_r\urcorner)\ (\mathcal{L}_{Ux}\ulcorner C_{cl}[\mathsf{var_x}\ ]\urcorner)$$

We now use extended top-down tree transducers, whose patterns are 'deep', that is, contain matching expressions within arbitrary context. As before, whenever a pattern, e.g., $C_{cl}[\mathsf{every_x}\ \ d_r]$, matches the source term, it is replaced with $\mathsf{U_x}\ d_r\ C_{cl}[\mathsf{var_x}]$, and the transformation is re-applied to its subterms. That is, $C_{cl}[\mathsf{every_x}\ d_r]$ on the left hand-side of the rule matches a tree that contains, somewhere inside, a sub-expression of the form $\mathsf{every_x}\ d_r$ (a branch headed by $\mathsf{every_x}$). On the right-hand side of the rule, $C_{cl}[\mathsf{var_x}]$ is the same tree in which $\mathsf{every_x}\ d_r$ subterm has been replaced with $\mathsf{var_x}$. Unlike $\mathcal{L}_{syn}$ above, the $\mathcal{L}_{Ux}$ transformation does not look like a context-free grammar. It is context-sensitive. The other difference is the presence of a

default rule: if $\mathcal{L}_{Ux}\ulcorner d\urcorner$ finds no match for d, $\mathcal{L}_{Ux}$ is repeated on subexpressions of d. In particular, $\mathcal{L}_{Ux}\ulcorner c\urcorner$ is the constant c itself (unless there is an explicit rule for that particular c). For $\mathcal{L}_{syn}$, which translates from one language, T_A, to another, T_S, the default rule does not make sense.

Our example $jgep$ matches the left-hand side of $\mathcal{L}_{Ux}$ immediately: d_r matches participant and C_{cl} is john (argp greet []), The result

$$(\mathsf{U_x}\ \ \mathsf{participant})\ \ (\mathsf{cl}\ \ \mathsf{john}\ \ (\mathsf{argp\ greet}\ \ \mathsf{var_x}))$$

is in effect the Quantifier Raising (QR) of "every participant", but in a rigorous, deterministic way. The intent of the new constants should become clear: $\mathsf{U_x}$ is to represent the raised quantifier, and $\mathsf{var_x}$ its trace. Unlike QR, the raised quantifier ($\mathsf{U_x}$ participant) lands not just on any suitable place. $\mathcal{L}_U$ puts it at the closest boundary marked by the clause-forming constant cl. $\mathcal{L}_U$ is type-preserving: it maps a well-typed term to also a well-typed term. Again unlike QR, we state the correctness properties such as type-preservation. The type preservation is the necessary condition for the correctness of the transformations.

To finally obtain the meaning we apply the transformation $\mathcal{L}_{sem}$:

$$\begin{array}{lcl}
\mathcal{L}_{sem}\ulcorner \mathsf{cl}\ d_1\ d_2\urcorner & \mapsto & \mathcal{L}_{sem}\ulcorner d_2\urcorner\ \mathcal{L}_{sem}\ulcorner d_1\urcorner \\
\mathcal{L}_{sem}\ulcorner \mathsf{argp}\ d_1\ d_2\urcorner & \mapsto & \mathcal{L}_{sem}\ulcorner d_1\urcorner\ \mathcal{L}_{sem}\ulcorner d_2\urcorner \\
\mathcal{L}_{sem}\ulcorner \mathsf{U_x}\ \ d_1\ d_2\urcorner & \mapsto & \forall_x\ \mathcal{L}_{sem}\ulcorner d_2\urcorner\ x \Rightarrow \mathcal{L}_{sem}\ulcorner d_2\urcorner \\
\mathcal{L}_{sem}\ulcorner \mathsf{var_x}\ \urcorner & \mapsto & x \\
\mathcal{L}_{sem}\ulcorner \mathsf{john}\ \urcorner & \mapsto & \text{john} \\
\mathcal{L}_{sem}\ulcorner \mathsf{participant}\ \urcorner & \mapsto & \text{participant} \\
 & & \ldots
\end{array}$$

that produces the logical formula representing the term's meaning. The transformation replaces john, etc. with the corresponding logical constants and $\mathsf{U_x}$ with the universal quantifier. Since $\mathcal{L}_{sem}$ translates one language, T_A, into a different one, T_L, this transformation, like $\mathcal{L}_{syn}$, has no default rule. If the source term does not match the pattern of any $\mathcal{L}_{sem}$ rule, the transformation is undefined. In particular, applying $\mathcal{L}_{sem}$ to the original $jgep$ term straight away is not defined because there is no rule for $\mathsf{every_x}$. The failure means that $jgep$ cannot be given meaning – directly. However, $\mathcal{L}_{sem}\ulcorner\mathcal{L}_{Ux}\ulcorner jgep\urcorner\urcorner$ is well-defined, resulting in

$$\forall_x\ \text{participant}\ x \Rightarrow (\text{greet}\ x\ \text{john})$$

3 Coordination in TS

We now apply TS to the analysis of (non-canonical) coordination. As a warm-up, we take the non-problematic "John tripped and fell," which is an example of the conventional VP coordination. We analyze it differently, however, as 'left-node raising' so to speak, to introduce the technique to be later used in right-node raising (RNR), argument cluster coordination (ACC) and gapping [1].

The abstract form of our example is

$$\mathsf{and_{S,VP}}\ (\mathsf{cl\ john\ tripped})\ \mathsf{fell}$$

The new constant $\mathsf{and_{S,VP}}$ has the type $S \to VP \to S$. As common, we assume a whole family of constants $\mathsf{and_{X,Y}}$ of different types. The constant $\mathsf{and_{S,VP}}$ – like $\mathsf{every_x}$ in the example of the previous section – is not in the domain of $\mathcal{L}_{sem}$. Therefore, to be able to derive the logical formula, we have to transform it away. The following transformation $\mathcal{L}_a$ does that:

$$\begin{aligned}\mathcal{L}_a{}^{\ulcorner}\mathsf{and_{S,VP}}\ (\mathsf{cl}\ d_{NP}\ d_{VP})\ d^{\urcorner} \quad &\mapsto \\ \mathsf{and}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d_{NP}\ d_{VP})^{\urcorner}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d\ d_{VP})^{\urcorner}\end{aligned}$$

The rule again is written in the form of extended top-down tree transducers: when the source term matches the rule's pattern, it is replaced with the right-hand-side of the rule. Again, d with various subscripts are meta-variables that stand for arbitrary subterms (tree branches). Like $\mathcal{L}_{Ux}$, there is a default rule: a term that does not match the rule undergoes $\mathcal{L}_a$ on its subterms, if any. Applying $\mathcal{L}_a$ to our T_A term transforms it to

$$\mathsf{and}\ (\mathsf{cl\ john\ tripped})\ (\mathsf{cl\ john\ fell})$$

where and is the ordinary coordination, of the type $S \to S \to S$, which can be given the meaning of propositional disjunction and which hence is in the domain of $\mathcal{L}_{sem}$. The result is straightforward to transform to a logical formula T_L.

[1] We may even analyze NP coordination as a sort of RNR: after all, "John and Mary left" can have the meaning of the conjunction of truth conditions of "John left" and "Mary left". Certainly, "John and Mary left" may also mean that "John and Mary", taken as a group, left. In the later case, the group can be referred as "they". Our analysis applies to the former (conjunction) case but not the latter. Hence we posit that 'and' is not only polytypic but also polysemic.

3.1 RNR in TS

Our next example is the proper RNR: "John likes and Mary hates Bill", whose abstract form is

$$\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}\ (\mathsf{john}, \mathsf{like})\ (\mathsf{cl}\ \mathsf{mary}\ (\mathsf{argp}\ \mathsf{hate}\ \mathsf{bill}))$$

We have added to T_A tuples (d, d) and tuple types (σ, σ). The constant $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$ has the type $(NP, TV) \to S \to S$. Whereas (cl mary (argp hate bill)) is the complete sentence, (john, like) is certainly not. It is not even a constituent; it is just a sequence of words: a cluster. Since we added to T_A tuples and new constants, we may need to extend our earlier transformation rules, specifically, $\mathcal{L}_{syn}$ for transforming into the surface form of the sentence T_S:

$$\begin{array}{lcl} \mathcal{L}_{syn}{\ulcorner}\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}\ \ d_1\ d_2{\urcorner} & \mapsto & \mathcal{L}_{syn}{\ulcorner}d_1{\urcorner} \cdot \texttt{"and"}\ \cdot \mathcal{L}_{syn}{\ulcorner}d_2{\urcorner} \\ \mathcal{L}_{syn}{\ulcorner}(d_1, d_2){\urcorner} & \mapsto & \mathcal{L}_{syn}{\ulcorner}d_1{\urcorner} \cdot \mathcal{L}_{syn}{\ulcorner}d_2{\urcorner} \end{array}$$

Applying $\mathcal{L}_{syn}$ to our T_A clearly gives "John likes and Mary hates Bill". This 'phonetic' transformation is dull and uninteresting, in contrast to the higher-order phonetics of [8].

Let us derive the meaning, the T_L formula, from the same T_A term. Before we can apply $\mathcal{L}_{sem}$ we need to transform away $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$, which is not in the domain of that transformation. We extend the $\mathcal{L}_a$ with a new clause:

$$\begin{array}{l} \mathcal{L}_a{\ulcorner}\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}\ (d_1, d_2)\ (\mathsf{cl}\ d\ C[\mathsf{argp}\ d_4\ d_5]){\urcorner} \quad \mapsto \\ \qquad \mathsf{and}\ \mathcal{L}_a{\ulcorner}(\mathsf{cl}\ d_1\ (\mathsf{argp}\ d_2\ d_5)){\urcorner}\ \mathcal{L}_a{\ulcorner}(\mathsf{cl}\ d\ C[\mathsf{argp}\ d_4\ d_5]){\urcorner} \end{array}$$

where d_1, d, d_5 have to be of the type NP and d_2 and d_4 of the type TV. The transformation is context-sensitive and type-directed. It may be regarded as matching of (d_1, d_2) against the complete sentence (the second argument of $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$). The matching is determined by the type of $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$. The parallel structure of the coordination is clearly visible.

Analyses of RNR without directional types (e.g, using ACG) run into trouble of over-generating "*John likes Bill and Mary hates". Although we can write the abstract form for that sentence as well:

$$\mathsf{and}_{\mathsf{S},(\mathsf{NP},\mathsf{TV})}\ (\mathsf{cl}\ \mathsf{john}\ (\mathsf{argp}\ \mathsf{like}\ \mathsf{bill}))\ (\mathsf{mary}, \mathsf{hate})$$

we do not provide the $\mathcal{L}_a$ rule with the constant $\mathsf{and}_{\mathsf{S},(\mathsf{NP},\mathsf{TV})}$. Since it remains uneliminated, $\mathcal{L}_{sem}$ cannot be applied and the meaning cannot be derived. In TS, transformations are partial and are not guaranteed to always succeed. The original sentence is considered

ungrammatical then. We discuss the choice of transformable $\mathsf{and}_{\mathsf{XY}}$ constants in §3.4.

Let us consider another well-known troublesome example, due to P. Dekker:

(1) *The mother of and John thinks that Mary left.

In categorial grammar approaches, 'the mother of' and 'John thinks that' may be given the same type, $(S/(N\backslash S))/N$. The two phrases may hence be coordinated, over-generating (1). In TS, 'the mother of' cannot be given any type at all (likewise, 'John thinks that' is not a constituent and has no type.) We can only treat 'the mother of' as a cluster, of the determiner, N and the proposition. We do provide the constant $\mathsf{and}_{(\mathsf{DET,N,POF}),\mathsf{S}}$ with the corresponding rule

$$\mathcal{L}_a{}^{\ulcorner}\mathsf{and}_{(\mathsf{DET,N,POF}),\mathsf{S}}\ (d_1, d_2, \mathsf{of})\ (C_{cl}[d_{det}\ (\mathsf{ppadj}\ d_n\ \mathsf{of}\ d_{np})])^{\urcorner} \quad \mapsto$$
$$\mathsf{and}\ \mathcal{L}_a{}^{\ulcorner}(C_{cl}[d_1\ (\mathsf{ppadj}\ d_2\ \mathsf{of}\ d_{np})])^{\urcorner}\ \mathcal{L}_a{}^{\ulcorner}(C_{cl}[d_{det}\ (\mathsf{ppadj}\ d_n\ \mathsf{of}\ d_{np})])^{\urcorner}$$

which can be used to analyze "The mother of, as well as the father of John died". The rule does not apply to the problematic (1) since there is no similar parallel structure of the of-headed PP.

3.2 Argument Cluster Coordination and Gapping

The same transformation idea also works for argument cluster coordination (ACC) and gappping. Take for example, "Mary liked Chicago and Bill Detroit", or, in the abstract form:

$$\mathsf{and}_{\mathsf{S},(\mathsf{NP,NP})}\ (\mathsf{cl}\ \mathsf{mary}\ (\mathsf{argp}\ \mathsf{liked}\ \mathsf{chicago}))\ (\mathsf{bill}, \mathsf{detroit})$$

The transformational rule for the constant $\mathsf{and}_{\mathsf{S},(\mathsf{NP,NP})}$ picks a suitable subterm that can relate two NPs from the left conjunct

$$\mathcal{L}_a{}^{\ulcorner}\mathsf{and}_{\mathsf{S},(\mathsf{NP,NP})}\ (\mathsf{cl}\ d\ C[\mathsf{argp}\ d_4\ d_5])\ (d_1, d_2)^{\urcorner} \quad \mapsto$$
$$\mathsf{and}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d\ C[\mathsf{argp}\ d_4\ d_5])^{\urcorner}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d_1\ C[(\mathsf{argp}\ d_4\ d_2)])^{\urcorner}$$

It turns our T_A term to

$$\mathsf{and}\ (\mathsf{cl}\ \mathsf{mary}\ (\mathsf{argp}\ \mathsf{liked}\ \mathsf{chicago}))\ \ (\mathsf{cl}\ \mathsf{bill}\ (\mathsf{argp}\ \mathsf{liked}\ \mathsf{detroit}))$$

with the clear meaning. The examples (2) and (4) of §1 are dealt with similarly. One may observe that the analysis of gapping is nearly the same as that of VP coordination, used in the warm-up example.

3.3 Coordination and Scoping

The interaction of non-canonical coordination with quantification is not much different from that of the ordinary coordination of two clauses. For example, take (2) of §1, whose abstract form is

$\mathsf{and}_{\mathsf{S},(\mathsf{PP},\mathsf{PP})}$
  (cl speaker (ppadv (ppadv (argp gave (a_x present)) (to robin))(on thu)))
  (to leslie, on fri)

contains two components to be eliminated by transformations: $\mathsf{and}_{\mathsf{S},(\mathsf{PP},\mathsf{PP})}$ and the QNP (a_x present). The latter is to be handled by $\mathcal{L}_E$, which is analogous to $\mathcal{L}_U$ but for the existential quantifier. The transformations $\mathcal{L}_a$ and $\mathcal{L}_E$ can be applied in either order, which corresponds to the wide- and narrow-scope–readings of (2). The narrow scope happens when $\mathcal{L}_a$ goes first, producing

and
  (cl speaker (ppadv (ppadv (argp gave (a_x present)) (to robin))(on thu)))
  (cl speaker (ppadv (ppadv (argp gave (a_x present)) (to leslie))(on fri)))

The $\mathcal{L}_{Ex}$ transformation then gives

and
  (E_x present (cl speaker (ppadv (ppadv (argp gave var_x) (to robin))(on thu))))
  (E_x present (cl speaker (ppadv (ppadv (argp gave var_x) (to leslie))(on fri))))

whose meaning is the conjunction of two existentially quantified formulas.

If $\mathcal{L}_{Ex}$ is applied first to the original sentence, we get

$\mathsf{and}_{\mathsf{S},(\mathsf{PP},\mathsf{PP})}$
  (E_x present (cl speaker (ppadv (ppadv (argp gave var_x) (to robin))(on thu))))
  (to leslie, on fri)

Strictly speaking, the rule analogous to $\mathcal{L}_a$ from §3.2 does not apply since the first conjunct now has the form $\mathsf{E}_\mathsf{x}\ d_r$ (cl d_1 d_2) rather than the bare (cl d_1 d_2). We have to hence generalize the rule to

$$\mathcal{L}_a{}^{\ulcorner}\mathsf{and}_{\mathsf{S},(\mathsf{PP},\mathsf{PP})}\ C_{ncl}\,[(\mathsf{cl}\ d\ C[\mathsf{ppadv}\ (\mathsf{ppadv}\ d_h\ d_4)\ d_5])]\ \ (d_1, d_2)^{\urcorner} \quad \mapsto$$
$$C_{ncl}[\mathsf{and}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d\ C[\mathsf{ppadv}\ (\mathsf{ppadv}\ d_h\ d_4)\ d_5])^{\urcorner}$$
$$\mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d\ C[\mathsf{ppadv}\ (\mathsf{ppadv}\ d_h\ d_1)\ d_2])^{\urcorner}]$$

effectively pulling out the context C_{ncl} – the sequence of $\mathsf{U}_\mathsf{x}\ d$ and $\mathsf{E}_\mathsf{x}\ d$ quantifiers and their restrictors – and coordinating underneath. The coordination thus receives narrow scope. Such pulling of the context may seem ad hoc; however, it is this general form of $\mathcal{L}_a$ rules that

gives the mechanism to account for the anomalous scope of negation in (7) of §1, repeated below.

(7) Pete wasn't called by Vanessa but rather John by Jesse.

The transformation involving the contrasting coordinating particle such as 'but rather' gets a chance to examine C_{ncl} and determine if there is a negation to contrast with:

$$\mathcal{L}_a{}^{\ulcorner}\mathsf{rather}_{\mathsf{S},(\mathsf{NP},\mathsf{PP})}\ (\mathsf{Neg}\ \ (\mathsf{cl}\ d\ C[\mathsf{ppadv}\ d_4\ d_5]))\ (d_1, d_2)^{\urcorner} \quad \mapsto$$
$$\mathsf{and}\ \ (\mathsf{Neg}\ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d\ C[\mathsf{ppadv}\ d_4\ d_5])^{\urcorner})\ \ \mathcal{L}_a{}^{\ulcorner}(\mathsf{cl}\ d_1\ (\mathsf{ppadv}\ d_4\ d_2))^{\urcorner}$$

where Neg is the constant analogous to U_{x}.

3.4 Discussion

We have presented the uniform analysis of both the canonical and non-canonical coordination, reducing the variety of coordination (VP, RNR, ACC, Gapping) to the choice of the coordinating constants $\mathsf{and}_{\mathsf{S},\mathsf{X}}$ or $\mathsf{and}_{\mathsf{X},\mathsf{S}}$ that adjoin material (often just a cluster of words) to a sentence. The transformation rules driven by the constants pick the pieces from the sentence to complete the material to a clause. We have thus provided a uniform *mechanism* of coordination. The corresponding policy is embodied in the coordinator constants like and and hence lexicalized.

There remains a question of a general principle/pattern that governs the choice of the constants. For example, the fact that in English the coordinated sentence appears on the right for RNR but on the left for ACC and Gapping boils down to the presence of $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$ and $\mathsf{and}_{\mathsf{S},(\mathsf{NP},\mathsf{NP})}$ and the absence of $\mathsf{and}_{\mathsf{S},(\mathsf{NP},\mathsf{TV})}$ and $\mathsf{and}_{(\mathsf{NP},\mathsf{NP}),\mathsf{S}}$. In contrast, one may say that this fact 'falls out' as a consequence of like-category coordination analyses in directional categorial grammars. One may also say that the like-category coordination is itself a postulate, which does not come from any general principle, but does have significant empirical justification. Like any empirical principle, it has exceptions: unlike-category coordination, e.g., "John saw the facts and that Mary had been right". Also, the like-category coordination leads to overgeneration, as we saw in the Dekker's example in §3.1.

Since our TS approach is still new, we have not yet accumulated enough empirical data to discern patterns and formulate postulates

that underlie the presence of coordination constants for some types and their absence for others. For now, we leave the question open.

4 Related Work

Our transformational approach is rooted in Transformational Generative Grammars [2, 3], later carried into Minimalism [4]. Our abstract form T_A is similar to the spell-out of Minimalism. However, whereas the spell-out is near culmination of a syntactic derivation for Minimalists, for us, it is just the beginning. We are not interested in how structure is created through a sequence of Merges from lexical selections. Rather, we consider our abstract form as given (by a parser) and investigate its transformations into a semantic form. Our transformations are hence all covert.

Closely related to TS is the work of Butler [1], who also obtains a semantic representation as a result of a transformation from a parsed tree. Unlike us, he has applied his approach to a wealth of empirical data in many languages and has truly achieved wide coverage. His transformations are rather complex and coarse, doing many things at once, and not typed. One may view TS as an attempt to re-engineer and understand Butler's approach and decompose his transformations into elementary steps.

We are grateful to the anonymous reviewer for pointing out the analysis of ACC and Gapping in [14].

> (1) The interpretation of an elliptical construction is obtained by uniformly substituting its immediate constituents into some immediately preceding structure, and computing the interpretation of the results. [14, p. 162, (119)]

We indeed share the underlying idea of picking and substituting of 'immediate constituents' into the coordinated material (understood at some level as an elliptical construction). The proposal of [14] remained rather informal; the present paper may be seen as an attempt to formalize the idea, as well as to extend it to scope phenomena.

There have been other attempts to solve the KLM problem without directional types (within the ACG-like formalisms). Kanazawa [5] proposes 'regular constraints' to prevent over-generation (which recall structural constraints in Government and Binding). This amounts

however to duplication of lexical entries. The approach [13] reins in the over-generation using subtyping. Either proposal can be classified as 'proof search' rather than computational like TS; in case of [13] with no guarantees that the proof search ever terminates (and, as the authors admitted, no good way to characterize the space of available derivations and detect over-generation).

5 Conclusions

We have demonstrated the transformational analyses of RNR and Gapping. The analyses make precise various eliding schemas, demanding type preservation. The asymmetry of the type of $\mathsf{and}_{(\mathsf{NP},\mathsf{TV}),\mathsf{S}}$ and similar constants is what lets us answer the Kubota, Levine and Moot challenge: how to prevent over-generation in analyses of RNR and gapping without directional types.

The idiosyncrasies of coordination are distilled to the ad hoc choice of constants $\mathsf{and}_{\mathsf{XY}}$. There are transformations for some types XY but not for the others. There may be a pattern there. Collecting the arbitrariness in one place might make the pattern easier to find. Being able to handle the entire ellipsis part of the FraCaS corpus seems the natural first step in searching for that pattern.

It is interesting to consider interpreting the "sequence of words" as a discontinuous sentence in the sense of Morrill [12].

Another future work task is to apply TS to more complicated scoping phenomena including 'same', 'different', 'the total of' – as well as to various wh-movement phenomena.

Acknowledgments I am very grateful to Leo Tingchen Hsu for numerous perceptive and stimulating discussions. I thank an anonymous reviewer for many very insightful and helpful comments. Numerous discussions with Yusuke Kubota, Bob Levine, Alastair Butler, Greg Kobele and the participants of the workshop "New Landscapes in Theoretical Computational Linguistics" (Ohio State University, October 14-16, 2016) are gratefully acknowledged.

Bibliography

[1] Alastair Butler. *Linguistic Expressions and Semantic Processing - A Practical Approach*. Springer, 2015.

[2] Noam Chomsky. *Aspects of a Theory of Syntax*. MIT Press, Cambridge, Mass., 1965.

[3] Noam Chomsky. *Lectures on Government and Binding*. Foris, Dordrecht, 1981.

[4] Noam Chomsky. *The Minimalist Program*. The MIT Press, Cambridge, Massachusetts, 1995.

[5] Makoto Kanazawa. Syntactic features for regular constraints and an approximation of directional slashes in abstract categorial grammars. In Kubota and Levine [11], pages 34–70.

[6] Oleg Kiselyov. Canonical constituents and non-canonical coordination - simple categorial grammar account. volume 9067 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2014.

[7] Oleg Kiselyov. Applicative abstract categorial grammars in full swing. In *Proc. LENLS 12*. November 2015.

[8] Yusuke Kubota and Robert Levine. Gapping as like-category coordination. In Denis Béchet and Alexander Dikovsky, editors, *Logical Aspects of Computational Linguistics: 7th International Conference*, pages 135–150. Springer, 2012.

[9] Yusuke Kubota and Robert Levine. Gapping as hypothetical reasoning. To appear in *Natural Language and Linguistic Theory*, available at `http://ling.auf.net/lingbuzz/002123`, 2014.

[10] Yusuke Kubota and Robert Levine. Against ellipsis: Arguments for the direct licensing of 'non-canonical' coordinations. *Linguistics and Philosophy*, 38(6):521–576, 2015.

[11] Yusuke Kubota and Robert Levine, editors. *Proceedings for ESSLLI 2015 Workshop 'Empirical Advances in Categorial Grammar'*. University of Tsukuba and Ohio State University, 2015.

[12] Glyn Morrill, Oriol Valentín, and Mario Fadda. The displacement calculus. *Journal of Logic, Language, and Information*,

20(1):1–48, 2011.

[13] Carl Pollard and Chris Worth. Coordination in linear categorial grammar with phenogrammatical subtyping. In Kubota and Levine [11], pages 162–182.

[14] Ivan A. Sag, Gerald Gazdar, Thomas Wasow, and Steven Weisler. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory*, 3(2):117–171, 1985.