A revised version is to appear in: D. Béchet and A. Dikovsky (eds.), *Logical Aspects of Computational Linguistics (LACL 2012)*, Lecture Notes in Artificial Intelligence Vol. 7351, pp. xxx-xxx, Springer, Berlin, Heidelberg, 2012.

CoTAGs and ACGs

Gregory M. Kobele[1] and Jens Michaelis[2]

[1] University of Chicago, Chicago, Illinois, USA
[2] Bielefeld University, Bielefeld, Germany

Abstract. Our main concern is to provide a complete picture of how *coTAGs*, as a particular variant within the general framework of *tree adjoining grammars (TAGs)*, can be captured under the notion of *abstract categorial grammars (ACGs)*. coTAGs have been introduced by Barker [1] as an "alternative conceptualization" in order to cope with the tension between the TAG-mantra of the "locality of syntactic dependencies" and the seeming non-locality of quantifier scope. We show how our formalization of Barker's proposal leads to a class of higher order ACGs. By taking this particular perspective, Barker's proposal turns out as a straightforward extension of the proposal of Pogodalla [11], where the former in addition to "simple" inverse scope phenomena also captures *inverse linking* and *non-inverse linking* phenomena.

1 Introduction

In [1], Barker sketches an alternative approach to semantics in the *tree adjoining grammar (TAG)*-framework,[3] which he refers to as *coTAG*-approach, after the extension of the modified *substitution* operation, *cosubstitution*. The presentation in [1] is simple and intuitive, tying the scope of a quantificational noun phrase together with its position in the derivation. The approach is presented via examples, however, and is not formalized. As far as the syntactic component is concerned, it does, interestingly, preserve the weak and strong generative capacity of "classical" TAGs, though, clearly, not the derivation sets, as we will make precise here.

Our contribution in this paper is to formalize coTAGs. We use the formalism of *abstract categorial grammars (ACGs)* [3], which has already been used to formalize TAGs in, e.g., [4, 10, 11]. We present a mapping from coTAG-lexica to ACGs, and demonstrate that it coincides with the informal examples from Barker's paper. The mapping has as a special case the one of de Groote and Pogodalla for TAGs. Of particular interest is the fact that our mapping reveals Barker's treatment of scope taking in coTAGs to be largely identical to the TAG-inspired ACG-analysis of Pogodalla [11].

The paper is structured as follows. We begin with an (informal) introduction to coTAGs in Sec. 2. Then, in Sec. 3, we present abstract categorial grammars, which we use to formalize coTAGs in Sec. 4. Section 5 is the conclusion.

[3] We assume the reader to be familiar with the basics of the TAG-framework. See e.g. [6] or [7] for an introduction to TAGs.

2 CoTAGs

For A a set of *atomic types*, $\mathcal{T}(A)$ is the set of *types over* A, the smallest superset of A closed under pair formation, i.e., $A \subseteq \mathcal{T}(A)$, and if $\alpha, \beta \in \mathcal{T}(A)$ then $(\alpha\,\beta) \in \mathcal{T}(A)$. We sometimes omit outer parentheses when writing types. We sometimes write $\alpha_1\,\alpha_2\,\alpha_3$ instead of $\alpha_1(\alpha_2\,\alpha_3)$, and $\alpha_1 \cdots \alpha_{k+1}$ instead of $\alpha_1(\alpha_2 \cdots \alpha_{k+1})$ for $k \geq 2$ and types $\alpha_1, \ldots, \alpha_{k+1} \in \mathcal{T}(A)$, where $\alpha_1(\alpha_2 \cdots \alpha_3) = \alpha_1(\alpha_2\,\alpha_3)$. We write $\alpha^{k+1}\,\beta$ instead of $\alpha\,\alpha^k\beta$ for $k \geq 1$ and types α, $\beta \in \mathcal{T}(A)$, where $\alpha^1 = \alpha$.

For $\mathtt{Cat}$ a set of *categories*, i.e., a set of *non-terminals* in the sense of a classical TAG, let $\tau_{\mathtt{Cat}} : \mathtt{Cat} \to \mathcal{T}(A_{\mathtt{Cat}})$ be a type assigning function with $A_{\mathtt{Cat}}$ a set of atomic types. $\tau_{\mathtt{Cat}}$ uniquely determines the function $\widehat{\tau}_{\mathtt{Cat}}$ given next, which assigns a type from $\mathcal{T}(A_{\mathtt{Cat}})$ to each $\delta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*\{\downarrow\}^?$, where $\uparrow$ and $\downarrow$ are two distinct new symbols not appearing in $\mathtt{Cat}$.[4] For each $\delta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*\{\downarrow\}^?$, we use its boldface variant, $\boldsymbol{\delta}$, to denote $\widehat{\tau}_{\mathtt{Cat}}(\delta)$: if $\delta \in \mathtt{Cat}$, we have $\boldsymbol{\delta} = \tau_{\mathtt{Cat}}(\delta)$. If $\delta = \zeta\downarrow$ for some $\zeta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$, we have $\boldsymbol{\delta} = \boldsymbol{\zeta}$. If $\delta = \zeta\uparrow\gamma$ for some $\zeta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$ and $\gamma \in \mathtt{Cat}$, we have $\boldsymbol{\delta} = ((\boldsymbol{\zeta}\,\boldsymbol{\gamma})\,\boldsymbol{\gamma})$, and we often write $\boldsymbol{\zeta\uparrow\gamma}$ instead of $((\boldsymbol{\zeta}\,\boldsymbol{\gamma})\,\boldsymbol{\gamma})$ in this case.

A *coTAG* can be defined as an octuple $\langle V_T, \mathtt{Cat}, A_{\mathtt{Cat}}, \widehat{\tau}_{\mathtt{Cat}}, \mathcal{I} \times \mathcal{I}', \mathcal{A} \times \mathcal{A}', \frown, \mathrm{S}\rangle$, where V_T, $\mathtt{Cat}$ and $A_{\mathtt{Cat}}$ are a set of *terminals*, a set of *categories* and a set of *atomic types*, respectively. $\widehat{\tau}_{\mathtt{Cat}}$ is the extension to the domain $\mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*\{\downarrow\}^?$ of a type assigning function $\tau_{\mathtt{Cat}} : \mathtt{Cat} \to \mathcal{T}(A_{\mathtt{Cat}})$ in the above sense. S is a distinguished category from $\mathtt{Cat}$, the *start symbol*. The set $(\mathcal{I} \times \mathcal{I}') \cup (\mathcal{A} \times \mathcal{A}')$ is a finite set of pairs of finite labeled trees which supplies the *lexical entries* of G.

The first component of a pair $\langle \alpha_{\mathrm{syn}}, \alpha_{\mathrm{sem}} \rangle \in (\mathcal{I} \times \mathcal{I}') \cup (\mathcal{A} \times \mathcal{A}')$ provides the syntactic, the second component the semantic representation of the corresponding lexical entry. Nodes in α_{syn} are linked to nodes in α_{sem} by the relation $\frown$. An operation applying to some node in α_{syn} must be accompanied by a parallel operation applying to the linked node(s) in α_{sem}. Regarding the projection to the first component of this general connection between syntactic and semantic representations, coTAGs are nearly identical to classical TAGs: labels of inner nodes are always from $\mathtt{Cat}$. The labels of the root and the foot node of a syntactic coTAG-auxiliary tree are, as in the regular TAG-case, from $\mathtt{Cat}$ and $\mathtt{Cat}\{^\star\}$, respectively, and up to the foot node indicating *-suffix both labels coincide. But whereas the root nodes of TAG-initial trees and the substitution nodes of arbitrary TAG-trees are labeled with elements from $\mathtt{Cat}$ and $\mathtt{Cat}\{\downarrow\}$, respectively, the root nodes of syntactic coTAG-initial trees and the substitution nodes of arbitrary syntactic coTAG-trees have labels from the sets $\mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$ and $\mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*\{\downarrow\}$, respectively.[5] That is to say, except for the differences with respect to the possible labeling of roots and substitution nodes, the 5-tuple

[4] For any set A, A^* is the set of all finite strings over A, including ϵ, the empty string. We identify A with the set of strings of length 1 over A, and take $A^?$ to denote an optional occurrence of an element from A (in a string), i.e., the set of strings $A \cup \{\epsilon\}$.

[5] For $\delta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$ and $\gamma \in \mathtt{Cat}$, we write $\delta\uparrow$ instead of $\delta\uparrow\gamma$, and $\boldsymbol{\delta}\uparrow$ instead of $\boldsymbol{\delta\uparrow\gamma}$ in case $\gamma = \mathrm{S}$.

$\langle V_T, \mathtt{Cat}, \mathcal{I}, \mathcal{A}, \mathrm{S}\rangle$ constitutes a classical TAG, in particular providing a set of *initial elementary* trees, $\mathcal{I}$, and a set of *auxiliary elementary* trees, $\mathcal{A}$.

In a somewhat relaxed sense, the projection to the second component provides a TAG as well. Given a lexical entry $\langle \alpha_{\mathrm{syn}}, \alpha_{\mathrm{sem}}\rangle \in (\mathcal{I} \times \mathcal{I}') \cup (\mathcal{A} \times \mathcal{A}')$, the relation $\frown$ links the root node of α_{syn} to the root node of α_{sem}, terminal nodes to terminal nodes, substitution nodes to substitution nodes and, in case α_{syn} is an auxiliary tree, the foot node to the foot node. For these nodes it holds that, if $\delta \in \mathtt{Cat}(\{\downarrow\}\mathtt{Cat})^*\{\uparrow\}^?$ is the syntactic label then $\boldsymbol{\delta} = \widehat{\tau}_{\mathtt{Cat}}(\delta)$ is the semantic label of the linked node, if $w \in V_T$ is the syntactic label then $w \in V_T$ is the semantic label too. More concretely, up to a certain extent, we may think of the 5-tuple $\langle V'_T, \mathcal{T}(A_{\mathtt{Cat}}), \mathcal{I}', \mathcal{A}', \mathbf{S}\rangle$ as a TAG, where $V'_T = V_T \cup \mathcal{X} \cup \mathit{Con} \cup \{\lambda\}$ with $\mathcal{X}$ being a denumerable set of variables, and Con being a set which consists at least of the "usual" logical connectives and quantifiers of FOL.

For each $\alpha_{\mathrm{sem}} \in \mathcal{I}' \cup \mathcal{A}'$ the node-labeling is constrained via $\widehat{\tau}_{\mathtt{Cat}}$ by the relation linking nodes of α_{sem} to those of the syntactic tree paired to α_{sem} in the lexicon. The "plain" yield of a α_{sem} is a string over $V'_T \cup \mathcal{T}(A_{\mathtt{Cat}})$. By definition we additionally demand the hierarchical tree structure to uniquely determine a closed well-typed lambda-term associated with the yield: if ν is an interior node in α_{sem} then the label of ν, $\mathit{label}(\nu)$ is a type from $\mathcal{T}(A_{\mathtt{Cat}})$, and ν has either one, two or three children. We distinguish these cases as i.1), i.2) and i.3), respectively.

i.1) Let μ be the single child of ν. μ's label, $\mathit{label}(\mu)$, is always in V'_T. $\mathit{label}(\nu)$ is virtually determining the type of $\mathit{label}(\mu)$, and for each node ν' of some $\alpha' \in \mathcal{I}' \cup \mathcal{A}'$ with a child of ν' labeled $\mathit{label}(\mu)$, $\mathit{label}(\nu) = \mathit{label}(\nu')$ holds.

i.2) Both children have a label from $\mathcal{T}(A_{\mathtt{Cat}})$. Say, the first child, μ_1, has label $\mathit{label}(\mu_1)$, and the second, μ_2, label $\mathit{label}(\mu_2)$. The labels are compatible with functional application in the sense that $\mathit{label}(\mu_1) = \mathit{label}(\mu_2)\mathit{label}(\nu)$ holds.

i.3) The first child, μ_0, is labeled λ. The second child, μ_1, and the third child, μ_2, are labeled by $\mathit{label}(\mu_1)$ and $\mathit{label}(\mu_2)$, respectively. As in i.2), $\mathit{label}(\mu_1)$ and $\mathit{label}(\mu_2)$ are from $\mathcal{T}(A_{\mathtt{Cat}})$, but now they virtually cope for lambda abstraction, i.e., $\mathit{label}(\nu) = (\mathit{label}(\mu_1)\mathit{label}(\mu_2))$ holds. In addition, μ_1, is necessarily dominating a single node labeled by a variable $x \in \mathcal{X}$. The logical scope of the thus virtually instantiated lambda abstraction over x is given by the subtree of ν rooted in μ_2.

If ν is a terminal node in $\alpha_{\mathrm{sem}} \in \mathcal{I}' \cup \mathcal{A}'$ and ν's label, $\mathit{label}(\nu)$, is from V'_T, ν is always the leftmost child of its parent node. In case $\mathit{label}(\nu) = \lambda$, ν has exactly two sister nodes. In case $\mathit{label}(\nu) \in V'_T - \{\lambda\}$, ν is the unique child of its parent. If ν is a terminal node in α_{sem} and its label is not from V'_T, then its label is from $\mathcal{T}(A_{\mathtt{Cat}})$.

In order to arrive at a closed well-typed lambda-term associated with the yield of some $\alpha_{\mathrm{sem}} \in \mathcal{I}' \cup \mathcal{A}'$, we have to replace each leaf-label from $\mathcal{T}(A_{\mathtt{Cat}})$ (i.e., each label of some substitution node and, if existing, the foot node) by a fresh variable of the corresponding type, and to lambda-abstract over this variable again.

As a "consequence" of the parallels between a TAG- and a coTAG-lexicon, the metaphor of constructing expressions in a coTAG is very similar to how it

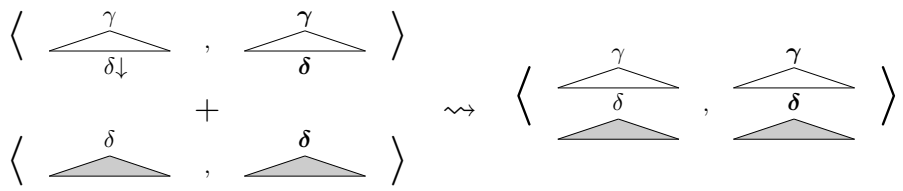

Fig. 1. *Substitution* schematically: syntactically, tree with root-label δ is substituted at leaf labeled $\delta\downarrow$ in tree with root-label γ; while semantically, the corresponding tree with root-label $\boldsymbol{\delta}$ is substituted at leaf labeled $\boldsymbol{\delta}$ in the corresponding tree with root-label $\boldsymbol{\gamma}$.

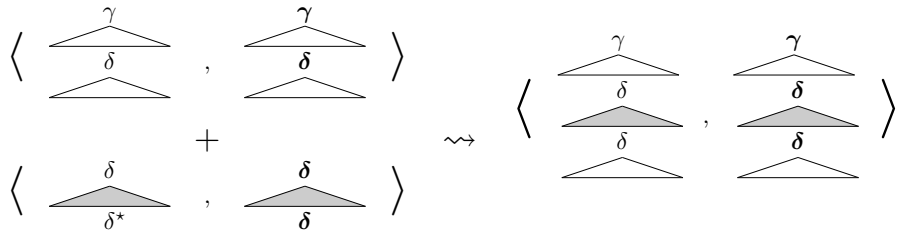

Fig. 2. *Adjoining* schematically: syntactically, tree with root-label δ and foot node-label $\delta^\star$ is adjoined at interior node labeled δ in tree with root-label γ; while semantically, the corresponding tree with root-label $\boldsymbol{\delta}$ and foot node-label $\boldsymbol{\delta}$ is adjoined at interior node labeled $\boldsymbol{\delta}$ in the corresponding tree with root-label $\boldsymbol{\gamma}$.

is in regular TAGs with one important difference. More concretely, we have the operations of *substitution* and *adjunction* in the "classical" sense, cf. Fig. 1 and 2. But in addition a derived structure with syntactic root-label $\delta\uparrow\gamma$ for some $\delta \in \texttt{Cat}(\{\uparrow\}\texttt{Cat})^*$ and $\gamma \in \texttt{Cat}$, can be *cosubstituted* into a derived structure with syntactic root-label γ at a leaf labeled $\delta\downarrow$ *at any point in the derivation.* Within the resulting semantic representation, new terminal leaves are introduced labeled by λ and a variable x. The operation is set up such that x is chosen to be "fresh," cf. Fig. 3.

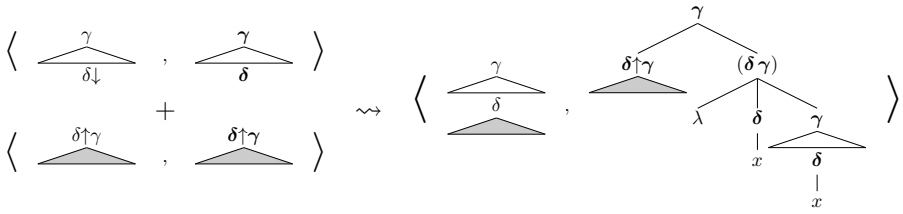

Fig. 3. *Cosubstitution* schematically: syntactically, tree with root-label $\delta\uparrow\gamma$ is cosubstituted at leaf labeled $\delta\downarrow$ in tree with root-label $\gamma \in \texttt{Cat}$; while semantically, the corresponding tree with root-label $\boldsymbol{\delta\uparrow\gamma}$ is "quantified in" at the root of the corresponding tree with root-label $\boldsymbol{\gamma}$.

The semantic result of applying two cosubstitution steps depends on the order in which those steps are applied, no matter whether the two steps do not derivationally depend on each other. Accordingly, we require a novel notion of derivation to represent this "timing" information. An ACG-based representation of this notion we will be concerned with in Sec. 4. We defer a somewhat more detailed semantic analysis to that section, but emphasize that—in the same way as outlined above for elementary semantic trees—for any given derived semantic tree, closed well-typed lambda-terms may be read off "left to right" from the yield taking into account the hierarchical tree structure.

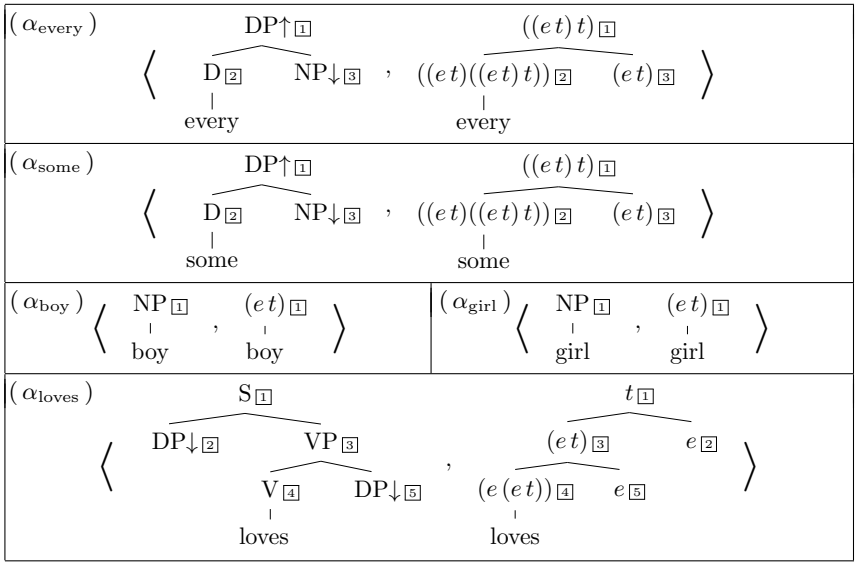

Fig. 4. The example coTAG G_{scope} (part 1)

As a concrete example, consider the grammar G_{scope} presented in Fig. 4 and 5, where S is supposed to be the start symbol, and e and t the atomic types.[6] Part 1 of G_{scope}, as given in Fig. 4, allows deriving the sentence *every boy loves some girl*. One can start deriving this sentence either by substituting *boy* into *every*, and then cosubstituting *every boy* into the subject position of *loves* as shown in Fig. 6, or by substituting *girl* into *some*, and then cosubstituting *some girl* into the object position of *loves* as shown in Fig. 7. Both complete derived pairs of structures for the sentence are given in Fig. 8. As desired, the derived syntactic surface trees are identical.

Part 2 of G_{scope}, as given in Fig. 5, would in principle allow us to derive an NP with an embedded PP like, e.g., *senator from every city* in two different ways

[6] Links will usually be marked with diacritics of the form $\boxed{n}$ for some $n \geq 1$. We may occasionally avoid explicitly mentioning the links between nodes of the syntactic and the semantic representation, when we think the canonical linking is obvious.

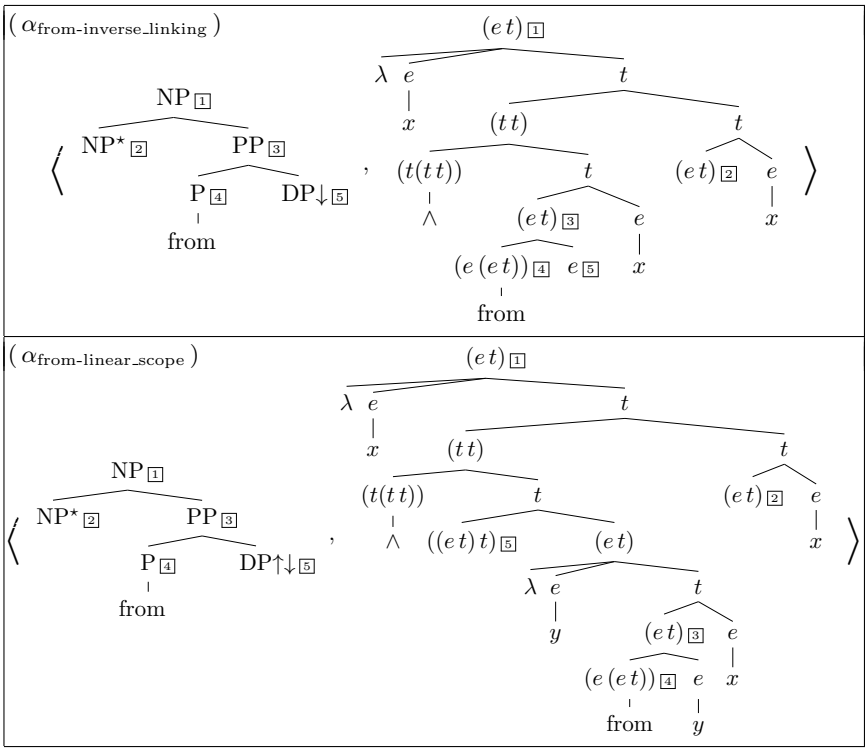

Fig. 5. The example coTAG G_{scope} (part 2)

providing us with the two different scope readings in a sentence like *a senator from every city sleeps*: the *inverse linking*-reading and the *linear scope*-reading. Note that, though not strictly linear, the lambda-terms (implicitly) associated with the semantic component are still *almost linear*, since the variables appearing more than once are dominated by an atomic type.[7]

3 Abstract Categorial Grammars

An *abstract categorial grammar (ACG)* [3] is a quadruple $\langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ consisting of an "alphabet" Σ_1 from which underlying structures are determined, an "alphabet" Σ_2 from which possible surface structures are determined, a pair of mappings $\mathcal{L}$ realizing underlying structures as surface structures, and a distinguished "start" symbol s provided by Σ_1. Particular to ACGs is that underlying

Also recall our convention to write $\delta\uparrow$ instead of $\delta\uparrow$S, and $\boldsymbol{\delta\uparrow}$ instead of $\boldsymbol{\delta\uparrow}\mathbf{S}$ for $\delta \in \mathtt{Cat}(\{\uparrow\}\mathtt{Cat})^*$ and the start symbol S.

[7] More concretely, the notion of *almost linear* employed here is the same as used by Kanazawa [8]: a lambda-term is *almost linear* if it is a lambda I-term such that any variable occurring free more than once in any subterm has an atomic type.

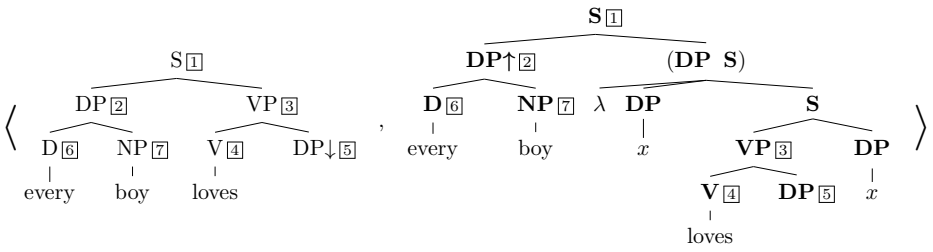

Fig. 6. Cosubstituting *every boy* into the subject position before filling the object position of *loves*: derived syntactic and semantic tree.

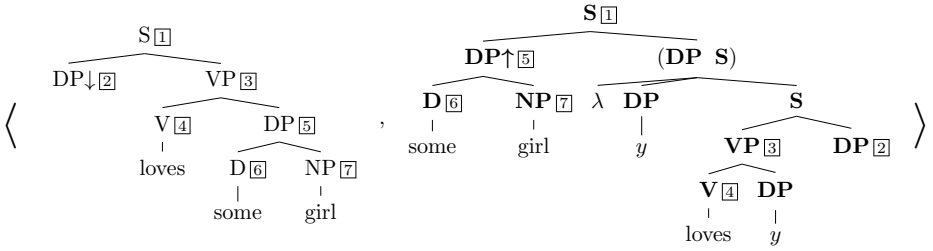

Fig. 7. Cosubstituting *some girl* into the object position before filling the subject position of *loves*: derived syntactic and semantic tree.

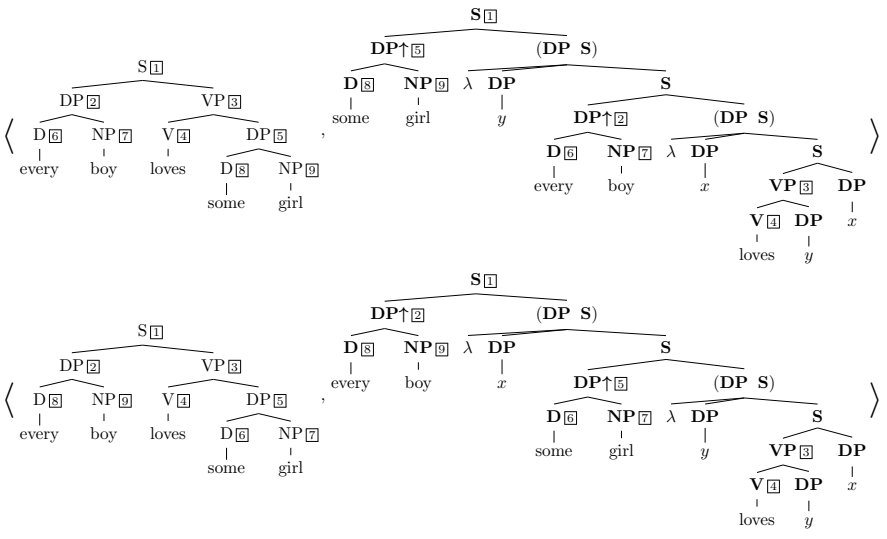

Fig. 8. The two complete derived pairs of structures of *every boy loves some girl*.

and surface structures are given as (almost) linear terms of the simply typed lambda calculus.[8] Accordingly, the "alphabets" are higher-order signatures of the form $\Sigma = \langle A, C, \tau \rangle$, where A is a finite set of *atomic types*, C is a finite set of *constants*, and $\tau : C \to \mathcal{T}(A)$ assigns a type to each constant.[9] Given a denumerably infinite set $\mathcal{X}$ of variables, we define a *(type) environment* to be a partial, finite mapping $\Gamma : \mathcal{X} \to \mathcal{T}(A)$, which we typically write as a list $x_1 : \alpha_1, \ldots, x_n : \alpha_n$. Given environments Γ and Δ, the composite type environment Γ, Δ is defined iff their domains are (almost) disjoint, in which case Γ, Δ is the union of Γ and Δ.[10] For $\Lambda(\Sigma)$, the set of *(untyped) lambda-terms* build on Σ,[11] we say that a term $M \in \Lambda(\Sigma)$ has type $\alpha \in \mathcal{T}(A)$ in environment Γ (written $\Gamma \vdash_\Sigma M : \alpha$) just in case it is derivable using the inference rules in Figure 9.

$$\vdash_\Sigma c : \tau(c) \qquad \text{for } c \in C \qquad\qquad x : \alpha \vdash_\Sigma x : \alpha \qquad \text{for } x \in \mathcal{X} \text{ and } \alpha \in \mathcal{T}(A)$$

$$\frac{\Gamma, x : \alpha \vdash_\Sigma M : \beta}{\Gamma \vdash_\Sigma (\lambda x.M) : (\alpha\beta)} \qquad\qquad \frac{\Gamma \vdash_\Sigma M : (\alpha\beta) \quad \Delta \vdash_\Sigma N : \alpha}{\Gamma, \Delta \vdash_\Sigma (MN) : \beta}$$

Fig. 9. ACG-inference rules

Given an ACG $G = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ with $\Sigma_i = \langle A_i, C_i, \tau_i \rangle$ and $s \in A_1$, the *abstract language* of G, $A(G)$, is the set $\{M \mid \vdash_{\Sigma_1} M : s\}$. The *object language* of G, $O(G)$, is the set $\{\hat{\theta}(M) \mid M \in A(G)\}$. More concretely, $O(G)$ is the image of $A(G)$ under the pair of mappings $\mathcal{L} = \langle \sigma, \theta \rangle$, referred to as the *lexicon* from Σ_1 to Σ_2, which meets the conditions:

1. $\sigma : A_1 \to \mathcal{T}(A_2)$ is the kernel of the type substitution $\hat{\sigma} : \mathcal{T}(A_1) \to \mathcal{T}(A_2)$ obeying $\hat{\sigma} \restriction A_1 = \sigma$, and $\hat{\sigma}((\alpha\beta)) = (\hat{\sigma}(\alpha)\hat{\sigma}(\beta))$
2. $\theta : C_1 \to \Lambda(\Sigma_2)$ specifies $\hat{\theta} : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ by setting $\hat{\theta}(c) = \theta(c)$, $\hat{\theta}(x) = x$, $\hat{\theta}(MN) = \hat{\theta}(M)\hat{\theta}(N)$, and $\hat{\theta}(\lambda x.M) = \lambda x.\hat{\theta}(M)$
3. $\vdash_{\Sigma_2} \theta(c) : \hat{\sigma}(\tau_1(c))$ for all $c \in C_1$

By abuse of notation, $\mathcal{L}$ will usually be used to denote both σ and θ, and also their respective extensions $\hat{\sigma}$ and $\hat{\theta}$.

---

[8] As to the notion of an *almost linear lambda-term* cf. fn. 7.

[9] Recall that $\mathcal{T}(A)$ is the smallest superset of A closed under pair formation, i.e., $A \subseteq \mathcal{T}(A)$, and if $\alpha, \beta \in \mathcal{T}(A)$ then $(\alpha\,\beta) \in \mathcal{T}(A)$.

[10] *Almost disjoint* is meant to denote the restriction that, if $x : \alpha$ belongs to the intersection $\Gamma \cap \Delta$ then the type assigned to x is atomic, i.e., $\alpha \in A$.

[11] That is, $\Lambda(\Sigma)$ is the smallest set such that $C \cup \mathcal{X} \subseteq \Lambda(\Sigma)$, and such that $(MN) \in \Lambda(\Sigma)$, and $(\lambda x.M) \in \Lambda(\Sigma)$ for $M, N \in \Lambda(\Sigma)$ and $x \in \mathcal{X}$. We omit outer parentheses when writing lambda-terms, and we write $M_1\,M_2\,M_3$ instead of $(M_1\,M_2)\,M_3$, and $M_1 \cdots M_{n+1}$ instead of $(M_1 \cdots M_n)M_{n+1}$ for $n \geq 2$ and $M_1, \ldots, M_{n+1} \in \Lambda(\Sigma)$, where $(M_1 \cdots M_2) = M_1M_2$. We write $\lambda x.M\,N$ instead of $\lambda x.(M\,N)$, $\lambda x_1 x_2.M$ instead of $\lambda x_1.\lambda x_2.M$, and $\lambda x_1 \ldots x_{n+1}.M$ instead of $\lambda x_1.\lambda x_2 \ldots \lambda x_n.M$ for $n \geq 2$, $x, x_1, \ldots, x_n \in \mathcal{X}$ and $M, N \in \Lambda(\Sigma)$, where $\lambda x_2 \ldots \lambda x_2.M = \lambda x_2.M$.

An ACG G belongs to the class $\mathcal{ACG}(m,n)$ iff the maximal order of the types of any of its abstract constants is less than or equal to m, and the maximal order of the type assigned to any constant by the lexicon is less than or equal to n. The order of a type α, $\mathrm{ord}(\alpha)$, is given recursively: For each atomic type a, $\mathrm{ord}(a) = 1$. For each two types α and β, $\mathrm{ord}((\alpha\beta))$ is the greater of $\mathrm{ord}(\beta)$ and $\mathrm{ord}(\alpha) + 1$. For each m, $\mathcal{ACG}(m)$ denotes the class of all m^{th}-order ACGs, i.e., $\mathcal{ACG}(m) = \bigcup_{n\geq 1} \mathcal{ACG}(m,n)$.

As shown by Salvati [12], if only ACGs of the form $\langle \Sigma_1, \Sigma_2, \mathcal{L}, s\rangle$ are considered where Σ_2 is a string signature,[12] $\mathcal{ACG}(2)$ derives exactly the string languages generated by set-local multicomponent TAGs, or likewise, a series of other weakly equivalent grammar formalisms. Kanazawa [9] proves that, similarly, if Σ_2 is a tree signature,[13] $\mathcal{ACG}(2)$ derives exactly the tree languages generated by, e.g., *context-free graph grammars* [2], or likewise, *hyperedge replacement grammars* [5].

One of the advantages of ACGs is that they provide a logical setting in which the abstract language can be used as a specification of the derivation set of a grammar instantiation of some grammar formalism, and that by applying two different lexicons to the abstract language, we can "simultaneously" obtain a syntactic object language and a semantic object language.

4 CoTAGs as ACGs

In order to translate a coTAG into an ACG, we build on the methods previously developed by de Groote and Pogodalla. De Groote [4] has shown how a regular TAG can be translated into a second-order ACG on trees. Pogodalla [10, 11] has shown how those techniques can be extended in order to define a third-order ACG which, "in parallel" to the syntactic derivation, by means of a different semantic object language provides the two different scope readings in a sentence like *every boy likes some girl*.

Let $G_{\text{coTAG}} = \langle V_T, \mathtt{Cat}, A_{\mathtt{Cat}}, \widehat{\tau}_{\mathtt{Cat}}, \mathcal{I}\times\mathcal{I}', \mathcal{A}\times\mathcal{A}', \frown, \mathrm{S}\rangle$ be a coTAG.

For expository reasons we assume that no node-label of any tree from $\mathcal{I}\cup\mathcal{A}$ is of the form $\mathrm{S}{\uparrow}\gamma$ or $\mathrm{S}{\uparrow}\gamma{\downarrow}$ for some $\gamma \in \mathtt{Cat}(\{\uparrow\}\mathtt{Cat})^*$. In other words, the simple category S "is never lifted."

First let $\Sigma_{\text{abs}} = \langle A_{\text{abs}}, C_{\text{abs}}, \tau_{\text{abs}}\rangle$ be the higher-order signature, where

$$A_{\text{abs}} = \{\boldsymbol{\delta}^\bullet, \boldsymbol{\delta}^\bullet_{\mathrm{A}} \mid \delta \in \mathtt{Cat}\}$$
$$\cup \left\{\boldsymbol{\zeta{\uparrow}\delta}^\bullet, \boldsymbol{\zeta}^\bullet, \boldsymbol{\delta}^\bullet \;\middle|\; \begin{array}{l}\zeta \in \mathtt{Cat}(\{\uparrow\}\mathtt{Cat})^* \text{ and } \delta \in \mathtt{Cat} \text{ with}\\ \zeta{\uparrow}\delta \text{ or } \zeta{\uparrow}\delta{\downarrow} \text{ labeling a node of some } \alpha \in \mathcal{I}\cup\mathcal{A}\end{array}\right\},^{14}$$
$$C_{\text{abs}} = \{id_X \mid X \in \mathtt{Cat}\} \cup \{\overline{\alpha} \mid \alpha \in \mathcal{I}\cup\mathcal{A}\}$$
$$\cup \{\overline{\overline{\alpha}} \mid \alpha \in \mathcal{I} \text{ and } \alpha\text{'s root-label belongs to } \mathtt{Cat}\{\uparrow\}\mathtt{Cat}(\{\uparrow\}\mathtt{Cat})^*\}$$

[12] That is, Σ_2 consists of the single atomic type o and assigns to each alphabet constant a the type $(o\,o)$

[13] That is, Σ_2 consists of the single atomic type o and assigns to each constant a of "rank" n the type o^{n+1}, where $o^1 = o$ and $o^{n+1} = (o\,o^n)$

[14] Following the notational convention introduced above, we write $\boldsymbol{\delta{\uparrow}}^\bullet$ instead of $\boldsymbol{\delta{\uparrow}\mathrm{S}}^\bullet$ for $\delta \in \mathtt{Cat}(\{\uparrow\}\mathtt{Cat})^*$ and the start symbol S.

In order to make the typing function τ_{abs} more precise, consider $\alpha \in \mathcal{I} \cup \mathcal{A}$ with root-label $\gamma \in \mathit{Cat}(\{\uparrow\}\mathit{Cat})^*$. Let $\nu_1, \ldots, \nu_m$ be the interior nodes of α except for the root, and let $\nu_{m+1}, \ldots, \nu_{m+n}$ be the substitution sites of α.[15] Furthermore, let $\gamma_i \in \mathit{Cat}$ be the label of ν_i for $1 \leq i \leq m$, and let $\delta_i{\downarrow}$ be the label of ν_{m+i} for $1 \leq i \leq n$, where $\delta_i \in \mathit{Cat}(\{\uparrow\}\mathit{Cat})^*$. The type of the constant $\overline{\alpha}$ is defined as follows: for $\alpha \in \mathcal{I}$, let $\tau_{\text{abs}}(\overline{\alpha}) = \boldsymbol{\gamma}_{\text{A}}^{\bullet}\boldsymbol{\gamma}_{1\text{A}}^{\bullet} \cdots \boldsymbol{\gamma}_{m\text{A}}^{\bullet}\boldsymbol{\delta}_1^{\bullet} \cdots \boldsymbol{\delta}_n^{\bullet}\boldsymbol{\gamma}^{\bullet}$. For $\alpha \in \mathcal{A}$, let $\tau_{\text{abs}}(\overline{\alpha}) = \boldsymbol{\gamma}_{\text{A}}^{\bullet}\boldsymbol{\gamma}_{1\text{A}}^{\bullet} \cdots \boldsymbol{\gamma}_{m\text{A}}^{\bullet}\boldsymbol{\delta}_1^{\bullet} \cdots \boldsymbol{\delta}_n^{\bullet}\boldsymbol{\gamma}_{\text{A}}^{\bullet}\boldsymbol{\gamma}_{\text{A}}^{\bullet}$. If $\alpha \in \mathcal{I}$, and if $\gamma = \zeta{\uparrow}\delta$ for some $\zeta \in \mathit{Cat}(\{\uparrow\}\mathit{Cat})^*$ and $\delta \in \mathit{Cat}$, the type of the additionally existing constant $\overline{\overline{\alpha}}$ is defined in the following way:[16] $\tau_{\text{abs}}(\overline{\overline{\alpha}}) = \boldsymbol{\gamma}_{1\text{A}}^{\bullet} \cdots \boldsymbol{\gamma}_{m\text{A}}^{\bullet}\boldsymbol{\delta}_1^{\bullet} \cdots \boldsymbol{\delta}_n^{\bullet}\boldsymbol{\gamma}^{\circ}$.[17] For each $X \in \mathit{Cat}$, the type of the constant id_X is defined by $\tau_{\text{abs}}(id_X) = \boldsymbol{X}_{\text{A}}^{\bullet}$.

In the way the abstract constants resulting from elementary trees are typed, adjunction at the adjunction sites of an elementary tree becomes obligatory. The abstract constants id_X for $X \in \mathit{Cat}$ have been introduced to provide the possibility of "vacuous" adjunction.

As to our example coTAG G_{scope}, the abstract typing function τ_{abs} assigns the following types to the abstract constants resulting from the lexical entries:[18]

$$\tau_{\text{abs}}(\overline{\overline{\alpha}}_{\text{every}}) = \mathbf{D}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}\ (\mathbf{DP}^{\bullet}\ \mathbf{S}^{\bullet})\ \mathbf{S}^{\bullet} \qquad \tau_{\text{abs}}(\overline{\alpha}_{\text{every}}) = \mathbf{D}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}\ \mathbf{DP}{\uparrow}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\overline{\alpha}}_{\text{some}}) = \mathbf{D}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}\ (\mathbf{DP}^{\bullet}\ \mathbf{S}^{\bullet})\ \mathbf{S}^{\bullet} \qquad \tau_{\text{abs}}(\overline{\alpha}_{\text{some}}) = \mathbf{D}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}\ \mathbf{DP}{\uparrow}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\alpha}_{\text{boy}}) = \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\alpha}_{\text{girl}}) = \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{NP}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\alpha}_{\text{loves}}) = \mathbf{S}_{\text{A}}^{\bullet}\ \mathbf{VP}_{\text{A}}^{\bullet}\ \mathbf{V}_{\text{A}}^{\bullet}\ \mathbf{DP}^{\bullet}\ \mathbf{DP}^{\bullet}\ \mathbf{S}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\alpha}_{\text{from\_inverse\_linking}}) = \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{PP}_{\text{A}}^{\bullet}\ \mathbf{P}_{\text{A}}^{\bullet}\ \mathbf{DP}^{\bullet}\ \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{NP}_{\text{A}}^{\bullet}$$

$$\tau_{\text{abs}}(\overline{\alpha}_{\text{from\_linear\_scope}}) = \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{PP}_{\text{A}}^{\bullet}\ \mathbf{P}_{\text{A}}^{\bullet}\ \mathbf{DP}{\uparrow}^{\bullet}\ \mathbf{NP}_{\text{A}}^{\bullet}\ \mathbf{NP}_{\text{A}}^{\bullet}$$

Representing derivations We next give the ACG $G_{\text{der}} = \langle \Sigma_{\text{abs}}, \Sigma_{\text{der}}, \mathcal{L}_{\text{der}}, \mathbf{S}^{\bullet} \rangle$. The higher-order signature Σ_{abs} provides us with an abstract language representing the coTAG-derivations including information on the derivational order. This is achieved qua lambda-terms whose order are greater than 2. The higher-order signature Σ_{der} provides us, via the lexicon $\mathcal{L}_{\text{der}}$, with the object language. In case we stick to coTAGs in a normalized form (see below), the object language represents the "plain" TAG-derivation trees of the coTAG, i.e., the object language provides representations of the derivations which (without distinguishing

[15] Regarding the tree structure, we assume the interior nodes $\nu_1, \ldots, \nu_m$ to be ordered "top-down, left-to-right," and the substitution nodes $\nu_{m+1}, \ldots, \nu_{m+n}$ "left-to-right.".

[16] For each $\gamma \in \mathit{Cat}(\{\uparrow\}\mathit{Cat})^*$ such that $\gamma = \zeta{\uparrow}\delta$ for some $\zeta \in \mathit{Cat}(\{\uparrow\}\mathit{Cat})^*$ and $\delta \in \mathit{Cat}$, we take the "circled" boldface version $\boldsymbol{\gamma}^{\circ}$ to denote the type $(\boldsymbol{\zeta}^{\bullet}\boldsymbol{\delta}^{\bullet})\,\boldsymbol{\delta}^{\bullet}$.

[17] That is, we allow adjunction at the root of an initial tree only if the root-label is a "simple" category.

[18] To avoid notational overload, we use the name of a lexical entry of G_{scope}, when referring to the abstract constant typed with regard to the first, i.e. the syntactic, component of the entry.

between the operations of *substitution* and *cosubstitution*) keep track only of which operation was applied at which node by using which trees, and which do not keep track of the relative order in which the operations took place.

More concretely, we have $\Sigma_{\mathrm{der}} = \langle A_{\mathrm{abs}}, \{\beta' \mid \beta \in C_{\mathrm{abs}}\}, \tau_{\mathrm{der}}\rangle$, and the lexicon $\mathcal{L}_{\mathrm{der}}$ of the ACG G_{der} and the typing function τ_{der} of the higher-order signature Σ_{der} are given in the following way: as a mapping from types to types, $\mathcal{L}_{\mathrm{der}}$ functions as identity mapping. For each $X \in \mathtt{Cat}$, the abstract constant id_X is mapped to $\lambda x.\, x$ under $\mathcal{L}_{\mathrm{der}}$. For $\alpha \in \mathcal{I} \cup \mathcal{A}$, we have $\tau_{\mathrm{der}}(\overline{\alpha}') = \tau_{\mathrm{abs}}(\overline{\alpha})$ and $\mathcal{L}_{\mathrm{der}}(\overline{\alpha}) = \overline{\alpha}'$. For $\alpha \in \mathcal{I}$ with root-label γ such that $\gamma = \zeta{\uparrow}\delta$ for some $\zeta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$ and $\delta \in \mathtt{Cat}$, and with $\tau_{\mathrm{abs}}(\overline{\alpha}) = \boldsymbol{\gamma}^{\bullet}_{1\mathrm{A}} \cdots \boldsymbol{\gamma}^{\bullet}_{m\mathrm{A}} \boldsymbol{\delta}^{\bullet}_{1} \cdots \boldsymbol{\delta}^{\bullet}_{n} \boldsymbol{\gamma}^{\circ}$, we also have $\tau_{\mathrm{der}}(\overline{\overline{\alpha}}') = \boldsymbol{\gamma}^{\bullet}_{1\mathrm{A}} \cdots \boldsymbol{\gamma}^{\bullet}_{m\mathrm{A}} \boldsymbol{\delta}^{\bullet}_{1} \cdots \boldsymbol{\delta}^{\bullet}_{n} \boldsymbol{\zeta}^{\bullet}$ and $\mathcal{L}_{\mathrm{der}}(\overline{\alpha}) = \lambda y_1 \ldots y_{m+n} f.\, f(\overline{\overline{\alpha}}' y_1 \cdots y_{m+n})$, where the variable f is of type $(\boldsymbol{\zeta}^{\bullet}\, \boldsymbol{\delta}^{\bullet})$. That is, in case the root-label of an elementary tree consists of a (possibly multiple) lifted type, we "undo" the (last) type lifting.

Put differently, the lexicon $\mathcal{L}_{\mathrm{der}}$ is quite simple, and is based on viewing higher-order constants as literally type lifted versions of themselves:[19] accordingly, an abstract constant as, e.g., $\overline{\overline{\alpha}}_{\mathrm{everyone}}$ of type $((\mathbf{DP}^{\bullet}\ \mathbf{S}^{\bullet})\ \mathbf{S}^{\bullet})$ would be mapped to $\lambda P.P\,\overline{\overline{\alpha}}'_{\mathrm{everyone}}$, where the variable P is of type $(\mathbf{DP}^{\bullet}\ \mathbf{S}^{\bullet})$, and where $\overline{\overline{\alpha}}'_{\mathrm{everyone}}$ is the object constant of type $\mathbf{DP}^{\bullet}$ associated with the elementary tree for *everyone* in the TAG-grammar obtained from the syntactic component of the corresponding lexical entry $\alpha_{\mathrm{everyone}}$ of the coTAG-grammar by removing all arrow annotations. Abstract constants without arrow annotations are simply mapped to "themselves" as is, e.g., demonstrated by $\mathcal{L}_{\mathrm{der}}(\overline{\alpha}_{\mathrm{loves}}) = \overline{\alpha}'_{\mathrm{loves}}$ and $\tau_{\mathrm{der}}(\overline{\alpha}'_{\mathrm{loves}}) = \tau_{\mathrm{abs}}(\overline{\alpha}_{\mathrm{loves}})$. Applying this lexicon to a simple example is illustrative:[20]

$$
\begin{aligned}
&\mathcal{L}_{\mathrm{der}}(\overline{\overline{\alpha}}_{\mathrm{everyone}}(\lambda x.\, \overline{\alpha}_{\mathrm{loves}}\, x\, \overline{\alpha}_{\mathrm{mary}})) \\
&\quad = \quad \mathcal{L}_{\mathrm{der}}(\overline{\overline{\alpha}}_{\mathrm{everyone}})\, \mathcal{L}_{\mathrm{der}}(\lambda x.\, \overline{\alpha}_{\mathrm{loves}}\, x\, \overline{\alpha}_{\mathrm{mary}}) \\
&\quad = \quad \mathcal{L}_{\mathrm{der}}(\overline{\overline{\alpha}}_{\mathrm{everyone}})\, (\lambda x.\, \mathcal{L}_{\mathrm{der}}(\overline{\alpha}_{\mathrm{loves}})\, \mathcal{L}_{\mathrm{der}}(x)\, \mathcal{L}_{\mathrm{der}}(\overline{\alpha}_{\mathrm{mary}})) \\
&\quad = \quad (\lambda P.\, P\, \overline{\overline{\alpha}}'_{\mathrm{everyone}})(\lambda x.\, \overline{\alpha}'_{\mathrm{loves}}\, x\, \overline{\alpha}'_{\mathrm{mary}}) \\
&\quad \twoheadrightarrow_{\beta} \quad (\lambda x.\, \overline{\alpha}'_{\mathrm{loves}}\, x\, \overline{\alpha}'_{\mathrm{mary}})\, \overline{\overline{\alpha}}'_{\mathrm{everyone}} \\
&\quad \twoheadrightarrow_{\beta} \quad \overline{\alpha}'_{\mathrm{loves}}\, \overline{\overline{\alpha}}'_{\mathrm{everyone}}\, \overline{\alpha}'_{\mathrm{mary}}
\end{aligned}
$$

Semantic representations In Sec. 2 we have explained, how the derived semantic trees of a coTAG determine closed well-typed lambda-terms associated with the yields of the trees, cf. page 2. The ACG $G_{\mathrm{sem}} = \langle \Sigma_{\mathrm{abs}}, \Sigma_{\mathrm{sem}}, \mathcal{L}_{\mathrm{sem}}, \mathbf{S}^{\bullet}\rangle$ will do nothing but making the lambda-terms of the finally derived semantic trees concrete as its object language. Therefore, the "flat" lambda-term representation

[19] In the same way as, e.g., the generalized quantifier $\lambda P.P(\mathbf{john})$ with variable P of type $(e\,t)$ is the type lifted version of the individual constant **john**.

[20] If we were more precise, we would actually be concerned with a lambda-term like $\overline{\overline{\alpha}}_{\mathrm{everyone}}(\lambda x.\, \overline{\alpha}_{\mathrm{loves}}\, id_{\mathrm{S}}\, id_{\mathrm{VP}}\, id_{\mathrm{V}}\, x\, \overline{\alpha}_{\mathrm{mary}})$. For better readability we leave out any possible instantiations of "vacuous" adjunction indicated by a functional application to a constant id_X.

of intermediately derived semantic trees—which takes into account only the "open slots" of the yield provided by coTAG-nonterminals—has to be enriched by information about those interior nodes which allow adjunction. We have to lambda-abstract about those sites by variables of appropriate type as well. As far as the general case is concerned, we will skip further technical details how to arrive at those lambda-terms, and look at our coTAG G_{scope} as an example case below. Let us assume here, that for each semantic elementary tree $\alpha_{\text{sem}} \in \mathcal{I}' \cup \mathcal{A}'$, $(\alpha_{\text{sem}})_\lambda$ is the corresponding lambda-term.

We set $\Sigma_{\text{sem}} = \langle A_{\mathcal{Cat}}, \{\mathbf{a} \,|\, a \in V_T \cup \mathit{Con}\}, \tau_{\text{sem}}\rangle$. In order to define τ_{sem} we assume, without loss of generality, that each $a \in V_T \cup \mathit{Con}$ appears as the leaf-label of some elementary semantic tree of our coTAG-lexicon. For $a \in V_T \cup \mathit{Con}$, we choose a node ν of some elementary semantic tree labeled a and let $\mathit{label}(\mu)$ be the label of the parent node of ν, μ. We set $\tau_{\text{sem}}(\mathbf{a}) = \mathit{label}(\mu)$.[21]

Defining the lexicon $\mathcal{L}_{\text{sem}}$, for each abstract atomic type $\boldsymbol{\delta}^\bullet$ arising from some $\delta \in \mathcal{Cat}(\{\uparrow\}\mathcal{Cat})^*$, we let $\mathcal{L}_{\text{sem}}(\boldsymbol{\delta}^\bullet) = \widehat{\tau}_{\mathcal{Cat}}(\delta)$, and if $\delta \in \mathcal{Cat}$, we also let $\mathcal{L}_{\text{sem}}(\boldsymbol{\delta}^\bullet_{\text{A}}) = \widehat{\tau}_{\mathcal{Cat}}(\delta)\ \widehat{\tau}_{\mathcal{Cat}}(\delta)$. For each abstract constant $\overline{\alpha}_{\text{syn}}$, respectively, $\overline{\overline{\alpha}}_{\text{syn}}$, arising from some lexical coTAG-entry $\langle \alpha_{\text{syn}}, \alpha_{\text{sem}}\rangle \in (\mathcal{I} \times \mathcal{I}') \cup (\mathcal{A} \cup \mathcal{A}')$ with $\alpha_{\text{syn}} \in \mathcal{I} \cup \mathcal{A}$ and $\alpha_{\text{sem}} \in \mathcal{I}' \cup \mathcal{A}'$, we let $\mathcal{L}_{\text{sem}}(\overline{\alpha}_{\text{syn}}) = (\alpha_{\text{sem}})_\lambda$, respectively, $\mathcal{L}_{\text{sem}}(\overline{\overline{\alpha}}_{\text{syn}}) = (\alpha_{\text{sem}})_\lambda$.

As an example consider the coTAG G_{scope} as given in Fig. 4 and 5. The function τ_{sem} assigning types to the (semantic) object constants is given by:

$$\begin{array}{lll}
\tau_{sem}(\mathbf{every}) = (e\ t)\ (e\ t)\ t & \tau_{sem}(\mathbf{boy}) = e\ t & \tau_{sem}(\mathbf{loves}) = e\ e\ t \\
\tau_{sem}(\mathbf{some}) = (e\ t)\ (e\ t)\ t & \tau_{sem}(\mathbf{girl}) = e\ t & \tau_{sem}(\mathbf{from}) = e\ e\ t \\
 & \tau_{sem}(\wedge) = t\ t\ t &
\end{array}$$

The lexicon, $\mathcal{L}_{\text{sem}}$, connecting abstract atomic types with (semantic) object types and abstract constants with (semantic) object lambda-terms is given by

$$\begin{array}{ll}
\mathcal{L}_{\text{sem}}(\mathbf{D}^\bullet) = (e\ t)\ (e\ t)\ t & \mathcal{L}_{\text{sem}}(\mathbf{D}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{D}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{D}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{DP}^\bullet) = e & \mathcal{L}_{\text{sem}}(\mathbf{DP}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{DP}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{DP}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{DP}{\uparrow}^\bullet) = (e\ t)\ t & \\
\mathcal{L}_{\text{sem}}(\mathbf{P}^\bullet) = e\ e\ t & \mathcal{L}_{\text{sem}}(\mathbf{P}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{P}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{P}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{PP}^\bullet) = e\ t & \mathcal{L}_{\text{sem}}(\mathbf{PP}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{PP}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{PP}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{V}^\bullet) = e\ e\ t & \mathcal{L}_{\text{sem}}(\mathbf{V}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{V}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{V}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{VP}^\bullet) = e\ t & \mathcal{L}_{\text{sem}}(\mathbf{VP}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{VP}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{VP}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{NP}^\bullet) = e\ t & \mathcal{L}_{\text{sem}}(\mathbf{NP}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{NP}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{NP}^\bullet) \\
\mathcal{L}_{\text{sem}}(\mathbf{S}^\bullet) = t & \mathcal{L}_{\text{sem}}(\mathbf{S}^\bullet_{\text{A}}) = \mathcal{L}_{\text{sem}}(\mathbf{S}^\bullet)\ \mathcal{L}_{\text{sem}}(\mathbf{S}^\bullet)
\end{array}$$

and furthermore, for

[21] By i.1) on page 2, $\mathit{label}(\mu)$ is uniquely determined independently of the choice of ν.

$$
\begin{array}{lll}
d \;:= \mathcal{L}_{\mathrm{sem}}(\mathbf{D}^\bullet) & p \;:= \mathcal{L}_{\mathrm{sem}}(\mathbf{P}^\bullet) & vp := \mathcal{L}_{\mathrm{sem}}(\mathbf{VP}^\bullet) \\
dp \;:= \mathcal{L}_{\mathrm{sem}}(\mathbf{DP}^\bullet) & pp := \mathcal{L}_{\mathrm{sem}}(\mathbf{PP}^\bullet) & np := \mathcal{L}_{\mathrm{sem}}(\mathbf{NP}^\bullet) \\
dp{\uparrow} := \mathcal{L}_{\mathrm{sem}}(\mathbf{DP}{\uparrow}^\bullet) & v \;:= \mathcal{L}_{\mathrm{sem}}(\mathbf{V}^\bullet) & s \;:= \mathcal{L}_{\mathrm{sem}}(\mathbf{S}^\bullet)
\end{array}
$$

by[22]

$$
\begin{array}{l}
\mathcal{L}_{\mathrm{sem}}(\overline{\overline{\alpha}}_{\mathrm{every}}) = \mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{every}}) = \lambda F^{(d\ d)} x^{np}.\,(\,F\ \mathbf{every}\,)\,x \\
\mathcal{L}_{\mathrm{sem}}(\overline{\overline{\alpha}}_{\mathrm{some}}) = \mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{some}}) = \lambda F^{(d\ d)} x^{np}.\,(\,F\ \mathbf{some}\,)\,x \\
\\
\mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{boy}}) \;= \lambda F^{(np\ np)}.\,F\ \mathbf{boy} \\
\mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{girl}}) \;= \lambda F^{(np\ np)}.\,F\ \mathbf{girl} \\
\\
\mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{loves}}) = \lambda F^{(s\ s)} G^{(vp\ vp)} H^{(v\ v)} y^{dp} x^{dp}.\,F(\,(\,G\,(\,(\,H\ \mathbf{loves}\,)\,y\,)\,)\,x\,) \\
\\
\mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{from\_inverse\_linking}}) = \lambda F^{(np\ np)} G^{(pp\ pp)} H^{(p\ p)} P^{np} y^{dp}. \\
\qquad\qquad F\,(\,\lambda x^{dp}.\,\wedge\,(\,G\,(\,(\,H\ \mathbf{from}\,)\,y\,)\,x\,)\,(\,P\,x\,)\,) \\
\\
\mathcal{L}_{\mathrm{sem}}(\overline{\alpha}_{\mathrm{from\_linear\_scope}}) \;= \lambda F^{(np\ np)} G^{(pp\ pp)} H^{(p\ p)} Q^{dp\uparrow} P^{np}. \\
\qquad\qquad F(\,\lambda x^{dp}.\,\wedge\,(\,Q\,(\,\lambda y^{dp}.\,G\,(\,(\,H\ \mathbf{from}\,)\,y\,)\,x)\,)\,(\,P\,x)\,)
\end{array}
$$

while, finally, we have

$$
\mathcal{L}_{\mathrm{sem}}(id_X) \;=\; \lambda x^{\mathcal{L}_{\mathrm{sem}}(\mathbf{X}^\bullet)}.\,x \qquad \text{for } X \in \mathtt{Cat}
$$

On (co)substitution nodes For $\langle V_T, \mathtt{Cat}, A_{\mathtt{Cat}}, \widehat{\tau}_{\mathtt{Cat}}, \mathcal{I}\times\mathcal{I}', \mathcal{A}\times\mathcal{A}', ^\frown, \mathrm{S}\rangle$, the coTAG G_{coTAG} we started with, the nodes of the derivable syntactic trees are labeled with strings from $\mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*\{\downarrow\}^?$. These labels come with seemingly implicit compositional structure by means of the occurrences of $\uparrow$ (and $\downarrow$): given some set of atomic types A_{Types}, and assigning a type $\boldsymbol{\gamma}^\blacklozenge$ from $\mathcal{T}(A_{Types})$ to each $\gamma \in \mathtt{Cat}$, it is straightforward to interpret $\uparrow$ as a type lifting operation (and $\downarrow$ as the identity function on types), thereby recursively assigning the type $\boldsymbol{\zeta}{\downarrow}^\blacklozenge := \boldsymbol{\zeta}^\blacklozenge$ to $\zeta\downarrow$, and $\boldsymbol{\zeta}{\uparrow}\boldsymbol{\delta}^\blacklozenge := ((\boldsymbol{\zeta}^\blacklozenge\,\boldsymbol{\delta}^\blacklozenge)\,\boldsymbol{\delta}^\blacklozenge)$ to $\zeta\uparrow\delta$ for $\zeta \in \mathtt{Cat}\,(\{\uparrow\}\mathtt{Cat}\,)^*$ and $\delta \in \mathtt{Cat}$.[23]

As far as the operation of substitution is concerned, we are not exploiting this structural potential in the translation to ACGs. Instead, given a substitution site with syntactic node label $\gamma\downarrow$ we are treating γ as an atomic category (label) with regard to the coTAG, irrespective of whether γ contains any $\uparrow$ characters. We are doing so in terms of the abstract atomic type $\gamma^\bullet$ belonging to A_{abs}. Looking

[22] As usual a superscript connected with a variable denotes the type of that variable. In the example, the variables F, G, H indicate potential adjunction sites. Again we use the name of a lexical entry of G_{scope}, when referring to the abstract constant typed with regard to the first component of the entry, cf. fn. 18.

[23] $\widehat{\tau}_{\mathtt{Cat}}$ as defined on page 2 is a particular instance of such a recursively defined function $\cdot^\blacklozenge$, assigning the types $\boldsymbol{\zeta}$ and $(\boldsymbol{\zeta}\,\boldsymbol{\delta})\,\boldsymbol{\delta}$ from $\mathcal{T}(A_{\mathtt{Cat}})$ to the same ζ and $\zeta\uparrow\delta$, respectively.

at the example above, instances of such a $\gamma^\bullet$ occur in terms of $\mathbf{DP}{\uparrow}^\bullet$ within the types assigned to $\overline{\alpha}_{\text{every}}$, $\overline{\alpha}_{\text{some}}$ and $\overline{\alpha}_{\text{from\_linear\_scope}}$ via τ_{abs}.

As far as the operation of cosubstitution is concerned, the compositional structure potential of a syntactic node label is exploited in the translation to ACGs only in its "simplest" non-recursive form: if there are any, only the last instance of $\uparrow$ within a syntactic root label is interpreted as a type lifting operation. More concretely, for $\zeta \in \mathit{Cat}\,(\{\uparrow\}\mathit{Cat}\,)^*$ and $\delta \in \mathit{Cat}$ such that $\zeta{\uparrow}\delta$ labels the root of a syntactic tree, the abstract type associated with that node is $(\boldsymbol{\zeta}^\bullet\,\boldsymbol{\delta}^\bullet)\,\boldsymbol{\delta}^\bullet$, where $\boldsymbol{\zeta}^\bullet$ and $\boldsymbol{\delta}^\bullet$ are atomic abstract types from A_{abs}. Looking at the example above, instances of such a $(\boldsymbol{\zeta}^\bullet\,\boldsymbol{\delta}^\bullet)\,\boldsymbol{\delta}^\bullet$ occur in terms of $(\mathbf{DP}^\bullet\,\mathbf{S}^\bullet)\,\mathbf{S}^\bullet$ within the types assigned to $\overline{\overline{\alpha}}_{\text{every}}$ and $\overline{\overline{\alpha}}_{\text{some}}$ via τ_{abs}.

Technically of course, it would be straightforwardly possible to exploit "all the way down" the compositional structure implicit in the syntactic node labels of the coTAG in an ACG-translation: starting from our Σ_{abs} this would essentially be achieved by

- assuming for each $\delta \in \mathit{Cat}$, $\boldsymbol{\delta}^\blacklozenge$ and $\boldsymbol{\delta}^\bullet_{\text{A}}$ to be atomic types,
- replacing A_{abs} by $\{\boldsymbol{\delta}^\blacklozenge, \boldsymbol{\delta}^\bullet_{\text{A}} \mid \delta \in \mathit{Cat}\}$, and
- replacing $\boldsymbol{\gamma}^\bullet$ by $\boldsymbol{\gamma}^\blacklozenge$ for each $\gamma \in \mathit{Cat}\,(\{\uparrow\}\mathit{Cat}\,)^*$, where $\boldsymbol{\gamma}^\blacklozenge \in \mathcal{T}(\{\boldsymbol{\delta}^\blacklozenge \mid \delta \in \mathit{Cat}\})$ is the type assigned to γ, recursively defined as above.

This potential alternative views a coTAG as an ACG of arbitrarily high, but lexically fixed, order. Although rather canonical from a formal perspective, this alternative encoding of coTAGs into ACGs is incorrect—cosubstitution as defined requires exactly third-order terms. The coTAG-grammar fragment above, e.g., is such that its alternative translation into an ACG does not preserve the form-meaning relation. This is due to the fact that higher-order types can be "lowered" by hypothetical reasoning, which cannot be simulated in the original coTAG. More concretely, in the alternative ACG-encoding, a substitution site with syntactic label DP$\uparrow\downarrow$ selects an argument of higher-order abstract type $(\mathbf{DP}^\blacklozenge\,\mathbf{S}^\blacklozenge)\;\mathbf{S}^\blacklozenge$, and not atomic type $\mathbf{DP}{\uparrow}^\bullet$. Therefore, in ACG-terms, the substitution site can be given as argument a term $\lambda P^{(\mathbf{DP}^\blacklozenge\,\mathbf{S}^\blacklozenge)}.\,P\,y$, where y is an unbound variable of type $\mathbf{DP}^\blacklozenge$. The variable y can be abstracted over at a later point in time, giving rise to a term of type $\mathbf{DP}^\blacklozenge\,\mathbf{S}^\blacklozenge$, which can then be the argument to the cosubstitutor of type $(\mathbf{DP}^\blacklozenge\,\mathbf{S}^\blacklozenge)\;\mathbf{S}^\blacklozenge$. Thus, the substitution for a cosubstitutor was only "tricked" by the combinator $\lambda P.\,P\,y$ into "thinking" that it had already been given the correct argument.

The lexical entry $\alpha_{\text{from\_linear\_scope}}$ from G_{scope} provides a case in point. This entry would be translated into the abstract constant $\overline{\alpha}_{\text{from\_linear\_scope}}$ of fourth-order type $\mathbf{NP}^\bullet_{\text{A}}\,\mathbf{PP}^\bullet_{\text{A}}\,\mathbf{P}^\bullet_{\text{A}}\,((\,\mathbf{DP}^\blacklozenge\,\mathbf{S}^\blacklozenge\,)\,\mathbf{S}^\blacklozenge\,)\,\mathbf{NP}^\bullet_{\text{A}}\,\mathbf{NP}^\bullet_{\text{A}}$, and as a consequence, the following term would be well-typed:[24]

$$\overline{\overline{\alpha}}_{\text{every}}\,\overline{\alpha}_{\text{city}}\,(\,\lambda y.\,\overline{\overline{\alpha}}_{\text{some}}\,(\,\overline{\alpha}_{\text{boy}}\,(\,\overline{\alpha}_{\text{from\_linear\_scope}}\,(\,\lambda P.\,P\,y\,)\,)\,)\,(\,\lambda x.\,\overline{\alpha}_{\text{left}}\,x\,)\,)\;,$$

[24] We owe this example to the anonymous reviewer mentioned in the acknowledgments. Again, for better readability we ignore "vacuous" adjunction indicated by a functional application to a constant id_X.

where the variables x and y are of type $\mathbf{DP}^{\blacklozenge}$, and the variable P is of type $\mathbf{DP}^{\blacklozenge}\,\mathbf{S}^{\blacklozenge}$. This term, however, evaluates semantically to the inverse scope reading of the noun phrase *some boy from every city*, despite the fact that the elementary tree $\alpha_{\text{from\_linear\_scope}}$ was used.

Thus, the alternative translation into higher-order ACGs is not faithful to Barker's original presentation. On the other hand, the higher-order ACG allows for a single lexical item, $\alpha_{\text{from\_linear\_scope}}$, to derive both inverse and linear scope readings, which might be seen as more elegant than the original coTAG-analysis. This behavior seems to be obtainable in the coTAG-formalism if we alter the definition of cosubstitution so as to also allow cosubstitution of trees with root-label $\zeta{\uparrow}\delta$ into nodes labeled $\zeta{\uparrow}\delta{\downarrow}$.

5 Conclusion

We have shown how Barker's ideas about coTAGs have a natural home in the ACG-formalism. Our formalization makes explicit the graph structure of coTAG-derivations, in particular the dependence in cosubstitution on both the substitution node (in a particular elementary tree) and the derivation over which it takes scope.

The ACG-perspective allows us to better understand the surprising fact that coTAGs have the same strong generative capacity as regular TAGs: it is due to the fact that the "lifted" underlying derivation is first lowered back into a regular TAG-derivation on the syntactic side, and then interpreted to obtain a derived tree. The crucial piece in this puzzle is our normal form theorem, which states that every coTAG (qua ACG) has an equivalent third-order variant.

This also highlights the fact that "strong generative capacity" in the TAG-sense, i.e. the sets of structures derivable, is not the most insightful measure of the complexity of a grammar formalism; rather it is the relation between derived objects and derivations (which stand proxy for meanings in a compositional system) which provides the most insight into the grammar formalism. According to this measure, coTAGs are much more complex than regular TAGs, whose derivation sets are regular tree languages, and therefore, second-order ACGs.

Acknowledgments We are grateful to an anonymous reviewer for pushing us to a much more rigorous presentation of Section 4. Any remaining lack of clarity and errors are due to us.

References

1. Barker, C.: Cosubstitution, derivational locality, and quantifier scope. In: Proceedings of the Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10), New Haven, CT, pp. 135–142 (2010)
2. Bauderon, M., Courcelle, B.: Graph expressions and graph rewriting. Mathematical Systems Theory 20, 83–127 (1987)

3. de Groote, P.: Towards abstract categorial grammars. In: 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001), Toulouse, pp. 252–259. ACL (2001)
4. de Groote, P.: Tree-adjoining grammars as abstract categorial grammars. In: Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+6), Venezia, pp. 145–150 (2002)
5. Habel, A., Kreowski, H.J.: Some structural aspects of hypergraph languages generated by hyperedge replacement. In: Brandenburg, F.J., Vidal-Naquet, G., Wirsing, M. (eds.) STACS 87, LNCS, Vol. 247, pp. 207–219. Springer, Berlin, Heidelberg (1987)
6. Joshi, A.K.: An introduction to tree adjoining grammars. In: Manaster-Ramer, A. (ed.) Mathematics of Language, pp. 87–114. John Benjamins, Amsterdam (1987)
7. Joshi, A.K., Schabes, Y.: Tree adjoining grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 69–124. Springer, Berlin, Heidelberg (1997)
8. Kanazawa, M.: Parsing and generation as datalog queries. In: 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007), Prague. pp. 176–183. ACL (2007)
9. Kanazawa, M.: Second-order abstract categorial grammars as hyperedge replacement grammars. Journal of Logic, Language and Information 19, 137–161 (2010)
10. Pogodalla, S.: Computing semantic representation. Towards ACG abstract terms as derivation trees. In: Proceedings of the Seventh International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+7), Vancouver, BC, pp. 64–71 (2004)
11. Pogodalla, S.: Ambiguïté de portée et approche fonctionnelle des grammaires d'arbres adjoints. In: Traitement Automatique des Langues Naturelles (TALN 2007), Toulouse (2007), 10 pages
12. Salvati, S.: Encoding second order string ACG with deterministic tree walking transducers. In: Wintner, S. (ed.) Proceedings of FG 2006: The 11th Conference on Formal Grammar, pp. 143–156. CSLI Publications, Stanford, CA (2007)