Derivational order and ACGs

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

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

ACG@10, Bordeaux, December 7−9, 2011

What good are ACGs?

- ACGs provide a way to understand MANY ideas from a uniform perspective
  - Architectures for grammar (SP)
    - Syntactico-centrism vs Parallelism (SP, PdG, & CP)
  - Grammar formalisms
    - CFHGs (MK)
    - CFG, LCFTG, MCFG (PdG, SP)
    - DTWT (SS)
    - MG (SS)

      ⋮
  - Distributional learning (RY)

  ⋮
- We want to add to the list:
  - Timing

Chris Barker and a new approach to TAG semantics

The basic idea of using cosubstitution to handle scope is that we can build the nuclear scope of a quantifier before the quantifier enters the derivation.

Chris Barker and a new approach to TAG semantics

The basic idea of using cosubstitution to handle scope is that we can build the nuclear scope of a quantifier before the quantifier enters the derivation.

In other words, the nuclear scope of a quantifier is quite simply and quite literally its syntactic and semantic argument.

Chris Barker and a new approach to TAG semantics

The basic idea of using cosubstitution to handle scope is that we can build the nuclear scope of a quantifier before the quantifier enters the derivation.

In other words, the nuclear scope of a quantifier is quite simply and quite literally its syntactic and semantic argument.

Thus on the cosubstitution approach, quantifier scope ambiguity is a matter of timing: quantifiers that enter later in the derivation take wider scope.

Chris Barker and a new approach to TAG semantics

The basic idea of using cosubstitution to handle scope is that we can build the nuclear scope of a quantifier before the quantifier enters the derivation.

In other words, the nuclear scope of a quantifier is quite simply and quite literally its syntactic and semantic argument.

Thus on the cosubstitution approach, quantifier scope ambiguity is a matter of timing: quantifiers that enter later in the derivation take wider scope.

In other words, co-TAG has exactly the same weak and strong generative capacity as TAG.

Actually, an old approach to semantics (and other things)

- Montague Grammar

(Montague)

> [...] ambiguity can arise even when there is no element of intensionality, simply because quantifying terms may be introduced in more than one order.

- Minimalist Grammars

(Gärtner & Michaelis; Kobele)

> [...] an expression may bind only those expressions that it c-commands when first merged

What's behind this?

The problem:

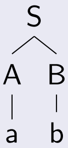
Sometimes we have a semantic ambiguity without any obvious corresponding difference in the derivation tree.

The idea:

Enrich the derivation tree with information about order.
(not only did I apply this rule, but I applied this rule *before that one*)

A concrete example

$$\begin{aligned} S &\rightarrow A \;\; B \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$



Example

The representation of a CFG derivation as a tree equates sequences of rewriting steps which rewrite nonterminal instances the same way. (Leftmost, rightmost, etc)

$$\langle S \Rightarrow AB \Rightarrow aB \Rightarrow ab \rangle \equiv \langle S \Rightarrow AB \Rightarrow Ab \Rightarrow ab \rangle$$

What's to come

The goals:

- Make sense of the idea of 'timing' in terms of derivations
- Understand (in particular) co-substitution in TAGs in these terms
- If SGC/WGC unchanged, what is going on?

Our main claims are that:

- Third order ACGs provide an elegant description of the derivations of context-free formalisms.
- "*Timing*" is not derivational order

Outline

1. Everyone's favourite example (Montague)
2. 3^{rd} order ACGs and context-free derivations
3. Cosubstitution in TAGs

Frameworks which make use of ‘late’ operations

Montague Grammar

- Quantifying In

Tree Adjoining Grammars

- co-substitution
- flexible composition

Minimalist Grammars

- Late adjunction
- ‘hypothetical reasoning’

Montague Grammar (MG)

- MG has order sensitive rules in the form of Quantifying In (QI), namely, S14:

Rules of quantification

S14. If $\alpha \in P_T$ and $\phi \in P_t$, then $F_{10,n}(\alpha, \phi) \in P_t$, where either (i) α does not have the form $\mathbf{he}_k$, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing the first occurrence of $\mathbf{he}_n$ or $\mathbf{him}_n$ by α and all other occurrences of $\mathbf{he}_n$ or $\mathbf{him}_n$ by $\left\{\begin{matrix}\mathbf{he}\\\mathbf{she}\\\mathbf{it}\end{matrix}\right\}$ or $\left\{\begin{matrix}\mathbf{him}\\\mathbf{her}\\\mathbf{it}\end{matrix}\right\}$ respectively, according as the gender of the first B_{CN} or B_T in α is $\left\{\begin{matrix}\text{masc.}\\\text{fem.}\\\text{neuter}\end{matrix}\right\}$, or
(ii) $\alpha = \mathbf{he}_k$, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing all occurrences of $\mathbf{he}_n$ or $\mathbf{him}_n$ by $\mathbf{he}_k$ or $\mathbf{him}_k$ respectively.

S15. If $\alpha \in P_T$ and $\zeta \in P_{CN}$, then $F_{10,n}(\alpha, \zeta) \in P_{CN}$.

S16. If $\alpha \in P_T$ and $\delta \in P_{IV}$, then $F_{10,n}(\alpha, \delta) \in P_{IV}$.

Montague Grammar (MG)

- MG has order sensitive rules in the form of Quantifying In (QI), namely, S14:
- QI has some complications we do not need to worry about right now: multiple occurrences of **he**$_n$ can be affected by rule $F_{10,n}$
- QI is order sensitive in the sense that if we have a derivation $d = C[F_{10,i}(\alpha, D[F_{10,j}(\beta, \phi)])]$, then α takes semantic scope over β

An Example - *De Re* vs *De Dicto* Readings

- John seeks a unicorn.

De Re

There is a particular unicorn that John is looking for.
$\exists y[\textsc{unicorn}(y) \,\&\, \textsc{try}(\textsc{find}(y))(\textsc{j})]$

De Dicto

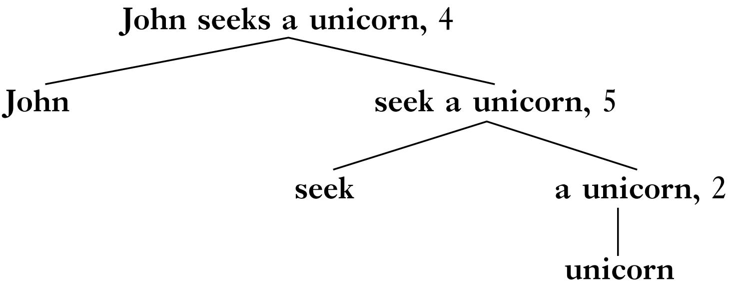
John will be satisfied with any unicorn.
$\textsc{try}(\exists y[\textsc{unicorn}(y) \,\&\, \textsc{find}(y)])(\textsc{j})$

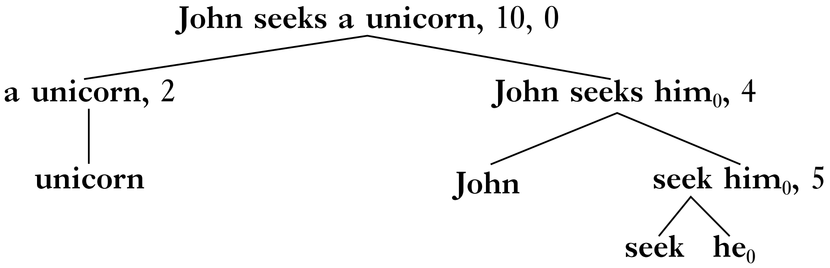
- Can be represented in terms of operator scope:

x seeks y

$\textsc{try}(\textsc{find}(\mathrm{y}))(\mathrm{x})$

Derivations for *De Re* vs *De Dicto* Readings





An ACG implementation (following de Groote 2001)

- A faithful ACG implementation of the derivation structure of MG might look as follows:
  - to each n-ary predicate (**seek**, **love**, …) we associate a function symbol of type $np^n \rightarrow s$
  - to each common noun (**boy**, **unicorn**, …) we associate a function symbol of type n
  - to each determiner (**some**, **every**, …) we associate a function symbol of type $n \rightarrow d$
  - we have a function symbol **QI** of type $(np \rightarrow s) \rightarrow (d \rightarrow s)$
- This allows us to have abstract terms like the following:
  - $\mathbf{QI}(\lambda y^{np}.\mathbf{QI}(\lambda x^{np}.\mathbf{seek}(x)(y))(\mathbf{some}(\mathbf{unicorn})))(\mathbf{every}(\mathbf{boy}))$

An ACG implementation

- The lexica are as follows:
  - on the concrete phonology side, the symbol **QI** is interpreted as the (forwards application) combinator $\lambda xy.x(y)$ (and everything else is interpreted in the obvious way, given that all abstract types *n*, *d*, *np* and *s* are interpreted as the object type $o \to o$, the type of strings).
  - on the concrete semantic side, the symbol **QI** is interpreted as the (backwards application) combinator $\lambda xy.y(x)$.

An ACG implementation (alternative)

- An alternative implementation substitutes for the type *d* the type $(np \to s) \to s$, and eliminates the symbol **QI**. Then we have abstract terms like:
  - **every**(**boy**)$(\lambda y^{np}.$**some**(**unicorn**)$(\lambda x^{np}.$**seek**$(x)(y)))$
- Under this alternative, the lexica are as follows:
  - The concrete phonological interpretation of a determiner **every** is as the function $\lambda x f.f(\texttt{every}\frown x)$
  - The concrete semantic interpretation of a determiner **every** is as the function $\lambda x f.\forall(x)(f)$

Summary

- The ACG implementation does currently not deal with the ignored complexities of QI.
- The straightforward implementation takes the abstract terms to be pretty much isomorphic to the concrete semantic terms.
- The big picture (that it admittedly seems is not very clear in this presentation) is that:
    - higher order constants mark scope positions, and apply to lambda abstracts which make the variables in lower positions accessible.
    - In other words, 'timing' requires of an expression that it simultaneously be connected to both an underlying (argument) position, and a 'surface' (timing) position.

Tree Adjoining Grammars

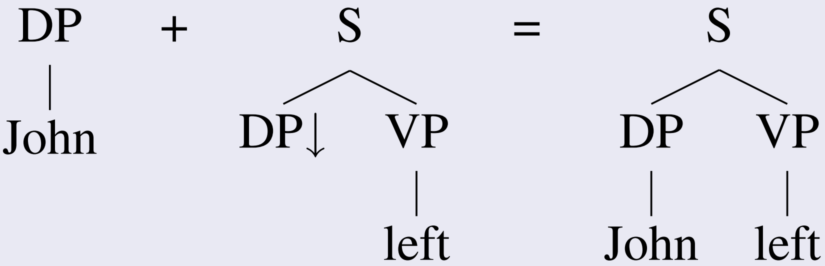
- Strongly equivalent to monadic linear CFTGs
  (Fujiyoshi & Kasai; Mönnich; Kepser & Rogers)
- Weakly equivalent to 2-MCFL$_{wn}$
  (Vijay-Shanker et al.; Seki et al.; Kanazawa)
- A TAG consists of

  initial trees: a finite set of trees with leaves labelled by either terminals or X↓, where X is a nonterminal symbol

  auxiliary trees: as before, but with exactly one leaf node labelled by X*, where X is the label of the root

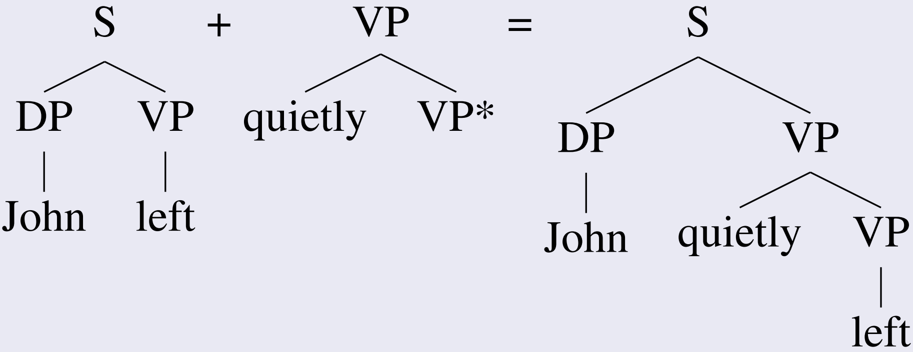
Tree Adjoining Grammars - Substitution

Substitute (1st Order Substitution)



Tree Adjoining Grammars - Adjunction

Adjoin (2nd Order Substitution)



Barker's Innovation: Unrestricted Derivation and Scope

Fact

Linear CFTGs derive the same tree languages under all (IO, OI, unrestricted) derivation modes.

Barker:

Γ scopes over Δ iff Γ was substituted in after a tree containing Δ was

$G = \langle N, T, P, s \rangle$ a linear CFTG

- N the ranked alphabet of nonterminals
- T the ranked alphabet of terminals
- s the start symbol of rank 0 from N
- P the production rules each of which of the form

$$a(x_1, \dots, x_n) \to t$$

  - $a \in N$ such that $rank(a) = n$
  - $x_1, \dots, x_n$ variables of rank 0
  - t a linear tree over $N \cup T \cup \{x_1, \dots, x_n\}$

- For expository reasons we assume that the start symbol $s \in N$ does not appear on the righthand side of any rule

Linear CFTGs as ACGs

Composition as second-order substitution (de Groote and Pogodalla 2004)

$G = \langle N, T, P, s \rangle$ a linear CFTG

Linear CFTGs as ACGs

Composition as second-order substitution (de Groote and Pogodalla 2004)

$G = \langle N, T, P, s\rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N, \{\overline{p} \mid p \in P\}, \tau\rangle$ higher-order linear signature

$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

$p \in P$ with skeleton $\langle a, a_1 \cdots a_m\rangle$

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle\, N \,,\, \{\, \overline{p} \mid p \in P \,\} \,,\, \tau \,\rangle$ higher-order linear signature

$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

$p \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$

- For $p = a(x_1, \ldots, x_n) \to t \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$
  by induction, linear λ-term $[\![p]\!] := \lambda y_1 \ldots y_m \,.\, \lambda x_1 \ldots x_n \,.\, |t|$

$G = \langle N, T, P, s\rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s\rangle$ associated 2nd order ACG

- $\Sigma_G = \langle\, N \,,\, \{\, \overline{p} \mid p \in P \,\} \,,\, \tau \,\rangle$ higher-order linear signature

  $$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

  $p \in P$ with skeleton $\langle a, a_1 \cdots a_m\rangle$

- For $p = a(x_1, \ldots, x_n) \to t \in P$ with skeleton $\langle a, a_1 \cdots a_m\rangle$
  by induction, linear λ-term $[\![p]\!] := \lambda y_1 \ldots y_m \,.\, \lambda x_1 \ldots x_n \,.\, |\, t \,|$

$\lvert x_i \rvert$	=	x_i	
$\lvert f(\,) \rvert$	=	$\lambda x \,.\, (\, f\, x \,)$	$f \in T$
$\lvert f(t_1, \ldots, t_k) \rvert$	=	$\lvert t_1\rvert + \ldots + \lvert t_k\rvert$	
$\lvert a_j(\,) \rvert$	=	y_j	a_j from skeleton
$\lvert a_j(t_1, \ldots, t_k) \rvert$	=	$y_j \; \lvert t_1\rvert \cdots \lvert t_k\rvert$	

Linear CFTGs as ACGs

Composition as second-order substitution (de Groote and Pogodalla 2004)

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N, \{ \overline{p} \mid p \in P \}, \tau \rangle$ higher-order linear signature

$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

$p \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$

- For $p = a(x_1, \ldots, x_n) \to t \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$
  by induction, linear λ-term $[\![p]\!] := \lambda y_1 \ldots y_m . \lambda x_1 \ldots x_n . |t|$

- $\mathcal{L}_G : \Sigma_G \to \Sigma_T$

$$\mathcal{L}_G(a) = string^{rank(a)} \to string \quad \text{for } a \in N$$

$$\mathcal{L}_G(\overline{p}) = [\![p]\!] \quad \text{for } p \in P$$

Linear CFTGs as ACGs

Composition as second-order substitution (de Groote and Pogodalla 2004)

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N , \{ \overline{p} \mid p \in P \} , \tau \rangle$ higher-order linear signature
$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$
$p \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$
- For $p = a(x_1, \dots, x_n) \to t \in P$ with skeleton $\langle a, a_1 \cdots a_m \rangle$ by induction, linear λ-term $[\![p]\!] := \lambda y_1 \dots y_m . \lambda x_1 \dots x_n . |t|$
- $\mathcal{L}_G : \Sigma_G \to \Sigma_T$ $\quad \mathcal{L}_G(a) = string^{rank(a)} \to string$ for $a \in N$
$\mathcal{L}_G(\overline{p}) = [\![p]\!]$ for $p \in P$
- $\Sigma_T = \langle \{ o \} , T , \widetilde{\tau} \rangle$ higher-order linear signature
$$\widetilde{\tau}(f) = string = o \to o \qquad f \in T$$

Linear CFTGs as ACGs

Composition as second-order substitution (de Groote and Pogodalla 2004)

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N, \{ \overline{p} \mid p \in P \}, \tau \rangle$ higher-order linear signature

  $$\tau(\overline{p}) = a_1 \rightarrow \cdots \rightarrow a_m \rightarrow a$$

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N, \{ \overline{p} \mid p \in P \}, \tau \rangle$ higher-order linear signature

$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

$\mathcal{G}'_G = \langle \Sigma'_G, \Sigma_T, \mathcal{L}'_G, s \rangle$ 3rd order ACG resulting from $\mathcal{G}_G$

Linear CFTGs as ACGs

Composition as third-order substitution (including derivation order)

$G = \langle N, T, P, s \rangle$ a linear CFTG

$\mathcal{G}_G = \langle \Sigma_G, \Sigma_T, \mathcal{L}_G, s \rangle$ associated 2nd order ACG

- $\Sigma_G = \langle N, \{ \overline{p} \mid p \in P \}, \tau \rangle$ higher-order linear signature

$$\tau(\overline{p}) = a_1 \to \cdots \to a_m \to a$$

$\mathcal{G}'_G = \langle \Sigma'_G, \Sigma_T, \mathcal{L}'_G, s \rangle$ 3rd order ACG resulting from $\mathcal{G}_G$

- $\Sigma'_G = \langle N, \{ \overline{p} \mid p \in P \}, \tau' \rangle$ higher-order linear signature

$$\tau'(\overline{p}) = \mathit{trans}(\tau(\overline{p})) = \mathit{trans}(a_1 \to \cdots \to a_m \to a)$$

$$\begin{array}{rcll} \mathit{trans}(s) & = & s & s \text{ the start symbol} \\ \mathit{trans}(a) & = & (a \to s) \to s & a \in N - \{s\} \\ \mathit{trans}(\alpha \to \beta) & = & \alpha \to \mathit{trans}(\beta) & \end{array}$$

Example grammar

CFTG

$$p_1 = S \to f(T(c), G(c))$$

$$p_2 = T(x) \to f(T(x), G(c))$$

$$p_3 = G(x) \to h(x)$$

$$p_4 = T(x) \to G(x)$$

2nd order ACG

$$\overline{p}_1 : T \to G \to S$$

$$\overline{p}_2 : T \to G \to T$$

$$\overline{p}_3 : G$$

$$\overline{p}_4 : G \to T$$

Example grammar

CFTG

$$p_1 = S \to f(T(c), G(c))$$
$$p_2 = T(x) \to f(T(x), G(c))$$
$$p_3 = G(x) \to h(x)$$
$$p_4 = T(x) \to G(x)$$

2nd order ACG

$$\overline{p}_1 : T \to G \to S$$
$$\overline{p}_2 : T \to G \to T$$
$$\overline{p}_3 : G$$
$$\overline{p}_4 : G \to T$$

3rd order ACG

$$\overline{p}_1 : T \to G \to S$$
$$\overline{p}_2 : T \to G \to (T \to S) \to S$$
$$\overline{p}_3 : (G \to S) \to S$$
$$\overline{p}_4 : G \to (T \to S) \to S$$

Example derivation — $p_1\,p_4\,p_3\,p_3$ left-to-right

$$\frac{\vdash \overline{p}_1 : T \to G \to S}{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}\ (\mathsf{var}\,,\mathsf{app})$$

$$\frac{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}\ (\mathsf{var}\,,\mathsf{app})$$

$$\frac{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}{g_1 : G \vdash \lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1 : T \to S}\ (\mathsf{abs})$$

$$\frac{\vdash \overline{p}_4 : G \to (T \to S) \to S}{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S}\ (\mathsf{var}\,,\mathsf{app})$$

$$\frac{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S \qquad g_1 : G \vdash \lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1 : T \to S}{g_1 : G\,,\, g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}\ (\mathsf{app})$$

$$\frac{g_1 : G\,,\, g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}{g_2 : G \vdash \lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : G \to S}\ (\mathsf{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad g_2 : G \vdash \lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : G \to S}{g_2 : G \vdash \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}\ (\mathsf{app})$$

$$\frac{g_2 : G \vdash \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}{\vdash \lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}\ (\mathsf{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad \vdash \lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}{\vdash \overline{p}_3 (\lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1))) : S}\ (\mathsf{app})$$

$$\frac{\vdash \overline{p}_1 : T \to G \to S}{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}\ (\text{var , app})$$

$$\frac{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}\ (\text{var , app})$$

$$\frac{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}{g_1 : G \vdash \lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1 : T \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_4 : G \to (T \to S) \to S}{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S}\ (\text{var , app})$$

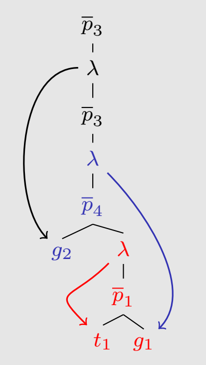
$$\frac{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S \qquad g_1 : G \vdash \lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1 : T \to S}{g_1 : G\,,\, g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}\ (\text{app})$$

$$\frac{g_1 : G\,,\, g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}{g_2 : G \vdash \lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : G \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad g_2 : G \vdash \lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1) : G \to S}{g_2 : G \vdash \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}\ (\text{app})$$

$$\frac{g_2 : G \vdash \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}{\vdash \lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad \vdash \lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}{\vdash \overline{p}_3 (\lambda g_2 \,.\, \overline{p}_3 (\lambda g_1 \,.\, \overline{p}_4\, g_2 (\lambda t_1 \,.\, \overline{p}_1\, t_1\, g_1))) : S}\ (\text{app})$$



Example derivation — $p_1\, p_3\, p_4\, p_3$ right-to-left

$$\dfrac{\vdash \overline{p}_1 : T \to G \to S}{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}\;(\text{var , app})$$

$$\dfrac{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}\;(\text{var , app})$$

$$\dfrac{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}{t_1 : T \vdash \lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1 : G \to S}\;(\text{abs})$$

$$\dfrac{\vdash \overline{p}_3 : (G \to S) \to S \qquad t_1 : T \vdash \lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1 : G \to S}{t_1 : T \vdash \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}\;(\text{var , app})$$

$$\dfrac{t_1 : T \vdash \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1) : S}{\vdash \lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1) : T \to S}\;(\text{abs})$$

$$\dfrac{\vdash \overline{p}_4 : G \to (T \to S) \to S}{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S}\;(\text{var , app})$$

$$\dfrac{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S \qquad \vdash \lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1) : T \to S}{g_2 : G \vdash \overline{p}_4\, g_2(\lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}\;(\text{app})$$

$$\dfrac{g_2 : G \vdash \overline{p}_4\, g_2(\lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1)) : S}{\vdash \lambda g_2 \,.\, \overline{p}_4\, g_2(\lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}\;(\text{abs})$$

$$\dfrac{\vdash \overline{p}_3 : (G \to S) \to S \qquad \vdash \lambda g_2 \,.\, \overline{p}_4\, g_2(\lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1)) : G \to S}{\vdash \overline{p}_3(\lambda g_2 \,.\, \overline{p}_4\, g_2(\lambda t_1 \,.\, \overline{p}_3(\lambda g_1 \,.\, \overline{p}_1\, t_1\, g_1))) : S}\;(\text{app})$$

Example derivation — $p_1\, p_3\, p_4\, p_3$ right-to-left

$$\frac{\vdash \overline{p}_1 : T \to G \to S}{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}\ (\text{var , app})$$

$$\frac{t_1 : T \vdash \overline{p}_1\, t_1 : G \to S}{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}\ (\text{var , app})$$

$$\frac{g_1 : G\,,\, t_1 : T \vdash \overline{p}_1\, t_1\, g_1 : S}{t_1 : T \vdash \lambda g_1 . \overline{p}_1\, t_1\, g_1 : G \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad t_1 : T \vdash \lambda g_1 . \overline{p}_1\, t_1\, g_1 : G \to S}{t_1 : T \vdash \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1) : S}\ (\text{var , app})$$

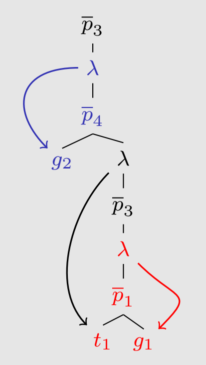
$$\frac{t_1 : T \vdash \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1) : S}{\vdash \lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1) : T \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_4 : G \to (T \to S) \to S}{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S}\ (\text{var , app})$$

$$\frac{g_2 : G \vdash \overline{p}_4\, g_2 : (T \to S) \to S \qquad \vdash \lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1) : T \to S}{g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1)) : S}\ (\text{app})$$

$$\frac{g_2 : G \vdash \overline{p}_4\, g_2 (\lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1)) : S}{\vdash \lambda g_2 . \overline{p}_4\, g_2 (\lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1)) : G \to S}\ (\text{abs})$$

$$\frac{\vdash \overline{p}_3 : (G \to S) \to S \qquad \vdash \lambda g_2 . \overline{p}_4\, g_2 (\lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1)) : G \to S}{\vdash \overline{p}_3 (\lambda g_2 . \overline{p}_4\, g_2 (\lambda t_1 . \overline{p}_3 (\lambda g_1 . \overline{p}_1\, t_1\, g_1))) : S}\ (\text{app})$$



Back towards the object language

The transformation g which identifies 'equivalent' derivations:

$$\bar{p} \longmapsto \begin{cases} \bar{p} & \text{if } p \in P \text{ is a non-lifted rule} \\ \lambda y_1 \ldots y_m f \,.\, f\,(\bar{p}\, y_1 \cdots y_m) & \text{otherwise} \end{cases}$$

- This actually maps a derivation with order information (qua third order lambda term) to a derivation tree, i.e.

  a representation which keeps track only of which rule was used to rewrite which nonterminal, and not the relative order in which the rules were used.

Example

The transformation g which identifies 'equivalent' derivations:

$$\overline{p} \longmapsto \begin{cases} \overline{p} & \text{if } p \in P \text{ is a non-lifted rule} \\ \lambda y_1 \dots y_m f \,.\, f\,(\overline{p}\, y_1 \cdots y_m) & \text{otherwise} \end{cases}$$

- $\overline{p}_3\,(\,\lambda g_2\,.\,\overline{p}_3\,(\,\lambda g_1\,.\,\overline{p}_4\, g_2\,(\,\lambda t_1\,.\,\overline{p}_1\, t_1\, g_1\,)\,)\,)$ left-to-right

  and

  $\overline{p}_3\,(\,\lambda g_2\,.\,\overline{p}_4\, g_2\,(\,\lambda t_1\,.\,\overline{p}_3\,(\,\lambda g_1\,.\,\overline{p}_1\, t_1\, g_1\,)\,)\,)$ right-to-left

  both are identified as

  $\overline{p}_1\,(\,\overline{p}_4\,\overline{p}_3\,)\,\overline{p}_3$ transformation under g

The object language

The object language of the new ACG is gotten by ‘composing’ the map g which turns a new abstract term into the old ‘IO’ one, and the original lexicon. (Note that this simply reuses the original ACG’s lexicon — the object language of the new ACG is obtained by first translating it back into the old ACG, and then interpreting the old ACG as usual.)

The object language

We need to prove that the translation $trans$ over types engenders an ACG whose abstract language exactly corresponds to the derivations of the original linear context-free tree grammar.

- One problem: the notion of derivation is not rich enough. We need to enrich it with information about which nonterminal is being rewritten. One way to do this is to redefine the notion of derivation:

$$\Rightarrow \subseteq (T_\Sigma \times P \times \mathbb{N}) \times T_\Sigma$$

That is, the one step derivation relation is between a tree (a sentential form), a rule, a natural number, and another tree:

$$C[A(t_1, \ldots, t_k)] \Rightarrow^p_n C[\,t\,[\,t_1, \ldots, t_k\,]\,]$$

if there are exactly $n - 1$ nonterminal symbols to the left of the bullet in $C[\bullet]$, and if there is some rule $p = A(x_1, \ldots, x_k) \to t$.

The object language

Now we want to say that each type derivation of an abstract term (a term in the abstract language) can be put into correspondence with a unique derivation in the CFTG, and what properties this correspondence should have.

- We will be interested in canonical (β-reduced, but η-long) terms. That is, in terms where all argument positions are explicitly filled, but where all applications have been computed.
- Note that the abstract language is a subset of the following:

$$L \ := \ \delta\, Y_1 \cdots Y_m \mid \rho\, Y_1 \cdots Y_m\, \lambda Y.L$$

  where δ is a rule rewriting the start symbol, and ρ is not.

The object language

- To define a mapping $\langle\!\langle \cdot \rangle\!\rangle$ from such terms to derivations in the underlying CFTG, we first need to be able to turn a term into a sentential form. We do this in the following way:
- For each nonterminal T in the CFTG, there is a type T in the ACG. We introduce a new zero-place constant $\overline{T}$ of type T for each nonterminal T. We define the lexicon $\mathcal{L}''$ to be the extension of $\mathcal{L}'$ which maps each $\overline{T}$ to

$$\mathcal{L}''(\overline{T}) \;=\; \lambda x_1 \dots x_{rank(T)} . T\, x_1 \cdots x_{rank(T)}$$

For $M \in \Lambda(\Sigma'_G)$ of atomic type with $FV(M) = \{y_1, \dots, y_m\}$ we define

$$[\![M]\!] := \mathcal{L}''((\lambda y_1, \dots, y_m . M)\, \overline{T}_1 \cdots \overline{T}_m\,)\,,$$

where $y_i : T_i$. Now $[\![M]\!]$ is simply the sentential form which M represents.

The object language

- We can define the translation from terms to derivations after introducing the following notation: Let $|\lambda x \,.\, M|$ denote the number of free variables which occur to the left of the first occurrence of x in M.

- Note, this definition treats the λ-term as a syntactic object — it distinguishes between terms which are β,η-equivalent. This is why we are working with the unique representative of a β,η-equivalence class which is β-reduced and η-long.

- Formally, define for any set B, any a being a constant or variable, and any lambda terms M and N:

$$
\begin{array}{rcl}
\mathit{str}(B, a) & = & \begin{cases} \varepsilon & \text{if } a \in B \\ a & \text{otherwise} \end{cases} \\
\mathit{str}(B, \lambda x \,.\, M) & = & \mathit{str}(B \cup \{x\}, M) \\
\mathit{str}(B, (MN)) & = & \mathit{str}(B, M)\widehat{\ }\mathit{str}(B, N)
\end{array}
$$

Then set $|\lambda x \,.\, M| := |u|$, where $\mathit{str}(C, M[x \to \bullet]) = u \bullet w$

The object language

- The translation can be given as

$$\langle\!\langle M \rangle\!\rangle = \begin{cases} S \Rightarrow^{\delta}_{1} [\![M]\!] & \text{if } M = \delta\, y_1 \cdots y_m \\ \langle\!\langle N \rangle\!\rangle \Rightarrow^{\rho}_{n+1} [\![M]\!] & \text{if } M = \rho\, y_1 \cdots y_m\, \lambda y.N \\ & \text{and if } n = |\lambda y.N| \end{cases}$$

Claim

For every term M of the abstract language $\mathcal{A}(\mathcal{G}'_G)$, $\langle\!\langle M \rangle\!\rangle$ is a derivation in G of $[\![M]\!]$.

Translating back to ACGs

- $\langle\!\langle\, S \Rightarrow^{\delta}_{1} D' \rangle\!\rangle^{-1} = [\![\, D' \,]\!]\,(\, \lambda \vec{y} \,.\, \delta\, \vec{y} \,)$
- $[\![\, \phi \Rightarrow^{\rho}_{n+1} D' \,]\!]\,(\, M \,) = [\![\, D' \,]\!]\,(\, \lambda\, \vec{x}\, \vec{u}\, \vec{v} \,.\, \rho\, \vec{x}\, \lambda\, z \,.\, (\, M\, \vec{u}\, z\, \vec{v}\,))$

  such that $n = |\lambda\, z \,.\, (\, M\, \vec{u}\, z\, \vec{v}\,)|$
- $[\![\, \phi \,]\!]\,(\, M \,) = M$

Claim

For every complete derivation D of t, $[\![\langle\!\langle D \rangle\!\rangle^{-1}]\!] = t$.
Moreover, $\langle\!\langle \cdot \rangle\!\rangle^{-1}$ is the inverse of $\langle\!\langle \cdot \rangle\!\rangle$.

Third order lambda terms really *are* nicer

- dequotienting of ‘equivalent’ derivations implicit in the use of derivation trees
- but easy to use because substitution is managed by β-reduction
- canonical terms correspond exactly to derivations

Barker's Analysis (qua CFTG)

- Some person left.

$$
\begin{aligned}
S &\to s(DP, v) \\
DP &\to d(det, NP) \\
NP &\to n
\end{aligned}
$$

- Some person from Jamaica left.

$$
\begin{aligned}
DP &\to d(det, NP'(NP)) \\
NP'(x) &\to NP'(np(x, pp(p, DP))) \\
NP'(x) &\to x \\
DP &\to name
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v) \\
&\Rightarrow s(d(det, NP'(NP)), v) \\
&\Rightarrow s(d(det, NP'(n)), v) \\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n))))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v)\\
&\Rightarrow s(d(det, NP'(NP)), v)\\
&\Rightarrow s(d(det, NP'(n)), v)\\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v) \\
&\Rightarrow s(d(det, NP'(NP)), v) \\
&\Rightarrow s(d(det, NP'(n)), v) \\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v) \\
&\Rightarrow s(d(det, NP'(NP)), v) \\
&\Rightarrow s(d(det, NP'(n)), v) \\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v) \\
&\Rightarrow s(d(det, NP'(NP)), v) \\
&\Rightarrow s(d(det, NP'(n)), v) \\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v)\\
&\Rightarrow s(d(det, NP'(NP)), v)\\
&\Rightarrow s(d(det, NP'(n)), v)\\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, {\color{blue}DP}))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, {\color{red}d(det, NP'(NP))}))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v) \\
&\Rightarrow s(d(det, NP'(NP)), v) \\
&\Rightarrow s(d(det, NP'(n)), v) \\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v) \\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

A Derivation

- Some person from every city left.

$$
\begin{aligned}
S &\Rightarrow s(DP, v)\\
&\Rightarrow s(d(det, NP'(NP)), v)\\
&\Rightarrow s(d(det, NP'(n)), v)\\
&\Rightarrow s(d(det, NP'(np(n, pp(p, DP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, DP))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP'(NP))))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, NP)))), v)\\
&\Rightarrow s(d(det, np(n, pp(p, d(det, n)))), v)
\end{aligned}
$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$S \to s(DP, v) \qquad (\rho_S)$$

$$DP \to d(det, NP'(NP)) \qquad (\rho_{DP})$$

$$NP \to n \qquad (\rho_N)$$

$$NP'(x) \to NP'(np(x, pp(p, DP))) \qquad (\rho_{NP'_a})$$

$$NP'(x) \to x \qquad (\rho_{NP'_b})$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$\begin{aligned}
S &\to s(DP, v) & DP \to S && (\rho_S)\\
DP &\to d(det, NP'(NP)) & && (\rho_{DP})\\
NP &\to n & && (\rho_N)\\
NP'(x) &\to NP'(np(x, pp(p, DP))) & && (\rho_{NP'_a})\\
NP'(x) &\to x & && (\rho_{NP'_b})
\end{aligned}$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$
\begin{array}{rlr}
S \rightarrow s(DP, v) & DP \rightarrow S & (\rho_S) \\
DP \rightarrow d(det, NP'(NP)) & NP' \rightarrow NP \rightarrow DP & (\rho_{DP}) \\
NP \rightarrow n & & (\rho_N) \\
NP'(x) \rightarrow NP'(np(x, pp(p, DP))) & & (\rho_{NP'_a}) \\
NP'(x) \rightarrow x & & (\rho_{NP'_b})
\end{array}
$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$
\begin{array}{rlr}
S \rightarrow s(DP, v) & DP \rightarrow S & (\rho_S) \\
DP \rightarrow d(det, NP'(NP)) & NP' \rightarrow NP \rightarrow DP & (\rho_{DP}) \\
NP \rightarrow n & NP & (\rho_N) \\
NP'(x) \rightarrow NP'(np(x, pp(p, DP))) & & (\rho_{NP'_a}) \\
NP'(x) \rightarrow x & & (\rho_{NP'_b})
\end{array}
$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$
\begin{array}{lrr}
S \to s(DP, v) & DP \to S & (\rho_S) \\
DP \to d(det, NP'(NP)) & NP' \to NP \to DP & (\rho_{DP}) \\
NP \to n & NP & (\rho_N) \\
NP'(x) \to NP'(np(x, pp(p, DP))) & NP' \to DP \to NP' & (\rho_{NP'_a}) \\
NP'(x) \to x & & (\rho_{NP'_b})
\end{array}
$$

The abstract language of Barker's TAG

- Here are the rules, and their second-order types:

$$
\begin{array}{rlr}
S \to s(DP, v) & DP \to S & (\rho_S) \\
DP \to d(det, NP'(NP)) & NP' \to NP \to DP & (\rho_{DP}) \\
NP \to n & NP & (\rho_N) \\
NP'(x) \to NP'(np(x, pp(p, DP))) & NP' \to DP \to NP' & (\rho_{NP'_a}) \\
NP'(x) \to x & NP' & (\rho_{NP'_b})
\end{array}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to \color{red}{S} \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Lifting Barker's TAG

And here the types lifted over S:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: (NP \to S) \to S \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: (NP \to S) \to S \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: (NP \to S) \to S \\
\rho_{NP'_a} &: NP' \to DP \to (NP' \to S) \to S \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: (NP \to S) \to S \\
\rho_{NP'_a} &: NP' \to DP \to (NP' \to S) \to S \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Lifting Barker's TAG

And here the types lifted over S:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: (NP \to S) \to S \\
\rho_{NP'_a} &: NP' \to DP \to (NP' \to S) \to S \\
\rho_{NP'_b} &: {\color{red}(NP' \to S) \to S}
\end{aligned}
$$

Derivations

- Some person left

  basic (2nd order):

$$\rho_S\,(\,\rho_{DP}\,(\,\rho_{NP'_b}\;\rho_{NP}\,)\,)$$

  lifted (3rd order):

$$\rho_N(\,\lambda x_N\,.\,\rho_{NP'_b}\,(\,\lambda x_{NP}\,.\,\rho_{DP}\,(\,x_{NP}\;x_N\;\lambda x_D\,.\,\rho_S\,x_D\,)\,)\,)$$

- Some person from every city left

  basic:

$$\rho_S\,(\,\rho_{DP}\,(\,\rho_{NP'_a}\,(\,\rho_{NP'_b}\;\rho_{DP}\,(\,\rho_{NP'_b}\;\rho_{NP}\,)\,)\,\rho_{NP}\,)\,)$$

  lifted

$$_{np}.\rho_{NP'_b}(\lambda x_{np'}.\rho_{DP}(x_{np'}\,x_{np}\,(\lambda x_d.\rho_{NP'_b}(\lambda x_{np'}.\rho_{NP'_a}(x_{np'}\,x_d\,(\lambda x_{np'}.\rho_N(\lambda x_{np}.\rho_{DP}(x_{np'}\,x_{np}\,(\lambda x_d.\rho_S\,x_d\,)))$$

The Problem of Inverse Linking

Observation:

derivation order does not permit surface scope in:

- *every person from a city left.*

The Problem of Inverse Linking

Observation:

derivation order does not permit surface scope in:

- *every person from a city left.*

The problem:

the most deeply embedded quantifier cannot be introduced until *after* its 'argument' position is present, which is introduced only once its containing DP is present

The Problem of Inverse Linking

Observation:

derivation order does not permit surface scope in:

- *every person from a city left.*

The problem:

the most deeply embedded quantifier cannot be introduced until *after* its 'argument' position is present, which is introduced only once its containing DP is present

Example

Every derivation begins:

$\langle\!\langle S \Rightarrow s(DP, v) \rangle\!\rangle^{-1} =$

$$\lambda x_D.\rho_S(x_D)$$

The Problem of Inverse Linking

Observation:

derivation order does not permit surface scope in:

- *every person from a city left.*

The problem:

the most deeply embedded quantifier cannot be introduced until *after* its 'argument' position is present, which is introduced only once its containing DP is present

Example

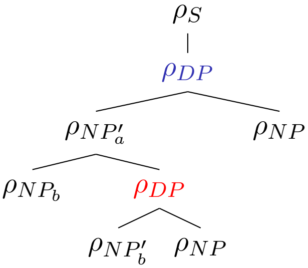
Every derivation begins:

$$\langle\!\langle S \Rightarrow s(DP, v) \Rightarrow s(d(det, NP'(NP)), v) \rangle\!\rangle^{-1} =$$

$$\lambda x_{NP'}\, x_N \,.\, \rho_{DP}\,(\, x_{NP'}\, x_N\, \lambda x_D . \rho_S(x_D)\,)$$

Constituency in derivations

A timing based approach to quantifier scope simply cannot derive surface scope relations between embedded scope-takers: the scope relations predicted are always inverted.



The possible derivations for t are given by $h(t)$, where $h(\sigma(t_1, \ldots, t_n)) := \sigma \cdot \bigsqcup\{h(t_1), \ldots, h(t_n)\}$

$$\rho_S \cdot (\rho_{DP} \cdot (\rho_{NP} \sqcup (\rho_{NP'_a} \cdot (\rho_{NP'_b} \sqcup (\rho_{DP} \cdot (\rho_{NP'_b} \sqcup \rho_{NP}))))))$$

Timing $\neq$ Derivation

- Barker wants scope relations such that $\rho_{DP} < \rho_{DP}$
- There is no derivation which provides this
- No system which derives this can be expressed in terms of derivations

Whatever people mean by "timing"

it is not describable in terms of rewriting order

Back to Barker

- We simply lifted all of our types
  - because we wanted to look at every possible derivation order
- Barker only allows for cosubstitution of certain initial trees (the DPs)
  - this corresponds to us only lifting types ending in DP
  - the resulting signature has *nothing* to do with derivations

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to DP \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$
\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: {\color{red}NP} \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}
$$

Only types ending in DP:

$$\begin{aligned} \rho_S &: DP \rightarrow S \\ \rho_{DP} &: NP' \rightarrow NP \rightarrow (DP \rightarrow S) \rightarrow S \\ \rho_N &: NP \\ \rho_{NP'_a} &: NP' \rightarrow DP \rightarrow NP' \\ \rho_{NP'_b} &: NP' \end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to \color{red}{NP'} \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: NP'
\end{aligned}$$

Selectively Lifting Barker's TAG

Only types ending in DP:

$$\begin{aligned}
\rho_S &: DP \to S \\
\rho_{DP} &: NP' \to NP \to (DP \to S) \to S \\
\rho_N &: NP \\
\rho_{NP'_a} &: NP' \to DP \to NP' \\
\rho_{NP'_b} &: \color{red}{NP'}
\end{aligned}$$

We *still* don't get the surface reading!

- but this time its easy to fix:

Barker's solution

allow a version of rule $\rho_{NP'_a}$ to directly select for a continuized DP:

$$\rho'_{NP'_a} := NP' \rightarrow ((DP \rightarrow S) \rightarrow S) \rightarrow NP$$

A lexical solution

introduce new atomic type XP, and duplicate rules as appropriate:

$$XP \rightarrow d(det, NP'(NP)) \qquad (\rho_{XP})$$

$$NP'(x) \rightarrow NP'(np(x, pp(p, XP))) \qquad (\rho_{NP'_c})$$

lexically interpret $\mathcal{L}(XP) = \mathcal{L}((DP \rightarrow S) \rightarrow S)$

Conclusions

Cotags

- leave the realm of context-free grammar formalisms
- WGC and SGC preservation
  - NOT because IO and OI coincide on linear CFTGs
  - instead, because spellout involves going back through the 2^{nd} order derivation term
- Complexity heuristics in ACG differ from those of TAG:
  - 'simpler' to deal with in-situ readings by making lexicon higher order than the abstract language . . . (??)

Conclusions

Timing

- intuitive and oft-occuring
- not related to rewriting strategies
  (we may have been the only ones to have thought this. . .)
- best seen as moving from *trees* to (3^{rd} order) *terms*

Thank you!