Traces Exist (Hypothetically)!

Carl Pollard

Structure and Evidence in Linguistics
Workshop in Honor of Ivan Sag
Stanford University

April 28, 2013

Traces in Transformational Grammar (1/2)

- Traces are usually thought to have been invented (discovered?) by linguists at MIT in the early 1970's:

  > WH-fronting could be formulated so that a phonetically null copy of the WH-word is left behind in its pre-fronting position. [Wasow 1972:139, attributed to Culicover (p.c.)]
  >
  > [A]ssuming that *wh*-Movement leaves a trace PRO, we might then stipulate that every rule that moves an item from an obligatory category (in the sense of Emonds (1970)) leaves a trace. [Chomsky 1973:135, fn. 49]

- Subsequently they became a mainstay of TG:

  > [D-structures] are mapped to S-structures by the rule Move-α, leaving traces coindexed with their antecedents .... [Chomsky 1981:5]

Traces in Transformational Grammar (2/2)

But even in TG, the ontological status of traces has not been completely straightforward:

> [T]he correct LF for (32)
>
> (32) *Who did Mary say that John kissed t*
>
> should be
>
> (37) *for which x, x a person, Mary said that John kissed [x]*
>
> The LF (37) has a terminal symbol, x, in the position of the NP source of *who*, but (32) has only a trace, i.e. only the structure $[_{\mathrm{NP}_i}\ e]$, where i is the index of *who*. [Chomsky 1977:83-84]

Traces in Phrase Structure Grammar (1/2)

- In Gazdar 1981, if A and B are syntactic categories, then so is A/B. Then the notion of trace is expressed as

  $$A/A \to t$$

  which is a lexical entry schema for the null string.
- Pollard and Sag's (1994:161) trace schema is the same as Gazdar's, recoded as an AVM:

  $$[\text{PHON } \langle\rangle, [\text{SYNSEM } [\text{LOC } \boxed{1}, \text{NONLOC}|\text{SLASH } \boxed{1}]]]$$

Traces in Phrase Structure Grammar (2/2)

- But Pollard and Sag (1994:378–387) eliminated traces in favor of three lexical rules responsible, respectively, for extraction of complements, subjects, and adjuncts.
- Sag and Fodor (1995) defended this analysis on empirical grounds, noting also the absence of (analogs of) traces in CCG andf LFG.
- Sag, Wasow, and Bender (2003) barely mention traces.

Natural Deduction

- *Natural deduction* (Gentzen 1934, Prawitz 1965) is a style of theorem proving characterized by the presence of inference rule schemas for introducing and eliminating logical connectives (examples coming right up).
- Below we'll focus on *implicative linear logic (ILL)*, which has just one connective $\multimap$ (linear implication).
- The premisses and conclusion of rules are *sequents* of the form $\Gamma \vdash A$, read 'A is deducible from the hypotheses Γ'.
    - A is a formula, called the *statement* of the sequent
    - Γ is a multiset of formulas, called the *context* of the sequent.
    - Commas in contexts represent multiset union.

Implicative Linear Logic (1/2)

- In ILL, the only rules are

Implication Elimination, aka Modus Ponens

$$\Gamma, \Delta \vdash B$$

$$\Gamma \vdash A \multimap B \qquad \Delta \vdash A$$

Implication Introduction, aka Hypothetical Proof

$$\Gamma \vdash A \multimap B$$

$$\Gamma, A \vdash B$$

- Each rule is a local tree with the daughter(s) labelled by premisses and the mother labelled by the conclusion.
- Contexts at each node represent undischarged hypotheses.
- There is also a logical axiom schema (Hypothesize):

$$A \vdash A$$

Implicative Linear Logic (2/2)

- With these, we can prove any ILL theorem, e.g. TR:

$$\vdash A \multimap (A \multimap B) \multimap B$$

$$A \vdash (A \multimap B) \multimap B$$

$$A, A \multimap B \vdash B$$

$$A \vdash A \qquad A \multimap B \vdash A \multimap B$$

- A proof is a tree.
- Each leaf is labelled by an axiom.
- Each nonleaf and its daughters instantiates one of the rules.
- The sequent labelling the root is the theorem proved.

ILL vs. PSG

- If only the natural deduction turnstile $\vdash$ and and Gazdar's slash / were the same thing, the Hypothesize axiom schema

$$A \vdash A$$

would be the same as Gazdar's syntactic category for traces

$$A/A$$

- That would only make sense if
  - a grammar was a natural deduction system
  - phrase structure trees were proof trees
  - linguistic expressions were sequents
  - lexical entries (not only traces) were axioms
- These things are all true!
- To see why, we have to reformulate PSG in terms of ILL.

The Curry-Howard Correspondence (1/2)

- Curry (1958) and Howard (1969) discovered a connection between implicative logic and lambda calculus: if we think of formulas as types, then a formula is a theorem iff there is a combinator (pure closed lambda term) of that type.
- For ILL, lambda terms are assigned to types/formulas is as follows:

$$x : A \vdash x : A$$

$$\Gamma, \Delta \vdash (M\ N) : B$$

$$\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A$$

$$\Gamma \vdash \lambda_x M : A \multimap B$$

$$\Gamma, x : A \vdash M : B$$

The Curry-Howard Correspondence (2/2)

- For example, the fact that TR is a theorem corresponds to the fact the combinator $\lambda_{xf}.(f\ x)$ has that type.
- We can see this by adding type annotations to the proof of TR we just gave:

$$\vdash \lambda_{xf}(f\ x) : A \multimap (A \multimap B) \multimap B$$

$$x : A \vdash \lambda_{f}(f\ x) : (A \multimap B) \multimap B$$

$$x : A, f : A \multimap B \vdash (f\ x) : B$$

$$f : A \multimap B \vdash f : A \multimap B \qquad x : A \multimap x : A$$

- This correspondence between theorems and terms is called the *Curry-Howard* correspondence.

Phenogrammar and Tectogrammar

- In his one foray into linguistics, Curry (1961) proposed that syntax should be bifurcated into *phenogrammatical structure* (roughly, surface form) and *tectogrammatical structure* (roughly, semantically motivated combinatorics).
- Curry's idea influenced PSGians (Reape, Kathol) and CGians (Dowty, Oehrle).
- In particular, Oehrle (1994) invented a kind of categorial grammar based on ILL, here called *linear grammar (LG)*.
- In the rest of this talk, I'll sketch how to logically reconstruct the PSG theory of UDCs, by identifying Gazdar's / with the natural deduction turnstile $\vdash$.

LG Basics: Phenogrammatical Types and Terms

- LG analyses consist of two simultaneous natural deduction proofs, one in the pheno dimension and one in the tecto dimension. (There is also a Montague-like semantic dimension, omitted here.)
- The only base type in the pheno logic is s (*string*).
- If A and B are pheno types, so is $A \to B$.
- The pheno proof is annotated with lambda terms, called *pheno terms*, that encode the surface form.
- There are pheno constants of type s correponding to lexical phonologies, such as he, is, easy, etc.
- There is also a pheno constant **e** of type s corresponding to the null string.
- There is an (infix) constant $\cdot$ of type $\mathrm{s} \to \mathrm{s} \to \mathrm{s}$ for concatenation.

LG Basics: Tectogrammatical Types

- The base types for the tecto logic are:

  $\mathrm{S}_{\mathbf{f}}$ (finite clause)

  $\mathrm{S}_{\mathbf{i}}$ (infinitive clause)

  $\mathrm{S}_{\mathbf{b}}$ (base-form clause)

  $\bar{\mathrm{Q}}$ (embedded interrogative clause)

  PrdA (predicative adjectival clause)

  $\mathrm{NP}_{\mathbf{n}}$ (nominative NP)

  $\mathrm{NP}_{\mathbf{a}}$ (accusative NP)

  $\mathrm{NP}_{\mathbf{it}}$ (dummy *it*)

  $\mathrm{PP}_{\mathbf{for}}$ (*for*-PP)

- If A and B are tecto types, so is $A \multimap B$.
- There is no need to distinguish between (categorial) / vs. \ (as in CCG or Lambek calculus) because constituent ordering is handled in the pheno component.

LG Basics: Nonlogical Axioms (Lexical Entries)

Types of pheno terms are omitted to save space.

$\textbf{she} = \vdash \text{she}; \text{NP}_{\mathbf{n}}$

$\textbf{he} = \vdash \text{he}; \text{NP}_{\mathbf{n}}$

$\textbf{him} = \vdash \text{him}; \text{NP}_{\mathbf{a}}$

$\textbf{her} = \vdash \text{her}; \text{NP}_{\mathbf{a}}$

$\textbf{it} = \vdash \text{it}; \text{NP}_{\mathbf{it}}$

$\textbf{pleases} = \vdash \lambda_{st}.t \cdot \text{pleases} \cdot s; \text{NP}_{\mathbf{a}} \multimap \text{NP}_{\mathbf{n}} \multimap \text{S}_{\mathbf{f}}$

$\textbf{please} = \vdash \lambda_{s}.\text{please} \cdot s; \text{NP}_{\mathbf{a}} \multimap \text{NP}_{\mathbf{n}} \multimap \text{S}_{\mathbf{b}}$

$\textbf{is} = \vdash \lambda_{st}.t \cdot \text{is} \cdot u; (A \multimap \text{PrdA}) \multimap A \multimap \text{S}_{\mathbf{f}}$

$\textbf{to} = \vdash \lambda_{s}.\text{to} \cdot s; (A \multimap \text{S}_{\mathbf{b}}) \multimap (A \multimap \text{S}_{\mathbf{i}})$

$\textbf{for} = \vdash \lambda_{s}.\text{for} \cdot s; \text{NP}_{\mathbf{a}} \multimap \text{PP}_{\mathbf{for}}$

$\textbf{easy}_{\mathbf{1}} = \vdash \lambda_{st}.\text{easy} \cdot s \cdot t; \text{PP}_{\mathbf{for}} \multimap (\text{NP}_{\mathbf{n}} \multimap \text{S}_{\mathbf{i}}) \multimap \text{NP}_{\mathbf{it}} \multimap \text{PrdA}$

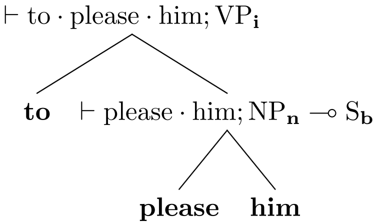
LG Basics: The Combine Rule

$$\Gamma, \Delta \vdash (M\ \ N); B$$

$$\Gamma \vdash M; A \multimap B \qquad \Delta \vdash N; A$$

- This is the LG version of Modus Ponens.
- It replaces all the PSG phrasal schemas.
- It is the only rule needed for analyzing local dependencies.
- Think of a sequent $\Gamma \vdash M; A \multimap B$ as $[\text{PHON } M; \text{HEAD } B; \text{SUBCAT } A; \text{SLASH } \Gamma]$
- Combine incorporates the effect of
  - the Head Feature Principle
  - the Valence Principle (but only one argument is discharged per rule application)
  - the GAP Principle (*sans* STOP-GAP, which is handled by the other rule).

to please him



Here and henceforth $\text{VP}_{\mathbf{i}}$ abbreviates $\text{NP}_{\mathbf{n}} \multimap \text{S}_{\mathbf{i}}$.

easy for her to please him

$$\vdash \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \text{him}; \text{NP}_{\textbf{it}} \multimap \text{PrdA}$$

$$\vdash \lambda_t \text{easy} \cdot \text{for} \cdot \text{her} \cdot t; \text{VP}_{\textbf{i}} \multimap \text{NP}_{\textbf{it}} \multimap \text{PrdA} \qquad \vdash \text{to} \cdot \text{please} \cdot \text{him}; \text{VP}_{\textbf{i}}$$

$$\textbf{easy}_{\textbf{1}} \qquad \vdash \text{for} \cdot \text{her}; \text{PP}_{\textbf{for}}$$

$$\textbf{for} \qquad \textbf{her}$$

It is easy for her to please him

$$\textbf{is} \quad \vdash \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \text{him}; \mathrm{NP}_{\textbf{it}} \multimap \mathrm{PrdA}$$

$$\textbf{it} \quad \vdash \lambda_t t \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \text{him}; \mathrm{NP}_{\textbf{it}} \multimap \mathrm{S}_{\mathbf{f}}$$

$$\vdash \text{it} \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \text{him}; \mathrm{S}_{\mathbf{f}}$$

LG Basics: The Stop-Gap Rule

$$\dfrac{\Gamma, t; A \vdash M; B}{\Gamma \vdash \lambda_t M; A \multimap B}$$

- This is the LG version of Hypothetical Proof.
- There is no PSG rule corresponding to this rule.
- Instead, the PSG counterpart is the STOP-GAP (or TO-BIND) feature on the lexical head of the Head-Filler Rule and lexical entries like *easy*.
- Stop-Gap discharges a hypothesis (trace) and lambda-binds the string variable t that it introduced.

LG Basics: Trace

$$t; A \vdash t; A$$

- This is the LG counterpart of the Hypothesize schema
- Here t is a variable of type s (string)
- A can be instantiated by any tecto type, e.g.

$$t; \mathrm{NP}_{\mathbf{a}} \vdash t; \mathrm{NP}_{\mathbf{a}}$$

- Think of $\mathrm{NP}_{\mathbf{a}} \vdash \mathrm{NP}_{\mathbf{a}}$ as LG-ese for $\mathrm{NP}_{\mathbf{a}}[\mathrm{SLASH}\ \langle \mathrm{NP}_{\mathbf{a}} \rangle]$.
- Equipped with Stop-Gap and Trace, we can analyze UDCs as soon as we add suitable lexical entries.

LG Basics: Lexical Entries for UDCs

$\textbf{whom} = \vdash \lambda_f.\text{whom} \cdot (f\ \mathbf{e}); (\text{NP}_{\mathbf{a}} \multimap \text{S}_{\mathbf{f}}) \multimap \bar{\text{Q}}$

$\textbf{easy}_{\mathbf{2}} = \vdash \lambda_{sf}.\text{easy} \cdot s \cdot (f\ \mathbf{e}); \text{PP}_{\mathbf{for}} \multimap (\text{NP}_{\mathbf{a}} \multimap \text{VP}_{\mathbf{i}}) \multimap \text{NP}_{\mathbf{n}} \multimap \text{PrdA}$

In both of these lexical entries:

- one of the arguments has an $\text{NP}_{\mathbf{a}}$ gap (which will have been discharged by an application of Stop-Gap)
- The bound variable f is of type $\text{s} \rightarrow \text{s}$ (functions from strings to strings), corresponding to the gappy argument
- when the lexical entry combines with that argument, the null string **e** is lambda-converted into the gap position!
- As much as I would like to take credit for it, this bit of pheno-technology was invented by Muskens (2007).

whom she pleases

$$\vdash \text{whom} \cdot \text{she} \cdot \text{pleases} \cdot \mathbf{e}; \bar{\mathrm{Q}}$$

$$\textbf{whom} \qquad \vdash \lambda_s \text{she} \cdot \text{pleases} \cdot s; \mathrm{NP}_{\mathbf{a}} \multimap \mathrm{S}_{\mathbf{f}}$$

$$s; \mathrm{NP}_{\mathbf{a}} \vdash \text{she} \cdot \text{pleases} \cdot s; \mathrm{S}_{\mathbf{f}}$$

$$\textbf{she} \qquad s; \mathrm{NP}_{\mathbf{a}} \vdash \lambda_t t \cdot \text{pleases} \cdot s; \mathrm{NP}_{\mathbf{n}} \multimap \mathrm{S}_{\mathbf{f}}$$

$$\textbf{pleases} \qquad s; \mathrm{NP}_{\mathbf{a}} \vdash s; \mathrm{NP}_{\mathbf{a}}$$

- The non-branching node is the instance of Stop-Gap that binds the $\mathrm{NP}_{\mathbf{a}}$ trace.
- That together with the instance of Combine just above it capture the effect of HPSG's Filler-Head rule.

to please t

$$\vdash \lambda_s \mathrm{to} \cdot \mathrm{please} \cdot s; \mathrm{NP}_{\mathbf{a}} \multimap \mathrm{VP}_{\mathbf{i}}$$

$$s; \mathrm{NP}_{\mathbf{a}} \vdash \mathrm{to} \cdot \mathrm{please} \cdot s; \mathrm{VP}_{\mathbf{i}}$$

$$\textbf{to} \qquad s; \mathrm{NP}_{\mathbf{a}} \vdash \mathrm{please} \cdot s; \mathrm{NP}_{\mathbf{n}} \multimap \mathrm{S}_{\mathbf{b}}$$

$$\textbf{please} \qquad s; \mathrm{NP}_{\mathbf{a}} \vdash s; \mathrm{NP}_{\mathbf{a}}$$

Again, the nonbranching node is the instance of Stop-Gap that binds the $\mathrm{NP}_{\mathbf{a}}$ trace.

easy for her to please t

- $\vdash \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \mathbf{e}; \text{AP}$
  - $\vdash \lambda_f \text{easy} \cdot \text{for} \cdot \text{her} \cdot (f\ \mathbf{e}); (\text{NP}_{\mathbf{a}} \multimap \text{VP}_{\mathbf{i}}) \multimap \text{AP}$
    - $\mathbf{easy_2}$
    - $\vdash \text{for} \cdot \text{her}; \text{PP}_{\mathbf{for}}$
      - **for**
      - **her**
  - $\vdash \lambda_s \text{to} \cdot \text{please} \cdot s; \text{NP}_{\mathbf{a}} \multimap \text{VP}_{\mathbf{i}}$

- Here AP abbreviates $\text{NP}_{\mathbf{n}} \multimap \text{PrdA}$.
- By the time *easy* combines with the infinitive VP, its $\text{NP}_{\mathbf{a}}$ gap has already been bound.
- So there is no need for *easy* to have a STOP-GAP feature.

He is easy for her to please t

$$\vdash \text{he} \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please}; \mathrm{S}_{\mathbf{f}}$$

$$\textbf{he} \qquad \vdash \lambda_t t \cdot \text{is} \cdot \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please}; \mathrm{NP}_{\mathbf{it}} \multimap \mathrm{S}_{\mathbf{f}}$$

$$\textbf{is} \qquad \vdash \text{easy} \cdot \text{for} \cdot \text{her} \cdot \text{to} \cdot \text{please} \cdot \mathbf{e}; \mathrm{AP}$$

Summary

We logically reconstructed PSG inside of linear grammar.

- Phrase structure trees become *natural-deduction proof trees*.
- Node labels become *sequents*.
- SLASH becomes the *turnstile* ($\vdash$) in sequents.
- SLASH values become the *contexts* in sequents.
- The valence features all become *linear implication* ($\multimap$).
- Traces become *hypotheses (logical axioms)*.
- Other lexical entries become *nonlogical axioms*.
- The phrasal schemas collapse into *Modus Ponens (Combine)*.
- The only other rule is *Hypothetical Proof (Stop-Gap)*, which does the work of PSG's STOP-GAP feature.

I wish we had known about natural deduction 30 years ago!

References

Chomsky, N. 1973. Conditions on transformations. In S. Anderson and P. Kiparsky, eds., *Festschrift for Morris Halle*. New York: Holt, Rinehart, and Winston, 232-285.

Chomsky, N. 1977. On *wh*-movement. In P. Culicover, T. Wasow, and A. Akmajian, eds., *Formal Syntax*. New York: Academic Press, 71-132.

Chomsky, N. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Curry, H. 1961. Some logical aspects of grammatical structure. In R. Jakobson, ed., *Structure of Language in its Mathematical Aspects*. Providence: American Mathematical Society, 56-68.

Curry, H. and R. Feys. 1958. *Combinatory Logic, Vol. 1*. North-Holland.

Dollop, S. and I. Slag. 1987. Realist unification grammar. *CSLI Monthly* 2(6):2-3.

Gazdar, G., E. Klein, G. Pullum, and I, Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Muskens, R. 2007. Separating syntax and combinatorics in categorisl grammar. *Research on Language and Computation* **5**(3):267-285.

Pollard, C. 1988. Categorial grammar and phrase structure grammar. In R. Oehrle, E. Bach, and D. Wheeler, eds..*Categorial Grammars and Natural Language structures*. Dordrecht:Reidel, 391-416.

Pollard, C. and I. Sag. 1994 *Head-Driven Phrase Structure Grammar*. Stanford: CSLI Publications and Chicago: University of Chicago Press.

Oehrle, R. 1994. Term-labelled categorial type systems. *Linguistics and Philosophy* **17**(6):633-678.

Sag, I. and J. Fodor. 1995. Extraction without traces. R. Aranovich et al., eds., *The Proceedings of the 13th West Coast Conference on Formal Linguistics'* . Stanford : CSLI Publications, 365-384.

Sag, I., T. Wasow, and E. Bender. 2003. *Syntactic Theory*. Stanford: CSLI Publications.

Wasow, T. 1972. *Anaphoric Relations in English*. Ph.D. dissertation, MIT.