# A PARALLEL-DERIVATIONAL ARCHITECTURE FOR THE SYNTAX-SEMANTICS INTERFACE

Carl Pollard
INRIA-Lorraine and Ohio State University

ESSLLI 2008 Workshop on
What Syntax Feeds Semantics
Hamburg, August 14, 2008

**These slides are available at:**

`http://www.ling.ohio-state.edu/~pollard/cvg/para-slides.pdf`

(1) **Back in 1970:**

- Montague's "Universal Grammar" and "English as a Formal Language" were published, proposing that NL syntactic derivations (analysis trees) and their meanings were constructed **in parallel**.

  In particular, there was nothing 'between' syntax and semantics.

- Chomsky's "Conditions on Transformations" (not published till 1973) introduced the **T-model**, in which interpretive rules applied between SS and LF:

$$\begin{tikzcd}
\text{Phonetics} & \text{PF} \arrow[l] & \text{SS} \arrow[l] \arrow[r] & \text{LF} \arrow[r] & \text{Semantics} \\
 & & \text{DS} \arrow[u] & & \\
 & & \text{LEX} \arrow[u] & &
\end{tikzcd}$$

(2) **The Cascade**

Straightening the right arm of the T and suppressing the left arm:

$$\begin{tikzcd}
\text{Semantics} \\
\text{LF} \arrow[u, "?"'] \\
\text{SS} \arrow[u, "C"'] \\
\text{DS} \arrow[u, "O"'] \\
\text{LEX} \arrow[u, "M"']
\end{tikzcd}$$

with the subscripts on the arrows distinguishing the three rule cycles (with more modern names) Merge, Overt Move, and Covert Move.

(3) **A Convergence of Views**

- The Cascade has long since been rejected—by all—because (in mainstream parlance) the three kinds of operations have to be intermingled: merges must be able to follow moves, and overt moves must be able to follow covert ones. Therefore:
- 
  - There is only a single cycle of operations.
  - DS and SS do not exist.
  - There are multiple points in a derivation where the syntax connect to the interface systems.
- The Minimalist Program (MP) is one framework for filling in the details of this consensus view.
- This talk is about a different one, worked out within the framework of **Extended Montague Grammar** (EMG) about 30 years ago.

(4) **Three Signal Achievments of EMG**

- Cooper's (1975) **storage** replaced **covert** movement.
- Gazdar's (1979) **linking schemata** replaced **overt** movement.
- Bach and Partee (1980) incorporated both into a PSG-based account of (what would later be called) **binding theory** facts, which anticipated later categorial treatments.

(5) **Why Reconstruct EMG?**

- EMG had already correctly perceived many of the main defects of the T-model and had good proposals for fixing them.
- But 30 years later, central EMG tenets (such as nonexistence of movement and of LF) remain outside the “mainstream”.
- So the case for EMG needs to be made anew.
- A promising approach is to reformulate the EMG ideas using an especially transparent formalism: **Gentzen natural deduction with Curry-Howard proof terms** (hereafter just ND).

(6) **Easier than it Sounds**

- The proof trees look just like familiar phrase markers.
- Each node in the tree is labelled with two terms, a syntactic one and a semantic one.
- The syntactic term is just a slightly upgraded version of a 1970's-style labelled bracketting.
- The semantic term is just an ordinary lambda term.
- The leaves are either lexical entries or traces.
- Each non-leaf node is licensed by a rule that constructs that nodes's syntactic (semantic) term from the syntactic (semantic) terms of the daughters.

(7) **Reformulating EMG using ND**

- We have **two** logics, each with its own ND proof theory, which specify (respectively) **candidate** syntactic and semantic terms.
- The **syntax-semantics interface** recursively defines the set of syntactic/semantic term-pairs that belong to the NL in question.
- We call those pairs the **signs** of the NL.
- The signs are the inputs to the interpretive interfaces:
  - the syntactic component is phonetically interpreted, and
  - the semantic component is semantically interpreted.
- We call this style of grammar **Convergent Grammar** (CVG).

(8) **Parallel-Derivational (PD) Artchitecture**

$$\begin{tikzcd}
 & \text{phonetics} & \\
 & \text{Syn} \arrow[u] & \\
\text{Syn candidates} \arrow[r] & + & \text{Sem candidates} \arrow[l] \\
 & \text{Sem} \arrow[d] & \\
 & \text{semantics} &
\end{tikzcd}$$

(9) **Time is Short**

- So if you want to know what the syntactic and semantic rules look like in isolation, you will have to read the handout.
- Here we skip straight to the syntax-semantics interface rules, which are just pairings of syntactic rules with semantic rules.
- Then we'll look at some representative analyses:

(10) **Some Lexical Entries (0-ary Rules)**

$\vdash \mathsf{Chris}, \mathsf{Chris'} : \mathrm{NP}, \mathrm{e}$

$\vdash \mathsf{everyone}, \mathsf{everyone'} : \mathrm{NP}, \mathrm{e}_{\mathrm{t}}^{\mathrm{t}} \dashv$

$\vdash \mathsf{someone}, \mathsf{someone'} : \mathrm{NP}, \mathrm{e}_{\mathrm{t}}^{\mathrm{t}}$

$\vdash \mathsf{liked}, \mathsf{like'} : \mathrm{NP} \multimap_{\mathrm{C}} \mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}, \mathrm{e} \to \mathrm{e} \to \mathrm{t}$

$\vdash \mathsf{thought}, \mathsf{think'} : \mathrm{S} \multimap_{\mathrm{C}} \mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}, \pi \to \mathrm{e} \to \mathrm{t}$

**Note:** Semantic types of the form $A_B^C$ are for in-situ operators that bind an $A$-variable in a $B$, forming a $C$.

This differs from Moortgat's $\mathsf{q}(A, B, C)$ or Barker-Shan's $C \mathbin{/\!\!/} (A \mathbin{\backslash\!\!\backslash} B)$ because those are **syntactic categories**: on our account the syntactic category of a QNP is just NP.

(11) **Schema $\mathbf{M}_{\mathrm{s}}$ (Subject Modus Ponens, version 1)**

If $\vdash a, c : A, C \dashv$ and $\vdash f, v : A \multimap_{\mathrm{S}} B, C \rightarrow D \dashv$,
then $\vdash ({}^{\mathrm{S}}\ a\ f), (v\ c) : B, D \dashv$

Heads combine with subjects semantically by function application.

(12) **Schema $\mathbf{M}_{\mathrm{s}}$ (Subject Modus Ponens, final version)**

If $\Gamma \vdash a, c : A, C \dashv \Delta$ and $\Gamma' \vdash f, v : A \multimap_{\mathrm{s}} B, C \rightarrow D \dashv \Delta'$,
then $\Gamma; \Gamma' \vdash ({}^{\mathrm{s}}\ a\ f), (v\ c) : B, D \dashv \Delta; \Delta'$

Heads combine with subjects semantically by function application.

Contexts (unbound traces) and co-contexts (Cooper-stored operators) get passed up (as in old-fashioned PSG).

(13) **Schema $\mathbf{M_c}$ (Complement Modus Ponens)**

If $\Gamma \vdash f, v : A \multimap_{\mathrm{C}} B, C \rightarrow D \dashv \Delta$ and $\Gamma' \vdash a, c : A, C \dashv \Delta'$,
then $\Gamma; \Gamma' \vdash (f\ a\ ^{\mathrm{C}}), (v\ c) : B, D \dashv \Delta; \Delta'$

Just like the preceding but for complements instead of subjects.

These schemata (and their counterparts for other grammatical functions) are our analogs of Merges in TG.

(14) **A Simple Sentence**

a. Chris thinks Kim likes Dana.

b. $\vdash$ ($^{\mathrm{S}}$ Chris (thinks ($^{\mathrm{S}}$ Kim (likes Dana $^{\mathrm{C}}$) $^{\mathrm{C}}$))) :
((think' ((like' Dana') Kim')) Chris') : $\mathrm{S}, \mathrm{t} \dashv$

(15) **Schema C (Cooper Storage)**

If $\Gamma \vdash a, b : A, B^D_C \dashv \Delta$, then $\Gamma \vdash a, x : A, B \dashv b_x : B^D_c; \Delta$ ($x$ fresh)

When a semantic operator is stored, nothing happens in the syntax.

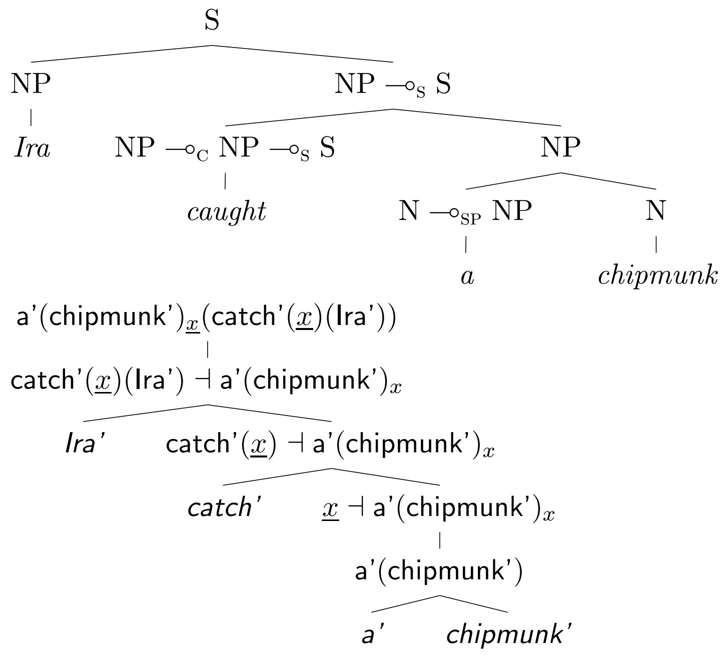
(16) **Schema R (Retrieval)**

If $\Gamma \vdash e, c[x] : E, C \dashv b_x : B^D_C; \Delta$ then $\Gamma \vdash e, (b_{\underline{x}} c[\underline{x}]) : E, D \dashv \Delta$

When a semantic operator is retrieved, nothing happens in the syntax.

These two schemata are our analog of Covert Movement in TG.

(17) **Cooper Storage, Natural-Deduction Style**


S
NP
$NP \multimap_S S$
*Ira*
$NP \multimap_C NP \multimap_S S$
NP
*caught*
$N \multimap_{SP} NP$
N
*a*
*chipmunk*
a'(chipmunk')$_{\underline{x}}$(catch'($\underline{x}$)(Ira'))
catch'($\underline{x}$)(Ira') ⊣ a'(chipmunk')$_x$
*Ira'*
catch'($\underline{x}$) ⊣ a'(chipmunk')$_x$
*catch'*
$\underline{x}$ ⊣ a'(chipmunk')$_x$
a'(chipmunk')
*a'*
*chipmunk'*

Terms of form $a_x b$ translate into typed lambda calculus as $a(\lambda_x.b)$.

(18) **Quantifier Scope Ambiguity**

a. Syntax (both readings):
($^{\mathrm{S}}$ Chris (thinks ($^{\mathrm{S}}$ Kim (likes everyone $^{\mathrm{C}}$) $^{\mathrm{C}}$))) : S

b. Semantics (scoped to lower clause):
((think' (everyone'$_{\underline{x}}$((like' $\underline{x}$) Kim'))) Chris')
TLC: think'$(\lambda_w(\forall_x($person'$(x)(w) \to$ like'$(x)($Kim'$)(w))))($Chris'$)$

c. Semantics (scoped to upper clause):
(everyone'$_{\underline{x}}$((think' ((like' $\underline{x}$) Kim')) Chris'))
TLC: $\lambda_w(\forall_x($person'$(x)(w) \to$ think'(like'$(x)($Kim'$))($Chris'$)(w)))$

**Note:** Meaning postulates and normalization are used to obtain the TLC translations of the CVG semantic terms.

(19) **Schema T (Trace)**

$t, x : A, B \vdash t, x : A, B \dashv$ ($t$ and $x$ fresh)

Traces are paired with semantic variables at birth.

Compare with the MP, where traces must undergo a multistage process of trace conversion in order to become semantically interpretable.

Logically, $t$ and $x$ are just variables, with no internal structure (the standard ND treatment of hypotheses in proofs).

(20) **Schema G (Gazdar Schema)**

If $\Gamma \vdash a, d : A^C_B, D^F_E \dashv \Delta$ and $t, x : B, D; \Gamma' \vdash b, e : B, E \dashv \Delta'$,
then $\Gamma; \Gamma' \vdash (a_t b), (d_x e) : C, F \dashv \Delta, \Delta'$
($t$ free in $b$, $x$ free in $e$)

This schema together with the Trace Schema are our analog of Covert Movement in TG.

'Overtly moved' signs are operators, both syntactically and semantically, and scope in parallel.

**Important**: The operator $a$ **binds** the trace $t$, but there is no construal of the words 'move' or 'copy' under which $a$ moved from the argument position $t$ occupies, or copied $t$.

(21) **Some *Wh*-Lexicon**

$$\vdash \textsf{whether}, \textsf{whether'} : \mathrm{S} \multimap_{\mathrm{M}} \mathrm{S}, \pi \to \kappa \dashv$$

$$\vdash \textsf{wondered}, \textsf{wonder'}_n : \mathrm{S} \multimap_{\mathrm{C}} \mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}, \kappa_n \to \iota \to \pi \dashv$$

$$\vdash \textsf{who}_{\textsf{filler}}, \textsf{who}^0 : \mathrm{NP}^{\mathrm{Q}}_{\mathrm{S}}, \iota^{\kappa_1}_{\pi} \dashv$$

$$\vdash \textsf{who}_{\textsf{in-situ}}, \textsf{who}^n : \mathrm{NP}, \iota^{\kappa_{n+1}}_{\kappa_n} \dashv \text{ (for } n > 0)$$

$$\vdash \textsf{what}_{\textsf{filler}}, \textsf{what}^0 : \mathrm{NP}^{\mathrm{Q}}_{\mathrm{S}}, \iota^{\kappa_1}_{\pi} \dashv$$

$$\vdash \textsf{what}_{\textsf{in-situ}}, \textsf{what}^n : \mathrm{NP}, \iota^{\kappa_{n+1}}_{\kappa_n} \dashv \text{ (for } n > 0)$$

(22) **Consequences of the Preceding Lexical Entries**

- There can be no purely in-situ interrogatives (leaving aside pragmatically restricted, intonationally marked ones which we cannot go into here):

  **I wonder Fido bit who?*

- A *wh*-expression cannot scope, either overtly or covertly, over a polar interrogative:

  **I wonder whether Fido bit who?*

  **I wonder who whether Fido bit?*

- In each constituent interrogative, only one 'overtly moved' *wh*-expression can take scope there:

  **I wonder who who(m) bit?*

(23) **More Consequences**

- Arbitrarily many in-situ *wh*-expressions can take their semantic scope at a given consituent interrogative:

  *Who gave what to who when*?

- There are (Baker) ambiguities that hinge on how high an in-situ *wh*-expression scopes:

  *Who wondered who bit who*?

- Even though subject *wh*-expressions might *look* in situ:

  *Who barked*?

  they aren't really; if they were, they could also scope higher to form imposssible embedded questions as in:

  **Kim wondered Chris thought who barked*?

  (Intended meaning: Kim wondered who Chris thought barked.)

(24) **Wh-In Situ Languages**

In languages without overt *wh*-movement, the counterpart of *who* is just an NP with **all** the meanings $\mathsf{who}^n$ $(n \geq 0)$, **including** $\mathsf{who}^0$.

That is: the difference between overt and covert *wh*-movement languages is in the lexicon.

(25) **An Embedded Polar Question**

a. Syntax: $\vdash$ (whether ($^{\mathrm{S}}$ Kim (likes Sandy $^{\mathrm{C}}$)) $^{\mathrm{C}}$) : S

b. Semantics: $\vdash$ (whether' (like' Sandy' Kim')) : $\kappa_0$

(26) **An Embedded Constituent Question**

a. Syntax: $\vdash$ [$\textsf{what}_{\textsf{filler}}$ ${}_t$(${}^{\mathrm{S}}$ $\textsf{Kim}$ ($\textsf{likes}$ $t$ ${}^{\mathrm{C}}$))] : S

b. Semantics: $\vdash$ ($\textsf{what}^0_y$(($\textsf{like'}$ $y$) ($\textsf{Kim'}$)) : $\kappa_1$

(27) **A Binary Constituent Question**

a. Syntax: $\vdash$ [$\textsf{who}_{\textsf{filler}}$ ${}_t$(${}^{\mathrm{S}}$ $t$ ($\textsf{likes}$ $\textsf{what}_{\textsf{in-situ}}$ ${}^{\mathrm{C}}$))] : S

b. Semantics: $\vdash$ ($\textsf{what}^1_{\underline{y}}$($\textsf{who}^0_x$(($\textsf{like'}$ $\underline{y}$) ($x$))) : $\kappa_2$

(28) **Baker Ambiguity**

a. $\vdash$ $[\textsf{who}_{\textsf{filler}}\ {}_{t}({}^{\mathrm{S}}\ t\ (\textsf{wonders}\ [\textsf{who}_{\textsf{filler}}\ {}_{t'}({}^{\mathrm{S}}\ t'\ (\textsf{likes}\ \textsf{what}_{\textsf{in-situ}}\ {}^{\mathrm{C}}))]\ {}^{\mathrm{C}}))] : \mathrm{S}$

b. $\vdash$ $(\textsf{who}^{0}_{x}((\textsf{wonder'}_{2}\ (\textsf{what}^{1}_{\underline{y}}(\textsf{who}^{0}_{z}((\textsf{like'}\ \underline{y})\ z))))\ x)) : \pi$

(E.g. Chris wonders who likes what.)

c. $\vdash$ $(\textsf{what}^{1}_{\underline{y}}(\textsf{who}^{0}_{x}((\textsf{wonder'}_{1}\ (\textsf{who}^{0}_{z}((\textsf{like'}\ \underline{y})\ z)))\ x)))) : \pi$

(E.g. Chris wonders who likes the books, and Kim wonders who likes the records.)

(29) **Raising of Two Quantifiers to Same Clause**

a. Syntax (both readings): $(^{\mathrm{S}}$ everyone (likes someone $^{\mathrm{C}}$)) : S

b. ∀∃-reading: (everyone'$_{\underline{x}}$(someone'$_{\underline{y}}$((like' $\underline{y}$) $\underline{x}$)))

c. ∃∀-reading: (someone'$_{\underline{y}}$(everyone'$_{\underline{x}}$((like' $\underline{y}$) $\underline{x}$)))

(30) **Abbreviated Notation for Functional Types**

Where $\sigma$ ranges over strings of types and $\epsilon$ is the null string:

i. $A_\epsilon =_{\text{def}} A$

ii. $A_{B\sigma} =_{\text{def}} B \rightarrow A_\sigma$ (e.g. $\text{t}_{\text{ee}} = \text{e} \rightarrow \text{e} \rightarrow \text{t}$)

iii. For $n \in \omega$, $A_n =_{\text{def}} A_\sigma$ where $\sigma$ is the string of e's of length $n$

**Example:** $\text{t}_2 =_{\text{def}} \text{t}_{\text{ee}} =_{\text{def}} \text{e} \rightarrow \text{e} \rightarrow \text{t}$.

(31) **A Refinement**

- Actually the QNP meanings have to be polymorphically typed to $\mathrm{e}^{\mathrm{t}_\sigma}_{\mathrm{t}_\sigma}$ where $\sigma$ ranges over strings of types, since quantifiers can retrieved not just at proposition nodes, but also at nodes with functional types whose final result type is proposition.
- An important case is $\sigma = \mathrm{e}$: quantifiers can be retrieved at nodes which are semantically individual properties ($\mathrm{t_e} = \mathrm{e} \rightarrow \mathrm{t}$), such as VPs and Ns:

  a. [Campaigning in every state] is prohibitively expensive.

  b. Every [owner of a donkey] walks.

(32) **An NP-Internal Scope Example**

a. The NP-internal-scope reading of the previous example

(b) Every owner of a donkey walks.

is analyzed unproblematically by retrieving a'(donkey') at the $\bar{\mathrm{N}}$ node and the every-quantifier at the S-node.

b. The resulting semantic term is

$\mathsf{every'}(\mathsf{a'}(\mathsf{donkey'})^{e}_{y}.\mathsf{own'}(y)(x))_{x}.\mathsf{walk'}$

c. This normalizes to the TLC term

$\forall_{x}.(\exists_{y}.\mathsf{donkey'}(y) \wedge \mathsf{own'}(y)(x)) \rightarrow \mathsf{walk'}(x)$

(33) **Scoping Out of** NP

a. The scoping-out-of-NP reading of

   (b) Some owner of every donkey walks.

   is analyzed unproblematically by scoping $\mathsf{some'}(\mathsf{own'}(y))_x$ over $\mathsf{walk'}(x)$ at the S node and then scoping $\mathsf{every'}(\mathsf{donkey'})_y$ over it.

b. The resulting semantic term is

   $\mathsf{every'}(\mathsf{donkey'})_y.\mathsf{some'}(\mathsf{own'}(y))_x.\mathsf{walk'}(x)$

c. This normalizes to the TLC term

   $\forall_y.\mathsf{donkey'}(y) \rightarrow \exists_x.\mathsf{own'}(y)(x) \wedge \mathsf{walk'}(x)$

(34) **A Scope "Reconstruction" Example (1/3)**

a. I wonder [how many cats]$_t$ John thought Mary saw $t$.

b. The interrogative part of the meaning of *how many* must scope at the intermediate clause (complement of *wonder*), but the cardinality part can scope in the lowest clause (complement of *thought*).

c. This is problematic for a model where QR follows Spellout, since we *hear* the cardinality word *many* in the intermediate clause.

d. The rules we already have analyze such examples unproblematically as long as we assign the right meaning to *how many*.

e. All we have to do is (a) posit a trace whose semantic variable has the type of a generalized quantifier, and (b) Cooper-store the trace's semantic term (that same quantifier variable).

(35) **A Scope Reconstruction Example (2/3)**

a. For specificity, we analyze cardinality determiners (e.g. *five*) semantically as $\mathsf{cd}(5)$ where $\mathsf{cd}$ is subject to the meaning postulate

   $\vdash \mathsf{cd} = \lambda_n.\lambda_P.\lambda_Q.\mathsf{card}(\lambda_y.P(y) \wedge Q(y)) \geq n$

b. The constant $\mathsf{howmany'}$ is subject to the meaning postulate

   $\vdash \mathsf{howmany'} = \lambda_P.\lambda_Z.\mathsf{which'}(\mathsf{number'})(\lambda_n.Z(\mathsf{cd}(n)(P)))$

   where $Z$ is a variable of type $\mathrm{e}^{\mathrm{t}}_{\mathrm{t}} \rightarrow \mathrm{t}$.

b. We let the semantic variable of the trace that *how many cats* will bind have the type of a generalized quantifier:

   $t, \mathsf{Q} : \mathrm{NP}, e^{\mathrm{t}}_{\mathrm{t}} \vdash t, \mathsf{Q} \dashv$

c. We immediately Cooper-store the trace's semantic term:

   $t, \mathsf{Q} : \mathrm{NP}, e^{\mathrm{t}}_{\mathrm{t}} \vdash t, x : \mathrm{NP}, \mathrm{e} \dashv \mathsf{Q}_x : e^{\mathrm{t}}_{\mathrm{t}}$

   Now $\mathsf{Q}$ is in both the context and the co-context simultaneously.

(36) **A Scope Reconstruction Example (3/3)**

a. Remember the example we are analyzing is

   I wonder [how many cats]$_t$ John thought Mary saw $t$.

b. After we retrieve $\mathsf{Q}$ (the semantic variable of the trace) from the co-context at the lowest clause, it is still in the context:

   $t, \mathsf{Q} \vdash (^{\mathrm{S}}\ \mathsf{Mary}\ (\mathsf{saw}\ t^{\mathrm{C}})), \mathsf{Q}_x.\mathsf{saw'}(x)(\mathsf{Mary'})$

c. At the *John thought Mary saw t* node, the semantic term is

   $\mathsf{think'}(\mathsf{Q}_x.\mathsf{see'}(x)(\mathsf{Mary'}))(\mathsf{John'})$

   and $\mathsf{Q}$ is still in the context.

d. Finally we use the Gazdar ('Overt Movement') schema to bind $\mathsf{Q}$ with the semantic term of *how many cats*, which yields:

   $\mathsf{which'}(\mathsf{number'})_x.\mathsf{think'}(\mathsf{cd}(n)(\mathsf{cat'})(\lambda_x.\mathsf{saw'}(x)(\mathsf{Mary'})))(\mathsf{John'})$

(37) **Parasitic Scope**

- Barker (2008) introduces this term to describe quantifiers such as *the same* and *different* whose ‘scope target does not exist until [another quantifier] takes its scope’.
- Other instances of this phenomenon include **superlatives** and elliptical constructions such as **phrasal comparatives**.
- Barker’s analysis uses **continuations** and **choice functions**.
- We propose an account based on a notion of **focus exploitation**.

(38) **Semantic Operizers**

- Recall that a semantic **operator** is a term whose type is of the form $A_B^C$.
- We define an **operizer** to be a functional term whose result type is an operator type.
- An operator can be thought of as a 0-ary operizer.
- Intuitively, an operizer is a 'movement trigger': it converts its argument into something that 'has to move' to take scope.

(39) **Some Signs with Operizer Semantics**

- ordinary determiners: type $(\mathrm{e} \rightarrow \mathrm{t}) \rightarrow \mathrm{e}^{\mathrm{t}}_{\mathrm{t}}$
- 'overtly moved' interrogative determiner *which*: type $(\mathrm{e} \rightarrow \mathrm{t}) \rightarrow \mathrm{e}^{\mathrm{e}\rightarrow\pi\rightarrow\mathrm{t}}_{\pi}$ (where $\pi =_{\mathrm{def}} \mathsf{s} \rightarrow \mathrm{t}$).
- (non-phrasal) comparative -*er*, assuming the *than*-phrase complement denotes a set of degrees: type $(\mathrm{d} \rightarrow \mathrm{t}) \rightarrow \mathrm{d}^{\mathrm{t}}_{\mathrm{t}}$.
- Following (in spirit) Moortgat 1991, we can analyze **pragmatic focus** as an intonationally realized phrasal affix whose semantics has the (polymorphic) operizer type $B \rightarrow B^{\mathrm{t}}_{\mathrm{t}}$.

(40) **Semantic Focus as an Operizer 'Wild Card'**

- We suggest treating **semantic focus** as an operizer 'wild card' whose instantiation depends on what other sign is **exploiting** it.
- Best-known is the case of 'particles' (*only*, *even*, *too*) discussed under the rubric of 'association with focus', where the **focus instantiator** (FI) is just the semantics of the particle itself.
- Here we consider more complex cases of **parasitic scope**, where the focus exploiter (FE) 'contributes' **two** operizers: one its own semantics and the other the FI; the focused phrase is called the **asscociate**.
- In still more complex—**elliptical**—cases to be treated elsewhere, the FI takes **two** arguments: the associate and the FE's (extraposed) complement, called the **remnant**.

(41) **A New Grammatical Function for Phrasal Affixation**

- We add to the inventory of gramfuns the name AFFIX (abbr. A), mnemonic for '(phrasal) affixation'.
- Correspondingly, we add a new 'flavor' of Modus Ponens to the syntactic (and interface) schemata ($\multimap_{\text{A}}$-Elimination).
- This is used to analyze intonationally realized phrasal affixes, Japanese and Korean case markers, Chinese sentence particles, English possessive -*'s*, etc.
- Lexical entry for English semantic focus:

  $\vdash \mathsf{foc}, \mathsf{foc'} : A \multimap_{\text{A}} A, B \rightarrow B^{\text{t}}_{\text{t}}$

(42) **Kim thinks Sandy makes the most**

a. First reading: Sandy makes the most, Kim thinks.

b. Second reading: The amount Kim thinks Sandy makes exceeds the amount Kim thinks anyone else makes.

c. Third reading: The amount Kim thinks Sandy makes exceeds the amount anyone else thinks Sandy makes.

(43) **Intuitive Explanation**

- The FE *the most* and the FI have adjacent scope ('parasitic scope' or 'tucking in').
- If **Kim** is focused, then they have to scope at the root clause (because operators can raise but not lower).
- If **Sandy** is focused, then there is ambiguity as to whether it scopes in the root clause or the complement clause.

(44) **Toward an Analysis of Superlatives**

a. **Fido** cost <u>the most</u>.

b. We take this to mean that Fido is the unique maximizer of the function that maps (relevant) entities to their prices.

c. We assume something's price is the maximum amount that it costs.

d. So our target semantics for this sentence is
$\mathsf{um}(\mathsf{Fido'})_{\underline{x}}.\mathsf{max}_{\underline{d}}.\mathsf{cost'}(\underline{d})(\underline{x})$
where the operizer $\mathsf{um}$ is subject to the meaning postulate

e. $\vdash \mathsf{um} = \lambda_x.\lambda_f.\forall_y((y \neq x) \rightarrow (f(x) > f(y))) : \mathrm{e} \rightarrow \mathrm{e}^{\mathrm{t}}_{\mathrm{d}}$

f. After normalization, (d) translates to:
$\forall_y((y \neq \mathsf{Fido'}) \rightarrow [\mathsf{max}(\lambda_d.\mathsf{cost'}(d)(\mathsf{Fido'})) > \mathsf{max}(\lambda_d.\mathsf{cost'}(d)(\mathsf{x}))])$

g. This is the semantics our theory will predict, as long as the semantics of *the most* is $\mathsf{max}$ and focus is instantiated as $\mathsf{um}$.

g. But how does focus get instantiated?

(45) **Instantiating Focus**

a. Lexical entries:

$\vdash \textsf{cost}, \textsf{cost'} : \mathrm{Deg} \multimap_{\mathrm{C}} \mathrm{NP} \multimap_{\mathrm{S}} \mathrm{S}$

$\vdash \textsf{the\_most}, \textsf{IF}(\textsf{um}) \cdot \textsf{max} : \mathrm{Deg}, \mathrm{d}^{\mathrm{d}}_{\mathrm{t}} \dashv$

The semantics here means: '$\textsf{max}$ directly outscoped by the result of instantiating focus as $\textsf{um}$'.

b. Focus Instantiation Semantic Schema (FI)

If $\Gamma \vdash a \dashv \textsf{foc'}(b)_x; \textsf{IF}(c) \cdot d_y; \Delta$,

then $\Gamma \vdash a \dashv c(b)_x \cdot d_y; \Delta$

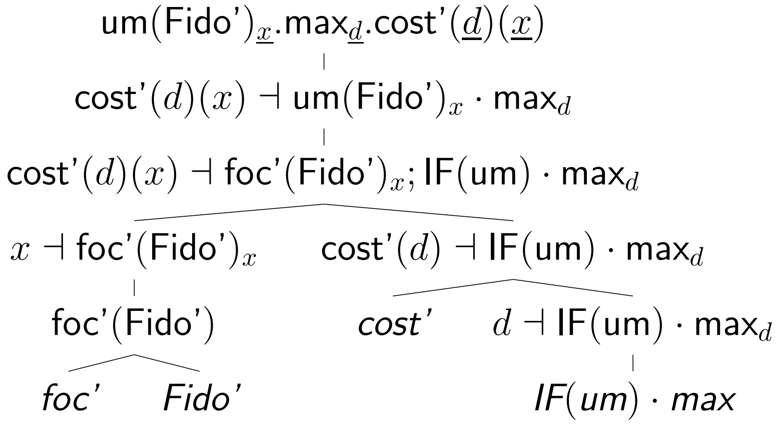
Note that in the corresponding interface schema, nothing happens in the syntax.

(46) **Analysis of a Superlative Sentence**

a. Syntax:

$(^{\text{S}}$ (foc Fido $^{\text{A}}$) (cost the_most $^{\text{C}}$))

b. Semantics:



$\mathsf{um(Fido')}_{\underline{x}}.\mathsf{max}_{\underline{d}}.\mathsf{cost'}(\underline{d})(\underline{x})$
$\mathsf{cost'}(d)(x) \dashv \mathsf{um(Fido')}_x \cdot \mathsf{max}_d$
$\mathsf{cost'}(d)(x) \dashv \mathsf{foc'(Fido')}_x; \mathsf{IF(um)} \cdot \mathsf{max}_d$
$x \dashv \mathsf{foc'(Fido')}_x$
$\mathsf{cost'}(d) \dashv \mathsf{IF(um)} \cdot \mathsf{max}_d$
$\mathsf{foc'(Fido')}$
*cost'*
$d \dashv \mathsf{IF(um)} \cdot \mathsf{max}_d$
*foc'*
*Fido'*
*IF*(*um*) · *max*

c. Normalized TLC translation:

$\forall_y((y \neq \mathsf{Fido'}) \rightarrow [\mathsf{max}(\lambda_d.\mathsf{cost'}(d)(\mathsf{Fido'})) > \mathsf{max}(\lambda_d.\mathsf{cost'}(d)(\mathsf{x}))])$

(47) ***The Same***

a. Plural-focus *the same*:

**Fido and Felix** got $\underline{\text{the same present}}$.

$\exists_y(\mathsf{present'}(y) \land \forall_x[(x <_{\mathsf{a}} \mathsf{Fido'} + \mathsf{Felix'}) \rightarrow \mathsf{get'}(y)(x)])$

Here + denotes Link join (plural formation), and $<_{\mathsf{a}}$ denotes the part-of relation between an atom and a plural.

b. Elliptical (associate-remnant) *the same*:

**Fido** got $\underline{\text{the same present}}$ *as* **Felix**.

$\exists_y(\mathsf{present'}(y) \land \mathsf{get'}(y)(\mathsf{Fido'}) \land \mathsf{get'}(y)(\mathsf{Felix'}))$

c. These sentences have equivalent truth conditions.

d. Here we only analyze plural-focus *the same*.

e. Elliptical *the same* and other associate-remnant constructions are analyzed in work in progress.

(48) **Analysis of Plural-Focus *The Same***

a. We cannot escape from positing a special coordination rule with semantics corresponding to Link join (plural formation).

b. We also need a new basic semantic type $\mathrm{e}'$ for plural entities.

c. Syntactically, plural-focus *the same* is just a determiner.

d. But semantically, it is an FE operizer:

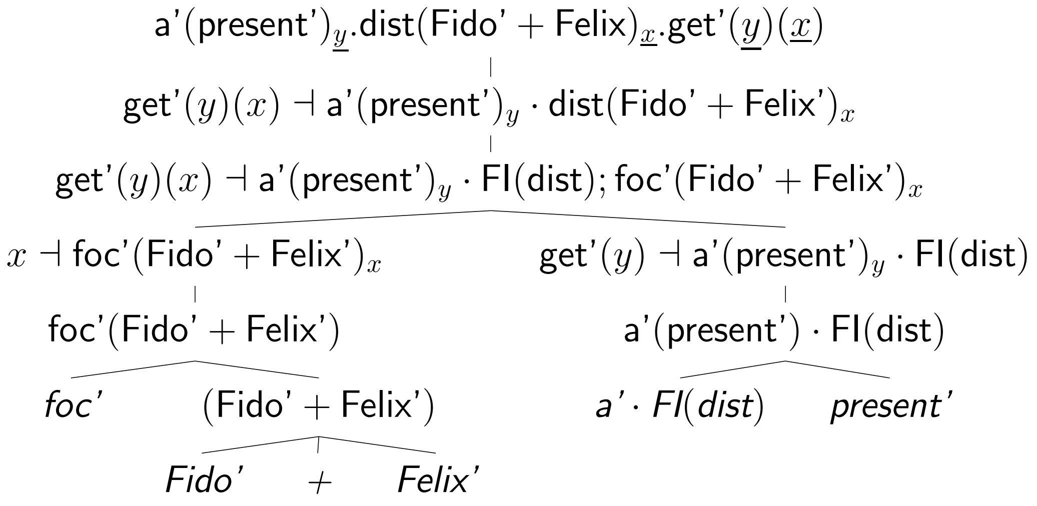
1. Its own semantics is the existential generalized determiner $\mathsf{a'}$.
2. The FI is the distributive operizer $\mathsf{dist}$ that converts a plural to a universal quantifier, characterized by the meaning postulate $\vdash \mathsf{dist} = \lambda_{x'}.\lambda_P.\forall_x((x <_{\mathsf{a}} x') \rightarrow P(x)) : \mathrm{e}' \rightarrow \mathrm{e}^{\mathrm{t}}_{\mathrm{t}}$
3. Unlike *the most*, in this case the FE outscopes the FI.

e. So the lexical entry for *the same* is:
$\vdash \mathsf{the\_same}, \mathsf{a'} \cdot \mathsf{FI}(\mathsf{dist}) : \mathrm{N} \multimap_{\mathrm{SP}} \mathrm{NP}, \mathrm{et} \rightarrow \mathrm{e}^{\mathrm{t}}_{\mathrm{t}}$

(49) **Analysis of a Plural-Focus *The Same* Sentence**

a. Syntax: ($^{\text{S}}$ (foc (Fido and Felix) $^{\text{A}}$) (got (the_same present $^{\text{SP}}$) $^{\text{C}}$))

b. Semantics:


$\mathsf{a'(present')}_{\underline{y}}.\mathsf{dist(Fido' + Felix)}_{\underline{x}}.\mathsf{get'}(\underline{y})(\underline{x})$
$\mathsf{get'}(y)(x) \dashv \mathsf{a'(present')}_y \cdot \mathsf{dist(Fido' + Felix')}_x$
$\mathsf{get'}(y)(x) \dashv \mathsf{a'(present')}_y \cdot \mathsf{FI(dist); foc'(Fido' + Felix')}_x$
$x \dashv \mathsf{foc'(Fido' + Felix')}_x$
$\mathsf{get'}(y) \dashv \mathsf{a'(present')}_y \cdot \mathsf{FI(dist)}$
$\mathsf{foc'(Fido' + Felix')}$
$\mathsf{a'(present')} \cdot \mathsf{FI(dist)}$
*foc'*
(Fido' + Felix')
*a'* · *FI*(*dist*)
*present'*
*Fido'*
+
*Felix'*

c. Normalized TLC translation:

$\exists_y(\mathsf{present'}(y) \wedge \forall_x[(x <_{\mathsf{a}} \mathsf{Fido' + Felix'}) \rightarrow \mathsf{get'}(y)(x)])$

(50) **The EMG Story Retold**

- Syntactic and semantic derivations are **parallel**, not cascaded.
- Derivations are **proofs**, not sequences of tree operations.
- **All** signs have a semantics ('it's phases all the way down').
- Traces are **ordinary logical variables**, not copies of their binders.
- **There is no 'Trace Conversion'**: traces are paired with semantic variables from birth.
- Merge is **Modus Ponens**.
- 'Overt Move' works as **Gazdar** said.
- 'Covert Move' works as **Cooper** said.
- Rules can intermingle because that's always the case in proofs.
- Interpretation of the semantic proof is **simple** and **explicit**.
- **There is no 'LF'** between syntax and semantics.