A derivational approach to phrasal spellout

Gregory M. Kobele

kobele@uchicago.edu
Computation Institute & Department of Linguistics
University of Chicago

BCGL 7, 2012
Brussels

What this talk is about

Hypothesis: "Interface Uniformity"

The interfaces to syntax have the same structure.

The differences between the syntax-semantics and the syntax-morphology interfaces lie only in the objects they are building.

Research Strategy

1. Study the structure of the syntax-semantics interface
2. Reflect these properties onto the syntax-morphology interface

Here we will look at idioms.

In a nutshell

Main Claim:

Morphology is insensitive to derived structure

Morphology only cares about DS.

Outline

Basic Assumptions

Direct Compositionality

The syntactic derivation is a recipe for constructing a semantic representation.

> *Semantic differences must stem from differences in the way expressions are built, not from differences in the surface (or LF) structure.*

Historical Antecedents

Generative Semantics

Deep structure is the structure interpreted.

Categorial Grammar

> Surface structure [is] no more than the trace of the algorithm that delivers the [...] interpretation (Steedman, 2000)

Direct Compositionality (in Minimalism)

Direct Compositionality

- Each lexical item has a semantic denotation
- Each operation (**merge**, **move**) is semantically interpretable;
  - $[\![\textbf{merge}]\!]$ combines two denotations into a single one
  - $[\![\textbf{move}]\!]$ maps a single denotation to another one

Denotations

- Expressions are associated with (bounded size) quantifier stores
- (Bounded size) stores faithfully implement Heim & Kratzer-style LF-interpretation (Kobele; 2006,2010)
- Can be encoded into the simply typed lambda calculus; (Kobele; 2012)

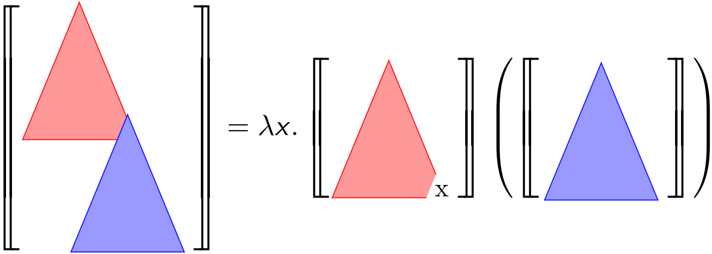
Direct Compositionality and Derivation Trees

Derivations are the structures interpreted

- We need a precise notion of what a derivation is

Complete Decomposability

If a derivation $d = M[N]$, then $[\![d]\!] = \lambda x. [\![M[x]]\!] ([\![N]\!])$



Representing derivations

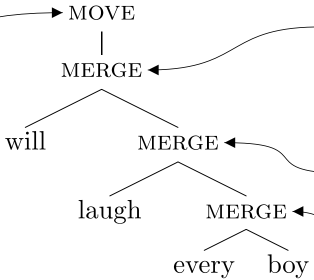
A derivation shows how a sentence is built up from the primitives of the grammar.

1. $[_{DP}$ every $[_{NP}$ boy $]]$ — *merge*(*every*, *boy*)
2. $[_{VP}$ laugh $[_{DP}$ every boy $]]$ — *merge*(*laugh*, #1)
3. $[_{IP}$ will $[_{VP}$ laugh $[_{DP}$ every boy $]]]$ — *merge*(*will*, #2)
4. $[_{IP}[_{DP}$ every boy $][_{I'}$ will $[_{VP}$ laugh $t]]]$ — *move*(#3)

Processes have structure

derivations like the above can be viewed as trees

Representing derivations



merge will and the VP just built

merge laugh and the DP every boy

merge every and boy

move the closest available thing (the DP every boy) to check the features of the current head (will)

The Syntax-Semantics Interface (I)

We typically see:

$[\![\sigma(t_1, \ldots, t_n)]\!] = \boldsymbol{\sigma}([\![t_1]\!], \ldots, [\![t_n]\!])$

This conflates two kinds of information:

1. *what the meanings of the formatives are*
2. *that there is an isomorphism between the syntactic structure and the semantic structure*

Reformulated:

1. replace each formative ϕ with its meaning $[\![\phi]\!]$
2. interpret immediate dominance as (uncurried) function application: $f(t_1, \ldots, t_n) \rightsquigarrow f(t_1) \cdots (t_n)$

Syntax-Semantics Interface (II)

The way to obtain a meaning from a structure is universal:

$$\langle \sigma(t_1, \ldots, t_n) \rangle_f = f_\sigma(\langle t_1 \rangle_f) \cdots (\langle t_n \rangle_f)$$

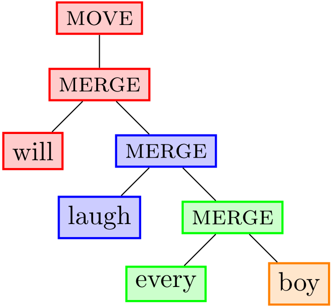
The language particular content of the interface:

a finite list of denotations for all formatives

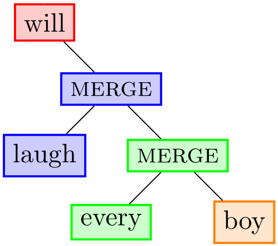
- **merge** and **move**
- all lexical items

But are $[\![$**merge**$]\!]$ *and* $[\![$**move**$]\!]$ *really language particular?*
(No.)

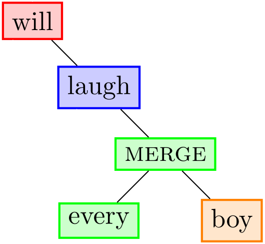
Alternative Representations (‘Mirror Theory’)



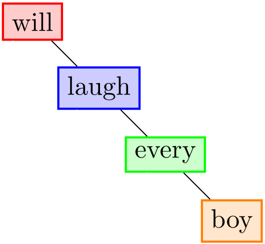
Alternative Representations ('Mirror Theory')



Alternative Representations ('Mirror Theory')



Alternative Representations ('Mirror Theory')



Syntax-Semantics Interface (III)

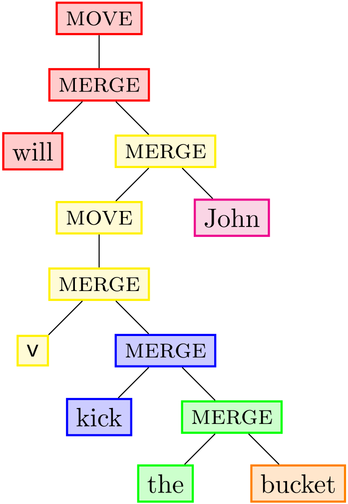
The language particular content of the interface:

a finite list Λ of denotations for

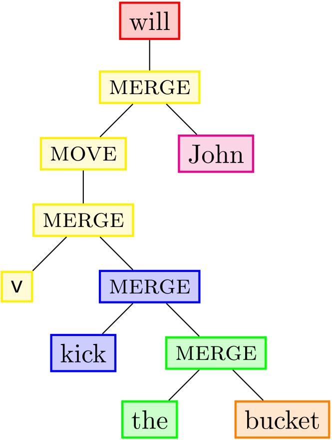
- all lexical items

A derivation is interpreted at the interface by replacing all of its formatives (lexical items) with their associated denotations.

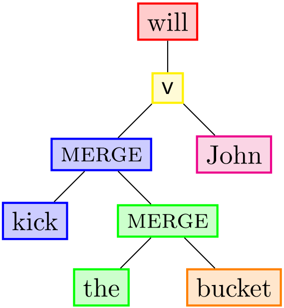
Interpreting chunks



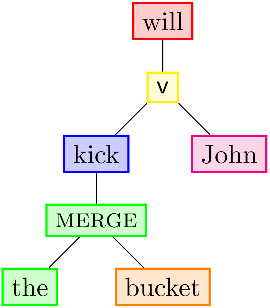
Interpreting chunks



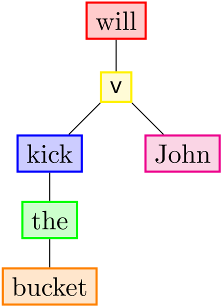
Interpreting chunks



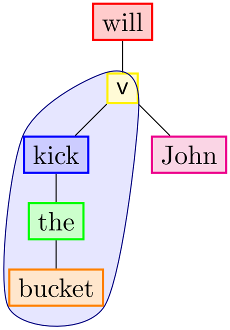
Interpreting chunks



Interpreting chunks



Interpreting chunks



Syntax-Semantics Interface (IV)

The language particular content of the interface:

a finite list Λ of denotations for

- all derivation chunks

A derivation is interpreted at the interface by replacing all of its chunks with their associated denotations.

Syntax-Semantics Interface (IV)

The language particular content of the interface:

a finite list Λ of denotations for

- all derivation chunks

A derivation is interpreted at the interface by replacing all of its chunks with their associated denotations.

This is a contradiction!

Syntax-Semantics Interface (IV)

The language particular content of the interface:

a finite list Λ of denotations for

- **some** derivation chunks

A derivation is interpreted at the interface by replacing all of its chunks with their associated denotations.

Syntax-Semantics Interface (IV)

The language particular content of the interface:

a finite list Λ of denotations for

- **some** derivation chunks

A derivation is interpreted at the interface by replacing all of its chunks with their associated denotations.

Nondeterministic!

Historical Antecedents

Fraser: Idioms are identical to non-idioms at DS

Koopman & Sportiche: "If X is the minimal constituent containing all the idiomatic material, the head of X is part of the idiom."

Jackendoff: Idioms are triples $\langle Phon, Syn, Sem \rangle$ of structured entities.

O'Grady: "An idiom's component parts must form a chain."

The main difference:

We are looking at the *derivation*

Historical Antecedents

Fraser: Idioms are identical to non-idioms at DS

Koopman & Sportiche: "If X is the minimal constituent containing all the idiomatic material, the head of X is part of the idiom."

Jackendoff: Idioms are triples $\langle Phon, Syn, Sem\rangle$ of structured entities.

O'Grady: "An idiom's component parts must form a chain."

A sequence of heads such that each is the head of a selected argument of another

The main difference:

We are looking at the *derivation*

Last Remarks

No derived idioms

Example

Raising verb and its derived subject cannot form an idiom.

Derivational patterns

Outline

Interface Uniformity

Interpretation

Interfaces are lists of associations between derivation pieces and things

Syntax-Semantics

- things are lambda terms over semantic domain
- 'chunks' of derivations correspond to *idioms/constructions*

Syntax-Morphology

- things are ???
- what do 'chunks' correspond to?

Interface Uniformity

Interpretation

Interfaces are lists of associations between derivation pieces and things

Syntax-Semantics

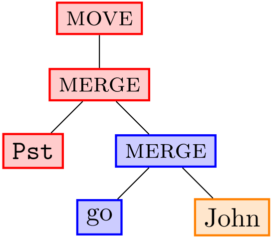
- things are lambda terms over semantic domain
- 'chunks' of derivations correspond to *idioms/constructions*

Syntax-Morphology

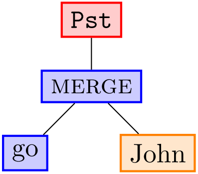
- things are ???
- what do 'chunks' correspond to?

A suppletive form is a chunk

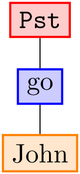
Suppletion



Suppletion



Suppletion

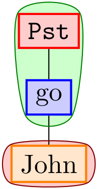


Interface

Pst
|
go $\rightsquigarrow$ "went"

John $\rightsquigarrow$ "John"

Suppletion



Interface

$$\begin{array}{c}\texttt{Pst}\\ | \\ \text{go}\end{array} \rightsquigarrow \text{“went”}$$

$$\text{John} \rightsquigarrow \text{“John”}$$

Suppletion

$$\begin{array}{c}\text{“went”}\\ |\\ \text{“John”}\end{array}$$

Interface

$$\begin{array}{c}\texttt{Pst}\\ |\\ \text{go}\end{array} \rightsquigarrow \text{“went”}$$

$$\text{John} \rightsquigarrow \text{“John”}$$

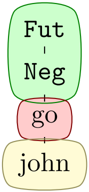
Suppletion

```
Fut
 |
Neg
 |
go
 |
john
```

Interface

Pst – go	$\rightsquigarrow$	"went"
John	$\rightsquigarrow$	"John"
Fut	$\rightsquigarrow$	"will"
Neg	$\rightsquigarrow$	"not"
Fut – Neg	$\rightsquigarrow$	"won't"
go	$\rightsquigarrow$	"go"

Suppletion



"won't"
"go"
"John"

Interface

Pst / go	⇝	"went"
John	⇝	"John"
Fut	⇝	"will"
Neg	⇝	"not"
Fut / Neg	⇝	"won't"
go	⇝	"go"

Suppletion

Fut
Neg
go
john

"won't"
"go"
"John"

"will"
"not"
"go"
"John"

Interface

Pst / go	$\rightsquigarrow$	"went"
John	$\rightsquigarrow$	"John"
Fut	$\rightsquigarrow$	"will"
Neg	$\rightsquigarrow$	"not"
Fut / Neg	$\rightsquigarrow$	"won't"
go	$\rightsquigarrow$	"go"

Prediction

If 'morphological idiom' = suppletion, then

Clitics cannot trigger suppletion

(Unless we hack the features in an otherwise unmotivated way)

???

'Lowering' vs 'Local Dislocation'

Embick & Noyer

- Jane is even prettier than Kim.
- Jane is even more naturally pretty than Kim.
- *Jane is even naturally prettier than Kim.

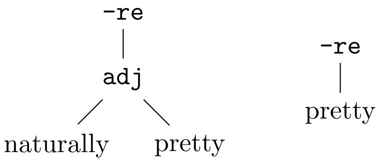
Evidence for sensitivity to order/adjacency (?)

‘Lowering’ vs ‘Local Dislocation’

Embick & Noyer

- Jane is even prettier than Kim.
- Jane is even more naturally pretty than Kim.
- *Jane is even naturally prettier than Kim.

Evidence for sensitivity to order/adjacency (?)



Differing Predictions

English-prime: English with mixed [A Adv] & [Adv A] order

- Jane is even prettier than Kim.
- Jane is even more naturally pretty than Kim.

1. Jane is even prettier naturally than Kim.
2. Jane is even more pretty naturally than Kim.

Embick & Noyer

1 is good

GK

1 is bad

Interpretation of morphological objects

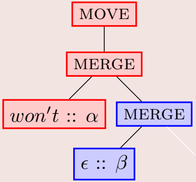
How is the following interpreted?

`Fut`
`Neg` $\rightsquigarrow$ "won't"

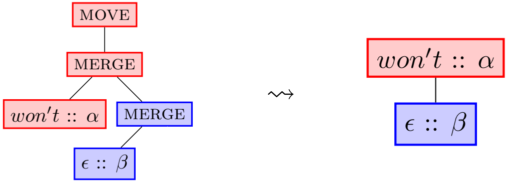
i.e. what does "won't" mean?

A simple answer:

It means the same thing that the following does, where α and β are the feature bundles of `Fut` and `Neg`, respectively.



Interpretation (II)



This looks familiar!

- Chomsky's *Strict Lexicalism*
- Brody's *Mirror Theoretic Spellout*

A MW is pronounced in its highest strong position

Formal Simulability Relations:

Chomsky $<$ Brody $<$ GK

Where's Morphology?

Answer: Inside the Interface

$$\begin{array}{c}\texttt{Fut}\\ |\\ \texttt{Neg}\end{array} \rightsquigarrow \begin{array}{c} won't :: \alpha \\ | \\ \epsilon :: \beta \end{array}$$

Regularities in '$\rightsquigarrow$':

- Paradigms
- DM operations
- Templates

Outline

Conclusion

Interface Uniformity

Interfaces interpret syntactic structure in the same way

Derivationalism

Derived structure is interface irrelevant

- Relations between theories (Mirror Theory, Strict Lexicalism)
- Location of morphology (at interface - spelling out '$\rightsquigarrow$')

Questions

- Is the interface sensitive to derived structure/order?
- Are the differences between morphological and semantic interfaces best viewed in terms of which chunks are 'idioms'?
- Can we formulate a learning theory which operates by breaking big interface chunks into smaller ones?

Outline

Inherited Features

He$_{3s,nom}$ is$_{pres,3s}$ happy.

$$\begin{array}{c}\texttt{Pres}\\ |\\ \text{be}\\ |\\ \text{happy}\\ |\\ \texttt{pro}_{3s}\end{array}$$

The fundamental claim:

Only deep configurations matter for morphology

Analytical Possibilities

Inherited Features

He$_{3s,nom}$ is$_{pres,3s}$ happy.

$\texttt{Pres}_{3s}$

be

happy

$\texttt{pro}_{3s,nom}$

The fundamental claim:

Only deep configurations matter for morphology

Analytical Possibilities

Checking: There are no inherited features.

Inherited Features

He$_{3s,nom}$ is$_{pres,3s}$ happy.

Pres
|
be
|
happy
|
pro$_{3s}$

The fundamental claim:

Only deep configurations matter for morphology

Analytical Possibilities

Checking: There are no inherited features.

Valuation: Interface objects are functions with inherited features as arguments

Inherited Features and Valuation

A closer look at `Pres`

feature matrix: `=v +k s`

Inherited Features and Valuation

A closer look at Pres

feature matrix: =v +k s

A closer look at (this) +k

- assigns *nominative* case
- inherits *person* and *number* features

$$\rightsquigarrow \quad \texttt{+k}\begin{bmatrix} \text{CASE} & : \texttt{nom} \\ \text{PERSON} & : \alpha \\ \text{NUMBER} & : \beta \end{bmatrix}$$

Inherited Features and Valuation

A closer look at `Pres`

feature matrix: `=v +k s`

A closer look at (this) `+k`

- assigns *nominative* case
- inherits *person* and *number* features

$$\rightsquigarrow \quad \texttt{+k}\begin{bmatrix} \text{CASE} & : \texttt{nom} \\ \text{PERSON} & : \alpha \\ \text{NUMBER} & : \beta \end{bmatrix}$$

$$\begin{array}{c}\texttt{Pres} \\ | \\ \text{be}\end{array} \rightsquigarrow \lambda\alpha, \beta, x : \langle v, \texttt{-k}[\text{PER} : \alpha, \text{NUM} : \beta]\rangle.\textbf{match } \alpha, \beta \textbf{ with}$$

$$\begin{array}{l} \mid 3, s \longrightarrow \text{"}is\text{"}(x) \\ \mid 1, s \longrightarrow \text{"}am\text{"}(x) \\ \mid \_, \_ \longrightarrow \text{"}are\text{"}(x) \end{array}$$

Outline

Constraints on Chunks at the Interfaces

Semantics

- Chunks are arbitrary
- Chunks can contain specifiers or complements
- Chunks are independent

Morphology

- Chunks are linear (i.e. non-branching treelets)
- Chunks go down complements (not specifiers)
- Chunks are uniform:

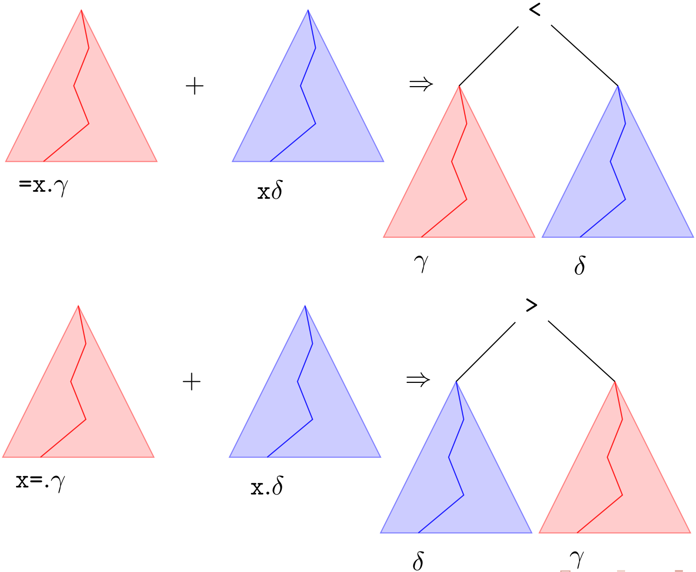
  Many chunks are present which differ just in the particular choice of content morphemes.
  (For many choices of V, T-v-V is an idiom)

Outline

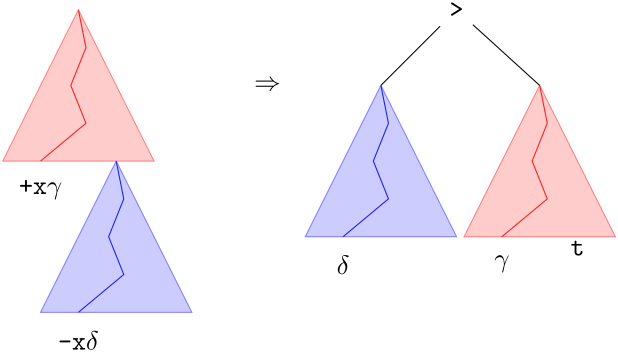
Minimalist Grammars

- To specify a grammar, we need to specify two things:
  1. The features
     (which features we will use in our grammar)
  2. The lexicon
     (which syntactic feature sequences are assigned to which words)

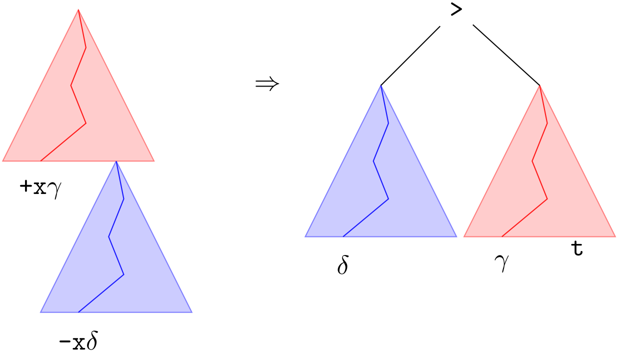
Merge



Move



Move



SMC

No other possible mover