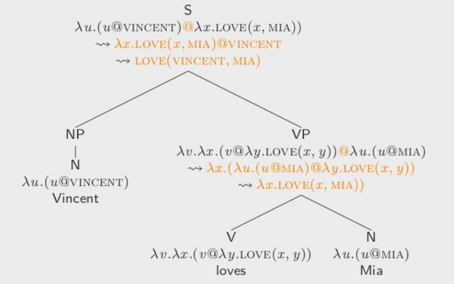
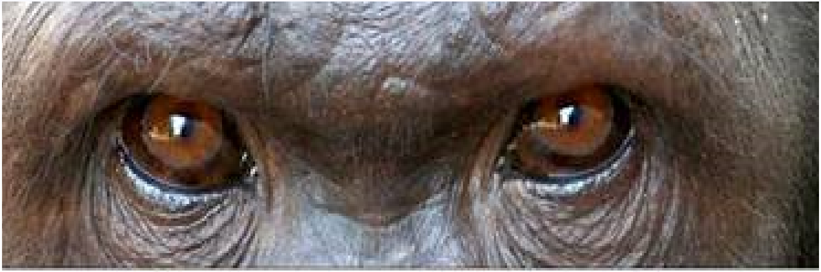


Logic & Language
Masterclass Logic – 2 March 2017

Jelle Zuidema

Institute for Logic, Language & Computation, UvA



Some facts about language: symbolicism

- Word meanings are conventional and arbitrary: even e.g. onomatopoeia: cock-a-doodle-do, cocorico, kukeleku
- Adult speakers know >>10000 words; children learn these words over the course of just a few years.
- Languages are transmitted culturally, and slowly change over the course of a number of generations, giving rise to an enormous variety of over > 6000 languages.

Some facts about language: phonemic coding

- Words are built-up from (meaningless) basic speech sounds, the phonemes;
- Phonemes are defined as the minimal difference in sound that corresponds to a difference in meaning. E.g. minimal pairs:

  bed bad bet bat /e/, /a/, /d/, /t/

- Phonemes are different in every language (and dialect), but phonemic coding is universal.

Some facts about language: compositional semantics

Sentences are built-up from meaningful words.

Words can be built from meaningful morphemes.
E.g.: “he walk-s”, disproof, disallow, rearrange

The meaning of a larger whole is determined by the meaning of the words and the way they are put together (compositionality),

i.e. word order and morphological marking and agreement: e.g. “In vino veritas est.”

Some facts about language: Hierarchical phrase structure

- *The man with the gun is in town*

Det N Prep Det N V Prep N

NP NP PP

PP

NP VP

S

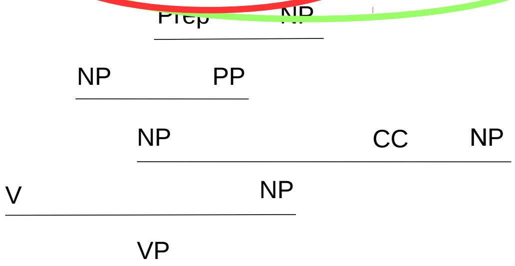
Some facts about language: Recursion

Phrases of category *X* can be embedded in a phrase of the same category *X.*

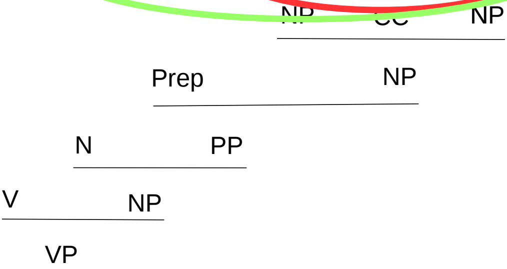
'the man', 'the gun' and 'the man with the gun' are all noun phrases (NP):

- *The man is in town*
- *The gun fell*

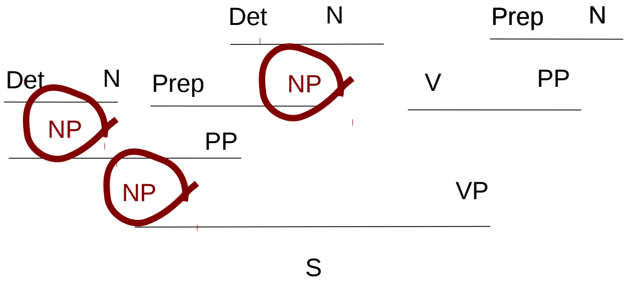
- *We ate pizza with ansjovis and soup*



- *We ate pizza with ansjovis and soup*



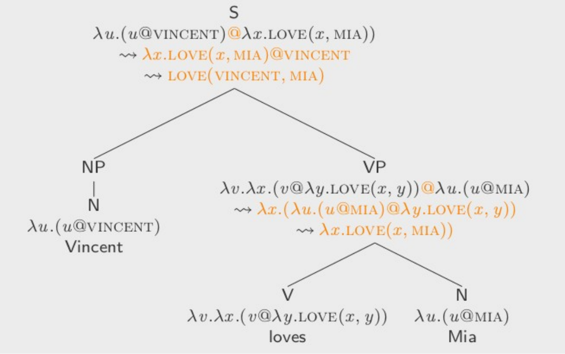
- *The man with the gun is in town*



- *We ate pizza with fork and knife*

“Structural Ambiguity”

Hierarchical structure + compositional semantics = hierarchical compositionality

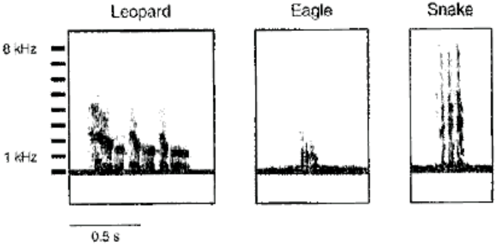


Hierarchical compositionality matters!

- Sentences have hierarchical structure:
  - Phrases can contain phrases ("hierarchy")
  - The embedded phrase can even be of the same type ("recursion")
- One sentence can often correspond to multiple different structure ("structural ambiguity")
  - Different structures correspond to different meanings

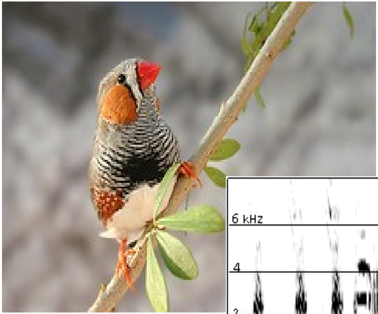
Arbitrariness: Vervet monkey alarm calls

(Struhsaker'67; Seyfarth, Cheney & Marler'80; squirrels: Sherman'77)

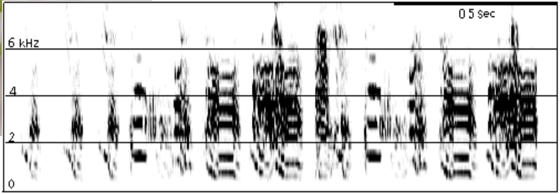


"chirp" "grunt" "chutter"

Cultural transmission, vocal learning, discreteness: songbirds

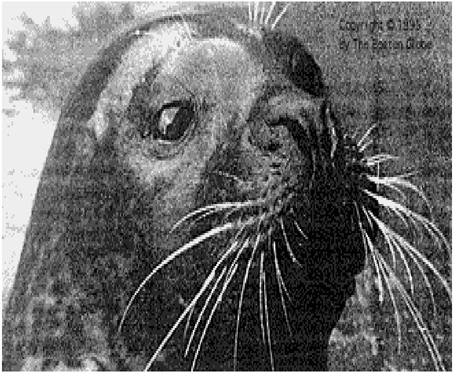


zebrafinch



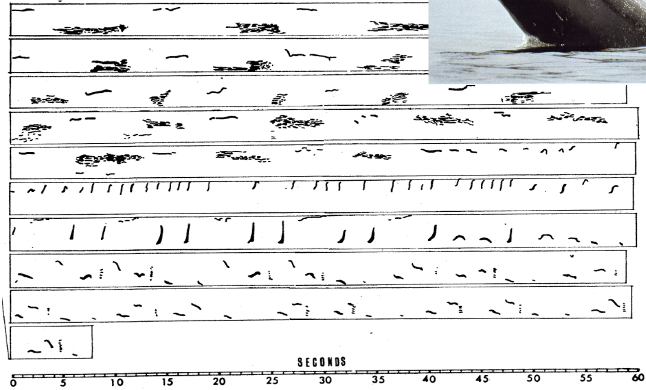
Doupe & Kuhl 1999

Vocal learning: seals



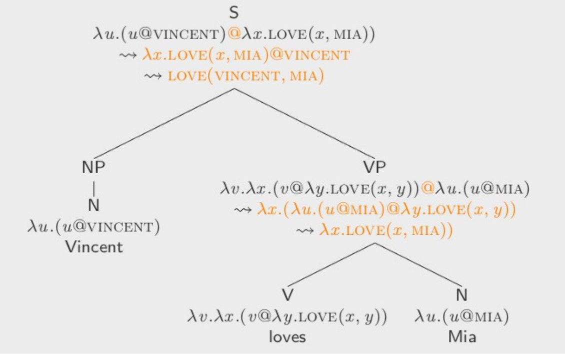
Hoover, the talking seal

Cultural transmission, vocal learning, discreteness: Humpback whales



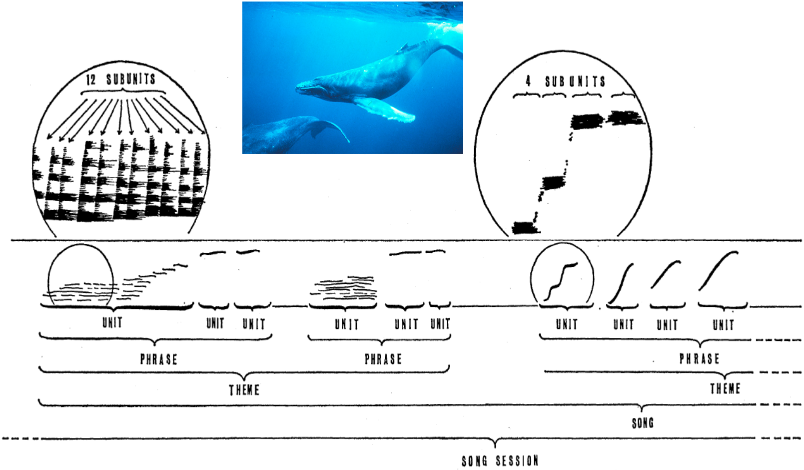
(Watlington'63; Payne & McVay'71)

Hierarchical structure + compositional semantics = hierarchical compositionality



Humpback whales: no semantics

Payne & McVay (1971)



Bee dance: non-hierarchical

(von Frisch'65, '74)

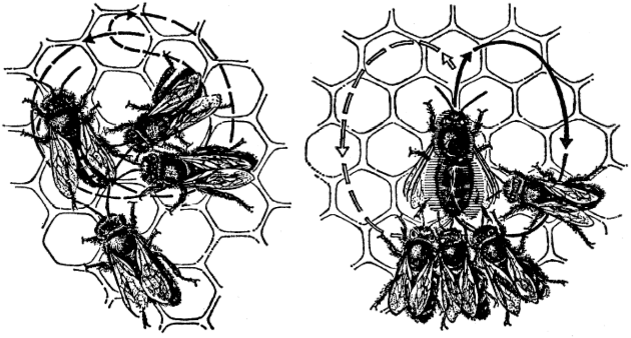


Fig. 1. Running curve of the bee (a) during round dance and (b) during tail-wagging dance. Bees that follow the dancer take in information.

von Frisch'74, *Science* 4152:664

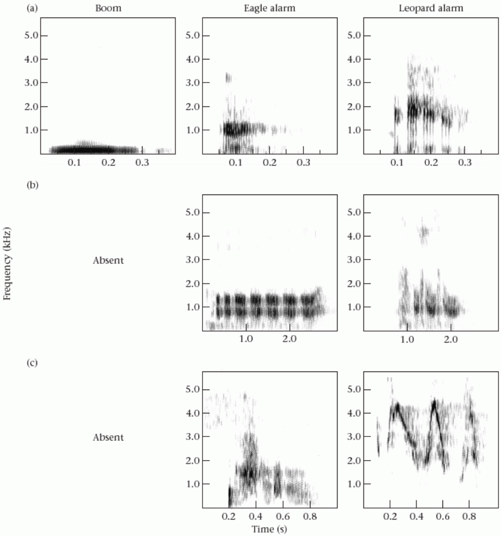
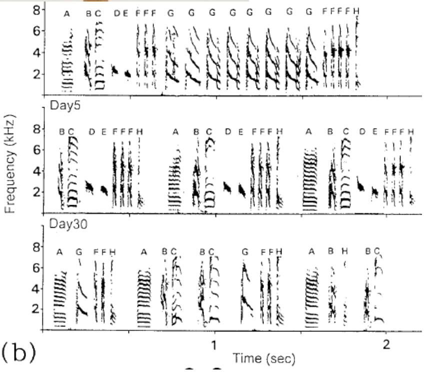


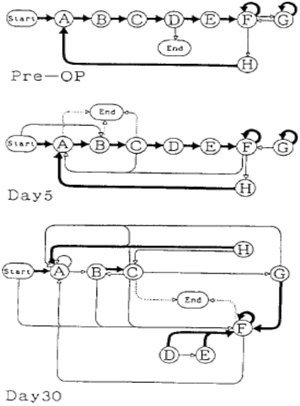
Figure 1. Spectrographic illustrations of the vocalizations used in this study. (a) Male Campbell's monkey, (b) male Diana monkey, (c) female Diana monkey. Male Campbell's and male Diana monkey alarm calls were used as playback stimuli (see text). Recordings were digitized at a sampling rate of 44 kHz (16 bits accuracy) and displayed using a 256-point Fourier transformation (Hamming window function) that resulted in wide-band spectrograms (analysis resolution: 700 Hz with 21.5 Hz/0.72 ms accuracy).

Monkey alarm-calls: non-hierarchical

(Zuberbühler'02 An. Beh. 63)

Bird song: No semantics





(Okanoya & Yamaguchi'97; *J. Neurobio. 33,4)*

First-order Predicate Logic

- Natural Language vs. Formal Languages
- Building blocks of FOL

Well-formed expressions

- Arithmetic
- First-order Predicate Logic
- Natural Language

Compositional Semantics

- Natural Language

What are the advantages of formal over natural languages?

If I am elected, I will make America great again.

$$EP(\mathbf{t}, \mathbf{a}) \Rightarrow MG(\mathbf{t}, \mathbf{a})$$

$$((16 - 2)/2) - 6$$

What are the building blocks of First Order Logic?

$$EP(\mathbf{t}, \mathbf{a}) \Rightarrow MG(\mathbf{t}, \mathbf{a})$$

$$\forall x \forall y (EP(x, y) \Rightarrow \neg MG(x, y))$$

$$\exists x (P(x) \wedge \neg Q(x))$$

$$\forall x (\text{POTUS}(x) \Leftrightarrow \text{CommanderInChief}(x))$$

$$\forall x\, (\text{DrinkingAlcohol}(x) \Rightarrow (\text{EighteenPlus}(x) \vee \text{BreakingTheLaw}(x)))$$

Building blocks of FOL

- Individual Constants such as **a**,**b**, max1, etc,
- Individual Variables such as w,x,y,z
- Predicates such as snore1, love1, P,Q,R,S
- Connectives $\vee$, $\wedge$, $\Leftarrow$, $\Leftrightarrow$
- Negation $\neg$
- Quantifiers $\exists$, $\forall$
- Brackets (,)

What makes a well-formed expression?

$$((16 - 2)/2) - 6$$

$$EP(\mathbf{t}, \mathbf{a}) \Rightarrow MG(\mathbf{t}, \mathbf{a})$$
$$\forall x \forall y EP(x, y) \Rightarrow \neg MG(x, y)$$

If I am elected, I will make America great again.

A Grammar for Arithmetic

1. Expression → Integer
2. Expression → (Expression Operator Expression)
3. Statement → Expression = Expression

A Grammar for Arithmetic

Rules

1. Exp $\rightarrow$ Integer
2. Exp $\rightarrow$ (Exp Op Exp)
3. Statement $\rightarrow$ Exp = Exp

Lexicon

1. Integer $\rightarrow$ 0, 1, 2, 3, 4, ...
2. Op $\rightarrow$ +, -, *, /, ...

0	Exp
R1	Integer
L1	3

A Grammar for Arithmetic

Rules

1. Exp → Integer
2. Exp → (Exp Op Exp)
3. Statement → Exp = Exp

Lexicon

1. Integer → 0, 1, 2, 3, 4, . . .
2. Op → +, -, *, /, . . .

0	Exp
R2	(Exp Op Exp)
R1	(Integer Op Exp)
L1	(7 Op Exp)
L2	(7 - Exp)
R2	(7 - (Exp Op Exp))
R1	(7 - (Integer Op Exp))
L1	(7 - (5 Op Exp))
L2	(7 - (5 + Exp))
R1	(7 - (5 + Integer))
L1	(7 - (5 + 9))

A Grammar for Arithmetic

Rules

1. Exp → Integer
2. Exp → (Exp Op Exp)
3. Statement → Exp = Exp

Lexicon

1. Integer → 0, 1, 2, 3, 4, . . .
2. Op → +, -, *, /, . . .

0	Statement
R3	Exp = Exp
R1	Integer = Exp
L1	7 = Exp
R1	7 = Integer
L1	7 = 5

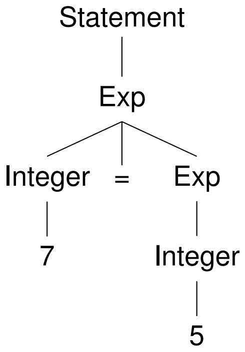
A Grammar for Arithmetic

Rules

1. Exp → Integer
2. Exp → (Exp Op Exp)
3. Statement → Exp = Exp

Lexicon

1. Integer → 0, 1, 2, 3, 4, . . .
2. Op → +, -, *, /, . . .



A Grammar for Arithmetic

Rules

1. Exp → Integer
2. Exp → (Exp Op Exp)
3. Statement → Exp = Exp

Lexicon

1. Integer → 0, 1, 2, 3, 4, . . .
2. Op → +, -, *, /, . . .

Question

How can you derive the expression below?

$$(2 - (3 + 7))$$

A Grammar without Brackets

Rules

1. Exp → Integer
2. Exp → Exp Op Exp
3. Statement → Exp = Exp

Lexicon

1. Integer → 0, 1, 2, 3, 4, ...
2. Op → +, -, *, /, ...

Question

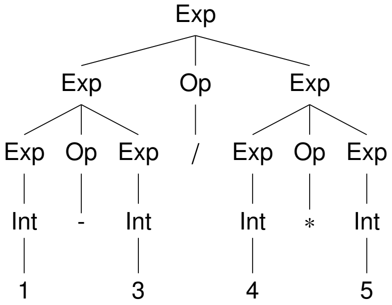
Can you give two different derivations of the expression below?

$$1 - 3/4 * 5$$

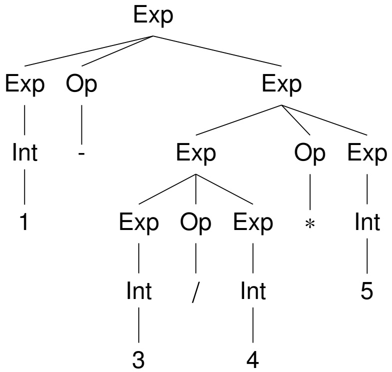
What are their solutions? Is there one that corresponds with the standard operator order (Brackets, Powers/Roots, Division/Multiplication, Addition/Substraction)?

Answer

Incorrect operator order



Correct operator order



Note: searching for a possible dervation (and tree structure) for a given expression is called ”parsing”.

What is a grammar that generates well-formed FOL-expressions?

$$EP(\mathbf{t}, \mathbf{a}) \Rightarrow MG(\mathbf{t}, \mathbf{a})$$

$$\forall x \forall y (EP(x, y) \Rightarrow \neg MG(x, y))$$

$$\exists x (P(x) \wedge \neg Q(x))$$

$$\forall x (\text{POTUS}(x) \Leftrightarrow \text{CommanderInChief}(x))$$

$$\forall x\, (\text{DrinkingAlcohol}(x) \Rightarrow (\text{EighteenPlus}(x) \vee \text{BreakingTheLaw}(x)))$$

What is a grammar that generates well-formed FOL-expressions?

1. S → Predicate(Term)
2. S → Predicate(Term,Term)
3. S → Predicate(Term,Term,Term)
4. S → (S Connective S)
5. S → Quantifier Variable S
6. S → Negation S
7. Term → Constant
8. Term → Variable

Well-formedness in Natural Language

- What is the shortest well-formed sentence?
- What are the equivalents of 1-place, 2-place and 3-place predicates?
- Proper nouns, nouns, determiners, intransitive verbs, transitive verbs, ditransitive verbs
- Adjectives, prepositions, adverbs?

Syntax with DCGs

```
s --> np, vp.              pn --> [vincent].
np --> pn.                 pn --> [mia].
vp --> iv.                 noun --> [woman].
vp --> tv, np.             noun --> [foot,massage].
np --> det, noun.          iv --> [snorts].
det --> [a].               iv --> [walks].
det --> [every].           tv --> [likes].
```

- complex syntactic categories: `s`, `np`, `vp`
- simple syntactic categories: `det`, `noun`, `iv`, `tv`
- lexical items: `a`, `every`, `vincent`, `mia`, `woman`, `footmassage`, `snorts`, `walks`, `likes`.

```
?- s([mia,likes,a,foot,massage],[]).        ?- np(X,[]).
true .                                       X = [vincent] ;
                                             X = [mia] ;
                                             X = [a, woman] ;
?- s([every,walks],[]).                      X = [a, foot, massage] ;
fail.                                        X = [every, woman] ;
                                             X = [every, foot, massage].
```

Compositionality

We want to be able to establish a systematic (non-arbitrary) relation between sentences and formulas.

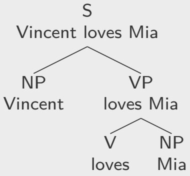
Vincent loves Mia	$? \rightsquigarrow$	$\textsc{love}(\textsc{vincent}, \textsc{mia})$
Everyone hates Butch	$? \rightsquigarrow$	$\forall x.\textsc{hate}(x, \textsc{butch})$

Intuitively, we know that the meaning of a sentence is based on the meaning of its bits and pieces (*compositionality*):

- we may be able to associate a representation with each lexical item, but how is this information combined?
- the meaning of a sentence is not only based on the words that make it up, but also on the ordering, grouping, and relations among such words
- the missing ingredient is a notion of **syntactic structure**.

Syntax and Compositional Semantics

As you know, syntax tells us how to hierarchically decompose a sentence into sub-parts that ultimately lead to the lexical items:



- If we associate a semantic representation with each lexical item, and...
- describe how the semantic representation of a syntactic constituent is to be built up from the representation of its sub-parts, then...
- we have at our disposal a compositional semantics: a systematic way of constructing semantic representations for sentences.

Semantic Construction

Now we have a plausible strategy for finding a way to systematically associate first-order semantic representations with sentences.

We need to:

1. Specify a reasonable syntax for the fragment of natural language of interest.
2. Specify semantic representations for the lexical items.
3. Specify how the semantic representation of a syntactic constituent is constructed in terms of the representations of its subparts.

Since we are interested in semantics, task 1 and 2 are where our real interests lie.

To handle task 1, we'll adopt a very simple solution: we'll use Definite Clause Grammars (DCGs), the built-in Prolog mechanism for grammar specification and parsing.

Syntax with DCGs

```
s --> np, vp.              pn --> [vincent].
np --> pn.                 pn --> [mia].
vp --> iv.                 noun --> [woman].
vp --> tv, np.             noun --> [foot,massage].
np --> det, noun.          iv --> [snorts].
det --> [a].               iv --> [walks].
det --> [every].           tv --> [likes].
```

- complex syntactic categories: `s`, `np`, `vp`
- simple syntactic categories: `det`, `noun`, `iv`, `tv`
- lexical items: `a`, `every`, `vincent`, `mia`, `woman`, `footmassage`, `snorts`, `walks`, `likes`.

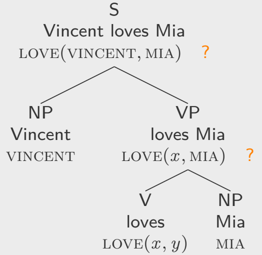
```
?- s([mia,likes,a,foot,massage],[]).        ?- np(X,[]).
true .                                      X = [vincent] ;
                                            X = [mia] ;
                                            X = [a, woman] ;
?- s([every,walks],[]).                     X = [a, foot, massage] ;
fail.                                       X = [every, woman] ;
                                            X = [every, foot, massage].
```

Semantic Construction

How shall we deal with tasks 2 and 3?

2. Specify semantic representations for the lexical items.
3. Specify how the semantic representation of a syntactic constituent is constructed in terms of the representations of its subparts.

Using plain FOL does not seem very handy...



we could represent lexical items with first-order terms and formulas, but how do we combine them?

we'd like to replace variables with terms, but how should we do that?

Fortunately, we can use a notational extension of FOL that will make these tasks easy: the *lambda calculus*.

The Lambda Calculus

Lambda Abstraction

We shall view the lambda calculus as a notational extension of FOL that allows us to bind variables with a new operator λ:

$$\lambda x.\textsc{woman}(x)$$

- the prefix $\lambda x.$ binds the occurrence of x in $\textsc{woman}(x)$
- we often say the that prefix $\lambda x.$ *abstracts over* x, and call expressions with such prefixes *lambda expressions* or *lambda abstractions*
- we can use one lambda expression as the body of another one:

$$\lambda x.\lambda y.\textsc{love}(x, y)$$

Functional Application

We can think of the lambda calculus as a tool dedicated to gluing together the items needed to build semantic representations.

$$\lambda x.\text{WOMAN}(x)$$

- the purpose of abstracting over variables is to mark the slots where we want substitutions to be made
  - * the binding of the free variable x in $\text{WOMAN}(x)$ indicates that WOMAN has an argument slot where we may perform substitutions
- lambda abstractions can be seen as *functors* that can be applied to *arguments* [we shall use the symbol @ for functional application]

$$\lambda x.\text{WOMAN}(x)@\text{MIA}$$

- a compound expression of this sort refers to the application of the functor $\lambda x.\text{WOMAN}(x)$ to the argument MIA.

β-conversion

Compound expressions $\mathcal{F}@\mathcal{A}$ can be seen as instructions to

- throw away the $\lambda x.$ prefix of the functor $\mathcal{F}$, and
- replace any occurrence of x bound by the λ-operator with the argument $\mathcal{A}$

This replacement or substitution process is called β-conversion:

$$\begin{array}{rcl} \lambda x.\text{WOMAN}(x)@\text{MIA} & \rightsquigarrow & \text{WOMAN}(\text{MIA}) \\ \lambda y.\lambda x.\text{HATE}(x,y)@\text{BUTCH} & \rightsquigarrow & \lambda x.\text{HATE}(x,\text{BUTCH}) \end{array}$$

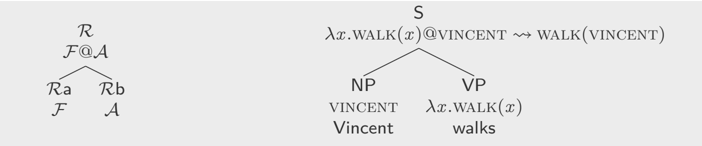
Note that the λ-operator can bind variables ranging over complex expressions: lambda abstractions can also act as arguments

$$\begin{array}{rcl} \lambda v.\exists x.(\text{BOXER}(x) \wedge v@x) \ @ \ \lambda x.\text{DANCE}(x) & \rightsquigarrow & \exists x.(\text{BOXER}(x) \wedge \lambda x.\text{DANCE}(x)@x) \\ & \rightsquigarrow & \exists x.(\text{BOXER}(x) \wedge \text{DANCE}(x)) \end{array}$$

Lambda Calculus for Semantic Construction

Lambda abstraction, functional application, and β-conversion are the main ingredients we need to deal with semantic construction:

- Once we have devised lambda abstractions to represent lexical items, we only need to use functional application and β-conversion to combine semantic representations compositionally.
  - ∗ Given a syntactic constituent $\mathcal{R}$ with subparts $\mathcal{R}$a and $\mathcal{R}$b, we need to specify which subpart is to be thought as the functor $\mathcal{F}$ and which as the argument $\mathcal{A}$.
  - ∗ We then construct the semantic representation of $\mathcal{R}$ by functional application $\mathcal{F}@\mathcal{A}$



A boxer walks

- S
  $\lambda v.\exists x.(\text{BOXER}(x) \wedge v@x)\ @\ \lambda x.\text{WALK}(x)$
  $\rightsquigarrow \exists x.(\text{BOXER}(x) \wedge \text{WALK}(x))$
  - NP
    $\lambda u.\lambda v.\exists x.(u@x \wedge v@x)\ @\ \lambda x.\text{BOXER}(x)$
    $\rightsquigarrow \lambda v.\exists x.(\text{BOXER}(x) \wedge v@x)$
    - Det
      $\lambda u.\lambda v.\exists x.(u@x \wedge v@x)$
      a
    - N
      $\lambda x.\text{BOXER}(x)$
      boxer
  - VP
    $\lambda x.\text{WALK}(x)$
    - V
      $\lambda x.\text{WALK}(x)$
      walks

Vincent loves Mia

S

$\lambda u.(u@\text{VINCENT})@\lambda x.\text{LOVE}(x, \text{MIA}))$

$\rightsquigarrow \lambda x.\text{LOVE}(x, \text{MIA})@\text{VINCENT}$

$\rightsquigarrow \text{LOVE}(\text{VINCENT}, \text{MIA})$

NP

N

$\lambda u.(u@\text{VINCENT})$

Vincent

VP

$\lambda v.\lambda x.(v@\lambda y.\text{LOVE}(x, y))@\lambda u.(u@\text{MIA})$

$\rightsquigarrow \lambda x.(\lambda u.(u@\text{MIA})@\lambda y.\text{LOVE}(x, y))$

$\rightsquigarrow \lambda x.\text{LOVE}(x, \text{MIA}))$

V

$\lambda v.\lambda x.(v@\lambda y.\text{LOVE}(x, y))$

loves

N

$\lambda u.(u@\text{MIA})$

Mia

Synchronicity

- The models for syntactic well-formedness and for sentence meaning (supporting inference; here: first order predicate logic, FOL) have different features;
- E.g., syntax in many natural languages is sensitive to word order, although word order rules differ from language to language. In predicate logic, clause order is not relevant.
- To compute the (FOL) meaning representation of a sentence, we need do do computations in the syntactic domain and *synchronously* perform computations in the meaning domain.
- Computationally challenging! Might this be unique to the human species?

Sue Savage-Rumbaugh

Apes: bonobo Kanzi

(Truswell, 2016)

645. (C) *I want Kanzi to grab Rose*.

(Kanzi turns around and grabs Rose on the leg, then walks away.)

581. (C) *Kanzi, tell Rose that you want to go outdoors*.

(Kanzi turns, looks at Rose, and gestures toward the play-yard door.) Rose looks in that direction and says, “You’re supposed to go over there?” (Kanzi heads toward the play-yard door, and Rose follows.)

In all these cases, however, Kanzi’s responses would be identical if he ignored the upstairs clauses, and just responded to the most embedded clause.

(Truswell, 2016)

428. (PC) *Give the water and the doggie to Rose*.

(Kanzi picks up the dog and hands it to Rose.)

526. (PC) *Give the lighter and the shoe to Rose*.

(Kanzi hands Rose the lighter, then points to some food in a bowl in the array that he would like to have to eat.)

281. (C) *Give me the milk and the lighter*.

(Kanzi does so.)

Kanzi's overall accuracy on the coordination construction is at chance level (25%).

Conclusions

Human language:

- Is an extremely complex and varied phenomenon;
- Is orders of magnitude more complex than any animal communication system discovered so far;
    - limited forms of referentiality, arbitrariness, discreteness, combinatoriality, but never in combination
- Above all, **hierarchical compositional semantics** is what makes it so powerful and unique
- Formal grammars + lambda calculus give us a first powerful model of hierarchical compositionality.