Lab class 1

Introduction, Lecture 1 & 2, assignment 1

Zeqi Zhao

Nov 8, 2023

Course Information

Lecturer: Zeqi Zhao (zeqi.zhao@uni-goettingen.de)

Lecture material (all available on Stud.IP):

- Slides
- Exercises (in class / after class)

To be prepared for the exams, you need to attend

- the lecture Tue.12–2pm
- my lab class and participate actively
- tutorials (highly recommended)

Course information

Assessment:

- credits, grades, moudles...
  See details in the syllabus of the lecture.
- submit assignments via CloCked (by Monday 23:59).

Office hours:

by appointments online or on site, just shoot me an Email

This lab class is designed to

make sure you become familiar with the materials from the lecture.

For each session, I will:

- go through the important things discussed in the lecture
- provide **in class exercises** to help you learn how to apply ***formal tools***

Formal semantics: What is it?

Core idea:

human languages (e.g. English) as **formal language**

A formal language is an abstract language consists of **strings of symbols** build over

- **a set of symbols** (***alphabet***)
- and **a set of formation rules**.

These rules specify which strings of symbols count as **well-formed**.

Toy formal language

Let's play around!

Imagine a formal language L_1 over an alphabet $\Sigma_1 = \{a, b\}$

Without any formation rules to constraint it, how many well-formed expressions belong to L_1?

Hint: Take the symbols contained in Σ_1 and perform **concatenation** (glue) repeatedly!

Toy formal language

Let's play around!

How many? The answer is **infinitely many**.

Let S_1 be the set of well-formed expressions of L_1:

$$S_1 = \left\{ \begin{array}{c} a \\ b \\ ab \\ ba \\ aa \\ bb \\ aab \\ aabb \\ abbaaabbaababb \\ \ldots \end{array} \right\}$$

Toy formal language

Let's add some rules!

Imagine a formal language L_2 over an alphabet $\Sigma_2 = \{a, b\}$

Formation rules:

- Rule of length: Not more than 2 symbols can be concatenated together.

How would the set of well-formed expressions look like?

Toy formal language

Let's add some rules!

Let S_2 be the set of well-formed expressions of L_2:

$$S_2 = \left\{ \begin{array}{c} a \\ b \\ ab \\ ba \\ aa \\ bb \end{array} \right\}$$

Now natural languages

If English is a formal language like L_2, *a* is an adjective (*'white'*), *b* is a noun (*'cat'*):

S_2 =

$$\left\{ \begin{array}{c} \text{white} \\ \text{cat} \\ \text{white cat} \\ \text{cat white} \\ \text{white white} \\ \text{cat cat} \end{array} \right\}$$

It seems like we need to add more rules.

Maybe we forbid the order *ba*? And forbid *aa* and *bb*?

Natural languages are very complicated

Forbidding *ba*, *aa*, *bb* would give us the wrong predictions:

- N+ADJ (*ba*):

  ’olive green’, ’baby blue’, ’cruelty-free’, ’sun-dried’...

- ADJ+ADJ (*aa*):

  ’a huge white cat’, ’a fancy red car’, ’John is Asian American’...

- N+AN (*bb*):

  ’baby stroller’, ’gas station’, ’house guest’...

The job of syntax

General question: How do we combine words into phrases and phrases into sentences?

hierarchy, categories, merge, movement...

Syntax: to study the rules for forming well-formed sentences structures

Data: native speakers's **wellformedness judgments**

(1) a. Colorless green ideas sleep furiously.
 b. ***Sleep green furiously ideas colorless.

We use * to mark a 'sentence' ungrammatical.

The job of semantics

General question: How do get the meaning of combined words/phrases and sentences?

Semantics data:

(2) ? Colorless green ideas sleep furiously.

(2) is for sure grammatical, but what does it mean?

Native speakers's semantic intuitions are hazy compared to wellformedness judgments.

The job of formal semantics

In formal semantics, we try our best to sharpen our intuitions and to capture ’meaning’ in a **precise** way.

We need formal concepts and techniques borrowed from mathematics and logic.

Truth-Conditional Semantics

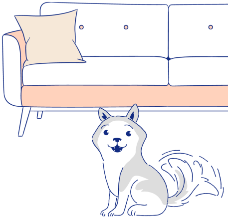
At this point, we focus on a relatively clear meaning:
Our ”**truth-conditional intuitions**”.

(3) The dog is sitting in front of the sofa.

A native speaker would know in what situations it is true and in what situations it is false.

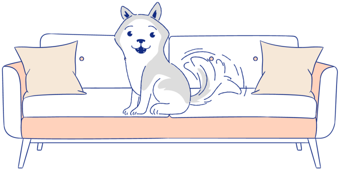
Truth-Conditions

(3) is true in this situation.



Truth-Conditions

(3) is false in this situation.



Truth-Conditions vs. Truth-Values

Note: Truth-values and truth-conditions are different things.

(3) is

> true iff there is a dog sitting in front of the sofa;
>
> false otherwise.

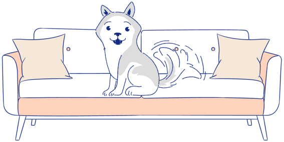
('iff' = 'if and only if')

In truth-conditional semantics, we see truth-conditions as the meaning of a sentence.

Truth-Conditions vs. Truth-Values

When a situation is given, a sentence **denotes a particular truth-value (0 or 1)**.

$[\![(3)]\!] = 1$ in



We use double brackets for the **interpretation function**:

$[\![(3)]\!]$ maps (3) to its meaning.

Goals of Truth-Conditional Semantics

There are infinitely many sentences and native speakers seem to be able to understand their meanings.

It's impossible to learn all sentence-meaning pairs by heart. So there must be a mechanism to **compute the truth-conditions of arbitrary sentences**.

Compositionality Principle

Hypothesis: Natural language semantics obeys the **compositionality principle**

Compositionality principle

The meaning of a sentence is a function of the meaning of its parts and its syntactic structure.

Compositionality Principle

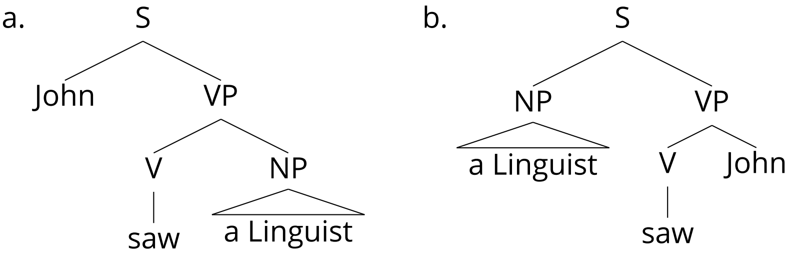
Compositionality principle

The meaning of a sentence is a function of **the meaning of its parts** and **its syntactic structure**.

To know the meaning of 'John saw a linguist', we need to know

- meanings of each individual words/morpheme (**Lexicon**),
- instructions about how to combine them (**Compositional Rules**),

The Role of Syntax



a and b have different truth-conditions, although they are made up of the same ingredients.

Literal meanings and entailment

Literal meaning:
by compositionality from the lexical meanings of individual words.

We use entailment to characterize literal meaning.

Anything not entailed by a sentence is not part of its literal meaning.

Relatedness as entailment

Entailment is a relation between sentences.

A sentence A entails a sentence B **iff whenever A is true, then B must also be true**.

Test: Contradiction

a. Eve is an intelligent student. **entails**
b. Eve is a student.

Proof:
John is an intelligent student, but it it is not the case that John is a student.

(we use '#' for semantically unacceptable.)

Assignment 1: Exercise 1

Is sentence (ii) an entailment of sentence (i)? Why/why not?

(1) a. i. Every man smokes.
   ii. Some man smokes.

b. i. Few men smoke.
   ii. Some men smoke

c. i. John and Mary are happy.
   ii. Mary is happy.

d. i. Neither John nor Mary is happy
   ii. It is not the case that both John and Mary are happy.

Assignment 1: Exercise 1

(1) b
i. Few men smoke.
ii. Some men smoke

Few [=*not any*] men smoke.
Few [=*not many*] men smoke.

Contradiction test:

Few men smoke, ***in fact***, it is not the case that some men smoke.

It seems like the 'not many' meaning can be cancelled, thus not part of the literal meaning.

Mathematical Concepts: Sets and functions

In formal semantics, we make use of mathematical concepts, most importantly sets and functions.

Why?

Sets can model our intuitions: Proper names

We have intuitions about what $[\![\textit{John}]\!]$ should be.
Native speakers use a **proper name** to **refer** to a particular individual/entity.

$[\![\textit{John}]\!] = \text{John}$

In set theory, we see proper names as denoting singleton sets, i.e., sets containing exactly one member.

$[\![\textit{John}]\!] = \{\text{John}\}$

Sets can model our intuitions: intransitive verbs

We have intuitions about what an intransitive verb $[\![smoke]\!]$ should be.

$[\![smoke]\!] =$ x: x smokes

In set theory, we see 'smoke' as denoting a set of people who smoke.

Two ways of defining compositional rules

To arrive at truth-conditions for 'John smokes', ther are two ways of defining compositional rules.

Option 1: we check set **membership**.

$$\left[\!\left[\overset{\frown}{A \quad B} \right]\!\right] = \text{true iff } [\![A]\!] \in [\![B]\!]$$
$$= \text{true iff } [\![A]\!] \text{ is an element of } [\![B]\!]$$

Two ways of defining compositional rules

Option 2: we check **subset relations**..

$$\left[\!\!\left[\widehat{\mathbf{A}\quad\mathbf{B}}\right]\!\!\right] = \text{true iff } [\![\mathbf{A}]\!] \subseteq [\![\mathbf{B}]\!]$$

In other words, we check whether every element of {John} is an element of {x: x smokes}.

This is what we need to do for *Excercise 2* and *3*.

Set Theory: A list to check

- Basics about set: member, empty set $\varnothing$, cardinality

- Two ways of defining sets: enumeration/abstraction

- Relations among sets: subset, proper subset, equivalence, power set

- Operations on sets: intersection, union, equivalence, set subtraction

(A detailed review of set theory will be uploaded; you can use it when preparing for exams.)

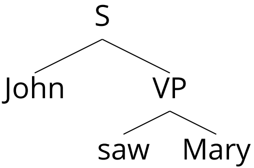
Exercise 4

6
a. $\{a, b, c\} = \{c, b, a\}$
b. $\{a, c\} \in \{a, b, c, d\}$
c. $\{c, d\} \subseteq \{a, b, c, d\}$
d. $\{a, b, c\} \subseteq \{\{a, b, c\}, \{d, e\}\}$

Why functions?

A problem with transitive verbs

Proper names and intransitive verbs can both be modeled using set theory. What about transitive verbs?



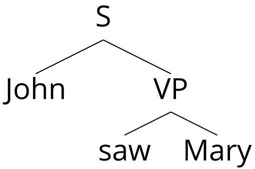
If VP denotes the set of people who saw Mary, ’saw’ cannot denote a set of individuals.

How do we combine ’saw’ and ’Mary’?

Why functions?

A problem with transitive verbs

We can also assume ⟦saw⟧ $= \{< x, y >: x \text{ saw } y\}$



But syntax tells us ’saw’ doesn’t combine with a pair $\langle$ John, Mary $\rangle$; it first combine with ’Mary’, then VP combines with ’John’.

Functions and relations

Another way to think about ’saw’ is a (binary) relation:

$R := \{<x, y> : x \text{ sees } y\}$

$<$ John, Mary $>\in R$

Functions are a special kind of relations: Each left component of an ordered pair in it has **exactly one** right component.

Think about the difference between ’*the brother of*’ and ’*the capital city of*’.

Ways of defining functions

We may define functions with lists, tables, or words.

- $F = \{< a, b >, < c, b >, < d, e >\}$
- $F = \left[\begin{array}{c} a \to b \\ c \to b \\ d \to e \end{array} \right]$

We can see functions are a special type of mapping where for each **input** there is a unique **output**.

Domain: The set of objects the function can take as inputs
Range: The set of objects the function returns as outputs.

- *F* is a function *f* with domain $\{$a, b, c$\}$ such that *f(a)=f=(c)=b* and *f(d)=e*

Functional application

With this very powerful tool of function, we are able to combine 'John' and 'smokes' via Functional application.

Imagine there are 3 boys John, Bill, Jack in a situation, John is the only one who smokes.

$$F = \begin{bmatrix} \text{John} \rightarrow 1 \\ \text{Bill} \rightarrow 0 \\ \text{Jack} \rightarrow 0 \end{bmatrix}$$

$$[\![\text{John smokes}]\!] = \begin{bmatrix} \text{John} \rightarrow 1 \\ \text{Bill} \rightarrow 0 \\ \text{Jack} \rightarrow 0 \end{bmatrix} \text{(John)=1}$$

Thanks!

Questions? Comments?