EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Semantics & Pragmatics SoSe 2022

Lecture 9: Formal Semantics (Summary)

24/05/2022, Christian Bentz

Overview

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Q&As

Tutorial 3

In Exercise 1 b) “Charles and John are brothers or cousins”, is inclusive “or” $\vee$ here also valid instead of exclusive “or” XOR?

Yes. In fact, strictly speaking, we said that we would translate “or” in English as $\vee$. However, in this case, our natural language understanding tells us that somebody cannot be somebody else’s cousin and brother at the same time. So this would require exclusive “or”. I would accept both in such a borderline case.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

General Background

Two Fundamental Concepts

Reference: How does the mapping between form and meaning work? Does it work at all?

tree

$\downarrow$

Compositionality: How are complex utterances built from smaller units? Are they built from smaller units at all?

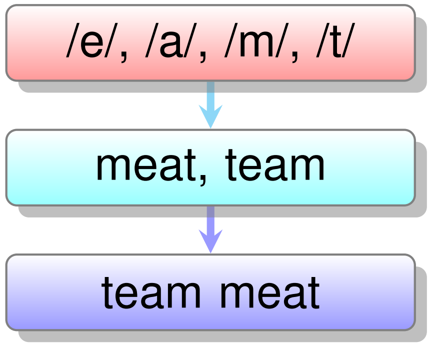
apple + tree

$\downarrow$

Duality of Patterning

"Language is structured on at least two levels (Hockett, 1960). On one level, a small number of meaningless building blocks (phonemes, or parts of syllables for instance) are combined into an unlimited set of utterances (words and morphemes). This is known as **combinatorial structure**. On the other level, meaningful building blocks (words and morphemes) are combined into larger meaningful utterances (phrases and sentences). This is known as **compositional structure**."

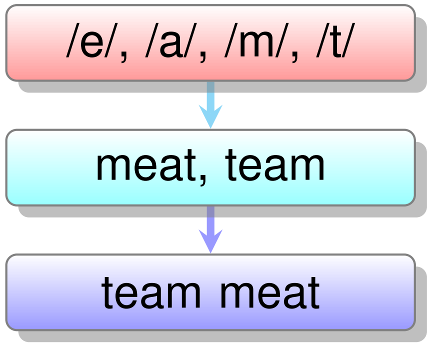
Little et al. (2017), p. 1.

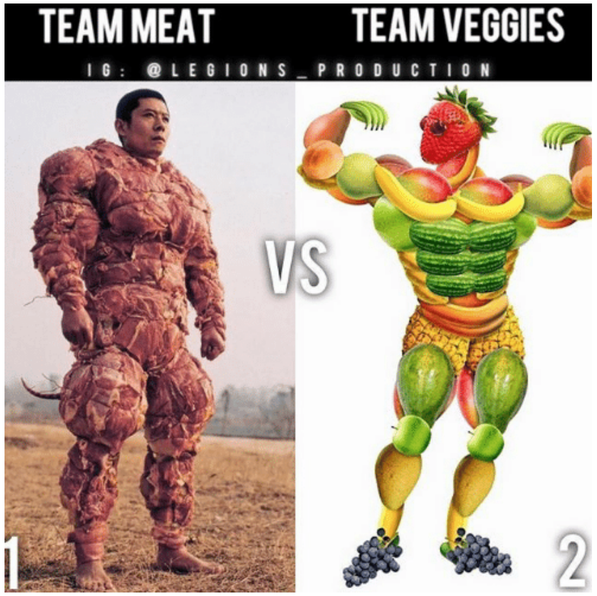


Duality of Patterning

"Language is structured on at least two levels (Hockett, 1960). On one level, a small number of meaningless building blocks (phonemes, or parts of syllables for instance) are combined into an unlimited set of utterances (words and morphemes). This is known as **combinatorial structure**. On the other level, meaningful building blocks (words and morphemes) are combined into larger meaningful utterances (phrases and sentences). This is known as **compositional structure**."

Little et al. (2017), p. 1.





EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

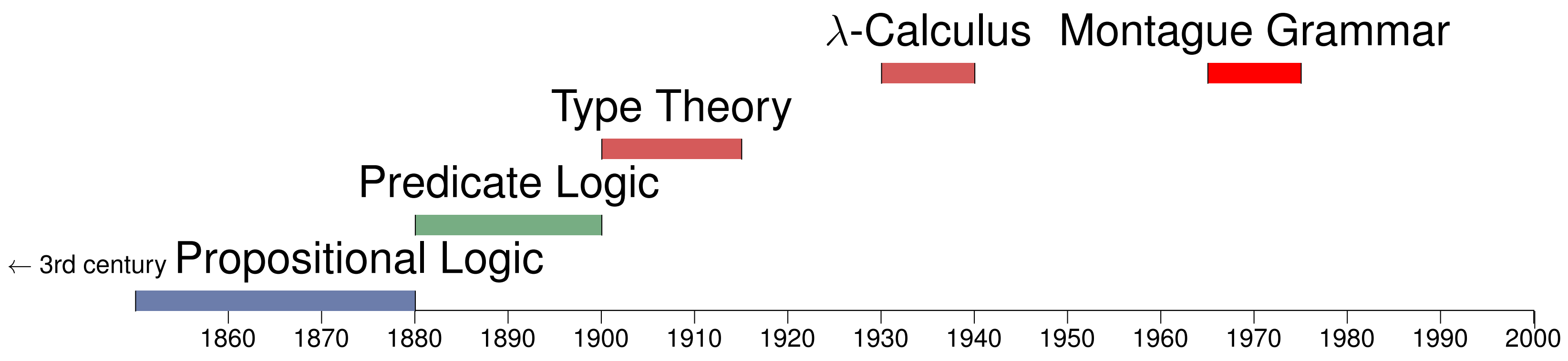
Faculty of Philosophy
General Linguistics

Historical Overview

Historical Overview

Some of the earliest proponents of each framework:

- **Propositional Logic**: Diodorus Cronus (died around 284 BCE at Alexandria in Egypt), Chrysippus (mid-3rd century).
- **Predicate Logic (1st and 2nd order)**: Frege (1879), Peirce (1885).
- **Type Theory**: Russell (1908).
- **λ-Calculus**: Church (1940).
- **Montague Grammar**: Montague (1970a, 1970b, 1973).



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 1: Propositional Logic

Formal Definition: Proposition

"The **proposition expressed by a sentence** is the **set of possible cases [situations]** of which that sentence is true."
Zimmermann & Sternefeld (2013), p. 141.

Coin-flip example:

situation	flip1	flip2
1	heads	heads
2	tails	tails
3	heads	tails
4	tails	heads

Sentence	**Proposition**
S_1: only one flip landed heads up	$[\![S_1]\!] = \{3, 4\}$
S_2: all flips landed heads up	$[\![S_2]\!] = \{1\}$
S_3: flips landed at least once tails up	$[\![S_3]\!] = \{2, 3, 4\}$
etc.	etc.

Propositional Formulas

“The propositional letters and the **composite expressions** which are formed from them by means of connectives are grouped together as *sentences* or **formulas**. We designate these by means of the letters ϕ and ψ, etc. For these **metavariables**, unlike the variables p, q, and r, there is no convention that different letters must designate different formulas.”

Gamut, L.T.F (1991). Volume 1, p. 29.

Examples:

$\phi \equiv$ p, q, r, etc.

$\phi \equiv \neg$p, $\neg$q, $\neg$r, etc.

$\phi \equiv$ p $\wedge$ q, p $\vee$ q, etc.

$\phi \equiv \neg(\neg p_1 \vee q_5) \rightarrow$ q, etc.

The Vocabulary

We can now define a **language L for propositional logic**. The “vocabulary” A of L consits of the propositional letters (e.g. p, q, r, etc.), the operators (e.g. $\neg$, $\wedge$, $\vee$, $\rightarrow$, etc.), as well as the round brackets ‘(’ and ‘)’. The latter are important to group certain letters and operators together. We thus have:

$$A = \{p, q, r, ..., \neg, \wedge, \vee, \rightarrow, ..., (,)\} \qquad (1)$$

The Syntax: Recursive Definition

Reminiscent of formal grammars of natural languages (see last years lecture on Phrase Structure Grammar), we now also need to define **syntactic rules** which allow for the symbols of the vocabulary to be combined yielding **well-formed expressions**. These rules are:

(i) Propositional letters in the vocabulary of *L* are formulas in *L*.

(ii) If ϕ is a formula in *L*, then $\neg\phi$ is too.

(iii) If ϕ and ψ are formulas in *L*, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ are too.[1]

(iv) Only that which can be generated by the clauses (i)-(iii) in a finite number of steps is a formula in *L*.

Gamut, L.T.F (1991). Volume 1, p. 35.

[1]We could also add the *exclusive or* here as a connective.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Examples of Valid and Invalid Formulas

Formula	Rule Applied
p ✓	(i)
¬¬¬q ✓	(i) and (ii)
$((\neg p \wedge q) \vee r)$ ✓	(i), (ii), and (iii)
$((\neg(p \vee q) \rightarrow \neg\neg\neg q) \leftrightarrow r)$ ✓	(i), (ii), and (iii)
pq x	–
$\neg(\neg\neg p)$ x	–
$\wedge p \neg q$ x	–
$\neg((p \wedge q \rightarrow r))$ x	–

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

The Semantics of Propositional Logic

“The valuations we have spoken of [i.e. truth valuations of formulas] can now, in the terms just introduced [i.e. functions], be described as (unary)[2] **functions mapping formulas onto truth values**. But not every function with formulas as its domain and truth values as its range will do. A valuation must agree with the **interpretations of the connectives** which are given in their truth tables.”

Gamut, L.T.F (1991). Volume 1, p. 35.

[2]An *unary* function is a function with a single argument, e.g. f(x). A *binary* function could be f(x,y), a *ternary* function f(x,y,z), etc.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Valuation Function

The valuation function V for each logical operator and logical formulas ϕ and ψ are then given as:

(i) Negation: $V(\neg\phi) = 1$ iff $V(\phi) = 0$,

(ii) Logical "and": $V(\phi \wedge \psi) = 1$ iff $V(\phi) = 1$ and $V(\psi) = 1$,

(iii) Inclusive "or": $V(\phi \vee \psi) = 1$ iff $V(\phi) = 1$ or $V(\psi) = 1$,

(iv) Material implication: $V(\phi \rightarrow \psi) = 0$ iff $V(\phi) = 1$ and $V(\psi) = 0$,

(v) Material equivalence: $V(\phi \leftrightarrow \psi) = 1$ iff $V(\phi) = V(\psi)$.

Gamut (1991). Volume I, p. 44.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 2: Predicate Logic

Propositional Logic vs. Predicate Logic

Commonalities:

- Usage of the same connectives and negation.

Differences:

- The introduction of **constants and variables** representing individuals and predicates to capture the main structural building blocks of sentences.
- The introduction of **quantifiers** to allow for quantified statements.

The Vocabulary

Similar as for propositional logic, we can define a **language *L* for predicate logic**. In this case, the "vocabulary" of *L* consits of

- a (potentially infinite) supply of **constant symbols** (e.g. a, b, c, etc.),
- a (potentially infinite) supply of **variable symbols** representing the constants (e.g. x, y, z, etc.),
- a (potentially infinite) supply of **predicate symbols** (e.g. A, B, C, etc.),
- the **connectives** (e.g. $\neg$, $\wedge$, $\vee$, $\rightarrow$, etc.),
- the **quantifiers** $\forall$ and $\exists$,
- as well as the round brackets '(' and ')'.
- (The equal sign '='.)

Translation Key

In order to translate a set of natural language sentences into predicate logic expressions unambiguously, we need a **translation key** listing the **predicates** and **constant symbols**.

Gamut, L.T.F (1991). Volume 1, p. 68.

English sentences:

(1) John is bigger than Peter or Peter is bigger than John.

(2) Alkibiades does not admire himself.

(3) If Socrates is a man, then he is mortal.

(4) Ammerbuch lies between Tübingen and Herrenberg.

(5) Socrates is a mortal man.

Translation key:

a_1: Alcibiades
a_2: Ammerbuch
j: John
p: Peter
s: Socrates
t: Tübingen
h: Herrenberg

Axy: x admires y
B_1xy: x is bigger than y
B_2xyz: x lies between y and z
M_1x: x is a man
M_2x: x is mortal

Translation Examples

We can then translate the natural language sentences into predicate logic by further identifying the logical operators, i.e. connectives and negation.

Gamut, L.T.F (1991). Volume 1, p. 68.

English sentences:

(1) John is bigger than Peter **or** John is bigger than Socrates.

(2) Alcibiades does **not** admire himself.

(3) **If** Socrates is a man, **then** he is mortal.

(4) Ammerbuch lies between Tübingen and Herrenberg.

(5) Socrates is a mortal man.

Translations:

(1) $B_1jp \vee B_1js$

(2) $\neg Aa_1a_1$

(3) $M_1s \rightarrow M_2s$

(4) B_2a_2th

(5) $M_1s \wedge M_2s$

The Syntax: Recursive Definition

Given the vocabulary of *L* we define the following clauses to create formulas of *L*.

(i) If A is an n-ary predicate letter in the vocabulary of *L*, and each of $t_1, \dots, t_n$ is a constant or a variable in the vocabulary of *L*, then $At_1, \dots, t_n$ is a formula in *L*.

(ii) If ϕ is a formula in *L*, then $\neg\phi$ is too.

(iii) If ϕ and ψ are formulas in *L*, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ are too.

(iv) If ϕ is a formula in *L* and *x* is a variable, then $\forall x\phi$ and $\exists x\phi$ are formulas in L.

(v) Only that which can be generated by the clauses (i)-(iv) in a finite number of steps is a formula in *L*.

Gamut, L.T.F (1991). Volume 1, p. 75.

Model Theory

"In order to develop and test a set of interpretive rules [...] it is important to provide very explicit descriptions for the test situations. As stated above, this kind of **description of a situation is called a model**, and must include two types of information: (i) **the domain**, i.e., the set of all individual entities in the situation; and (ii) the **denotation sets for the basic vocabulary items** [constant symbols, predicates] in the expressions being analyzed."

Kroeger (2019). Analyzing meaning, p. 240.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 3: Second-Order Logic

First-Order Logic vs. Second-Order Logic

Commonalities:

- Usage of the same **logical operators** (connectives, negation, quantifiers).
- Generally similar syntax and valuation of expressions.

Differences:

- Introducing **first-order predicate variables**, and **second-order predicates**.

Beyond First-Order Predicate Logic

We have seen that predicate logic is an extension of propositional logic, by introducing predicates and quantifiers. **Predicate logic** might itself be superseded by another logical system, called **second-order logic**.

Gamut, L.T.F (1991). Volume 1, p. 168.

Take the following English sentences:

(1) Mars is red.
(2) Red is a color.
(3) Mars has a color.
(4) John has at least one thing in common with Peter.

How can we translate these into logical expressions?

First-Order and Second-Order Logic

A **second-order logic** language L' is then an extension to a standard predicate logic language L by adding second-order predicates to L. The original language L is then sometimes referred to as **first-order logic** language.

Further Examples:

(5) $\exists X(\mathcal{C}X \wedge Xm)$ (English sentence: “Mars has a color.”)

(6) $\exists X(Xj \wedge Xp)$ (English sentence: “John has at least one thing in common with Peter.”)

(7) $\exists \mathcal{X}(\mathcal{X}R \wedge \mathcal{X}G)$ (English sentence: “Red has something (a property) in common with green.”)

Vocabulary (special to Second-Order Logic)

The vocabulary extensions to fit **second-order logic requirements** are:

- A (potentially infinite) supply of **first-order predicate variables** (e.g. X, Y, Z, etc.), which are necessary to quantify over first-order predicates,
- a (potentially infinite) supply of **second-order predicate constants** (e.g. $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, etc.).

If we wanted to take it even at a higher-order level we could also have:

- a (potentially infinite) supply of **second-order predicate variables** (e.g. $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, etc.) to stand in for second-order predicates.

The Syntax: Recursive Definition

Given the vocabulary of L we then define the following clauses to create formulas of *L*:

(i) If A is an n-ary **first-order** predicate letter/constant in *L*, and $t_1, \ldots, t_n$ are individual terms in *L*, then $At_1, \ldots, t_n$ is an (atomic) formula in *L*;

(ii) If X is a [**first-order**] predicate variable and t is an individual term in *L*, then Xt is an atomic formula in *L*;

(iii) If $\mathcal{A}$ is an n-ary **second-order** predicate letter/constant in *L*, and $T_1, \ldots, T_n$ are **first-order unary** predicate constants, or predicate variables, in *L*, then $\mathcal{A}T_1, \ldots, T_n$ is an (atomic) formula in *L*;

(iv) If ϕ is a formula in *L*, then $\neg\phi$ is too;

(v) If ϕ and ψ are formulas in *L*, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$ are too.

The Syntax: Recursive Definition

Given the vocabulary of L we then define the following clauses to create formulas of *L*:

(vi) If *x* is an individual variable ϕ is a formula in *L*, then $\forall x\phi$ and $\exists x\phi$ are also formulas in L;

(vii) If *X* is a [first-order] predicate variable, and ϕ is a formula in *L*, then $\forall X\phi$ and $\exists X\phi$ are also formulas in L;

(viii) Only that which can be generated by the clauses (i)-(vii) in a finite number of steps is a formula in *L*.

Gamut, L.T.F (1991). Volume 1, p. 170.

Note: In the above clauses (i) and (ii), the word "term" is used, which has not been defined by us before. In the context here, suffices to say that it includes both constants and variables (of constants), i.e. a, b, c, etc. and x, y, z, etc.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 4: Type Theory

Standard Logic vs. Typed Logic

Commonalities:

- Usage of the same **logical operators** (connectives, negation, quantifiers).

Differences:

- Introduction of a **potentially infinite number of types** defined for logical constants and variables which we can quantify over. Note that this makes typed logic a **higher-order logic**.

Definition: The Syntax of Types

For the set of types $\mathbb{T}$ we define that:

(i) $e, t \in \mathbb{T}$,

(ii) if $a, b \in \mathbb{T}$, then $\langle a, b \rangle \in \mathbb{T}$,

(iii) nothing is an element of $\mathbb{T}$ except on the basis of clauses (i) and (ii).

Gamut (1991), Volume 2, p. 79.

Note: *a* and *b* above are variables which stand in for all kinds of types. This means we can create an infinite number of types by recursively applying clause (ii). For example:

Applying (ii) to $a = e$ and $b = t$ yields $\langle e, t \rangle$
Applying (ii) to $a = \langle e, t \rangle$ and $b = t$ yields $\langle \langle e, t \rangle, t \rangle$
Applying (ii) to $a = e$ and $b = \langle \langle e, t \rangle, t \rangle$ yields $\langle e, \langle \langle e, t \rangle, t \rangle \rangle$
etc.

Definition: Functional Application

How do we derive one type of expression from another?

“[...] if α is an expression of type $\langle a, b\rangle$ and β is an expression of type a, then $\alpha(\beta)$ is of type b.”

Gamut (1991), Volume 2, p. 79.

Examples

If $\alpha = \langle e, t\rangle$ and $\beta = e$ then $\alpha(\beta) = t$.
If $\alpha = \langle\langle e, t\rangle, \langle e, t\rangle\rangle$ and $\beta = \langle e, t\rangle$ then $\alpha(\beta) = \langle e, t\rangle$.
If $\alpha = \langle t, \langle t, e\rangle\rangle$ and $\beta = t$ then $\alpha(\beta) = \langle t, e\rangle$.

However,

If $\alpha = \langle t, \langle t, e\rangle\rangle$ and $\beta = \langle t, e\rangle$ then $\alpha(\beta)$ is **not defined**.

The Syntax: Recursive Definition

The clauses for the syntax of a type-theoretic language are then:

(i) If α is a variable or a constant of type a in *L* [i.e. v_a or c_a], then α is an expression of type a in *L*.

(ii) If α is an expression of type $\langle a, b \rangle$ in *L*, and β is an expression of type a in *L*, then $(\alpha(\beta))$ is an expression of type b in *L*.

(iii) If ϕ and ψ are expressions of type *t* in *L* (i.e. formulas in L), then so are $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, and $(\phi \leftrightarrow \psi)$.

(iv) If ϕ is an expression of type *t* in *L* and v is a variable (of arbitrary type a), then $\forall v\phi$ and $\exists v\phi$ are expression of type *t* in *L*.

(v) If α and β are expressions in *L* which belong to the same (arbitrary) type, then $(\alpha = \beta)$ is an expression of type *t* in *L*.

(vi) Every expression *L* is to be constructed by means of (i)-(v) in a finite number of steps.

Gamut (1991), Volume 2, p. 81-82.

Examples of Valid and Invalid Expressions

Definition of Types

Assume j is of type e (i.e. representing an entity), x is of type e, A is of type $\langle e, t\rangle$ (i.e. a first order one-place predicate), B is of type $\langle e, \langle e, t\rangle\rangle$ (i.e. a first-order two-place predicate), and $\mathcal{C}$ is of type $\langle\langle e, t\rangle, t\rangle$ (i.e. a second-order one-place predicate).

Expressions	**Clause Applied**
j ✓	(i)
A ✓	(i)
A(j) ✓	(i) and (ii)
(B(j))(x) ✓ alternative notation: B(j)(x)	(i) and (ii)
$\mathcal{C}$(B(j)) ✓	(i) and (ii)
A(j) $\wedge$ $\mathcal{C}$(A)✓	(i), (ii), and (iii)
$\forall$xA(x)✓	(i), (ii), and (iv)
Aj x	–
B(A) x	–
$\forall$x$\mathcal{C}$(x) x	–

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 5: λ-Calculus

The Syntax: Adding the λ-clause

We simply add another clause to the **type-theoretic language** syntax:

(vii) If α is an expression of type a in L, and v is a variable of type b, then λv(α) is an expression of type $\langle b, a \rangle$ in L.[3]

Gamut (1991), Volume 2, p. 104.

[3] I added the brackets around α here, since at least in some cases these are necessary to disambiguate.

Examples of λ-Abstractions

Assume a, b and x, y are of type e; A is of type $\langle e, t\rangle$; B is of type $\langle e, \langle e, t\rangle\rangle$; and X is of type $\langle e, t\rangle$.

Expressions	**Types**	**λ-Abstraction**	**Types**
x ✓	e	λx(x)	$\langle e, e\rangle$
A(x)✓	t	λx(A(x))	$\langle e, t\rangle$
B(y)(x) ✓	t	λx(B(y)(x)) or λy(B(y)(x))	$\langle e, t\rangle$
B(a)(x) ✓	t	λx(B(a)(x))	$\langle e, t\rangle$
$\forall$xB(x)(y)✓	t	λy($\forall$xB(x)(y))	$\langle e, t\rangle$
X(a) ✓	t	λX(X(a))	$\langle\langle e, t\rangle, t\rangle$
X(a) $\wedge$ X(b)✓	t	λX(X(a) $\wedge$ X(b))	$\langle\langle e, t\rangle, t\rangle$

Note: In our practical usage of the type-theoretic language, variables are mostly defined to have type e (i.e. x, y, z, etc.). In some cases, they might be of type $\langle e, t\rangle$, namely, if they refer to predicate variables (X, Y, Z, etc.). Hence, λ-abstraction essentially amounts to **adding an e or $\langle e, t\rangle$ as a "prefix"** to the type of the expression that is abstracted over.

λ-Conversion (aka β-Reduction)

Informally speaking, **λ-conversion**[4] is the process whereby we reduce the λ-statement by removing the λ-operator (and the variable directly following it) and pluging an expression (in the simplest case a constant c, or a predicate constant C) into **every occurrence of the variable which is bound by the λ-operator**.

Typed expression	**λ-Abstraction (over x or X)**	**λ-Conversion (with c or C over x or X)**
S(x)	λx(S(x))	λx(S(x))(c) = S(c)
S(x) $\wedge$ D(x)	λx(S(x) $\wedge$ D(x))	λx(S(x) $\wedge$ D(x))(c) = S(c) $\wedge$ D(c)
X(a) $\wedge$ X(b)	λX(X(a) $\wedge$ X(b))	λX(X(a) $\wedge$ X(b))(C) = C(a) $\wedge$ C(b)

[4] The term λ-conversion is not to be confused with α-conversion. The latter refers to replacing one variable for another.

Why is λ-calculus needed?

If our aim is to model not only full sentences and formulas representing predicates, but also parts of sentences, and even individual words, by using in a unified account, then λ-abstraction and λ-conversion are possible solutions. Thus, λ-calculus allows us to capture the **compositionality of language**.

English sentence	**Typed expression**
John smokes and drinks.	$\lambda x(S(x) \land D(x))(j) = S(j) \land D(j)$
John smokes	$\lambda x(S(x))(j) = S(j)$
smokes	$\lambda x(S(x))$
drinks	$\lambda x(D(x))$
smokes and drinks	$\lambda x(S(x) \land D(x))$

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Summary Formal Semantics

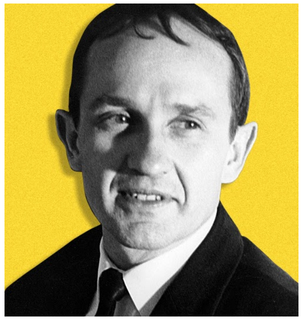
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Translation Summary

Natural Language	PL	FOL	SOL	TL
John smokes.	p	Sj	Sj	$S(j)$
John smokes and drinks.	$p \wedge q$	$Sj \wedge Dj$	$Sj \wedge Dj$	$S(j) \wedge D(j)$
Jumbo likes Bambi.	r	Ljb	Ljb	$L(b)(j)$
Every man walks.	p_1	$\forall x(Mx \rightarrow Wx)$	$\forall x(Mx \rightarrow Wx)$	$\forall x(M(x) \rightarrow W(x))$
Red is a color.	q_1	Cr	$\mathcal{C}R$	$\mathcal{C}(R)$
smokes and drinks	_	_	_	$\lambda x(S(x) \wedge D(x))$
every man	_	_	_	$\lambda X(\forall x(M(x) \rightarrow X(x)))$
every	_	_	_	$\lambda Y(\lambda X(\forall x(Y(x) \rightarrow X(x))))$
is	_	_	_	$\lambda X(\lambda x(X(x)))$

PL: Propositional Logic
FOL: First-Order Predicate Logic
SOL: Second-Order Predicate Logic
TL: Typed Logic (Higher-Order) with λ-calculus

Montague Grammar (Tensed Intensional Logic)



1 Every variable and constant of type a is in ME_a.
2 If $\alpha \in \mathrm{ME}_a$ and u is a variable of type b, then $\lambda u\alpha \in \mathrm{ME}_{\langle b,\, a\rangle}$.
3 If $\alpha \in \mathrm{ME}_{\langle a,b\rangle}$ and $\beta \in \mathrm{ME}_a$, then $\alpha(\beta) \in \mathrm{ME}_b$.
4 If $\alpha, \beta \in \mathrm{ME}_a$, then $\alpha = \beta \in \mathrm{ME}_t$.
5 If $\phi, \psi \in \mathrm{ME}_t$ and u is a variable, then $\neg\phi, [\phi \wedge \psi], [\phi \vee \psi], [\phi \rightarrow \psi], [\phi \leftrightarrow \psi]$, $\bigvee u\phi, \bigwedge u\phi, \Box\phi, W\phi, H\phi \in \mathrm{ME}_t$.
6 If $\alpha \in \mathrm{ME}_a$, then $[\hat{}\alpha] \in \mathrm{ME}_{\langle s,a\rangle}$.
7 If $\alpha \in \mathrm{ME}_{\langle s,\, a\rangle}$, then $[\check{}\alpha] \in \mathrm{ME}_a$.
8 Nothing is in any set ME_a except as required by 1–7.[7]

Montague (1973). The proper treatment of quantification in ordinary English, p. 23.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

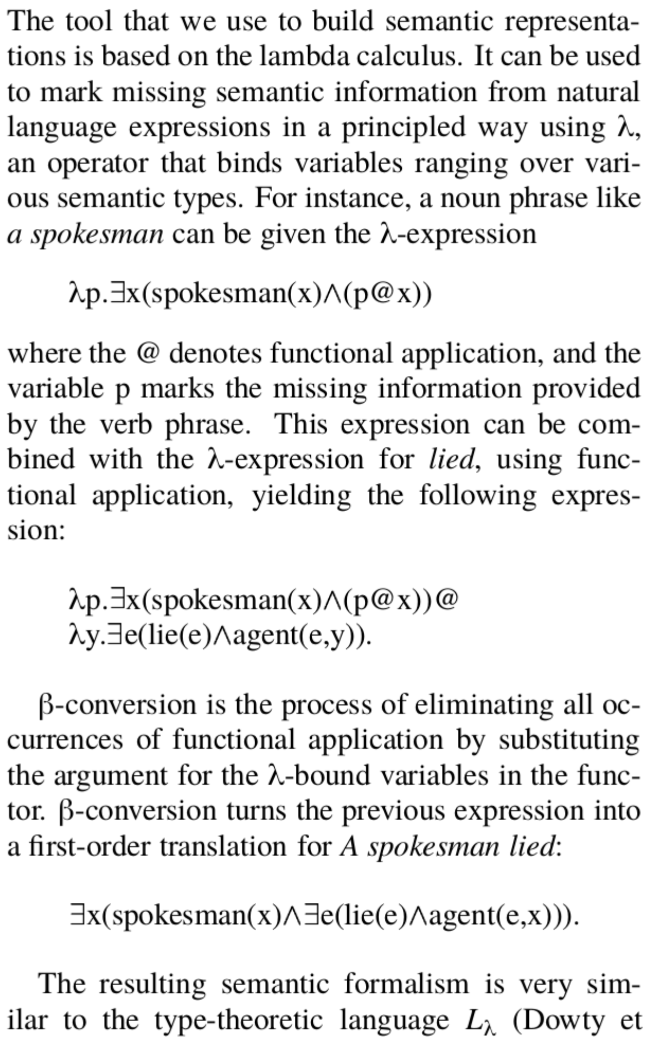
Faculty of Philosophy
General Linguistics

Beyond Compositionality?

λ-calculus in NLP

Especially in the pre-neural-net era, NLP models sometimes aimed to model **symbolic compositionality** explicitly by using, for instance, λ-calculus.

Bos et al. (2004). Wide-coverage semantic representations from a CCG parser.



The tool that we use to build semantic representations is based on the lambda calculus. It can be used to mark missing semantic information from natural language expressions in a principled way using λ, an operator that binds variables ranging over various semantic types. For instance, a noun phrase like *a spokesman* can be given the λ-expression

$$\lambda p.\exists x(\mathrm{spokesman}(x)\wedge(p@x))$$

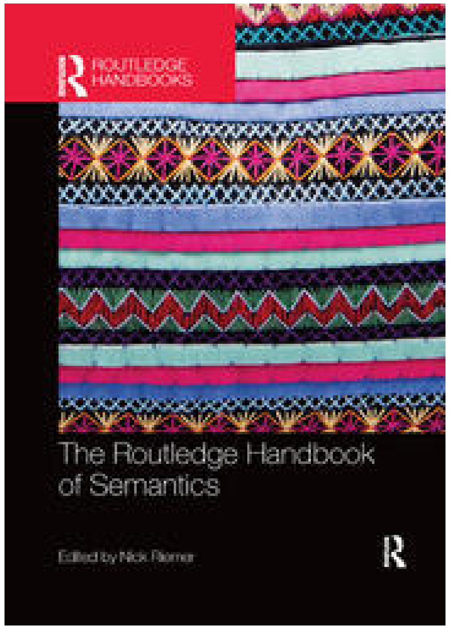
where the @ denotes functional application, and the variable p marks the missing information provided by the verb phrase. This expression can be combined with the λ-expression for *lied*, using functional application, yielding the following expression:

$$\lambda p.\exists x(\mathrm{spokesman}(x)\wedge(p@x))@ \\ \lambda y.\exists e(\mathrm{lie}(e)\wedge\mathrm{agent}(e,y)).$$

β-conversion is the process of eliminating all occurrences of functional application by substituting the argument for the λ-bound variables in the functor. β-conversion turns the previous expression into a first-order translation for *A spokesman lied*:

$$\exists x(\mathrm{spokesman}(x)\wedge\exists e(\mathrm{lie}(e)\wedge\mathrm{agent}(e,x))).$$

The resulting semantic formalism is very similar to the type-theoretic language L_λ (Dowty et



The principle of compositionality is widely acknowledged to be a foundational claim in formal semantics [...] And yet, there are many ways in which natural languages depart from formal languages [...]

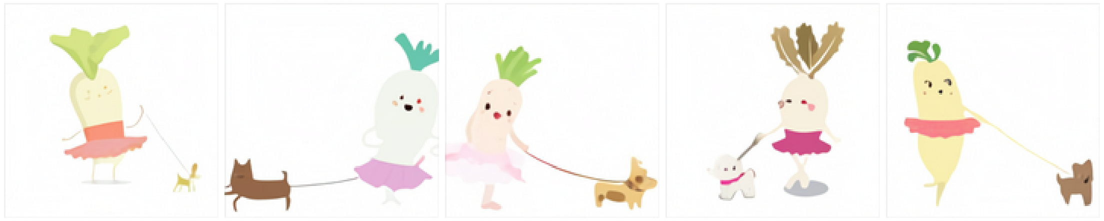
The apparently noncompositional meaning evident in idioms, discontinuous semantic units, and complex words must be addressed, and the contribution of argument structure constructions, intonation, and non-linguistic context [...] must be taken into account.

Goldberg, A. (2015). Compositionality.

Compositionality without Symbolic Composition?

TEXT PROMPT an illustration of a baby daikon radish in a tutu walking a dog

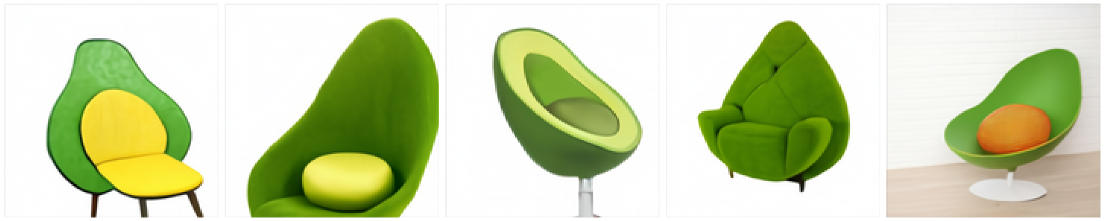
AI-GENERATED IMAGES



Edit prompt or view more images↓

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES



Edit prompt or view more images↓

https://openai.com/blog/dall-e/
https://fh295.github.io/noncompositional.html

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

References

References

Bos, J., Clark, S., Steedman, M., Curran, J. R., & Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics* (pp. 1240-1246).

Church, Alonzo (1940). A formulation of the Simple Theory of Types. *The Journal of Symbolic Logic* 5(2), 56–68.

Frege, Gottlob (1879). *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert/Halle.

Gamut, L.T.F (1991). *Logic, Language, and Meaning. Volume 1: Introduction to Logic*. Chicago: University of Chicago Press.

Gamut, L.T.F (1991). *Logic, Language, and Meaning. Volume 2: Intensional Logic and Logical Grammar*. Chicago: University of Chicago Press.

Montague, Richard (1970a). English as a formal language. *Linguaggi nella Società e nella Tecnica*, Milan, Edizioni di Comunita, 189–224.

Montague, Richard (1970b). Universal Grammar. *Theoria* 36, 373–398.

Montague, Richard (1973). The proper treatment of quantification in Ordinary English. In: J. Hintikka, J. Moravcsik, and P. Suppes (eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*. Dordrecht, D. Reidel Publishing Company, 221-242.

Peirce, Charles S. (1885). On the Algebra of Logic: A Contribution to the Philosophy of Notation, *American Journal of Mathematics* 7, 18–202.

Russell, Bertrand (1908). Mathematical Logic as based on the Theory of Types. *American Journal of Mathematics* 3, 222–262.

Thank You.

Contact:

Faculty of Philosophy
General Linguistics
Dr. Christian Bentz
SFS Wilhelmstraße 19-23, Room 1.15
chris@christianbentz.de

Office hours:
During term: Wednesdays 10-11am
Out of term: arrange via e-mail