EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Syntax & Semantics WS2019/2020

Lecture 22: Syntax & Semantics Interface

31/01/2020, Christian Bentz

Overview

Semantics Lectures

- Lecture 18: Introduction to Semantics
  Kroeger (2019). Chapters 1-2.
- Lecture 19: Word Meaning
  Kroeger (2019). Chapter 5-6.
- Lecture 20: Propositional Logic
  Kroeger (2019). Chapter 3-4; and Zimmermann & Sternefeld Chapter 7.
- Lecture 21: Predicate Logic
  Kroeger (2019). Chapter 4; and Zimmermann & Sternefeld Chapter 10 (p. 244-258).
- **Lecture 22: Syntax & Semantics Interface**
  Kearns (2011). Semantics. Second Edition. Chapter 4.; Zimmermann & Sternefeld (2013), Chapter 4.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 1: Recap of Lecture 21

“[...] fand ich ein Hindernis in der **Unzulänglichkeit der Sprache**, die bei aller entstehenden Schwerfälligkeit des Ausdruckes doch, je verwickelter die Beziehungen wurden, desto weniger die Genauigkeit erreichen liess, welche mein Zweck verlangte. Aus diesem Bedürfnisse ging der Gedanke der vorliegenden **Begriffsschrift** hervor.”

Frege (1879). Begriffsschrift: Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, p. X.

Translation: [...] I found the **inadequacy of language** to be an obstacle; no matter how unwieldy the expressions I was ready to accept, I was less and less able, as the relations became more and more complex, to attain the precision that my purpose required. This **deficiency** led me to the idea of the present **ideography**.

BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

FORMELSPRACHE

DES REINEN DENKENS.

VON

Dr. GOTTLOB FREGE,

PRIVATDOCENTEN DER MATHEMATIK AN DER UNIVERSITÄT JENA.

HALLE a/S.

VERLAG VON LOUIS NEBERT.

1879.

Logical Symbols

The following types of logical symbols are relevant for our analyses:

- **Logical operators (connectives)** equivalent to the ones defined in propositional logic: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$
- The **quantifier** symbols: $\forall$ (universal quantifier), $\exists$ (existential quantifier)
- An infinite set of variables: x, y, z, etc.[1]
- Parentheses '()' and brackets '[]'[2]

[1]This set is called *Var* in Zimmermann & Sternefeld (2013), p. 244.

[2]Beware: In the propositional logic notation, we used parentheses '()' for disambiguating the reading of a propositional logic expression as in $(p \rightarrow q) \wedge q$. However, in the predicate logic notation, parentheses can also have a different function (see below).

Logical Symbols: Quantifiers

"Standard predicate logic makes use of two quantifier symbols: the **Universal Quantifier** ∀, and the **Existential Quantifier** ∃. As the mathematical examples [below] illustrate, these quantifier symbols **must introduce a variable**, and this variable is said to be bound by the quantifier."

Kroeger (2019) Analyzing meaning, p. 69.

Examples:	Quantifier notation:
For all x it is the case that x plus x equals x times two.	$\forall$ x [x+x = 2x]
There is some y for which y plus four equals y divided by three.	$\exists$ y [y+4 = y/3]

Note: The square brackets are used here to illustrate the formulation that the quantifier scopes over.

Non-Logical Symbols: Predicates

Predicate symbols: these are typically given as upper case letters, and reflect relations between n elements, where $n \geq 0$, and $n \in \mathbb{N}$ (i.e. natural numbers). These are also called **n-ary** or **n-place predicate symbols**: $P(x)$, $P(x, y)$, $Q(x, y)$, etc.

Examples:	Predicate notation:
x snores	P(x)$\equiv$ SNORE(x)
x is honest	Q(x)$\equiv$ HONEST(x)
x sees y	R(x,y)$\equiv$ SEE(x,y)
x gives y z	S(x,y,z)$\equiv$ GIVE(x,y,z)

The single upper case letter notation is used by Zimmermann & Sternefeld (2013), the all capital notation is used by Kroeger (2019). Yet another notation involving primes (e.g. snore′was used earlier in the lecture following Müller (2019). In the following we will use the notation by Kroeger.

Non-Logical Symbols: Functions

Function symbols are different from predicates since they do not denote a relation between the variables, but they **map the variables to unique values**. Importantly, a function with $n = 0$, i.e. zero valence, is called a **constant symbol** and denotes for example an individual or object.

Examples:	Function notation:
Socrates	s
Paris	p
a crocodile	c
father of x	f(x)

Note: *s*, *j*, *p*, and *c* are *constant symbols* here, i.e. strictly speaking zero valence functions, while $f(x)$ is a monovalent function. It is important to realize that while lower case letters are used for both *constant symbols* and *variables* (i.e. *x*), they represent different elements of predicate logic. The convention here is to use the *first letter of the respective name in lower case* as a constant symbol, while variables start at *x*.

Predicates and Quantifiers

Importantly, formulating predicates which involve **quantifications** requires the usage of particular logical operators, since quantifiers require variables, and the variables then need to be further linked to predicates via logical operators.

(1) *All students are weary.* $\forall x[\text{STUDENT}(x) \rightarrow \text{WEARY}(x)]$
lit. “For all x it is the case that if x is a student, then x is weary.”

(2) *Some men snore.* $\exists x[\text{MAN}(x) \wedge \text{SNORE}(x)]$
lit. “There exists some x for which it is the case that x is a man and x snores.”[3]

(3) No crocodile is warm-blooded. $\neg\exists x[\text{CROCODILE}(x) \wedge \text{WARM-BLOODED}(x)]$
lit. “It is not the case that there is some x for which x is a crocodile and x is warm-blooded.”

[3]Note that while the plural *men* suggests that we are talking about 2 or more individuals, the predicate logic formulation is valid for 1 or more individual(s).

Multi-Valent Predicates and Quantifiers

“When a quantifier combines with another quantifier, with negation, or with various other elements [...], it can give rise to **ambiguities of scope.**”

Kroeger (2019). Analyzing meaning, p. 72.

(4) *Some man loves every woman.*

i. $\exists x[\mathrm{MAN}(x) \wedge (\forall y[\mathrm{WOMAN}(y) \rightarrow \mathrm{LOVE}(x,y)])]$
lit. “Fore some x it is the case that x is a man and [for all y it is the case that y is a woman and x loves y].”

ii. $\forall y[\mathrm{WOMAN}(y) \rightarrow (\exists x[\mathrm{MAN}(x) \wedge \mathrm{LOVE}(x,y)])]$
lit. “For all y it is the case that if y is a woman then there is an x which is a man and x loves y.”

(5) *All that glitters is not gold.*

i. $\forall x[\mathrm{GLITTER}(x) \rightarrow \neg\mathrm{GOLD}(x)]$
lit. “For all x it is the case that if x glitters then x is not gold.”

ii. $\neg\forall x[\mathrm{GLITTER}(x) \rightarrow \mathrm{GOLD}(x)]$
lit. “It is not the case for all x that if x glitters then x is gold.”

Note: In the first case the ambiguity is between whether the existential quantifier scopes over the universal quantifier, or the other way around. In the second example the ambiguity is whether the negation scopes over the universal quantifier or the other way around.

Example Denotations

Let us further assume the **denotation sets** of three predicates and three constant symbols. These denotation sets specify which individuals of U a particular expression can possibly denote.

$[\![\text{MAN}]\!]$ = {King Henry VIII, Thomas Moore}
$[\![\text{WOMAN}]\!]$ = {Anne Boleyn}
$[\![\text{SNORE}]\!]$ = {King Henry VIII}
$[\![\text{h}]\!]$ = {King Henry VIII}
$[\![\text{a}]\!]$ = {Anne Boleyn}
$[\![\text{t}]\!]$ = {Thomas Moore}

Example Model Evaluation

Based on our **example model**, consisting of the example domain and the example universal set, we can now evaluate the truth values of predicate logic expressions. One-place predicates are evaluated by whether the constant symbol is a member of the denotation set of the predicate. Logical operators are evaluated the same way as in propositional logic. Quantifiers are evaluated according to subset relations.

See Kroeger (2019). Analyzing meaning, p. 241.

English sentence	logical form	interpretation	truth value
a. *Thomas More is a man.*	MAN(t)	Thomas More $\in [\![\text{MAN}]\!]$	T
b. *Anne Boleyn is a man or a woman.*	MAN(a) $\vee$ WOMAN(a)	Anne Boleyn $\in ([\![\text{MAN}]\!] \cup [\![\text{WOMAN}]\!])$	T
c. *Henry VIII is a man who snores.*	MAN(h) $\wedge$ SNORE(h)	Henry VIII $\in ([\![\text{MAN}]\!] \cap [\![\text{SNORE}]\!])$	T
d. *All men snore.*	$\forall$x[MAN(x) $\longrightarrow$ SNORE(x)]	$[\![\text{MAN}]\!] \subseteq [\![\text{SNORE}]\!]$	F
e. *No women snore.*	$\neg\exists$x[WOMAN(x) $\wedge$ SNORE(x)]	$[\![\text{WOMAN}]\!] \cap [\![\text{SNORE}]\!] = \varnothing$	T

Section 2: Valency in Syntax and Semantics

Formal Definition: Extensions

Remember from Lecture 1 that within **denotational semantics** meaning is construed as the mapping between a given word and the real-world object it refers to (reference theory of meaning). More generally, words, phrases or sentences are said to have **extensions**, i.e. real-world situations they refer to.

Zimmermann & Sternefeld (2013), p. 71.

Type of expression	**Type of extension**	**Example**	**Extension of example**
proper name	individual	*Paul*	Paul McCartney
definite description	individual	*the biggest German city*	Berlin
noun	set of individuals	*table*	the set of tables
intransitive verb	set of individuals	*sleep*	the set of sleepers
transitive verb	set of pairs of individuals	*eat*	the set of pairs ⟨*eater*, *eaten*⟩
ditransitive verbs	set of triples of individuals	*give*	the set of triples ⟨*donator*, *recipient*, *donation*⟩

Formal Definition: Extensions

"Let us denote the **extension** of an expression *A* by putting double brackets '$[\![\,]\!]$' around *A*, as is standard in semantics. The extension of an expression depends on the **situation *s*** talked about when uttering *A* ; so we add the index *s* to the closing bracket."

Zimmermann & Sternefeld (2013), p. 85.

$[\![\text{Paul}]\!]_s = [\![\text{p}]\!]_s = \text{Paul McCartney}$[4]
$[\![\text{table}]\!]_s = [\![\text{TABLE}]\!]_s = \{\text{table}_1, \text{table}_2, \text{table}_3, \ldots, \text{table}_n\}$[5]
$[\![\text{sleep}]\!]_s = [\![\text{SLEEP}]\!]_s = \{\text{sleeper}_1, \text{sleeper}_2, \text{sleeper}_3, \ldots, \text{sleeper}_n\}$
$[\![\text{eat}]\!]_s = [\![\text{EAT}]\!]_s =$
$\{\langle\text{eater}_1, \text{eaten}_1\rangle, \langle\text{eater}_2, \text{eaten}_2\rangle, \ldots, \langle\text{eater}_n, \text{eaten}_n\rangle\}$

[4]Zimmermann & Sternefeld just put the full proper name in brackets here, Kroeger follows another convention and just put the first letter in lower case, e.g. $[\![\text{p}]\!]_s$.

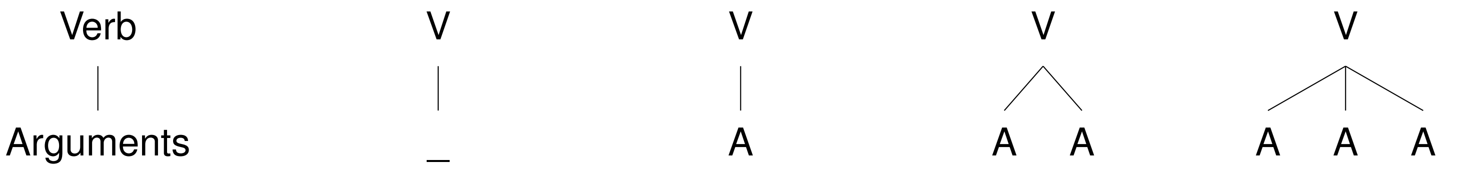
[5]Kroeger (2019) uses upper case notation for both nouns and predicates, e.g. TABLE and SLEEP respectively.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Valence according to Tesnière (Syntax Lecture 3)

“Nous avons vu qu’il y avait des verbes sans actant, des verbes à un actant, des verbes à deux actants et des verbes à trois actants.”

Tesnière (1959). Éléments de syntaxe structurale, p. 238.



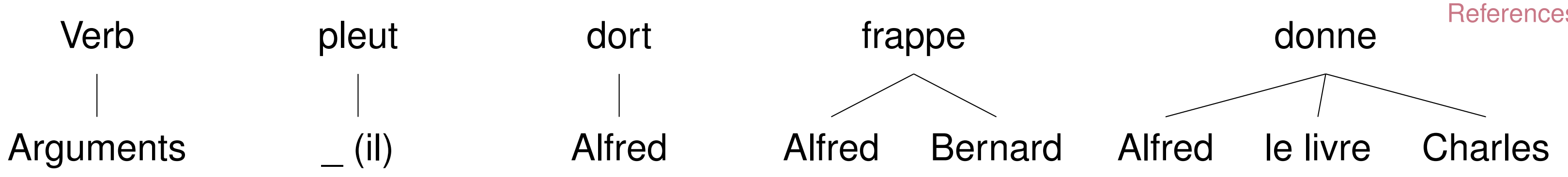
Sentence Type:	impersonal sentence	intransitive sentence	transitive sentence	ditransitive sentence
Valency:	avalent (0)	monovalent (1), one-place predicate	bivalent (2), two-place predicate	trivalent (3), three-place predicate

Note: MÃ¼ller states that the pronouns in expletives (e.g. *it rains*) should be considered obligatory arguments of the verb, while Tesnière explicitely calls them “sans actant”.

Valence according to Tesnière (Syntax Lecture 3)

“Nous avons vu qu’il y avait des verbes sans actant, des verbes à un actant, des verbes à deux actants et des verbes à trois actants.”

Tesnière (1959). Éléments de syntaxe structurale, p. 238.



Example:	*il pleut* “it rains”	*Alfred dort* “Alfred sleeps”	*Alfred frappe Bernard* “Alfred hits Bernard”	*Aflred donne le livre à Charles* “Alfred gives the book to Charles”

Valency in Semantics

"[...] one may detect an increasing complexity concerning the so-called **valency of verbs** [...] Corresponding to these types of predicates there are **three-place tuples (triples)**, **two-place tuples (pairs)** and **one-place tuples (individuals)**."

Parallelism between valency and type of extension:
The extension of an *n*-place verb is always a set of *n*-tuples.
Zimmermann & Sternefeld (2013). Introduction to semantics, p. 72.

Verb	**Valency**	**Extension**
sleep	monovalent	$[\![\text{SLEEP}]\!]_s = \{\text{sleeper}_1, \text{sleeper}_2, \ldots, \text{sleeper}_m\}$
see	bivalent	$[\![\text{SEE}]\!]_s = \{\langle\text{seer}_1, \text{seen}_1\rangle, \ldots, \langle\text{seer}_m, \text{seen}_m\rangle\}$
give	trivalent	$[\![\text{GIVE}]\!]_s = \{\langle\text{giver}_1, \text{receiver}_1, \text{given}_1\rangle, \ldots, \langle\text{giver}_m, \text{receiver}_m, \text{given}_m\rangle\}$

Note: We use *m* instead of *n* here as an index, in order to not confuse it with the *n* representing the valency.

Filling of Arguments/Gaps

As the arguments of an *n*-place verb are "filled in", the extensions change according to how many *components*[6] are in the tuples.[7]

Zimmermann & Sternefeld (2013). Introduction to semantics, p. 72.

Verb or VP	Valency	Extension
_ shows _ _	3	set of all triples $\langle a, b, c\rangle$ where a shows b c
_ shows the president _	2	set of all pairs $\langle a, c\rangle$ where a shows the president c
_ shows the president the Vatican Palace	1	set of all individuals (1-tuples) $\langle a\rangle$ where a shows the president the Vatican Palace
The Pope shows the president the Vatican Palace	0	set of all 0-tuples $\langle\rangle$ where the Pope shows the president the Vatican Palace

[6]Zimmermann & Sternefeld (2013), p. 67 point out that we speak of *components* of tuples (ordered lists), but *elements* of sets.

[7]Note: the individuals (constant symbols) are here given as *a*, *b*, and *c*. In the Kroeger (2019) notation, we would use p_1, p_2, v (the first letter of the respective name).

Interlude: 0-Valence and Truth Values

If we have a complete sentence with all arguments filled, then the verb strictly speaking has **zero valence**, and the extension of the sentence is the **set of zero-tuples**. This might seem strange at first, but note that this leads to *Frege's Generalization*, namely that the **extension of a sentence is its truth value**.

Zimmermann & Sternefeld (2013), p. 74.

S: *The Pope shows the president the Vatican Palace*.

$[\![S]\!]_s = \{\varnothing\} \equiv 1 \equiv T$, with *s* being a situation in which the Pope *actually* shows the president the Vatican Palace.

$[\![S]\!]_s = \varnothing \equiv 0 \equiv F$, with *s* being a situation in which the Pope *does not* show the president the Vatican Palace.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 3: Formal Composition

Combinatoriality in Semantics

(6)

Kim	sieh-t	ein-en	groß-en
kim	see-PRS.3SG	DET.INDF-ACC.SG	big-ACC.SG
Baum			
tree.ACC.SG			

"Kim sees a big tree"
$\exists x[\text{TREE}(x) \wedge \text{SEE}(k,x)]$

In the example above, the meaning of the overall sentence arguably derives as a *combination* of the meanings of the individiual parts.

Formal Composition

"**Compositional semantic theories** assume that the syntax and semantics work in parallel. For each *phrase structure rule* that combines two expressions into a larger phrase, there is a corresponding *semantic rule* which combines the meanings of the parts into the meaning of the newly formed expression."

Kearns (2011). Semantics, p. 57.

Semantic Types

"Linguistic expressions are classified into their **semantic types** according to the kind of denotation they have. The two most basic denotation types are **type e**, the type of **entities**, and **type t**, the type of **truth values**."

Kearns (2011). Semantics, p. 57.

Type of expression	**Type of extension**	**Semantic type**	**Example**
proper name	individual (entity)	**e**	$[\![\text{Paul}]\!]_s$=Paul McCartney
...	...	...	...
sentence	truth value	**t**	$[\![\text{Paul is happy}]\!]_s \in \{0, 1\}$

Functional Application

"[...] a function binds arguments together into a statement. From this insight, Frege proposed that all semantic composition is **functional application**. Functional application is just the combination of a function with an argument."

Kearns (2011), p. 58.

Formal Definition

"We can define the following **combinatorial rule** for [...] typed expressions:

If α is of type $\langle b, a\rangle$ and β of type b, then $\alpha(\beta)$ is of type a.

This type of combination is called **functional application**."

Müller (2019), p. 188.

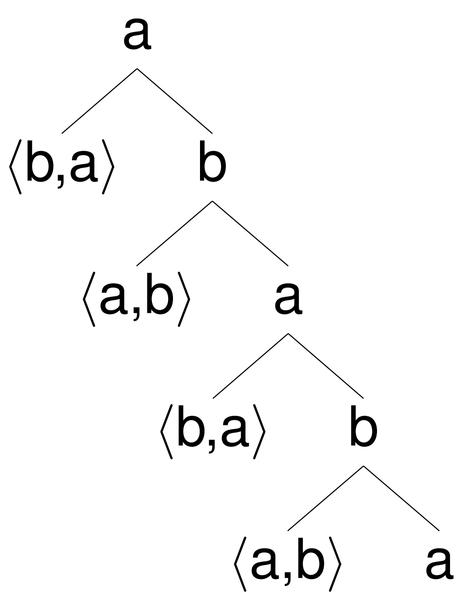
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Example: Recursive Application

$$\begin{tikzcd} & \alpha(\beta) = \text{a} \arrow[dl, no head] \arrow[dr, no head] & \\ \alpha = \langle \text{b,a} \rangle & & \beta = \text{b} \end{tikzcd}$$

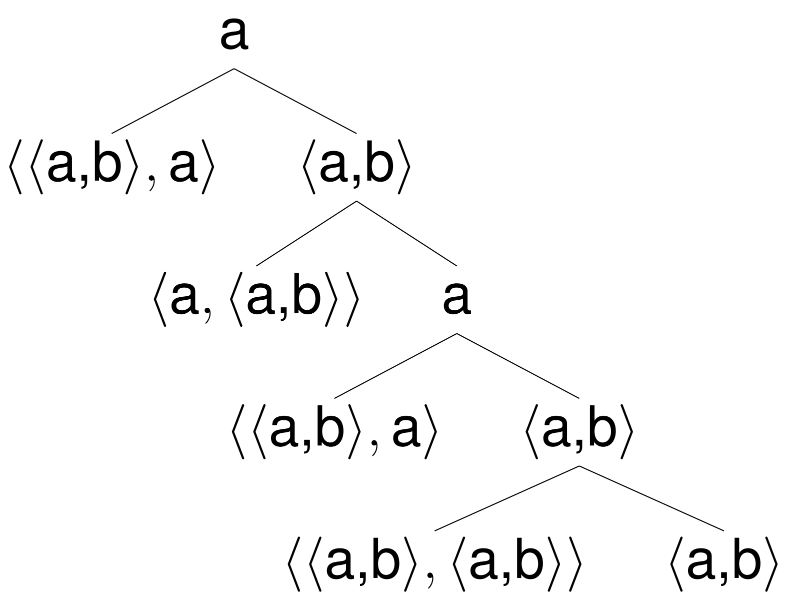
Note: The **functional application** of the component *b* to the tuple $\langle b, a \rangle$ is a mapping from *b* to *a* (this is how mathematical functions are defined, see also Kroeger (2019), p. 235 on relations and functions). For illustration, this might be thought of as an inference: the tuple expresses *if b then a*. b expresses *b is the case*, hence we get *a*. Importantly, it is always the *left component* in a tuple that is the argument, and the *right component* is the outcome *value*.

Example: Recursive Application



Note: We can apply functional application recursively add infinitum to create a **binary tree**. Binarization is a **fixed constraint in type-theoretic semantic analysis**. Note that it *a* and *b* always switch here, since it is always the left component in the tuple that is the argument.

Example: Recursive Application



Note: Binarization does **not** mean that there are only a maximum of two components in each overall tuple. Instead there can be infinitely many 2-tuple embeddings. But each individual tuple can only have two components. Hence, we can built more complex semantic types out of the two basic types *e* and *t*.

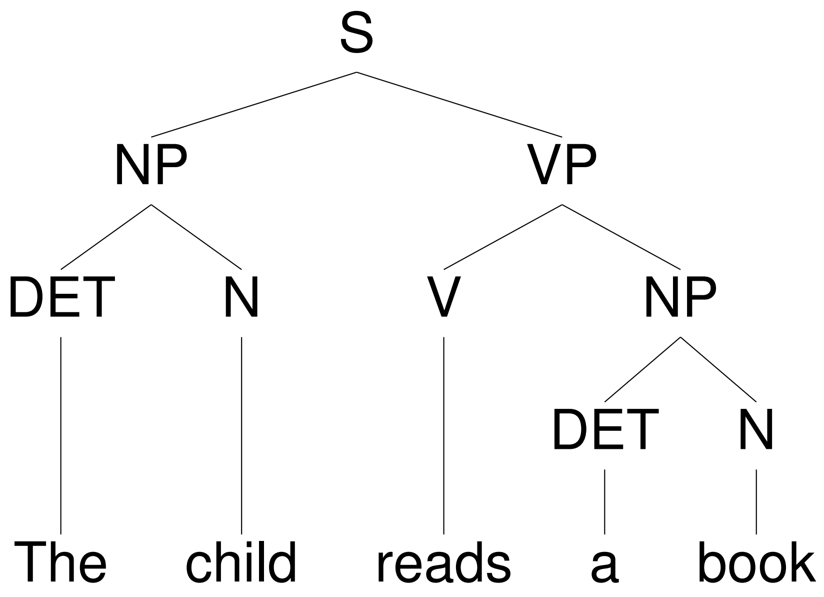
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

Section 4: Translating Syntactic into Semantic Trees

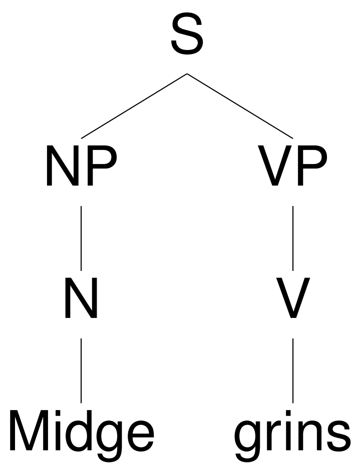
Interlude: Syntax Trees

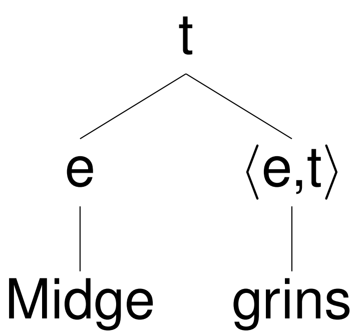
We will now translate **syntactic trees** into **type-theoretic trees** that are eventually used in semantic analyses to compose the meaning of constituents and whole sentences. Note: While often X-bar theoretic trees are used in parallel to semantic analyses, we will use simple PSG trees here for illustration (see also Kearns (2011), p. 59). Importantly, these need to be **binarized trees**.



Semantic Types: One-Place Predicates

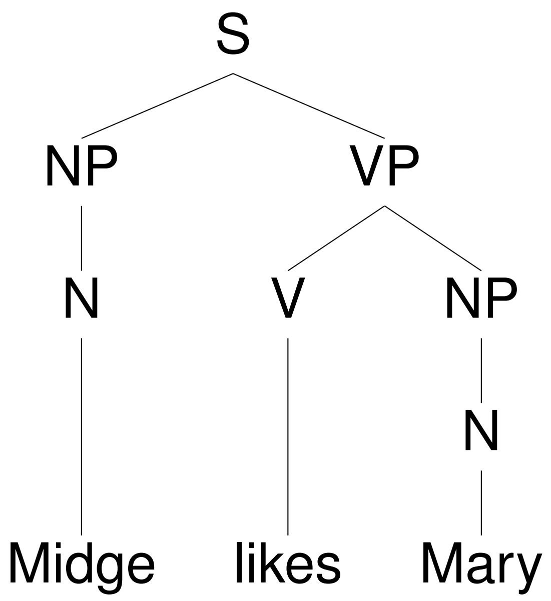
An **intransitive verb** requires **one argument** to be filled in order to form a full sentence, hence it is of the **type** ⟨**e,t**⟩. Remember that the argument is on the left side of the tuple, hence the component of type *entity* (e) is left.

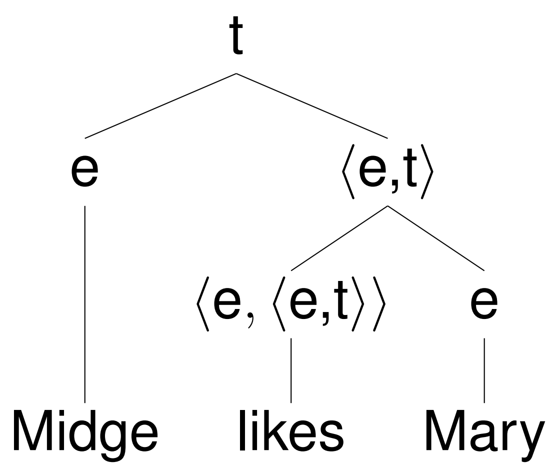




Semantic Types: Two-Place Predicates

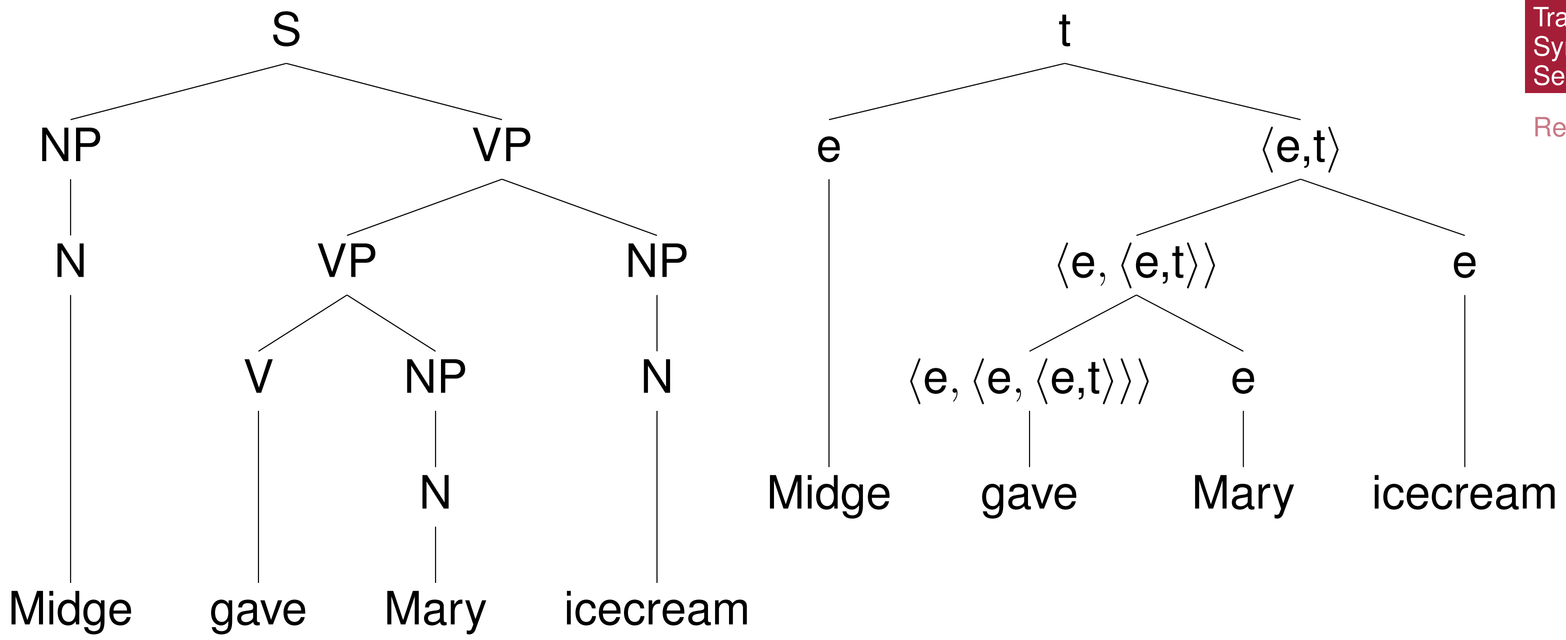
A **transitive verb** requires **two arguments** to be filled in order to form a full sentence, hence it is of the **type** ⟨**e**, ⟨**e,t**⟩⟩.





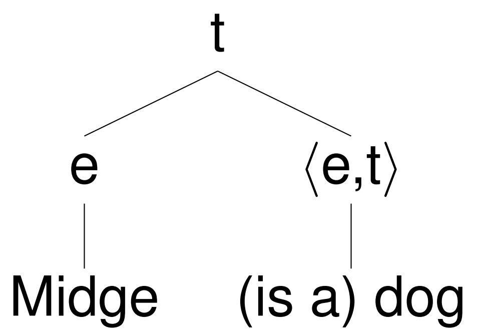
Semantic Types: Three-Place Predicates

A **ditransitive verb** requires **three arguments** to be filled in order to form a full sentence, hence it is of the **type** $\langle \mathbf{e}, \langle \mathbf{e}, \langle \mathbf{e,t} \rangle\rangle\rangle$.



Semantic Types: Nouns

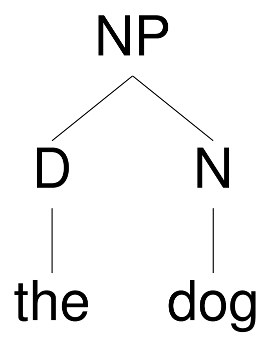
Common nouns are of type **type ⟨e,t⟩**. This might seem counterintuitive at first sight, but the idea here is that nouns are essentially like **one-place predicates**, in the sense that they require a concrete entity (e) to form a basic existential statement (with a copular) which can be true or false.

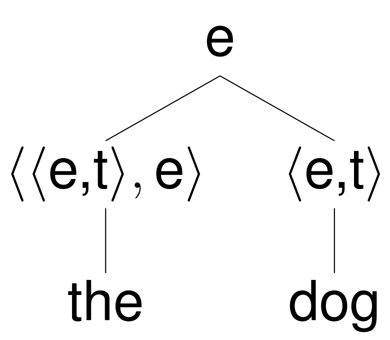


Note: This corresponds to the predicate logic formulation DOG(m), where the copular and the indefinite determiner are also dropped. As pointed out earlier in the lecture, the copular is a problematic and controversial element to analyze within syntactic theories, hence, the syntactic tree is not given here.

Semantic Types: NPs and Determiners

NPs are of **type e**, i.e. referring to a concrete entity. Note that it follows from this definition and the definition of common nouns above that **determiners** then have to be of **type** $\langle\langle e,t\rangle, e\rangle$.

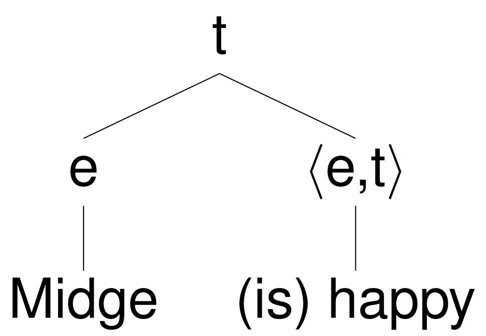




Note: This illustrates the semantic argument for why phrases consisting of nouns and determiners are considered NPs, rather than DPs. It seems clear that an NP can refer to a concrete entity (e), but if we consider the determiner to be the head of the phrase, it is not clear what the reference would be.

Semantic Types: Adjectives

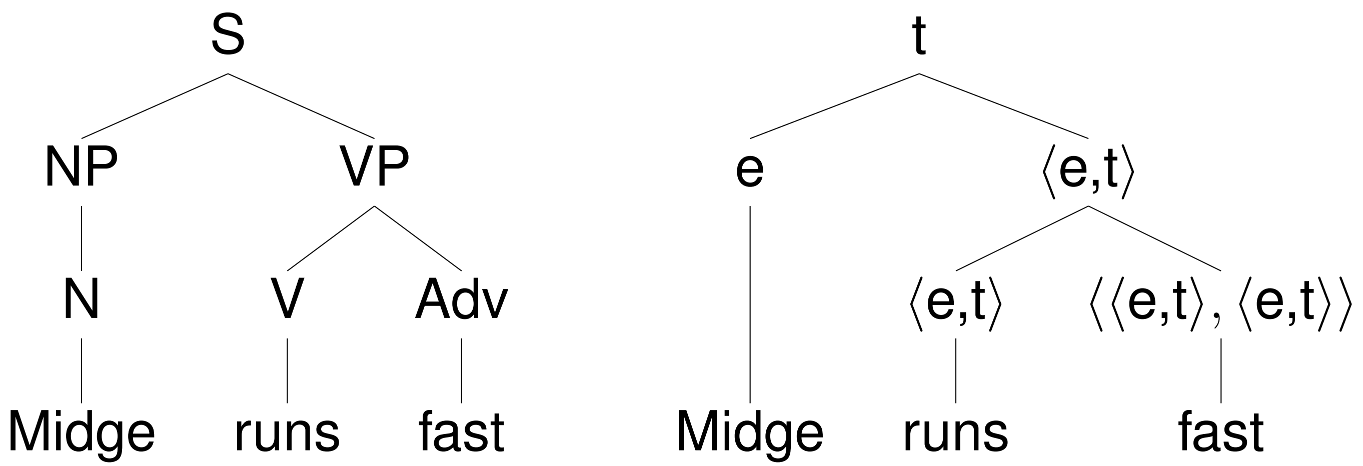
Similar to common nouns, **adjectives** are considered to be of **type ⟨e,t⟩**. The same argument applies: they require a concrete entity (e) to form a basic existential statement (with a copular) which can be true or false.



Note: We are not dealing with NPs here where the adjective modifies a noun as in *happy dog*. This can only be dealt with when we extend the analyses to Lambda calculus.

Semantic Types: Adverbs

Adverbs are considered type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. Note that similar as for determiners, this is a logical consequence of the definition of other types, i.e. the definition of a one-place predicate modified by an adverb.



Summary: Semantic Types

Type of Expression	Semantic Type
Proper names	e
Sentences	t
Nouns	$\langle e,t \rangle$
Adjectives	$\langle e,t \rangle$
One-Place Predicates	$\langle e,t \rangle$
Two-Place Predicates	$\langle e, \langle e,t \rangle \rangle$
Three-Place Predicates	$\langle e, \langle e, \langle e,t \rangle \rangle \rangle$
Determiners	$\langle \langle e,t \rangle, e \rangle$
Adverbs	$\langle \langle e,t \rangle, \langle e,t \rangle \rangle$

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Philosophy
General Linguistics

References

References

Kearns, Kate (2011). *Semantics.* Second edition. Palgrave Macmillan.

Kroeger, Paul R. (2019). *Analyzing meaning. An introduction to semantics and pragmatics.* Second corrected and slightly revised version. Berlin: Language Science Press.

Zimmermann, Thomas E. & Sternefeld, Wolfgang (2013). *Introduction to semantics. An essential guide to the composition of meaning.* Mouton de Gruyter.

Thank You.

Contact:

Faculty of Philosophy
General Linguistics
Dr. Christian Bentz
SFS Wihlemstraße 19-23, Room 1.24
chris@christianbentz.de

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