Semantics lab class (Course 2)

Lecture 4, assignment 4

Zeqi Zhao

Session 3

November 22, 2023

Our agenda today

- Recap of last session.
- Assignment 3

- Also something new:

Negation, Coordination, **semantic types, type-driven interpretation**, **λ-notation and conversion**

- Some exercise to help you with assignment 4

Recap

Our semantic theory interpret any syntactic structure with two branches in terms of

Function Application.

If α has the form $[_{S}\ \beta\ \gamma]$, then $[\![\alpha]\!] = [\![\gamma]\!]([\![\beta]\!])$.

Since i**ntransitive verbs** can be interpreted as both a function and a set, there must be some kind of relationship between sets and functions.

We use **characteristic function** to express the membership of the elements of any set S. This means, there is a ***one-on-one correspondence*** between sets and their characteristic functions.

Recap

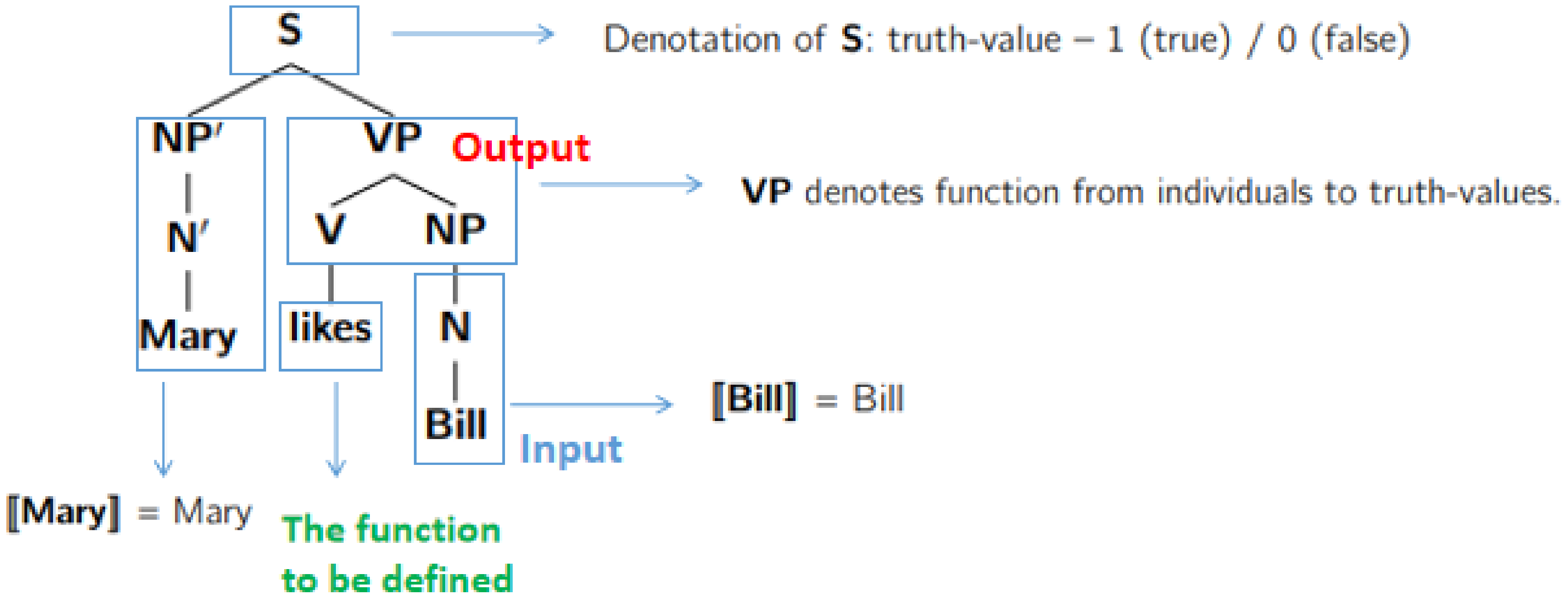
The meaning of **intransitive verbs:** Sets or characteristic functions.

This is because intransitive verbs expresses a property which may be true of some individuals but not of others in a given situation.

For **transitive verbs,** their denotation is not a set of individuals, but a set of **ordered pairs**. In other words, transitive verbs can been seen as **binary function** that takes two arguments.

Recap

However, our syntax and compositionality tell us, transitive verbs denote a unary function takes exactly one individual as argument. Therefore, we need **Schonfinkelization** to turn n-ary functions into unary functions.



Recap

Schonfinkelization/currying: Turning n-ary functions into unary functions.

N-ary

$$\begin{bmatrix} <\text{Fiona, Fiona}> \to 0 \\ <\text{Fiona, Patsy}> \to 1 \\ <\text{Fiona, Jenny}> \to 0 \\ <\text{Patsy, Fiona}> \to 0 \\ <\text{Patsy, Patsy}> \to 0 \\ <\text{Patsy, Jenny}> \to 1 \\ <\text{Jenny, Fiona}> \to 0 \\ <\text{Jenny, Patsy}> \to 0 \\ <\text{Jenny, Jenny}> \to 1 \end{bmatrix}$$

Left-to-right unary

$$\begin{bmatrix} \text{Fiona} \to \begin{bmatrix} \text{Fiona} \to 0 \\ \text{Patsy} \to 1 \\ \text{Jenny} \to 0 \end{bmatrix} \\ \text{Patsy} \to \begin{bmatrix} \text{Fiona} \to 0 \\ \text{Patsy} \to 0 \\ \text{Jenny} \to 1 \end{bmatrix} \\ \text{Jenny} \to \begin{bmatrix} \text{Fiona} \to 0 \\ \text{Patsy} \to 0 \\ \text{Jenny} \to 1 \end{bmatrix} \end{bmatrix}$$

The “likers” The ones being liked

Right-to-left unary

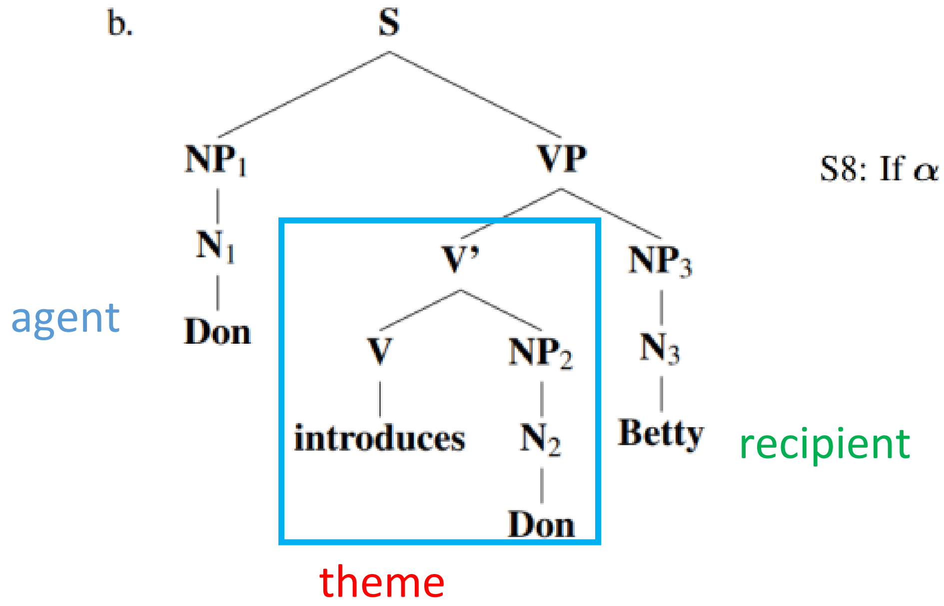
$$\begin{bmatrix} \text{Fiona} \to \begin{bmatrix} \text{Fiona} \to 0 \\ \text{Patsy} \to 0 \\ \text{Jenny} \to 0 \end{bmatrix} \\ \text{Patsy} \to \begin{bmatrix} \text{Fiona} \to 1 \\ \text{Patsy} \to 0 \\ \text{Jenny} \to 0 \end{bmatrix} \\ \text{Jenny} \to \begin{bmatrix} \text{Fiona} \to 0 \\ \text{Patsy} \to 1 \\ \text{Jenny} \to 1 \end{bmatrix} \end{bmatrix}$$

The ones being liked The “likers”

Assignment 3: Exercise 1

(1) a. **Don introduces himself to Betty.**

b.



S8: If α has the form $\overset{\textbf{V'}}{\beta \quad \gamma}$, then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.

agent theme recipient

agent theme recipient

$[\![\textbf{introduce}]\!] = f : D \to \left\{g : D \to \{h : D \to \{0, 1\}\}\right\}$

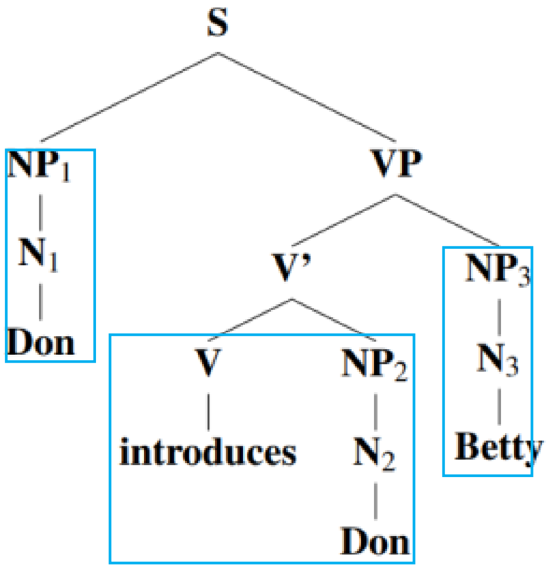
For all $x, y, z \in D$, $f(x)(y)(z) = 1$ iff z introduces x to y

S8: If α has the form $\overset{\textbf{V'}}{\beta \quad \gamma}$, then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$.

Assignment 3: Exercise 1

(1) a. **Don introduces himself to Betty.**

b.



Bottom up

$\llbracket V \rrbracket = \llbracket \textbf{introduce} \rrbracket$ (S5)

$\llbracket NP_1 \rrbracket = \llbracket N_1 \rrbracket = \llbracket \textbf{Don} \rrbracket$ (S4, S2)

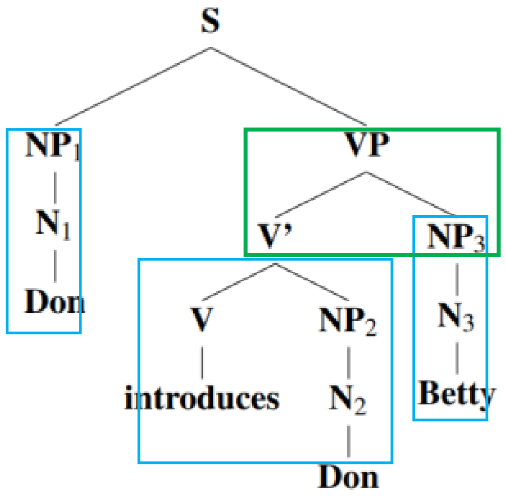
$\llbracket NP_2 \rrbracket = \llbracket N_2 \rrbracket = \llbracket \textbf{Don} \rrbracket$ (S4, S2)

$\llbracket NP_3 \rrbracket = \llbracket N_3 \rrbracket = \llbracket \textbf{Betty} \rrbracket$ (S4, S2)

Assignment 3: Exercise 1

(1) a. **Don introduces himself to Betty.**

b.



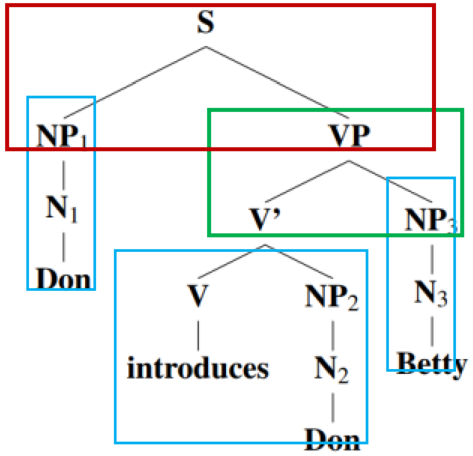
$$
\begin{array}{lr}
[\![\mathbf{V}]\!] = [\![\mathbf{introduce}]\!] & (S5) \\
[\![\mathbf{NP}_1]\!] = [\![\mathbf{N}_1]\!] = [\![\mathbf{Don}]\!] & (S4, S2) \\
[\![\mathbf{NP}_2]\!] = [\![\mathbf{N}_2]\!] = [\![\mathbf{Don}]\!] & (S4, S2) \\
[\![\mathbf{NP}_3]\!] = [\![\mathbf{N}_3]\!] = [\![\mathbf{Betty}]\!] & (S4, S2)
\end{array}
$$

$$
\begin{array}{lr}
[\![\mathbf{V'}]\!] = [\![\mathbf{V}]\!]([\![\mathbf{NP}_2]\!]) & (S8) \\
\quad = [\![\mathbf{introduce}]\!]([\![\mathbf{Don}]\!]) & ([\![\mathbf{V}]\!], [\![\mathbf{NP}_2]\!]) \\
[\![\mathbf{VP}]\!] = [\![\mathbf{V'}]\!]([\![\mathbf{NP}_3]\!]) & (S7) \\
\quad = [\![\mathbf{introduce}]\!]([\![\mathbf{Don}]\!])([\![\mathbf{Betty}]\!]) & ([\![\mathbf{V'}]\!], [\![\mathbf{NP}_3]\!])
\end{array}
$$

Assignment 3: Exercise 1

(1) a. **Don introduces himself to Betty.**

b.



$$\llbracket \textbf{V} \rrbracket = \llbracket \textbf{introduce} \rrbracket \qquad \text{(S5)}$$
$$\llbracket \textbf{NP}_1 \rrbracket = \llbracket \textbf{N}_1 \rrbracket = \llbracket \textbf{Don} \rrbracket \qquad \text{(S4, S2)}$$
$$\llbracket \textbf{NP}_2 \rrbracket = \llbracket \textbf{N}_2 \rrbracket = \llbracket \textbf{Don} \rrbracket \qquad \text{(S4, S2)}$$
$$\llbracket \textbf{NP}_3 \rrbracket = \llbracket \textbf{N}_3 \rrbracket = \llbracket \textbf{Betty} \rrbracket \qquad \text{(S4, S2)}$$

$$\begin{aligned}
\llbracket \textbf{V'} \rrbracket &= \llbracket \textbf{V} \rrbracket(\llbracket \textbf{NP}_2 \rrbracket) && \text{(S8)}\\
&= \llbracket \textbf{introduce} \rrbracket(\llbracket \textbf{Don} \rrbracket) && (\llbracket \textbf{V} \rrbracket, \llbracket \textbf{NP}_2 \rrbracket)\\
\llbracket \textbf{VP} \rrbracket &= \llbracket \textbf{V'} \rrbracket(\llbracket \textbf{NP}_3 \rrbracket) && \text{(S7)}\\
&= \llbracket \textbf{introduce} \rrbracket(\llbracket \textbf{Don} \rrbracket)(\llbracket \textbf{Betty} \rrbracket) && (\llbracket \textbf{V'} \rrbracket, \llbracket \textbf{NP}_3 \rrbracket)
\end{aligned}$$

$$\begin{aligned}
\llbracket \textbf{S} \rrbracket &= \llbracket \textbf{VP} \rrbracket(\llbracket \textbf{NP}_1 \rrbracket) && \text{(S1)}\\
&= \llbracket \textbf{introduce} \rrbracket(\llbracket \textbf{Don} \rrbracket)(\llbracket \textbf{Betty} \rrbracket)(\llbracket \textbf{Don} \rrbracket) && (\llbracket \textbf{VP} \rrbracket, \llbracket \textbf{NP}_1 \rrbracket)\\
&= \left[\begin{array}{l} f : D \to \left\{ g : D \to \{ h : D \to \{0,1\} \} \right\} \\ \quad \text{For all } x, y, z \in D,\ f(x)(y)(z) = 1 \text{ iff } z \text{ introduces } x \text{ to } y \end{array} \right] (\text{Don})(\text{Betty})(\text{Don}) && \\
& && \text{(lexicon)}\\
&= \left[\begin{array}{l} g : D \to \{ h : D \to \{0,1\} \} \\ \quad \text{For all } y, z \in D,\ f(y)(z) = 1 \text{ iff } z \text{ introduces Don to } y \end{array} \right] (\text{Betty})(\text{Don}) && \\
&= \left[\begin{array}{l} h : D \to \{0,1\} \\ \quad \text{For all } z \in D,\ f(z) = 1 \text{ iff } z \text{ introduces Don to Betty} \end{array} \right] (\text{Don}) && \\
&= 1 \text{ iff Don introduces Don to Betty} &&
\end{aligned}$$

Assignment 3: Exercise 2

Exercise 2 Assume $D = \{\text{Don, Betty}\}$. The ternary relation $R_{introduce}$ is defined as in (2). The first ordered triple in (2) says that Don introduces himself to Betty, the second one that Don introduces Betty to herself, the third one that Betty introduces herself to Don.

(2) $R_{introduce} = \{\langle \text{Don, Don, Betty} \rangle, \langle \text{Don, Betty, Betty} \rangle, \langle \text{Betty, Betty, Don} \rangle\}$

agent theme recipient

The characteristic function of $R_{introduce}$ is:

$$f_{introduce} := \left[\begin{array}{lcl} \langle \text{Don, Don, Don} \rangle & \rightarrow & 0 \\ \langle \text{Don, Don, Betty} \rangle & \rightarrow & 1 \\ \langle \text{Don, Betty, Don} \rangle & \rightarrow & 0 \\ \langle \text{Don, Betty, Betty} \rangle & \rightarrow & 1 \\ \langle \text{Betty, Don, Don} \rangle & \rightarrow & 0 \\ \langle \text{Betty, Don, Betty} \rangle & \rightarrow & 0 \\ \langle \text{Betty, Betty, Don} \rangle & \rightarrow & 1 \\ \langle \text{Betty, Betty, Betty} \rangle & \rightarrow & 0 \end{array} \right]$$

agent theme recipient

Assignment 3: Exercise 2

(i) Is $[\![\textbf{S}]\!]$ true in the situation defined by $R_{introduce}$?

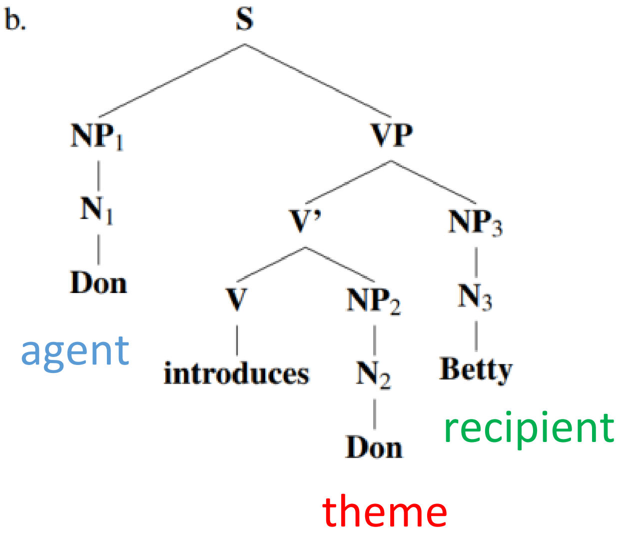
The characteristic function of $R_{introduce}$ is:

$$f_{introduce} := \left[\begin{array}{lcc} \langle \text{Don, Don, Don} \rangle & \rightarrow & 0 \\ \langle \text{Don, Don, Betty} \rangle & \rightarrow & 1 \\ \langle \text{Don, Betty, Don} \rangle & \rightarrow & 0 \\ \langle \text{Don, Betty, Betty} \rangle & \rightarrow & 1 \\ \langle \text{Betty, Don, Don} \rangle & \rightarrow & 0 \\ \langle \text{Betty, Don, Betty} \rangle & \rightarrow & 0 \\ \langle \text{Betty, Betty, Don} \rangle & \rightarrow & 1 \\ \langle \text{Betty, Betty, Betty} \rangle & \rightarrow & 0 \end{array}\right]$$

agent theme recipient

(1) a. **Don introduces himself to Betty.**

b.



Assignment 3: Exercise 2

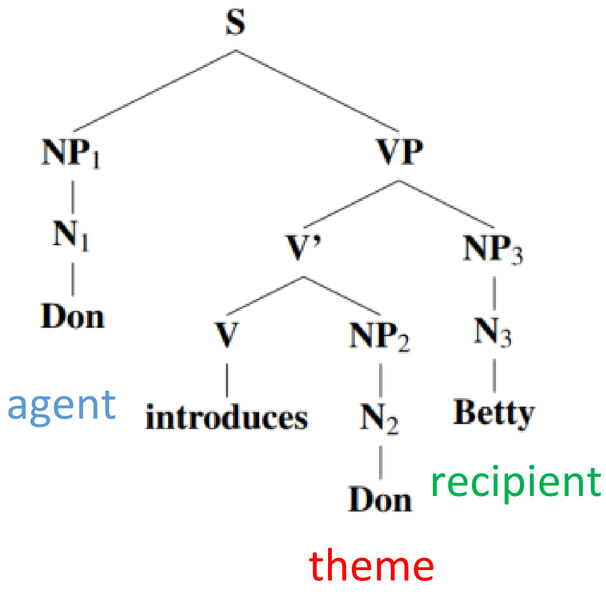
(iii) Which schönfinkelization

$$[\![\textbf{introduce}]\!] = \begin{bmatrix} \text{Don} \to \begin{bmatrix} \text{Don} \to \begin{bmatrix} \text{Don} \to 0 \\ \text{Betty} \to 0 \end{bmatrix} \\ \text{Betty} \to \begin{bmatrix} \text{Don} \to 1 \\ \text{Betty} \to 0 \end{bmatrix} \end{bmatrix} \\ \text{Betty} \to \begin{bmatrix} \text{Don} \to \begin{bmatrix} \text{Don} \to 0 \\ \text{Betty} \to 1 \end{bmatrix} \\ \text{Betty} \to \begin{bmatrix} \text{Don} \to 1 \\ \text{Betty} \to 0 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

theme recipient agent

(1) a. **Don introduces himself to Betty.**

b.



Left to right schönfinkelization:

agent theme recipient

Right to left schönfinkelization: DO

recipient theme agent

Any questions about assignment 3?

What kind of denotation type? Let's make a list

Sentences: the set of truth-values {0,1}

Proper names *Chomsky*: individuals.

e-type/referential NPs *The cat*: Individuals.

Common nouns *cat*: Functions from D to {0, 1}.

Intransitive verbs *smokes*: Functions from D to {0, 1}.

Predicative ADJ *green*: Functions from D to {0, 1}.

Transitive verbs *loves*: Functions from D to functions from D to {0, 1}.

What about negation?

The logical negation operator: **¬ 'not'**

If [[John smokes]]=1, then [[It is not the case that John smokes]]=0.

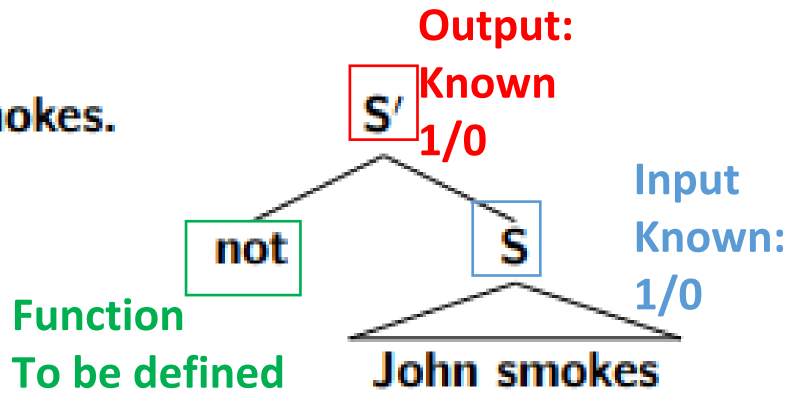
If [[John smokes]]=0, then [[It is not the case that John smokes]]=1.

It seems that any proposition *p* must have **the opposite truth value** from its negation ***¬p*** *(read* 'not *p*').

Neg as function

Function application: Using function as a tool to interpret any syntactic structure with two branches.

(1) It is not the case that John smokes.



[[not]]= $\begin{bmatrix} 1 \rightarrow 0 \\ 0 \rightarrow 1 \end{bmatrix} = f : \{0,1\} \rightarrow \{0,1\}$

For all $x \in \{0,1\}$, $f(x) = 1$ iff $x = 0$

What kind of denotation type does [[not]] have?

Expanding our list...

Sentences: the set of truth-values {0,1}

Proper names *Chomsky*: individuals.

e-type/referential NPs *The cat*: Individuals.

Common nouns *cat*: Functions from D to {0, 1}.

Intransitive verbs *smokes*: Functions from D to {0, 1}.

Predicative ADJ *green*: Functions from D to {0, 1}.

Transitive verbs *loves*: Functions from D to functions from D to {0, 1}.

Negation operators ***not* '¬': Function from {0, 1} to {0, 1}.**

What about connectives?

There are lots of connectives:

and, or, but, because.......

Our concern: The **logical operators** which are **truth-functional,** like ***and, or, not.***

(2) a. I'm a student *and* I love semantics.

b. I'm a student *but* I love semantics.

c. I'm a student *or* I love semantics.

d. It is not the case that I'm a student *and* I love semantics.

(2a) and (2b) have the same **truth-conditional meaning**, but not (2c) and (2d).

Connectives *and/or*

The logical connectives: and, or

p= I'm a student   q= I love semantics

and $p \wedge q$ read ‘p and q’

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

(Inclusive) or $p \vee q$ read ‘p or q'

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Two kinds of *'or':* Exclusive/inclusive (extra)

In spoken English we often use *or* to mean 'either … or … but not both'.

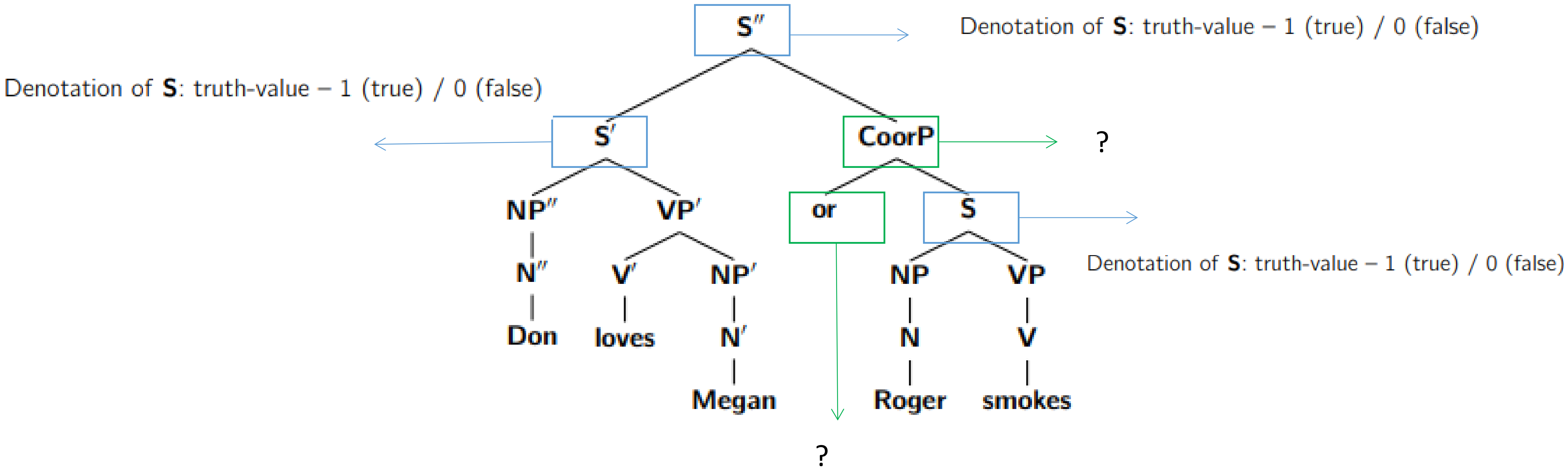
p= Would you like tee? q= Would you like coffe?

Exclusive or p XOR q

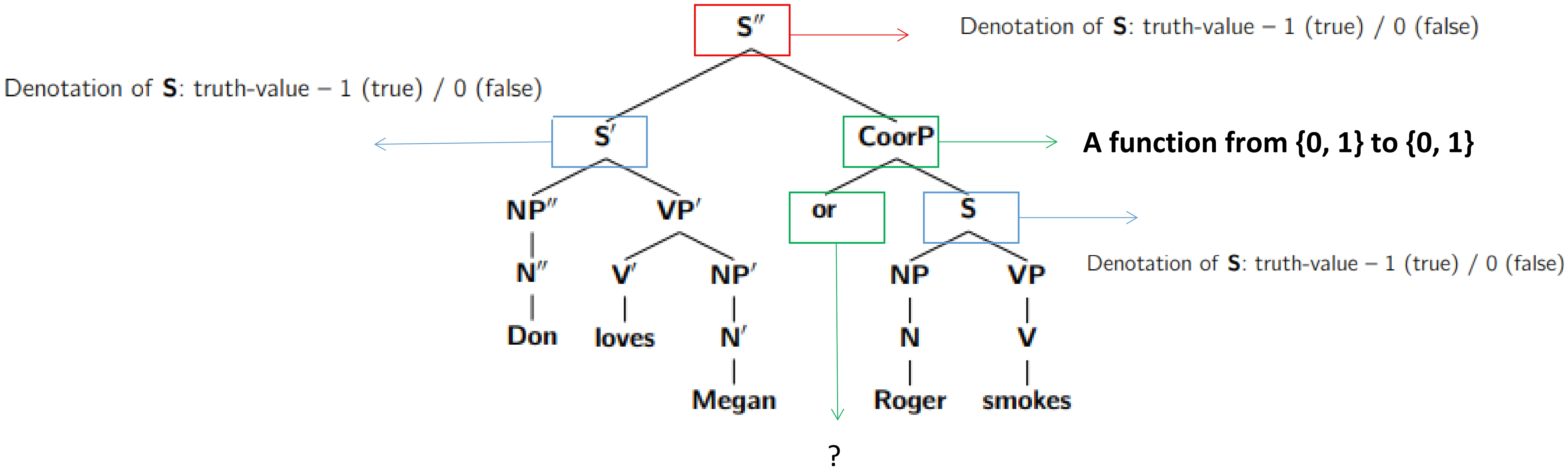
p	q	p XOR q
T	T	F
T	F	T
F	T	T
F	F	F

<u>For this course, we only use "or" in a **inclusive** sense.</u>

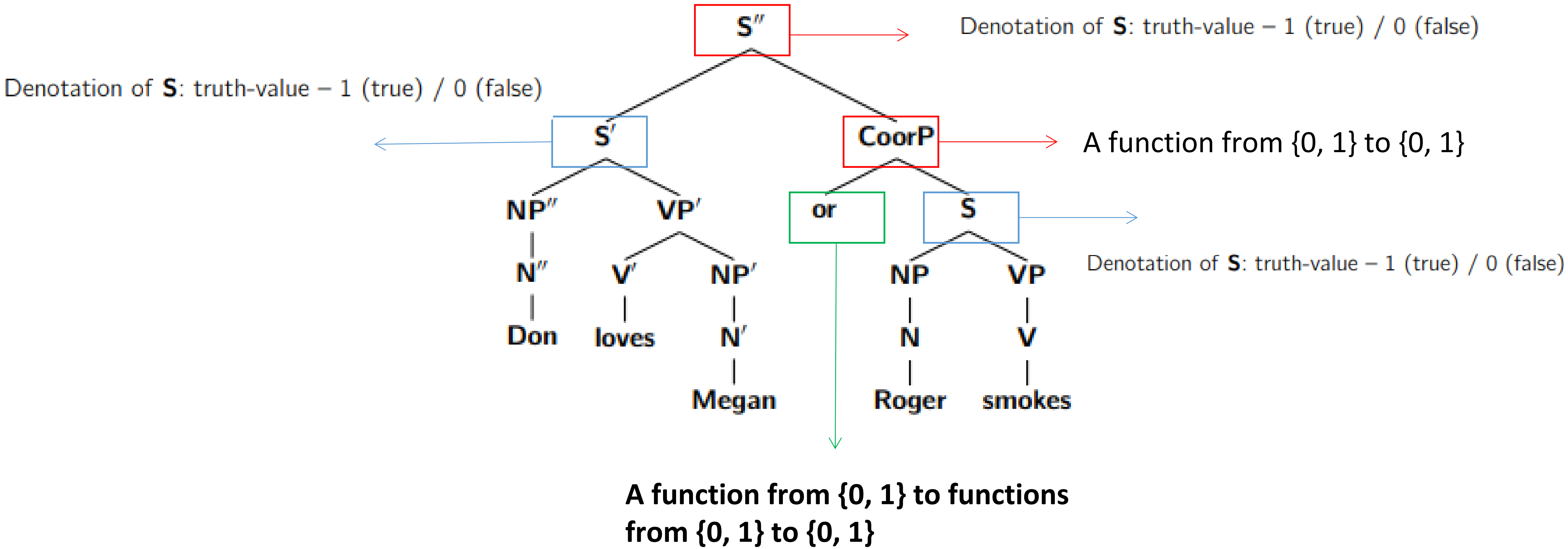
What kind of denotation type?



Filling in the blanks: The CoordinationP



Filling in the blanks: The Connective *or*



Expanding our list...

Sentences: the set of truth-values {0,1}

Proper names *Chomsky*: individuals.

e-type/referential NPs *the cat*: Individuals.

Common nouns *cat*: Functions from D to {0, 1}.

Intransitive verbs *smokes*: Functions from D to {0, 1}.

Predicative ADJ *green*: Functions from D to {0, 1}.

Transitive verbs *loves*: Functions from D to functions from D to {0, 1}.

Negation operators ***not*** **'¬': Function from {0, 1} to {0, 1}.**

Connectives ***or/and*****: Functions from {0, 1} to functions from {0, 1} to {0, 1}**

CoorP: Function from {0, 1} to {0, 1}.

Categorizing denotation types

Our list is getting longer and more complex. Is there a way to express the denotation types more straightforwardly?

Syntactic Category	Denotation types
S	Truth values {0, 1}
Proper name, e-type/referential NP	Individual
Common noun, $V_{intransitive}$, Predicative ADJ	Functions from D to {0, 1} (**characteristic function**)
$V_{transitive}$	Functions from D to functions from D to {0, 1}.
Neg, CoorP	Function from {0, 1} to {0, 1}
or/and	Functions from {0, 1} to functions from {0, 1} to {0, 1}

Semantic types and denotation domains

A more straightforward way: Define two basic types. With the help of function, we can recursively define complex types.

e ('entity') stands for individual
t stands for truth-value

Basic types

$\langle \sigma, \tau \rangle$ stands for 'function from type σ to type τ'

Functional types

We can also use this type notation for the types of domains:

a. $D_t = \{1, 0\}$

b. $D_e = \{x : x \text{ is an entity}\}$

c. $D_{\langle \alpha, \beta \rangle} = \{f \mid f : D_\alpha \to D_\beta\}$ (functions from things of type α to things of type β)

Inventory of semantic types

Syntactic Category	Denotation types	Semantic type	
S	Truth values {0, 1}	**t**	Basic types
Proper name, e-type/referential NP	Individual	**e**	
Common noun, $V_{intransitive}$, Predicative ADJ	Functions from D_e to D_t {0, 1} (**characteristic function**)	<e,t>	Functional types
$V_{transitive}$	Functions from D_e to functions from D_e to {0, 1}.	<e, <e,t>>	
Neg, CoorP	Function from {0, 1} to {0, 1}	<t,t>	
or/and	Functions from {0, 1} to functions from {0, 1} to {0, 1}	<t,<t,t>>	

Exercise 8: Semantic types

(3) Specify the semantic types.

a. [[John Lennon]]

b. [[the girl who stole my bike]]

c. [[a student in Göttingen]]

d. [[I love semantics]]

e. [[and]]

f. [[and]] ([[I love semantics]])

f. [[not]]

g. [[happy]]

h. [[dances]]

i. [[love]]

j. [[love]] ([[John]])

k. [[love]] ([[John]]) ([[Mary]])

l. [[not]] ([[I love semantics]])

Tips for specifying semantic types

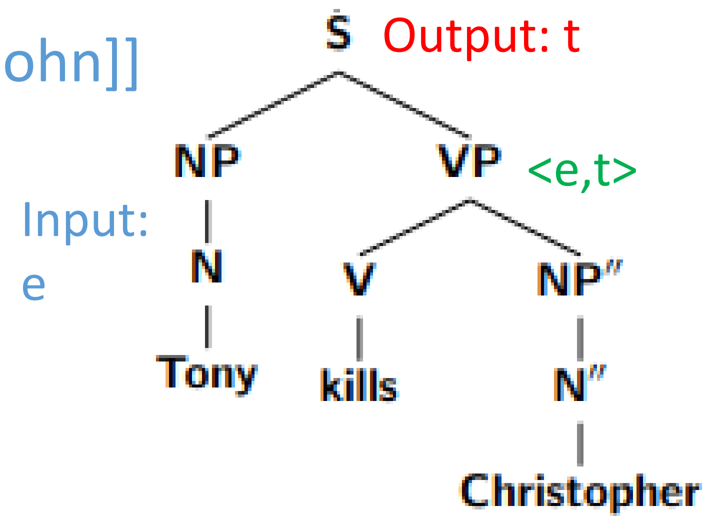
Tip 1: By definition, **all characteristic functions are type <e,t>.**

a. **Common noun**: cat, a student, a member of the Beatles, a smart linguist

b. **V**$_{\textbf{intransitive}}$: smokes, dances, runs, lives in New York

c. **Predicative ADJ**: happy, green

d. **[[V**$_{\textbf{transitive}}$**P]]**: [[likes]] ([[John]])

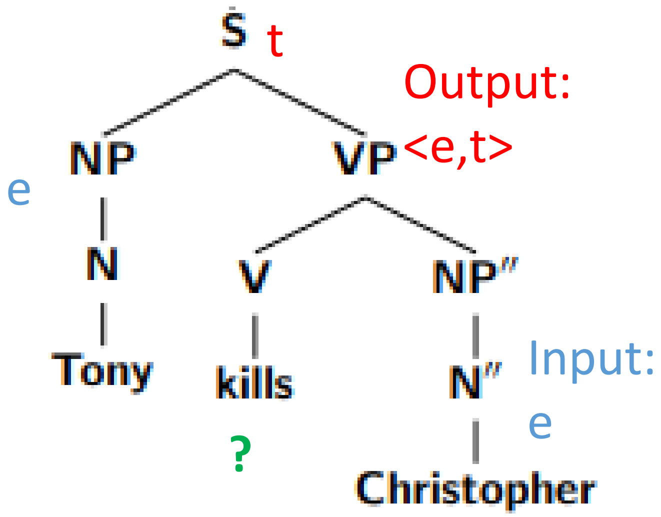


Tips for specifying semantic types

Tip 2: No need to memorize our list. If you don't remember a certain type, just mark the nodes you do know and then calculate the unknown.

The only thing you need to kow:

- e for individuals, in D_e
- t for truth values, in $\{1, 0\}$



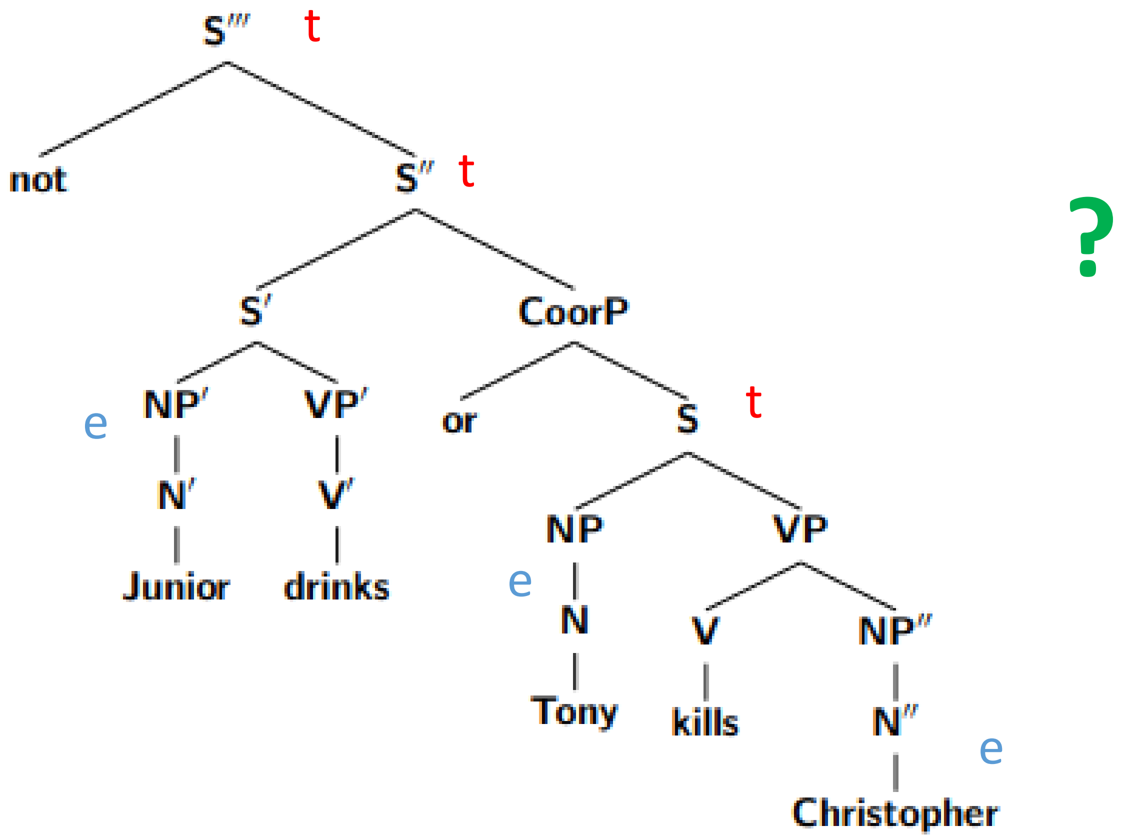
[[kill]] denotes a function from D_e to $D_{<e,t>}$

Therefore, the semantic type of [[kill]] is <e,<e,t>>

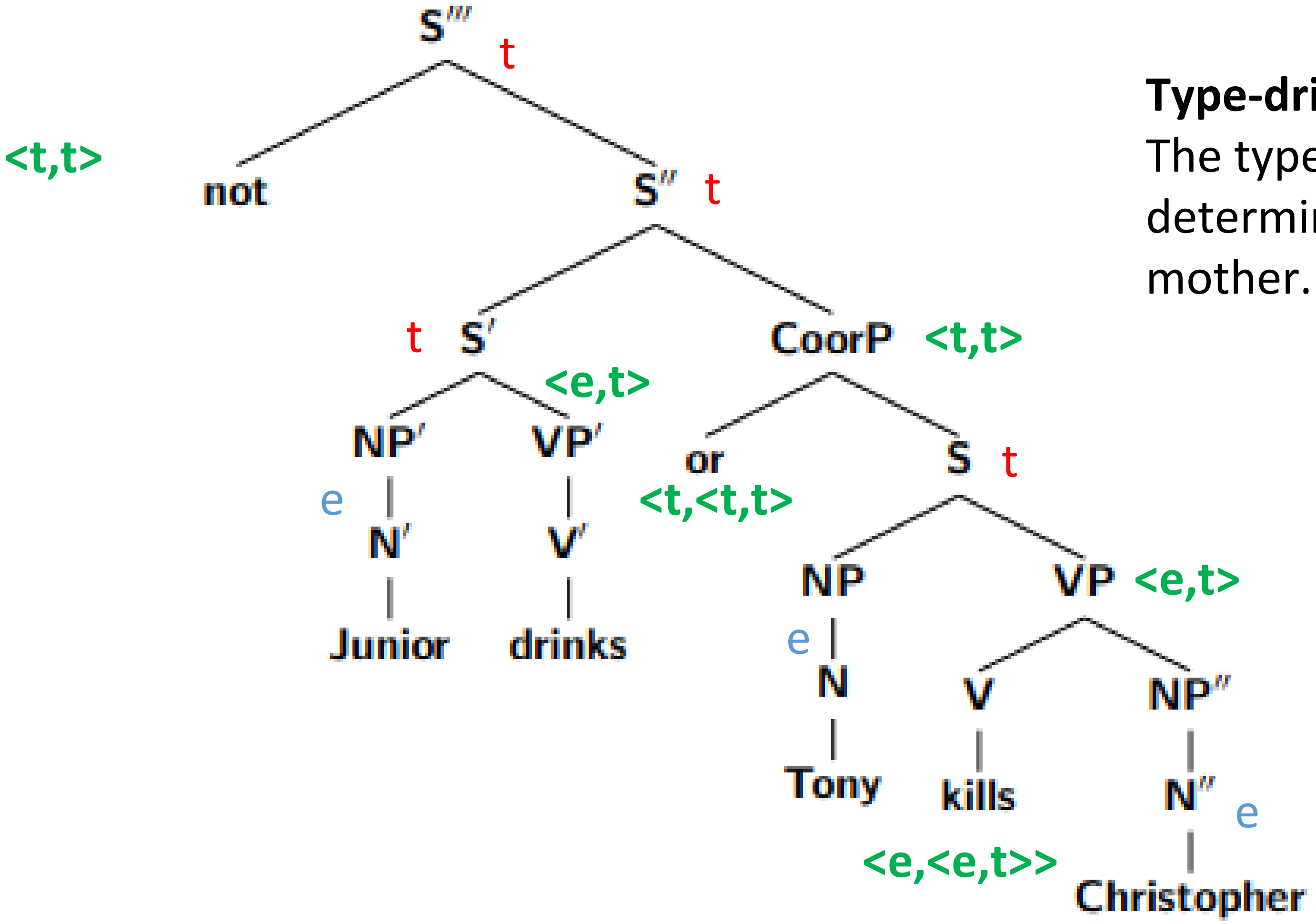
One more example

Situation: On the day of the exam, you realized that these are the only semantic types you can remember:

– e for individuals, in D_e
– t for truth values, in $\{1, 0\}$



No need to panic. Just fill in the blanks



Type-driven interpretation:
The types of the daughter nodes determine the type of their mother.

Solutions: Excersise 8

(3) a. [[John Lennon]] e

b. [[the girl who stole my bike]] e

c. [[a student in Göttingen]] <e,t>

d. [[I love semantics]] t

e. [[and]] <t,<t,t>>

f. [[and]] ([[I love semantics]]) <t,t>

f. [[not]] <t,t>

g. [[happy]] <e,t>

h. [[dances]] <e,t>

i. [[love]] <e,<e,t>>

j. [[love]] ([[John]]) <e,t>

k. [[love]] ([[John]]) ([[Mary]]) t

l. [[not]] ([[I love semantics]]) t

How we used to define functions

$$[\![\textbf{loves}]\!] = f : D \rightarrow \{g : g \text{ is a function from } D \text{ to } \{0, 1\}\}$$
$$\text{For all } x, y \in D, f(x)(y) = 1 \text{ iff } y \text{ loves } x$$

Too much writing! And this is only one simple function.

To much when writing derivations.

How we used to define functions

With our new type notation, we can simplify the mapping of function as in (4):

(4) $[\![\textbf{loves}]\!] = \boxed{f : D_e \rightarrow D_{\langle e,t \rangle}}$

For all $x, y \in D_e$, $f(x)(y) = 1$ iff y loves x

Still too much writing!

Lambda notation

λ-notation is very convenient for defining functions.

General format: $[\lambda\alpha : \phi \, . \, \gamma]$

α = argument variable (x for arbitrary objects in the following)

ϕ = domain condition (introduces condition on x)

γ = value description (specifies the value of the function for x)

Algebraic notation: $f(x) = x+1$

Set theoretic notation: $F := \{\langle x, x+1 \rangle : x \in N\}$

λ-notation: $F := \lambda x{:}\, x \in N.\ x+1$ or $\boldsymbol{F := \lambda x \in D_N.}\ x+1$

<u>We name the function as:</u> 'The function which maps every x such that x is a natural number to x+1.'

What about natural language?

F **:=** $\boldsymbol{\lambda x \in D_N \,.\, x+1}$

Read as: 'The function which maps every x such that x is a natural number to x+1.'

<u>Reading convention:</u>

‘the function which maps every variable such that domain to value.’

Problem with semantic function:

[[smoke]] = $\lambda x \in D_e$. x smokes

This is a function of type <e,t>.

Wrong reading: ‘The function which maps every x such that x ∊ D to x smokes.’

Our semantic theory is truth-value based

Syntactic Category	Denotation types	Semantic type
S	Truth values {0, 1}	**t**
Proper name, e-type/referential NP	Individual	**e**
Common noun, V_i, Predicative ADJ	Functions from D to {0, 1} (**characteristic function**)	<e,t>
V_t	Functions from D to functions from D to {0, 1}.	<e, <e,t>>
Neg, CoorP	Function from {0, 1} to {0, 1}	<t,t>
or/and	Functions from {0, 1} to functions from {0, 1} to {0, 1}	<t,<t,t>>

Two reading conventions for lambda-terms

$$[\lambda\alpha : \phi \,.\, \gamma]$$

α = argument variable (x for arbitrary objects in the following)

ϕ = domain condition (introduces condition on x)

γ = value description (specifies the value of the function for x)

For functions with D_t {0,1} as their range: Reading 1
'the function which maps every variable such that domain to 1 if truth conditions, and to 0 otherwise.'

For other functions: Reading 2
'the function which maps every variable such that domain to value.'

Examples: How do we name functions?

- F_1=: $\lambda x \in D_N$. $x + 4$ Domain: N Range: N

Read: 'The function which maps every x such that x is a natural number to x +4.'

- F_2=: $\lambda x \in D_e$. the age of x Range: {x: x is the age of an individual}

Read: 'The function which maps every x such that x is a individual to the age of x.'

Examples: How do we name functions?

- F_3=: $\lambda x : x \in D_e$. x smokes   Range: {0,1}

Read: 'The (smallest) function which maps every x such that x is a individual to 1 iff x smokes, and to 0 otherwise.'

- F_3=: $\lambda x \in D_e$. [$\lambda y \in D_e$. y kisses x]   <e,<e, t>>   Range: {g : g is a function from D to {0, 1}}

Read: 'The function which maps every x such that x is an individual to the function which maps every y such that y is an individual to 1 iff y kisses x , and to 0 otherwise.'

Exercise 9: Name the functions

(5) a. $\lambda x \in D_e$. the mother of x

b. $\lambda x \in D_e$. x

c. $\lambda x \in D_e$. Chomsky

d. $\lambda x \in D_e$. [$\lambda y \in D_e$. y saw x]

e. $\lambda X \subseteq D_e$. [$\lambda y \in D_e$. $y \notin x$]

f. $\lambda x \in D_e$. [$\lambda y \in D_e$. [$\lambda z \in D_e$. z introduced y to x]

A tip for reading the functions

(5) a. $\lambda x \in D_e$. the mother of x

b. $\lambda x \in D_e$. x

c. $\lambda x \in D_e$. Chomsky

d. $\lambda x \in D_e$. [$\lambda y \in D_e$. y saw x]

e. $\lambda X \subseteq D_e$. [$\lambda y \in D_e$. $y \notin X$]

f. $\lambda x \in D_e$. [$\lambda y \in D_e$. [$\lambda z \in D_e$. z introduced y to x]]

Solutions: Exercise 9

(5) a. $\lambda x \in D_e$. the mother of x

The function which maps every x such that x is an individual to x's mother.'

b. $\lambda x \in D_e$. x

The function which maps every x such that x is an individual to x itself.'

c. $\lambda x \in D_e$. Chomsky

The function which maps every x such that x is an individual to Chomsky.'

Solutions: Exercise 9

(5) d. $\lambda x \in D_e$. [$\lambda y \in D_e$. y saw x]

'The function which maps every x such that x is an individual to the function which maps every y such that y is an individual to 1 if y saw x , and to 0 otherwise.'

e. $\lambda x \subseteq D_e$. [$\lambda y \in D_e$. y $\notin$ x]

'The function which maps every *X* such that *X* is a subset of the domain of individuals to the function that maps every individual y to 1 iff y is not a member of *X*'

f. $\lambda x \in D_e$. [$\lambda y \in D_e$. [$\lambda z \in D_e$. z introduced y to x]

The function which maps every x such that x is an individual to the function which maps every y such that y is an individual to the function which maps every z such that z is an individual to 1 if z introduced y to x, and to 0 otherwise.

Lambda conversion

Conversion: Computing with lambda terms

λ-notation: $F := \lambda x{:}\ x \in N.\ x^2$

When x=2,

then $[\lambda x{:}\ x \in N.\ x^2]\ (2)$

$= 2^2$

$= 4$

Step 1: Delete the λ, the variable, and the period.

Step 2: Replace all variables after the period by the argument.

Step 3: If possible, simplify the resulting expression.

Lambda conversion for natural language

Step 1: Delete the λ, the variable, and the period.

Step 2: Replace all variables after the period by the argument.

Step 3: If possible, simplify the resulting expression.

When x= Mary, then

[$\lambda x \in D_e$. the mother of x] (Mary)

= the mother of Mary

[$\lambda x : x \in D_e$. x smokes] (Mary)

= 1 iff Mary smokes

More than one arguments?

Apply the one argument which is to the most left first to the variable to the most left.

$[\lambda x \in D_e\ .\ [\lambda y \in D_e\ .\ [\lambda z \in D_e\ .\ z \text{ introduced } x \text{ to } y]]]$ (Ann)(Sue) (Mary)

$= [\lambda y \in D_e\ .\ [\lambda z \in D_e\ .\ z \text{ introduced Ann to } y]]$ (Sue) (Mary)

$= [\lambda z \in D_e\ .\ z \text{ introduced Ann to Sue}]$ (Mary)

= 1 iff Mary introduced Ann to Sue

Exercise 10: Simplify this function.

$[\lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, [\lambda z \in D_e \,.\, z \text{ introduced } y \text{ to } x] \text{ (Ann)}]] \text{ (Sue)}$

Tip: Argument and function are close "friends".

[λx] (argument to replace x)

Simplify this function.

Solution

$$[\lambda x \in D_e \,.\, [\lambda y \in D_e \,.\, [\lambda z \in D_e \,.\, z \text{ introduced } y \text{ to } x]\,(\text{Ann})]]\,(\text{Sue})$$

$$= [\lambda x \in D_e .\, [\lambda y \in D_e \,.\, \text{Ann introduced } y \text{ to } x]]\,(\text{Sue})$$

$$= \lambda y \in D_e \,.\, \text{Ann introduced } y \text{ to Sue}$$

Exercise 10: Function-valued functions

$$\boxed{[\lambda f \in D_{\langle e,t\rangle} \,.\, \boxed{[\lambda x \in D_e \,.\, f(x) = 1]}]}\boxed{([\lambda y \in D_e \,.\, y \text{ is blond}])}$$ **A char-function**

Simplify this function.

Solution

$$[\lambda f \in D_{\langle e,t \rangle} \ . \ [\lambda x \in D_e \ . \ f(x) = 1]]([\lambda y \in D_e \ . \ y \text{ is blond}])$$

$$=$$

$$[\lambda x \in D_e \ . \ [\lambda y \in D_e \ . \ y \text{ is blond}](x) = 1]$$

Next time...

Empty expressions, non-verbal predicates, modification

Thanks and see you next week!