

Tutorial 'Introduction to Semantic Theory' (No. 4)

Lecture 4

Zeqi Zhao

Session 3

November 15, 2019

Our agenda today

- Recap of last session.
- Also something new:
Negation, Coordination, **semantic types, type-driven interpretation**, λ -notation and conversion
- Some exercise to help you with assignment 4

For next time: Assignment 3, also the homework from last session

Remaining questions from previous assignments

Recap

Our semantic theory interpret any syntactic structure with two branches in terms of **Function Application**.

If α has the form
$$\begin{array}{c} S \\ \swarrow \quad \searrow \\ \beta \quad \gamma \end{array}$$
, then $[\alpha] = [\gamma]([\beta])$.

Since **intransitive 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

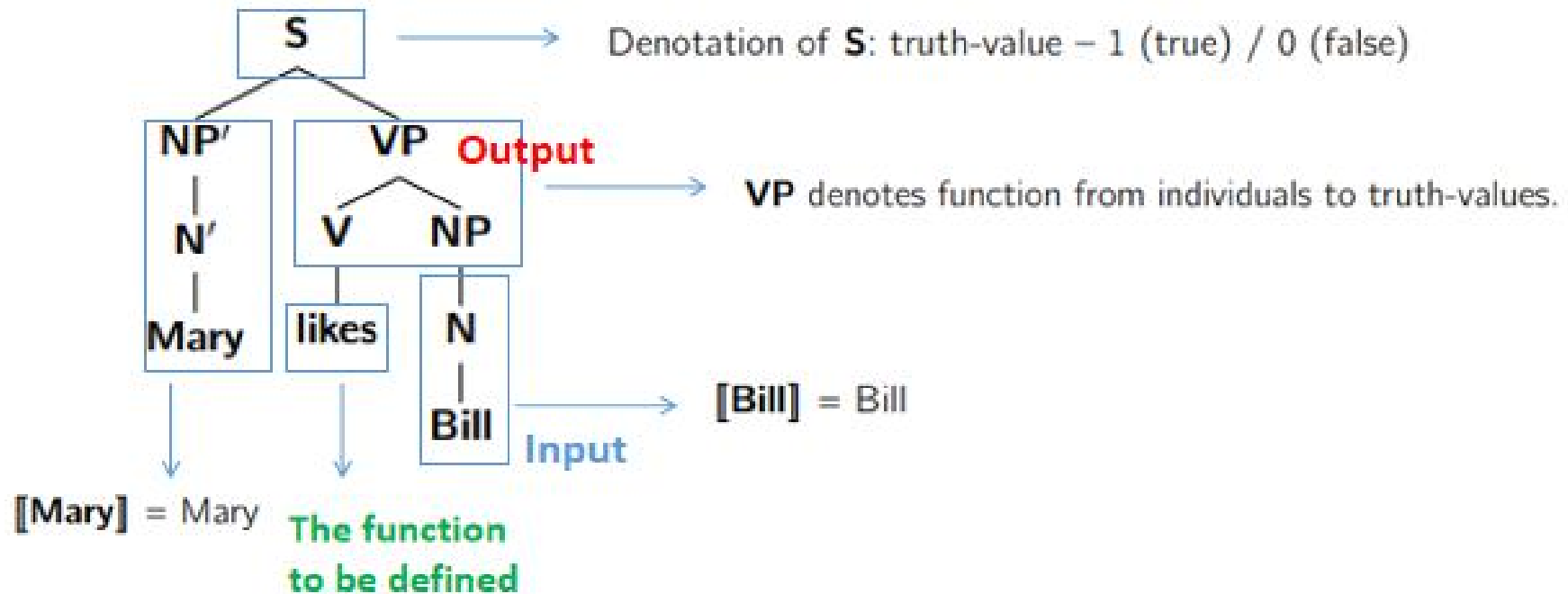
The meaning of **intransitive verbs** can be captured by either sets and or characteristic functions. This is because intransitive verbs expresses a property which may be true of some individuals but not of others in a given situation.

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

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.

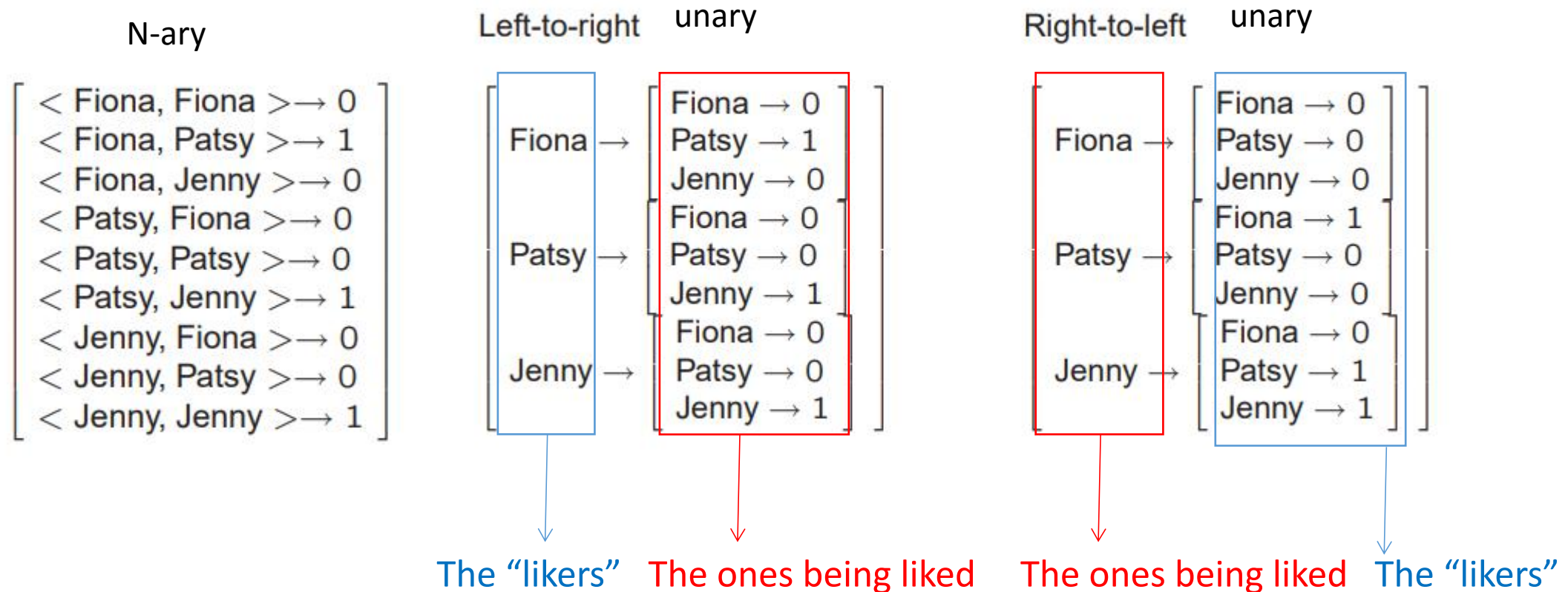
Recap

However, our syntax and compositionality tell us, transitive verbs denote a unary function that 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.



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: \neg 'not'

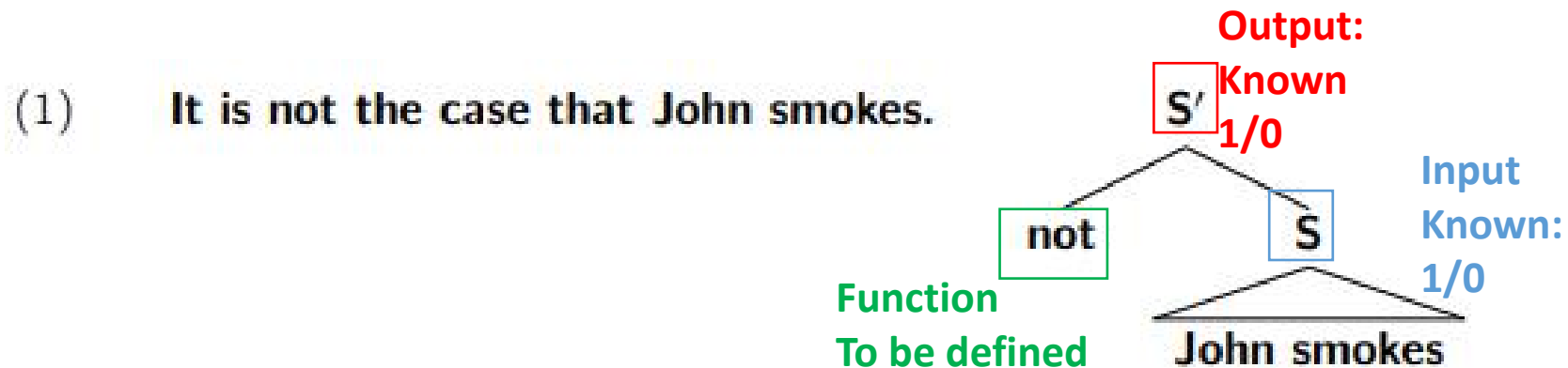
If $[[\text{John smokes}]] = 1$, then $[[\text{It is not the case that John smokes}]] = 0$.

If $[[\text{John smokes}]] = 0$, then $[[\text{It is not the case that John smokes}]] = 1$.

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

Neg as function

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



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

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

What kind of denotation type does $[[\text{not}]]$ 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* \neg : 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= Don loves Megan q= Roger smokes

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'.

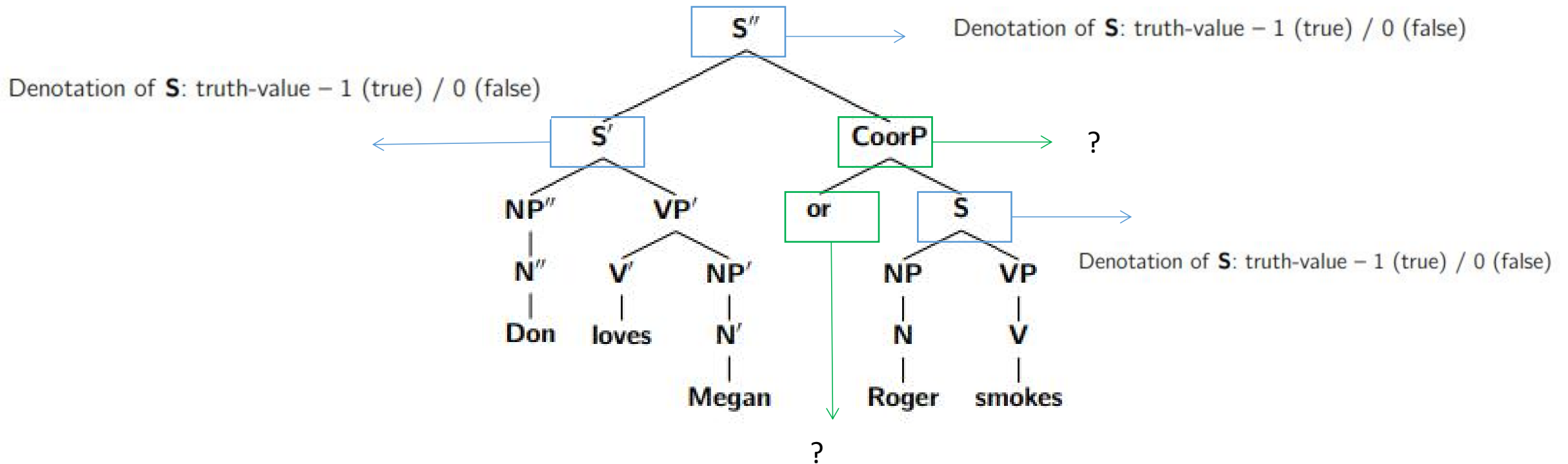
p = Would you like tee? q = Would you like coffe?

Exclusive or $p \text{ XOR } q$

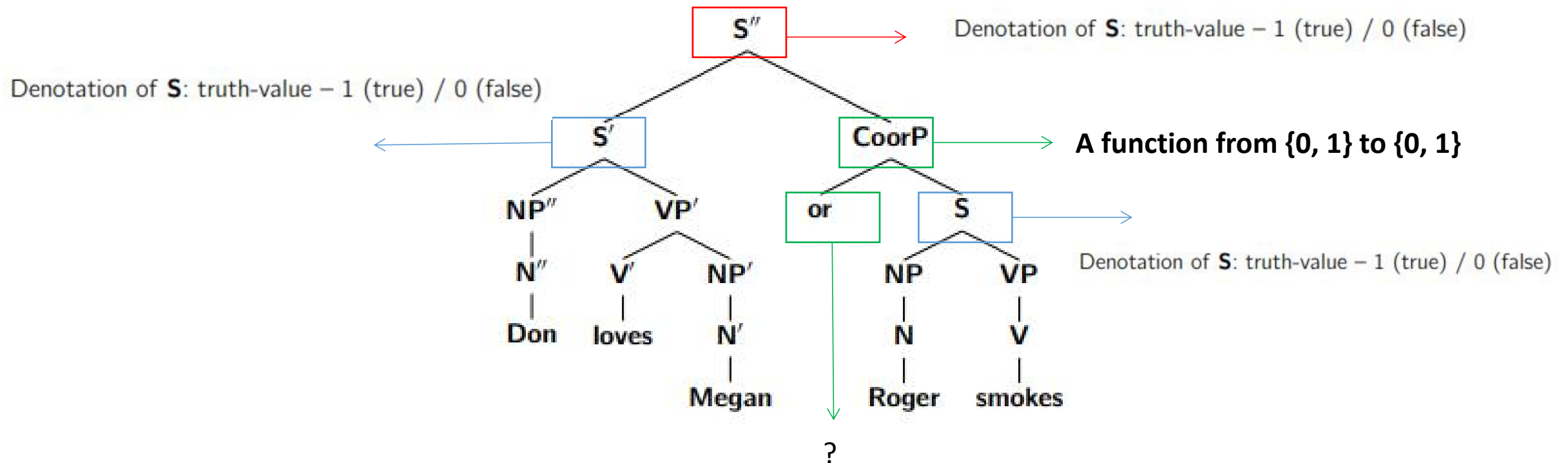
p	q	$p \text{ XOR } q$
T	T	F
T	F	T
F	T	T
F	F	F

For this course, we only use “or” in a **inclusive** sense.

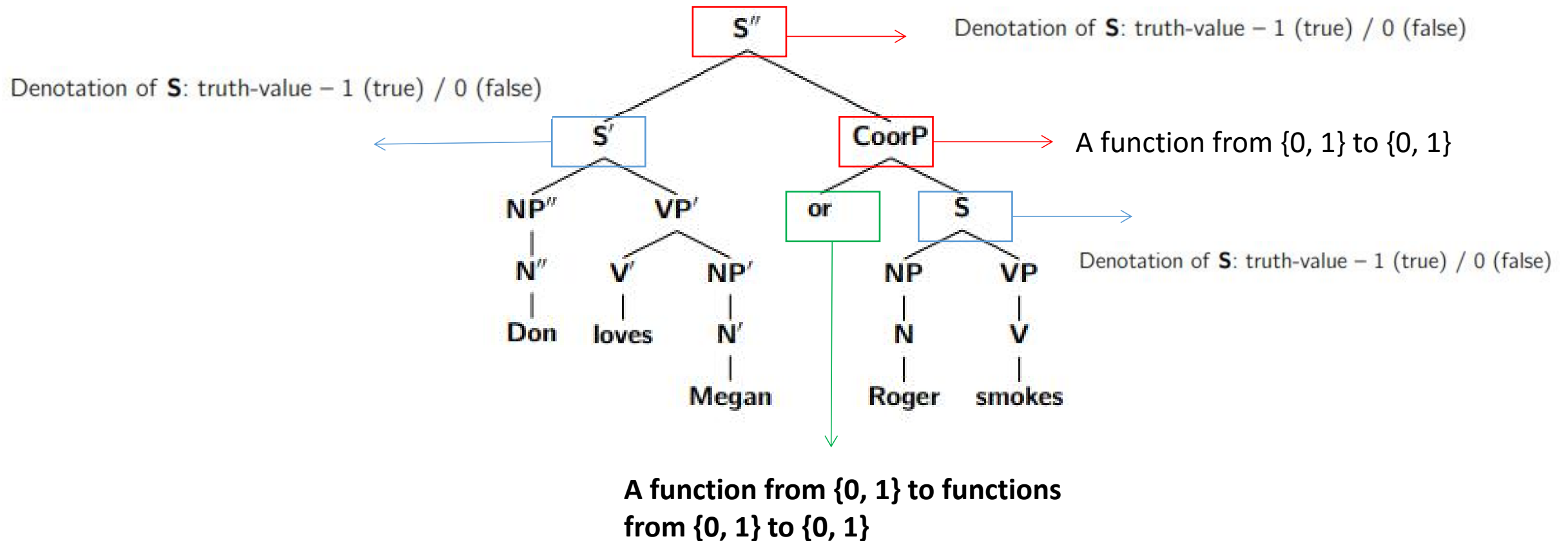
What kind of denotation type?



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_i , Predicative ADJ	Functions from D to $\{0, 1\}$ (characteristic function)
V_t	Functions from D to functions from D to $\{0, 1\}$.
Neg, CoordP	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. $D_{\langle \alpha, \beta \rangle} = \{f \mid f : D_\alpha \rightarrow D_\beta\}$ (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
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, CoordP	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>>

Basic types

Functional
types

Excercise 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]])

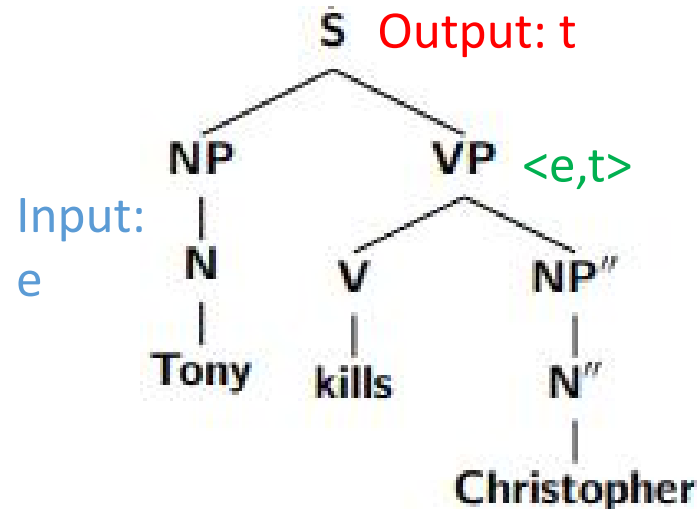
k. [[love]] ([[John]]) ([[Mary]])

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

Tips for specifying semantic types

Tip 1: By definition, **all characteristic functions are type $\langle e, t \rangle$** .

- a. **Common noun**: cat, a student, a member of the Beatles, a smart linguist
- b. **V_i** : smokes, dances, runs, lives in New York
- c. **Predicative ADJ**: happy, green
- d. **$[[V_tP]]$** : **[[likes]]** (**[[John]]**)

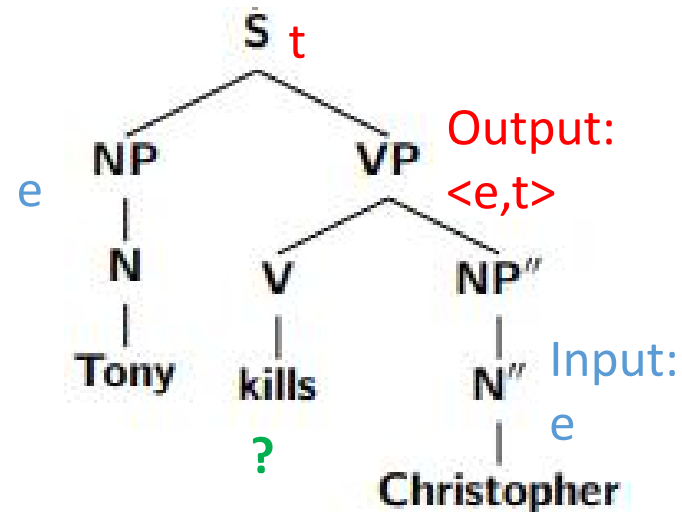


Tips for specifying semantic types

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 know:

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



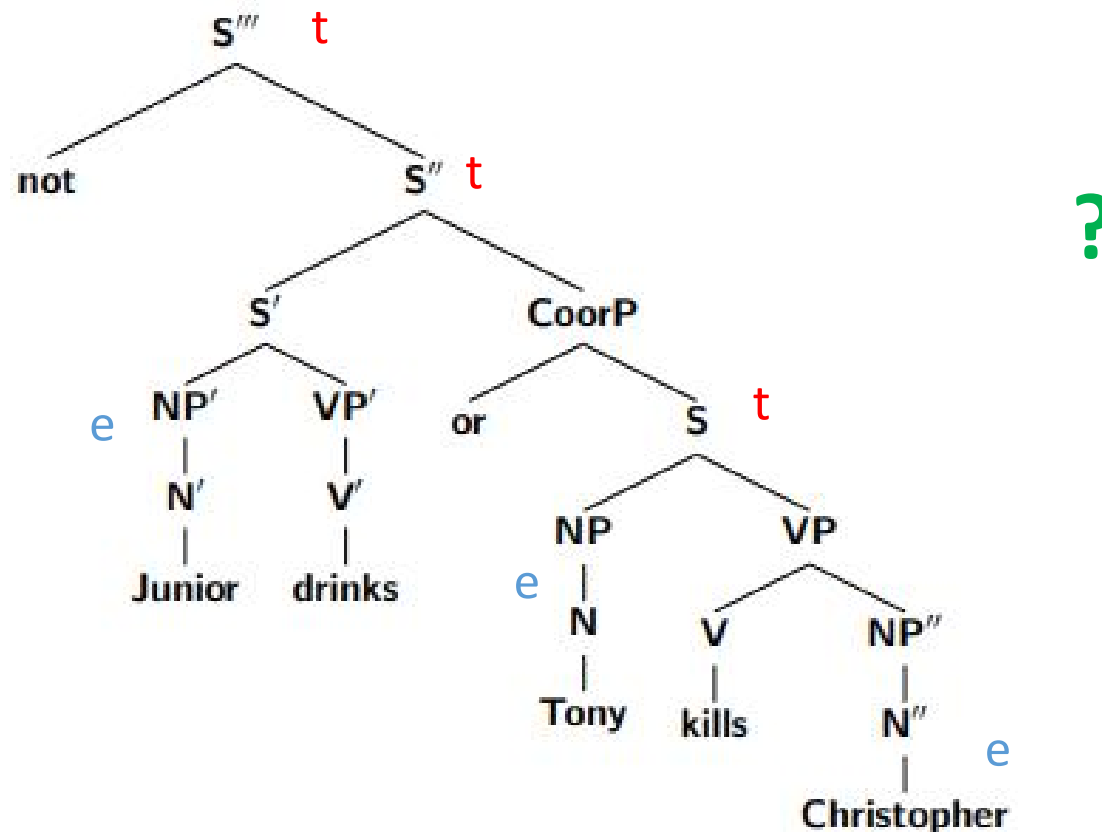
[[kill]] denotes a function from D_e to $D_{\langle e, t \rangle}$

Therefore, the semantic type of [[kill]] is $\langle e, \langle e, t \rangle \rangle$

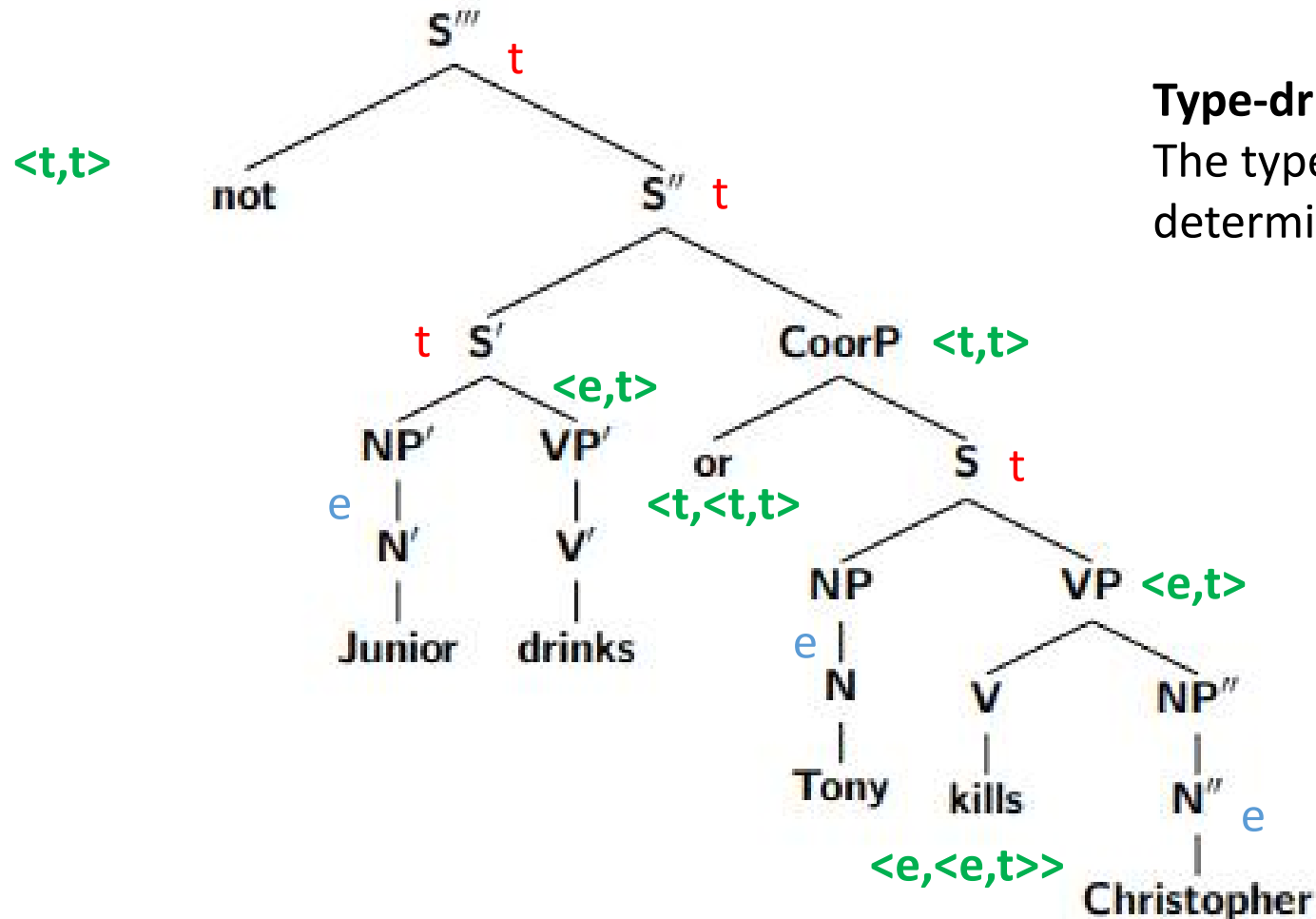
One more example

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: Exercise 8

- (3) a. [[John Lennon]] e
b. [[the girl who stole my bike]] e
c. [[a student in Göttingen]] $\langle e, t \rangle$
d. [[I love semantics]] t
e. [[and]] $\langle t, \langle t, t \rangle \rangle$
f. [[and]] ([[I love semantics]]) $\langle t, t \rangle$

- f. [[not]] $\langle t, t \rangle$
g. [[happy]] $\langle e, t \rangle$
h. [[dances]] $\langle e, t \rangle$
i. [[love]] $\langle e, \langle e, t \rangle \rangle$
j. [[love]] ([[John]]) $\langle e, t \rangle$
k. [[love]] ([[John]]) ([[Mary]]) t
l. [[not]] ([[I love semantics]]) t

How we used to define functions

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

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



How we used to define functions

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

(4) $\llbracket \text{loves} \rrbracket = f : D_e \rightarrow D_{\langle e, t \rangle}$
For all $x, y \in D_e$, $f(x)(y) = 1$ iff y loves x

Still too much writing! Farewell, again!



Lambda notation

λ -notation is very convenient for functions defining.

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: $f(x) = x^2$

Set theoretic notation: $F := \{ \langle x, x^2 \rangle : x \in \mathbb{N} \}$

λ -notation: $F := [\lambda x : x \in \mathbb{N}. x^2]$ or $F := \lambda x \in \mathbb{D}_N. x^2$

We name the function as: 'The **(smallest)** function which maps every x such that x is a natural number to x^2 (x squared).'

What about natural language?

$$F := \lambda x \in D_N . x^2$$

Read as: 'The (smallest) function which maps every x such that x is a natural number to x^2 (x squared).'

Reading convention:

'the (smallest) function which maps every variable such that domain to value.'

Problem with semantic function:

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

This is a function of type $\langle e, t \rangle$.

Wrong reading: 'The (smallest) function which maps every x such that $x \in D$ 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)	$\langle e, t \rangle$
V_t	Functions from D to functions from D to {0, 1}.	$\langle e, \langle e, t \rangle \rangle$
Neg, CoordP	Function from {0, 1} to {0, 1}	$\langle t, t \rangle$
or/and	Functions from {0, 1} to functions from {0, 1} to {0, 1}	$\langle t, \langle t, t \rangle \rangle$

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 $\{0,1\}$ as their range: Reading 1

‘the (smallest) function which maps every variable such that domain to 1 if value, and to 0 otherwise.’

For other functions: Reading 2

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

Examples: How do we name functions?

- $F_1 =: \lambda x \in D_N . x + 4$ Range: \mathbb{N}

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

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

Read: 'The (smallest) function which maps every x such that x is an individual to the age of x .'

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

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

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

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

Excercise 9: Name the functions

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

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

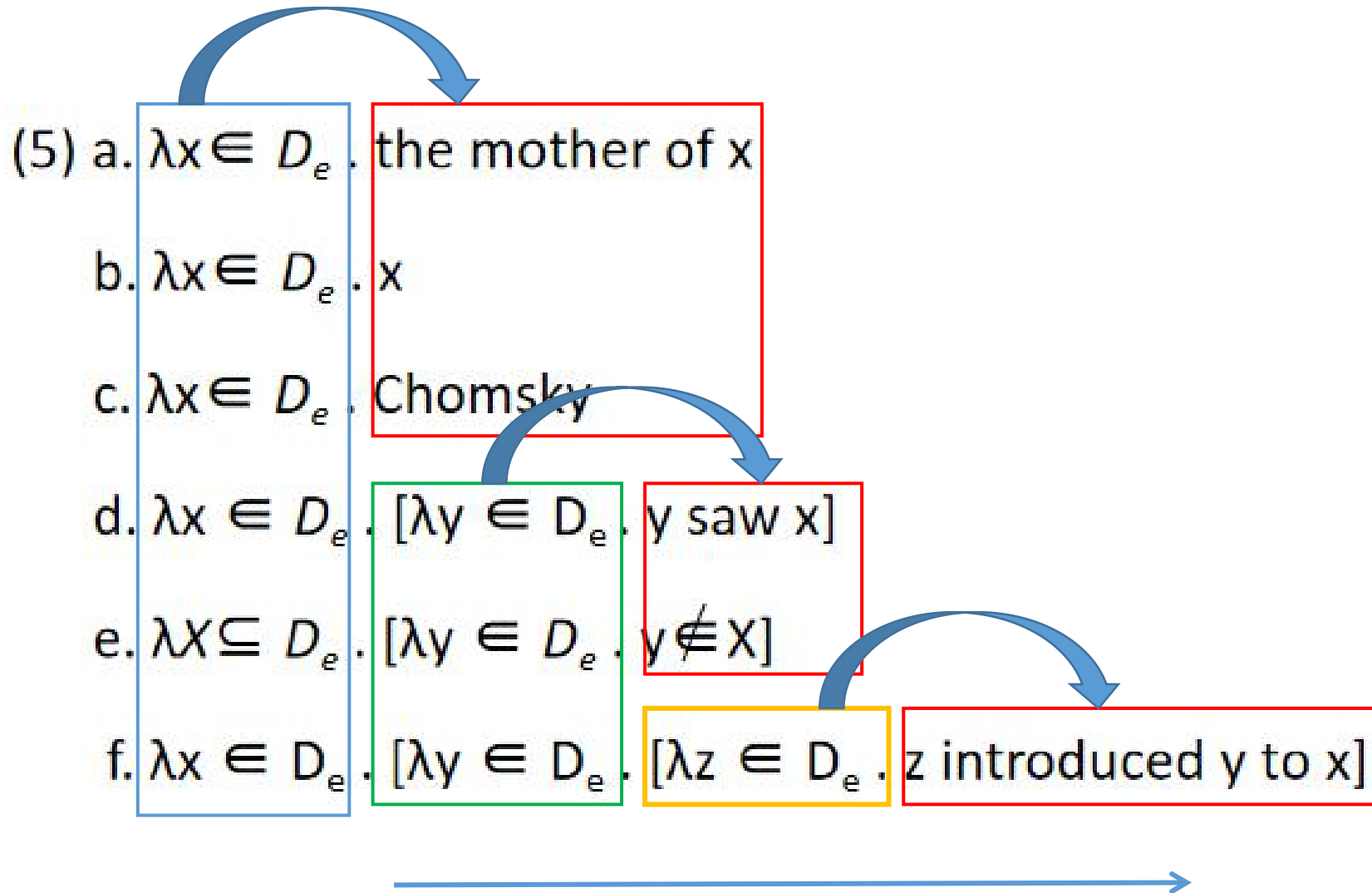
c. $\lambda x \in D_e . \text{Chomsky}$

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

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

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

A tip for reading the functions



Solutions: Exercise 9

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

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

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

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

c. $\lambda x \in D_e . \text{Chomsky}$

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

Solutions: Exercise 9

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

'The (smallest) function which maps every x such that x is an individual to the (smallest) 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 (smallest) 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. $\lambda x \in D_e . [\lambda y \in D_e . [\lambda z \in D_e . z \text{ introduced } y \text{ to } x]]$

The (smallest) function which maps every x such that x is an individual to the (smallest) function which maps every y such that y is an individual to the (smallest) 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: $F := \lambda x: x \in \mathbb{N}. x^2$

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

$$= 2^2$$

$$= 4$$

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 $x = \text{Mary}$, then

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

= the mother of Mary

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

= 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.



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



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



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

$= 1$ iff Mary introduced Ann to Sue

Exercise 10: Be careful of the “[]”

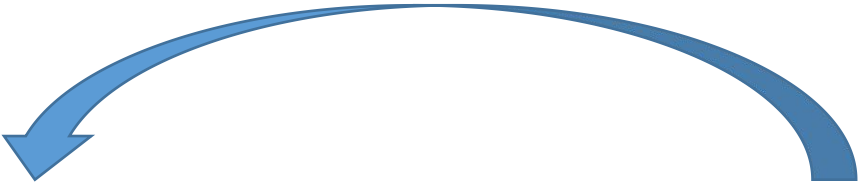
$[\lambda x \in D_e . [\lambda y \in D_e . [\lambda z \in D_e . z \text{ introduced } y \text{ to } x] (\text{Ann})]] (\text{Sue})$

Tip: Argument and function are close “friends”.

$[\lambda x]$ (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

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

A char-function

Simplify this function.

Solution

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

=

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

Next time...

Textbooks (relevant chapters posted on StudIP)

H&K 4.1–4.3

Assignment 3, also the homework from last session

Remaining questions from previous assignments

Empty expressions, non-verbal predicates, modification

Thanks and see you next week!