Semantik, Modul 1003

More on predicates, Modification

Heim & Kratzer (1998), ch. 3 - 4.2

Leipzig University

April 25^{th}, 2024

Fabian Heck
Slides by Imke Driemel

Recap: λ-calculus, semantic types

- last week we introduced the λ-calculus: a simplified notation for functions

(1) a. $\boxed{\lambda\alpha : \phi[\gamma]}$

b. $\boxed{\lambda \text{ argument} : \text{domain condition } [\text{value description}]}$

- α is the argument of the function; ϕ is used to specify the domain that α denotes in; γ specifies the value that the function assigns to α
- ϕ uses the semantic types $\langle e \rangle$ and $\langle t \rangle$ (or other types) to make clear what kind of argument a functions needs

(2) $[\![\textit{cries}]\!] = \lambda x : x \in D_{\langle e \rangle}. [x \text{ cries}]$

- in order to get to the value of a function, we apply the λ-term to an argument (functional application) by substituting the variable the λ-term introduces with that argument $\rightarrow$ *λ-conversion*

(3) $[\![\textit{cries}]\!]([\![\textit{Mo}]\!]) = \lambda x_{\langle e \rangle}[\text{cries}(x)](\text{Mo}) = \text{cries}(\text{Mo})$

Recap: λ-calculus, semantic types

- semantic types help us keep track of the semantic objects we are referring to
- complex types consist of an input and an output: $\langle \underbrace{\sigma}_{in}, \underbrace{\langle \sigma, \tau \rangle}_{out} \rangle$
- we looked at intransitive, transitive, and ditransitive predicates, as well as negation

(4) $\underbrace{[\![\textit{cries}]\!]}_{\langle e,t\rangle} = \underbrace{\lambda x : x \in D_{\langle e\rangle}}_{\langle e,} \underbrace{[\text{ x cries }]}_{t\rangle}$

(5) $\underbrace{[\![\textit{dislike}]\!]}_{\langle e,\langle e,t\rangle\rangle} = \underbrace{\lambda x : x \in D_{\langle e\rangle}}_{\langle e,}. [\underbrace{\lambda y : y \in D_{\langle e\rangle}}_{\langle e,}. \underbrace{\text{y dislikes x}]}_{t\rangle\rangle}$

(6) $\underbrace{[\![\textit{introduce}]\!]}_{\langle e,\langle e,\langle e,t\rangle\rangle\rangle} = \underbrace{\lambda x : x \in D_{\langle e\rangle}}_{\langle e,}. [\underbrace{\lambda y : y \in D_{\langle e\rangle}}_{\langle e,}. [\underbrace{\lambda z : z \in D_{\langle e\rangle}}_{\langle e,}. \underbrace{\text{z introduces x to y}]]}_{t\rangle\rangle\rangle}$

(7) $\underbrace{[\![\textit{nicht}]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} = \underbrace{\lambda Q : Q \in D_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,}. [\underbrace{\lambda x : x \in D_{\langle e\rangle}}_{\langle e,}. \underbrace{\neg Q(x)]}_{t\rangle\rangle}$

(weil) Peter nicht weint

Recap: λ-calculus, semantic types

- we also noticed that some lexical items do not contribute any meaning
- since compositionality requires to interpret any node in the tree, we assigned *identity functions* to them, i.e. functions which take an argument and give the same argument back
- examples are the preposition *to* in ditransitve clauses and the copula *be*

(8) a. Mo introduces Lee **to** Jean.

b. $\underbrace{[\![to]\!]}_{\langle e,e\rangle} = \underbrace{\lambda y_{\langle e\rangle}}_{\langle e,} \underbrace{[y]}_{e\rangle}$ *a function from individuals to individuals*

(9) a. Mo **is** happy.

b. $\underbrace{[\![be]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} = \underbrace{\lambda P_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,} \underbrace{[P]}_{\langle e,t\rangle\rangle}$ *a function from properties to properties*

- compare the copula to **negation** which does contribute meaning

(10) $\underbrace{[\![nicht]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} = \underbrace{\lambda Q : Q \in D_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,} . \underbrace{[\neg Q]}_{\langle e,t\rangle\rangle}$

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ *λ-C*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ λ-*C*
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12) a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ λ-*C*
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12)

a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ *λ-C*
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![not\ happy]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda x_{\langle e \rangle}[\neg happy(x)])$ *FA*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12)

a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ λ-*C*
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![not\ happy]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda x_{\langle e \rangle}[\neg happy(x)])$ *FA*

g. $[\![is]\!]([\![not\ happy]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)]$ λ-*C*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12)

a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$ λ-*C*
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![not\ happy]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda x_{\langle e \rangle}[\neg happy(x)])$ *FA*

g. $[\![is]\!]([\![not\ happy]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)]$ λ-*C*

h. $[\![Mo]\!] = \text{Mo}$ *Lex*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12)

a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$ *λ-C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![not\ happy]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda x_{\langle e \rangle}[\neg happy(x)])$ *FA*

g. $[\![is]\!]([\![not\ happy]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)]$ *λ-C*

h. $[\![Mo]\!] = \text{Mo}$ *Lex*

i. $[\![VP]\!]([\![Mo]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)](\text{Mo})$ *FA*

Exercise

(11) $[_S$ Mo $[_{VP}$ is not happy]]

Calculate the denotation of S above!

(12)

a. $[\![happy]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)]$ *Lex*

b. $[\![not]\!] = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)]$ *Lex*

c. $[\![not]\!]([\![happy]\!]) = \lambda P_{\langle e,t \rangle} \lambda x_{\langle e \rangle}[\neg P(x)](\lambda y_{\langle e \rangle}[\text{happy}(y)])$ *FA*

d. $[\![not]\!]([\![happy]\!]) = \lambda x_{\langle e \rangle}[\neg \lambda y_{\langle e \rangle}[happy(y)](x)]$
$= \lambda x_{\langle e \rangle}[\neg happy(x)]$ *λ-C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![not\ happy]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda x_{\langle e \rangle}[\neg happy(x)])$ *FA*

g. $[\![is]\!]([\![not\ happy]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)]$ *λ-C*

h. $[\![Mo]\!] = \text{Mo}$ *Lex*

i. $[\![VP]\!]([\![Mo]\!]) = \lambda x_{\langle e \rangle}[\neg happy(x)](\text{Mo})$ *FA*

j. $[\![S]\!] = \neg happy(Mo) = 1$ iff Mo is not happy *λ-C*

Non-verbal predicates

Intransitive predicates are one-place verbal predicates:

(13) $[\![\textit{cries}]\!] = \lambda y_{\langle e \rangle}[\text{cries}(y)] = 1$ iff y cries

We have already looked at adjectival predicates, they look very similar. With the help of the copula, which is an identity function, we can use them exactly like intransitive predicates.

(14) $[\![\textit{happy}]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)] = 1$ iff y is happy

Non-verbal predicates

Intransitive predicates are one-place verbal predicates:

(13) $[\![\textit{cries}]\!] = \lambda y_{\langle e \rangle}[\text{cries}(y)] = 1$ iff y cries

We have already looked at adjectival predicates, they look very similar. With the help of the copula, which is an identity function, we can use them exactly like intransitive predicates.

(14) $[\![\textit{happy}]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)] = 1$ iff y is happy

It turns out that nominals can also be used as predicates:

(15) Mo is a student.

(16) $[\![\textit{student}]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)] = 1$ iff y is a student

Non-verbal predicates

Intransitive predicates are one-place verbal predicates:

(13) $[\![\textit{cries}]\!] = \lambda y_{\langle e \rangle}[\text{cries}(y)] = 1$ iff y cries

We have already looked at adjectival predicates, they look very similar. With the help of the copula, which is an identity function, we can use them exactly like intransitive predicates.

(14) $[\![\textit{happy}]\!] = \lambda y_{\langle e \rangle}[\text{happy}(y)] = 1$ iff y is happy

It turns out that nominals can also be used as predicates:

(15) Mo is a student.

(16) $[\![\textit{student}]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)] = 1$ iff y is a student

We will (for now) assume that the denotation of *a* is also an identity function:

(17) $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student$]]$

Calculate the denotation of S above!

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19) a. $[\![\textit{student}]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19) a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student$]]$

Calculate the denotation of S above!

(19) a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student$]]$

Calculate the denotation of S above!

(19) a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19) a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19)

a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ λ-*C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![a\ student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19)

a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![a\ student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

g. $[\![is]\!]([\![a\ student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19)
a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![a\ student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

g. $[\![is]\!]([\![a\ student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *λ-C*

h. $[\![Mo]\!] = \text{Mo}$ *Lex*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student]]

Calculate the denotation of S above!

(19)

a. $[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$ *Lex*

b. $[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

c. $[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

d. $[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ λ-*C*

e. $[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$ *Lex*

f. $[\![is]\!]([\![a\ student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$ *FA*

g. $[\![is]\!]([\![a\ student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$ λ-*C*

h. $[\![Mo]\!] = \text{Mo}$ *Lex*

i. $[\![VP]\!]([\![Mo]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)](\text{Mo})$ *FA*

Exercise

(18) $[_S$ Mo $[_{VP}$ is a student$]]$

Calculate the denotation of S above!

(19)

a.	$[\![student]\!] = \lambda y_{\langle e \rangle}[\text{student}(y)]$	*Lex*
b.	$[\![a]\!] = \lambda P_{\langle e,t \rangle}[P]$	*Lex*
c.	$[\![a]\!]([\![student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$	*FA*
d.	$[\![a]\!]([\![student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$	λ-*C*
e.	$[\![is]\!] = \lambda P_{\langle e,t \rangle}[P]$	*Lex*
f.	$[\![is]\!]([\![a\ student]\!]) = \lambda P_{\langle e,t \rangle}[P](\lambda y_{\langle e \rangle}[\text{student}(y)])$	*FA*
g.	$[\![is]\!]([\![a\ student]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)]$	λ-*C*
h.	$[\![Mo]\!] = \text{Mo}$	*Lex*
i.	$[\![VP]\!]([\![Mo]\!]) = \lambda y_{\langle e \rangle}[\text{student}(y)](\text{Mo})$	*FA*
j.	$[\![S]\!] = student(Mo) = 1$ iff Mo is a student	λ-*C*

Non-verbal predicates

Set-notation:

(20) $[\![\textit{cries}]\!] = \{x \mid x \text{ cries}\}$ *the set of all x such that x cries*

(21) $[\![\textit{happy}]\!] = \{x \mid x \text{ is happy}\}$ *the set of all x such that x is happy*

(22) $[\![\textit{student}]\!] = \{x \mid x \text{ is a student}\}$ *the set of all x such that x is a student*

Non-verbal predicates

Set-notation:

(20) $[\![\textit{cries}]\!]$ = {x | x cries} *the set of all x such that x cries*

(21) $[\![\textit{happy}]\!]$ = {x | x is happy} *the set of all x such that x is happy*

(22) $[\![\textit{student}]\!]$ = {x | x is a student} *the set of all x such that x is a student*

One-place predicates are also called **properties**. The copula is therefore a function from properties to properties.

(23) $\underbrace{[\![be]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} = \underbrace{\lambda P_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,} \underbrace{[P]}_{\langle e,t\rangle\rangle}$

Non-verbal predicates

Set-notation:

(20) $[\![\textit{cries}]\!] = \{x \mid x \text{ cries}\}$ *the set of all x such that x cries*

(21) $[\![\textit{happy}]\!] = \{x \mid x \text{ is happy}\}$ *the set of all x such that x is happy*

(22) $[\![\textit{student}]\!] = \{x \mid x \text{ is a student}\}$ *the set of all x such that x is a student*

One-place predicates are also called **properties**. The copula is therefore a function from properties to properties.

(23) $\underbrace{[\![\textit{be}]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} = \underbrace{\lambda P_{\langle e,t\rangle}}_{\langle\langle e,t\rangle,} \underbrace{[P]}_{\langle e,t\rangle\rangle}$

Not every adjective is a property, one example is *proud* which is a two-place predicate. The preposition *of* has a denotation like *to*.

(24) $\underbrace{[\![\textit{Mo}]\!]}_{\langle e\rangle}$ $\underbrace{[\![\textit{is}]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ $\underbrace{[\![\textit{proud}]\!]}_{\langle e,\langle e,t\rangle\rangle}$ $\underbrace{[\![\textit{of}]\!]}_{\langle e,e\rangle}$ $\underbrace{[\![\textit{Dee}]\!]}_{\langle e\rangle}$

Non-verbal predicates

Prepositions have so far been treated as identity functions, i.e. they do not contribute meaning.

(25) $\underbrace{[\![to]\!]}_{\langle e,e\rangle} = \underbrace{\lambda y_{\langle e\rangle}}_{\langle e,} \underbrace{[y]}_{e\rangle}$ *a function from individuals to individuals*

(26) $\underbrace{[\![of]\!]}_{\langle e,e\rangle} = \underbrace{\lambda y_{\langle e\rangle}}_{\langle e,} \underbrace{[y]}_{e\rangle}$ *a function from individuals to individuals*

Non-verbal predicates

Prepositions have so far been treated as identity functions, i.e. they do not contribute meaning.

(25) $\underbrace{[\![to]\!]}_{\langle e,e\rangle} = \underbrace{\lambda y_{\langle e\rangle}}_{\langle e,} \underbrace{[y]}_{e\rangle}$ *a function from individuals to individuals*

(26) $\underbrace{[\![of]\!]}_{\langle e,e\rangle} = \underbrace{\lambda y_{\langle e\rangle}}_{\langle e,} \underbrace{[y]}_{e\rangle}$ *a function from individuals to individuals*

This is not true for every preposition.

Why is it not a good idea to treat *with* and *above* in the sentences below as identity functions?

(27) a. Dee is with Mo.

b. Spiderman is above Hulk.

Non-verbal predicates

Assigning identity functions to *with* and *above* would result in a type clash. Moreover, they do seem to contribute meaning.

(28) a. $\underbrace{[\![Dee]\!]}_{\langle e \rangle} \quad \underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \quad \underbrace{[\![with]\!]}_{\langle ?? \rangle} \underbrace{[\![Mo]\!]}_{\langle e \rangle}.$

b. $\underbrace{[\![Spiderman]\!]}_{\langle e \rangle} \quad \underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \underbrace{[\![above]\!]}_{\langle ?? \rangle} \underbrace{[\![Hulk]\!]}_{\langle e \rangle}.$

Non-verbal predicates

Assigning identity functions to *with* and *above* would result in a type clash. Moreover, they do seem to contribute meaning.

(28) a. $\underbrace{[\![Dee]\!]}_{\langle e \rangle} \quad \underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \quad \underbrace{[\![with]\!]}_{\langle ?? \rangle} \underbrace{[\![Mo]\!]}_{\langle e \rangle}.$

b. $\underbrace{[\![Spiderman]\!]}_{\langle e \rangle} \quad \underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \quad \underbrace{[\![above]\!]}_{\langle ?? \rangle} \underbrace{[\![Hulk]\!]}_{\langle e \rangle}.$

Therefore, we treat *with* and *above* as two-place predicates.

(29) a. $[\![with]\!] = \lambda y_{\langle e \rangle} \lambda x_{\langle e \rangle} [\text{with}(x, y)] = 1$ iff x is with y

b. $[\![above]\!] = \lambda y_{\langle e \rangle} \lambda x_{\langle e \rangle} [\text{above}(x, y)] = 1$ iff x is above y

Non-verbal predicates

Assigning identity functions to *with* and *above* would result in a type clash. Moreover, they do seem to contribute meaning.

(28) a. $\underbrace{[\![\textit{Dee}]\!]}_{\langle e\rangle}\ \underbrace{[\![\textit{is}]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}\ \underbrace{[\![\textit{with}]\!]}_{\langle ??\rangle}\ \underbrace{[\![\textit{Mo}]\!]}_{\langle e\rangle}$.

b. $\underbrace{[\![\textit{Spiderman}]\!]}_{\langle e\rangle}\ \underbrace{[\![\textit{is}]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}\ \underbrace{[\![\textit{above}]\!]}_{\langle ??\rangle}\ \underbrace{[\![\textit{Hulk}]\!]}_{\langle e\rangle}$.

Therefore, we treat *with* and *above* as two-place predicates.

(29) a. $[\![\textit{with}]\!] = \lambda y_{\langle e\rangle}\lambda x_{\langle e\rangle}[\text{with}(x, y)] = 1$ iff x is with y

b. $[\![\textit{above}]\!] = \lambda y_{\langle e\rangle}\lambda x_{\langle e\rangle}[\text{above}(x, y)] = 1$ iff x is above y

What is the type we need to assign to *with* and *above*?

Non-verbal predicates

Assigning identity functions to *with* and *above* would result in a type clash. Moreover, they do seem to contribute meaning.

(28) a. $\underbrace{[\![Dee]\!]}_{\langle e \rangle}$ $\underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ $\underbrace{[\![with]\!]}_{\langle ?? \rangle}$ $\underbrace{[\![Mo]\!]}_{\langle e \rangle}$.

b. $\underbrace{[\![Spiderman]\!]}_{\langle e \rangle}$ $\underbrace{[\![is]\!]}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ $\underbrace{[\![above]\!]}_{\langle ?? \rangle}$ $\underbrace{[\![Hulk]\!]}_{\langle e \rangle}$.

Therefore, we treat *with* and *above* as two-place predicates.

(29) a. $[\![with]\!] = \lambda y_{\langle e \rangle} \lambda x_{\langle e \rangle}$[with($x$, y)] = 1 iff x is with y

b. $[\![above]\!] = \lambda y_{\langle e \rangle} \lambda x_{\langle e \rangle}$[above($x$, y)] = 1 iff x is above y

What is the type we need to assign to *with* and *above*?
$\langle e, \langle e, t \rangle \rangle$

New topic

Modification

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

c. It is surprising that Susan, from Nebraska, finds it cold in here.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*

d. Peter fährt das Auto vorsichtig.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*

d. Peter fährt das Auto vorsichtig. *modifier*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30) a. I live in a part of Europe. *argument*

b. I live in a city in France. *modifier*

c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*

d. Peter fährt das Auto vorsichtig. *modifier*

e. I met the delightful mother of Susan.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)
a. I live in a part of Europe. *argument*
b. I live in a city in France. *modifier*
c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*
d. Peter fährt das Auto vorsichtig. *modifier*
e. I met the delightful mother of Susan. *argument*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)
a. I live in a part of Europe. *argument*
b. I live in a city in France. *modifier*
c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*
d. Peter fährt das Auto vorsichtig. *modifier*
e. I met the delightful mother of Susan. *argument*
f. I met the delightful mother of Susan.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)
a. I live in a part of Europe. *argument*
b. I live in a city in France. *modifier*
c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*
d. Peter fährt das Auto vorsichtig. *modifier*
e. I met the delightful mother of Susan. *argument*
f. I met the delightful mother of Susan. *modifier*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)
a. I live in a part of Europe. *argument*
b. I live in a city in France. *modifier*
c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*
d. Peter fährt das Auto vorsichtig. *modifier*
e. I met the delightful mother of Susan. *argument*
f. I met the delightful mother of Susan. *modifier*
g. I met the younger sister of Susan.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)

a.	I live in a part of Europe.	*argument*
b.	I live in a city in France.	*modifier*
c.	It is surprising that Susan, from Nebraska, finds it cold in here.	*modifier*
d.	Peter fährt das Auto vorsichtig.	*modifier*
e.	I met the delightful mother of Susan.	*argument*
f.	I met the delightful mother of Susan.	*modifier*
g.	I met the younger sister of Susan.	*modifier?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)	a.	I live in a part of Europe.	*argument*
	b.	I live in a city in France.	*modifier*
	c.	It is surprising that Susan, from Nebraska, finds it cold in here.	*modifier*
	d.	Peter fährt das Auto vorsichtig.	*modifier*
	e.	I met the delightful mother of Susan.	*argument*
	f.	I met the delightful mother of Susan.	*modifier*
	g.	I met the younger sister of Susan.	*modifier?*
	h.	We sold the car to the dealer for 2000 euros.	

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)	a.	I live in a part of Europe.	*argument*
	b.	I live in a city in France.	*modifier*
	c.	It is surprising that Susan, from Nebraska, finds it cold in here.	*modifier*
	d.	Peter fährt das Auto vorsichtig.	*modifier*
	e.	I met the delightful mother of Susan.	*argument*
	f.	I met the delightful mother of Susan.	*modifier*
	g.	I met the younger sister of Susan.	*modifier?*
	h.	We sold the car to the dealer for 2000 euros.	*argument?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)
a. I live in a part of Europe. *argument*
b. I live in a city in France. *modifier*
c. It is surprising that Susan, from Nebraska, finds it cold in here. *modifier*
d. Peter fährt das Auto vorsichtig. *modifier*
e. I met the delightful mother of Susan. *argument*
f. I met the delightful mother of Susan. *modifier*
g. I met the younger sister of Susan. *modifier?*
h. We sold the car to the dealer for 2000 euros. *argument?*
i. We sold the car to the dealer for 2000 euros.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

First try:
A modifier adds additional, non-essential descriptive content to that contributed by the expression that it combines with. An argument adds required content.

(30)	a.	I live in a part of Europe.	*argument*
	b.	I live in a city in France.	*modifier*
	c.	It is surprising that Susan, from Nebraska, finds it cold in here.	*modifier*
	d.	Peter fährt das Auto vorsichtig.	*modifier*
	e.	I met the delightful mother of Susan.	*argument*
	f.	I met the delightful mother of Susan.	*modifier*
	g.	I met the younger sister of Susan.	*modifier?*
	h.	We sold the car to the dealer for 2000 euros.	*argument?*
	i.	We sold the car to the dealer for 2000 euros.	*modifier?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer).

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house *modifier or argument?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house *modifier or argument?*

d. ein stabil-gebautes Haus

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house *modifier or argument?*

d. ein stabil-gebautes Haus *modifier or argument?*

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house *modifier or argument?*

d. ein stabil-gebautes Haus *modifier or argument?*

Conclusion: Neither the non-essential nature nor the syntax of a modifier helps us distinguish modifiers from arguments.

Modification

The term modifier (and modification, the function a modifier carries out) is difficult to define in intuitive terms.

Second try:
A modifier is syntactically not obligatory, in contrast to arguments.

(31) a. We sold the car (to the dealer). *modifier or argument?*

b. Ich habe schon (Mittag) gegessen. *modifier or argument?*

c. a well-built house *modifier or argument?*

d. ein stabil-gebautes Haus *modifier or argument?*

Conclusion: Neither the non-essential nature nor the syntax of a modifier helps us distinguish modifiers from arguments.

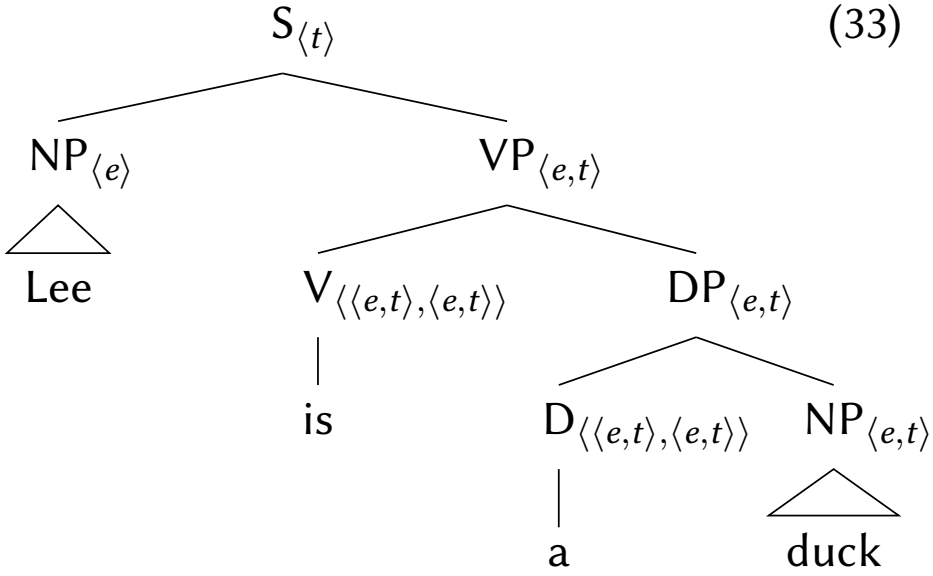
New Idea: The distinction between modifiers and arguments lies in the saturation of a predicate!

- an argument saturates the predicate it combines with
- a modifier does not

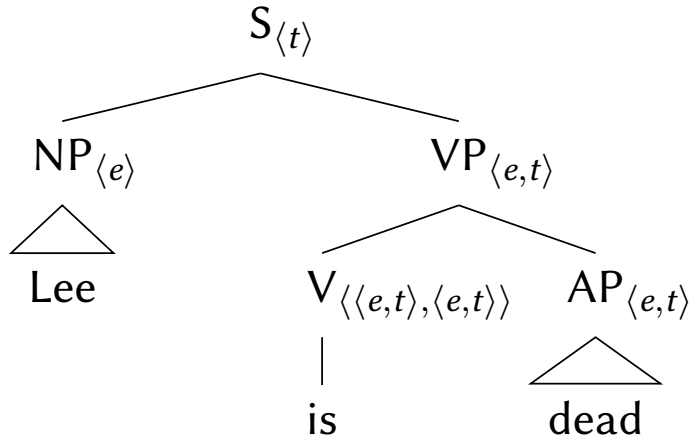
Modification

Let us go back to semantic types. So far we have looked at properties (one-place predicates) separately.

(32)



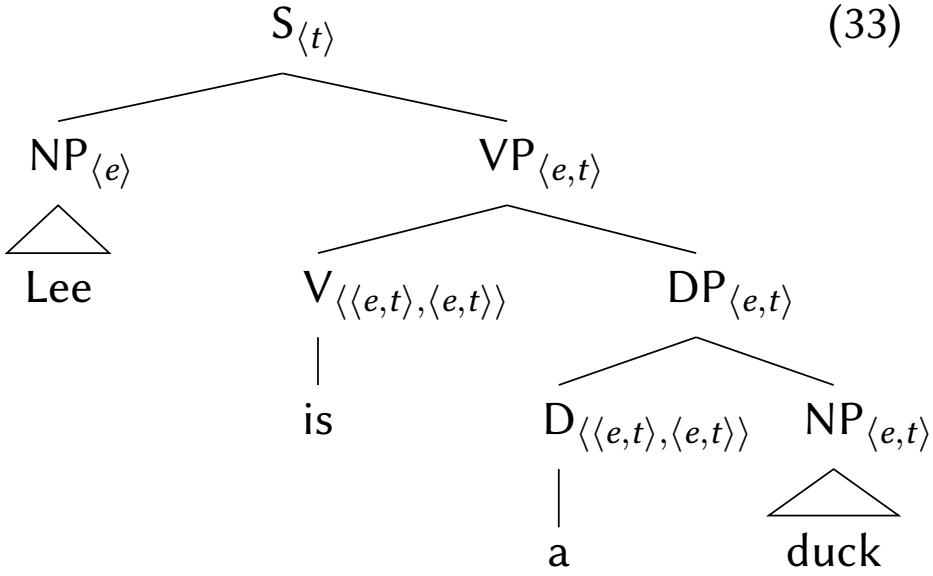
(33)



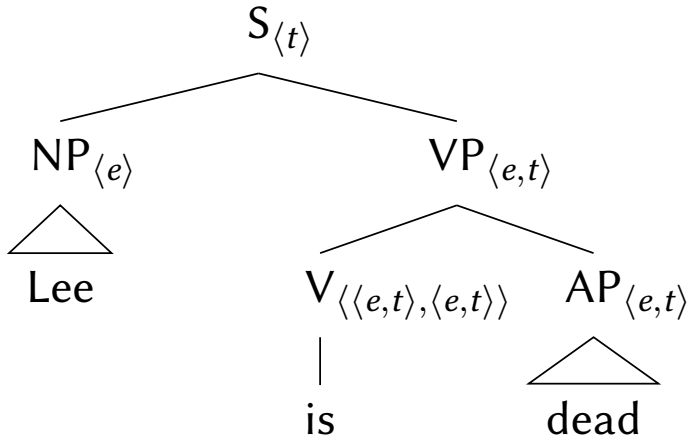
Modification

Let us go back to semantic types. So far we have looked at properties (one-place predicates) separately.

(32)



(33)



Can we derive the denotation for the following sentence?

(34) Lee is a dead duck.

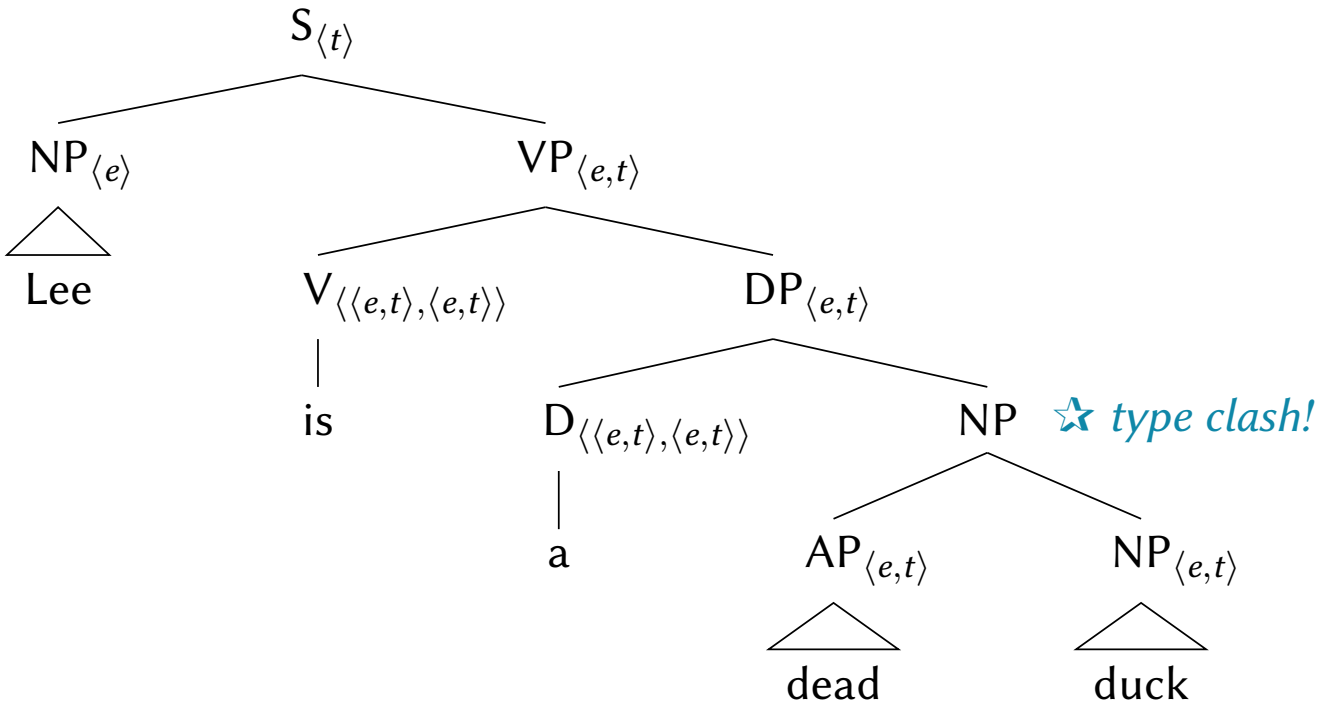
Modification

Can we derive the denotation for the following sentence?

(35) Lee is a dead duck.

No! There is no way of combining *dead* and *duck* directly via functional application.

(36)



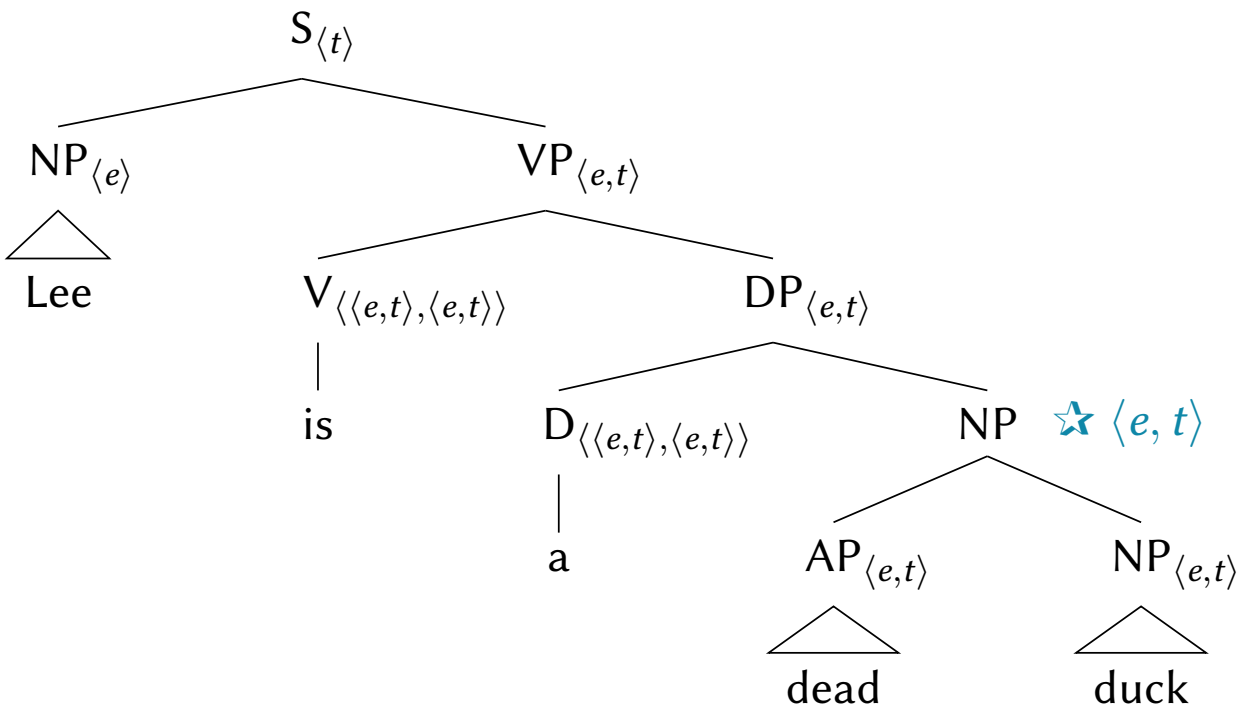
Modification

Arguments reduce the arity of a predicate they combine with; modifiers leave it unchanged.

Modifiers *(McNally, 2016, 243)*

An expression that combines with an unsaturated expression to form another unsaturated expression of the same type.

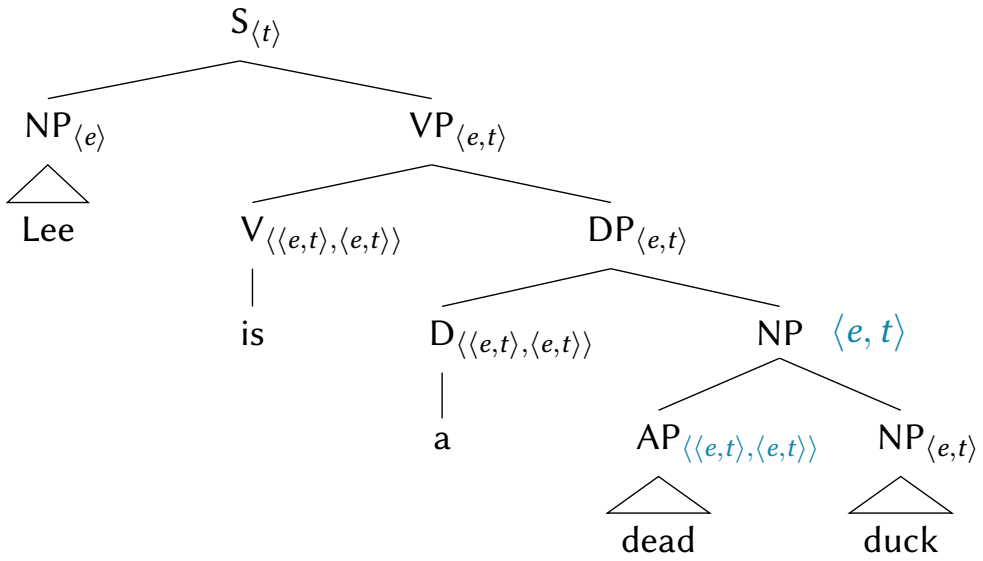
(37)



Modification: the lexical approach

One way of giving *dead duck* the type $\langle e, t\rangle$ is by changing the semantics of *dead* (or *duck*) into $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

(38)



(39) a. $[\![duck]\!] = \lambda x_{\langle e\rangle}[\text{duck}(x)]$ *Lex*

b. $[\![dead_{attr}]\!] = \lambda P_{\langle e,t\rangle}\lambda y_{\langle e\rangle}[\text{dead}(y) \wedge P(y)]$ *Lex*

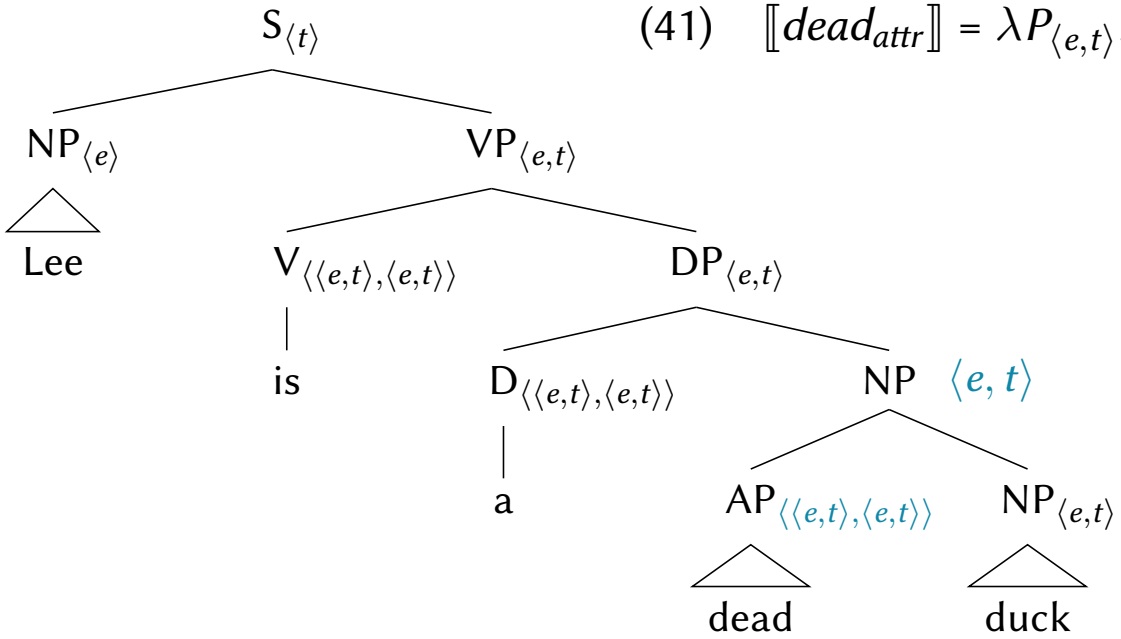
c. $[\![dead_{attr}]\!]([\![duck]\!]) = \lambda P_{\langle e,t\rangle}\lambda y_{\langle e\rangle}[\text{dead}(y) \wedge P(y)](\lambda x_{\langle e\rangle}[\text{duck}(x)])$ *FA*

d. $[\![dead_{attr}\ duck]\!] = \lambda y_{\langle e\rangle}[\text{dead}(y) \wedge \lambda x_{\langle e\rangle}[\text{duck}(x)](y)]$
$= \lambda y_{\langle e\rangle}[\text{dead}(y) \wedge \text{duck}(y)]$ *λ-C*

Modification: the lexical approach

One way of giving *dead duck* the type $\langle e, t\rangle$ is by changing the semantics of *dead* (or *duck*) into $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

(40)

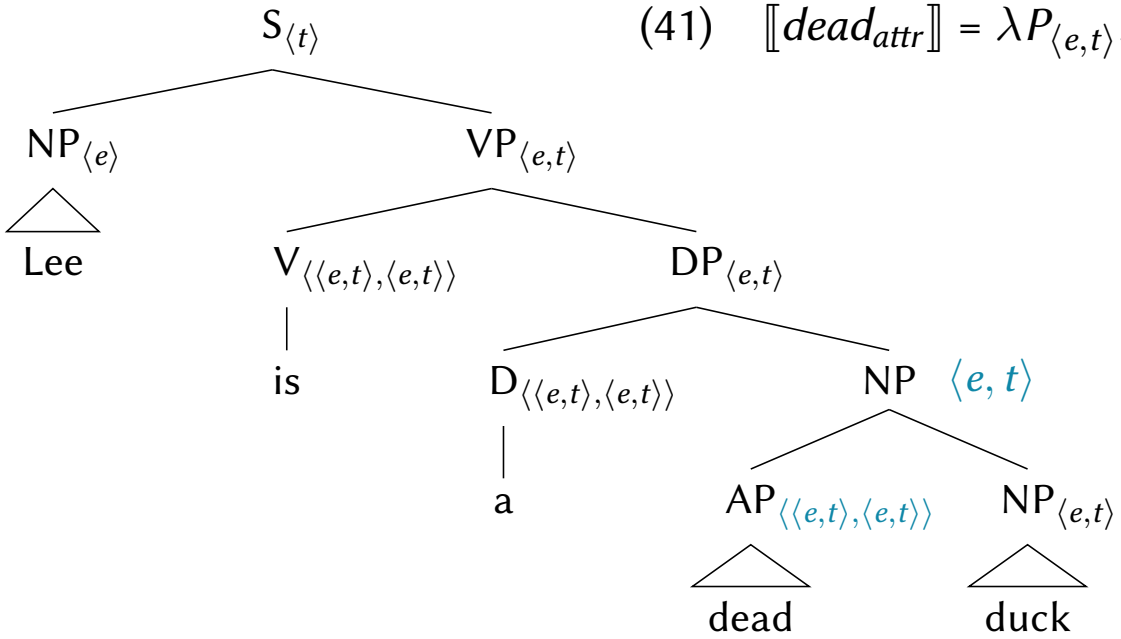


(41) $[\![dead_{attr}]\!] = \lambda P_{\langle e,t\rangle} \lambda x_{\langle e\rangle} [\text{dead}(x) \wedge P(x)]$

Modification: the lexical approach

One way of giving *dead duck* the type $\langle e, t\rangle$ is by changing the semantics of *dead* (or *duck*) into $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

(40)



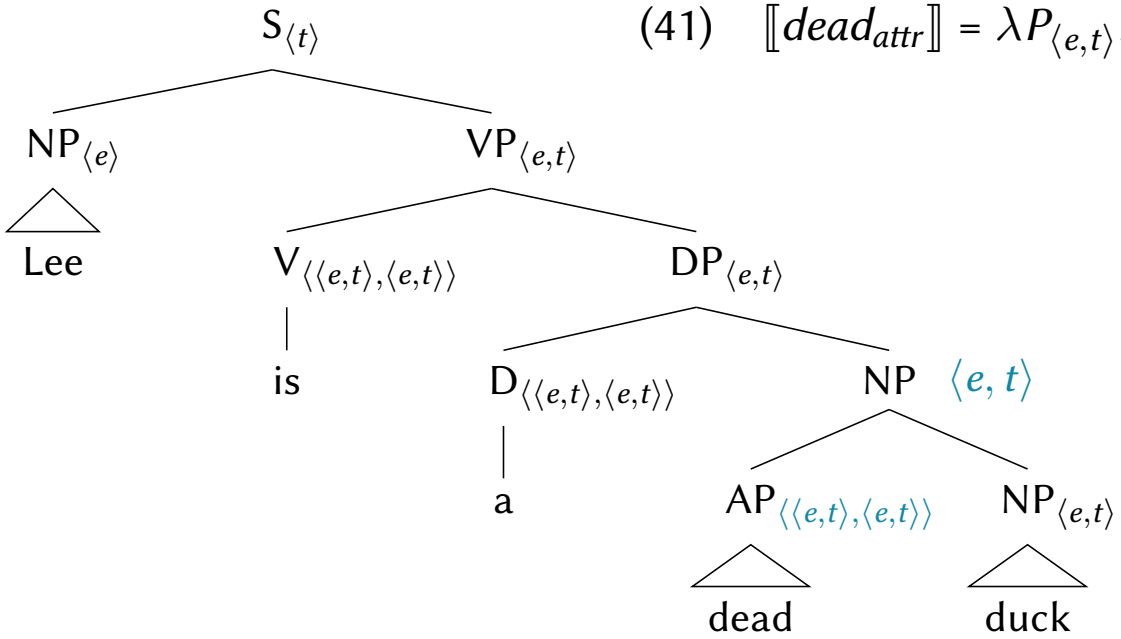
(41) $[\![dead_{attr}]\!] = \lambda P_{\langle e,t\rangle} \lambda x_{\langle e\rangle} [\text{dead}(x) \wedge P(x)]$

Advantage:
Semantic composition is done exclusively by functional application (Frege's conjecture).

Modification: the lexical approach

One way of giving *dead duck* the type $\langle e, t\rangle$ is by changing the semantics of *dead* (or *duck*) into $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

(40)



(41) $[\![dead_{attr}]\!] = \lambda P_{\langle e,t\rangle} \lambda x_{\langle e\rangle} [\text{dead}(x) \wedge P(x)]$

Advantage:

Semantic composition is done exclusively by functional application (Frege's conjecture).

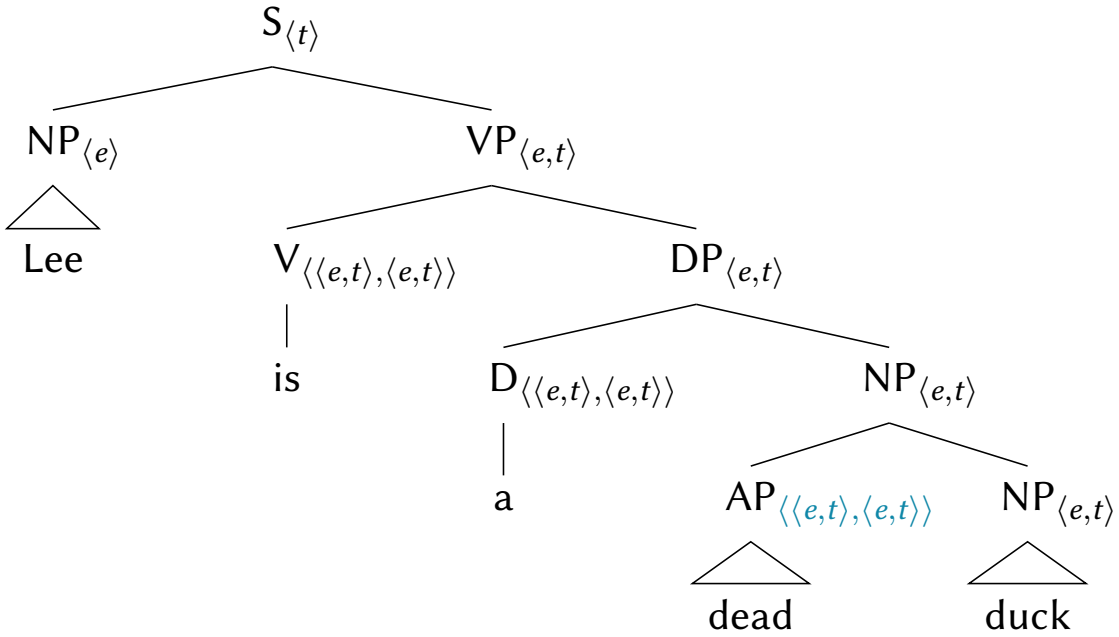
Disadvantage:

Adjectives are ambiguous between a predicative meaning and an attributive meaning.

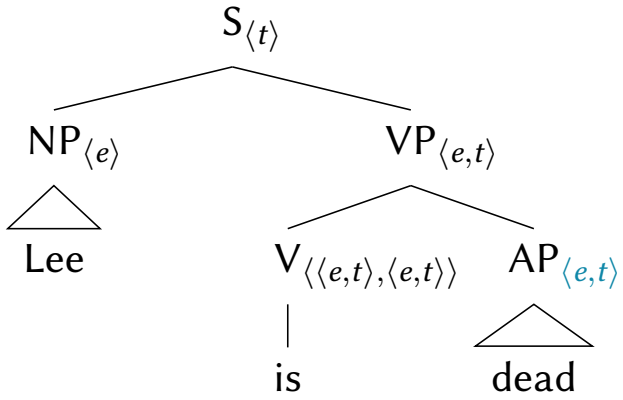
Modification: the lexical approach

Ambiguous meaning for *dead*:

(42) $[\![dead_{attr}]\!] = \lambda P_{\langle e,t\rangle} \lambda x_{\langle e\rangle} [\text{dead}(x) \wedge P(x)]$



(43) $[\![dead_{pred}]\!] = \lambda x_{\langle e\rangle} [\text{dead}(x)]$



References

McNally, L. (2016). Modification. In Aloni, M. and Dekker, P., editors, *The Cambridge Handbook of Formal Semantics*, Cambridge Handbooks in Language and Linguistics, pages 442–464. Cambridge University Press, Cambridge.