Syntax-Semantics Interface

λ ^

λ lambda

Manfred Sailer

2.11.2016

Homework

- Read Chapter 2, Sections ?? of Levine et al. (in prep.) (available on olat as lrs-vol1-161020.pdf)
- Watch the videos below
- Provide one true and two false atomic formula with respect to the model from our class meeting and provide a step-by-step computation of the truth values of these three formulae.

O logic robin

λ

World

O language Robin

$kiss_2(sheldon, penny)$

$[\![kiss_2(sheldon, penny)]\!] = true$

iff $\langle [\![sheldon]\!], [\![penny]\!] \rangle$ is in $[\![kiss_2]\!]$

iff $\langle I(sheldon), I(penny) \rangle$ is in $I(kiss_2)$

iff $\langle$  $\rangle$ is $\{\langle x, y \rangle \mid x \text{ kissed } y\}$

Since this is not the case,

$[\![kiss_2(sheldon, penny)]\!]$ is false

$[\![wait\text{-}tables_1(sheldon)]\!] = true$

iff $\langle [\![sheldon]\!] \rangle$ is in $[\![wait\text{-}tables_1]\!]$

iff $\langle I(sheldon) \rangle$ is in $I(waits\text{-}tables_1)$

iff $\langle \underline{sheldon} \rangle$ is in $\{\langle penny \rangle\}$

this is not the case, so the formula is false

is false

L. is a scientist and P. is a scientist

$[\![\text{scientist}_1(\text{leonard}) \wedge \text{scientist}_1(\text{penny})]\!]$

iff $[\![sc_1(e)]\!] = \text{true}$ <u>and</u> $= \text{true}$

$[\![sc_1(p)]\!] = \text{true}$.

iff $\langle [\![\text{leonard}]\!] \rangle$ is in $[\![\text{scient}]\!]$ $\overset{\text{and}}{\cancel{A}}$ $\langle [\![\text{penny}]\!] \rangle$ is in $[\![sc]\!]$

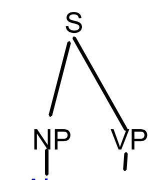
$[\![\varphi \wedge \psi]\!] = \text{true}$

phi psi

iff $[\![\varphi]\!] = \text{true}$ <u>and</u> $[\![\psi]\!] = \text{true}$

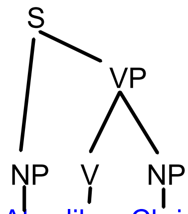
Syntax-semantics interface

snore1(~~alex~~ penny)



~~Alex~~ Penny snored.

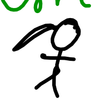
like2(alex,chris)



Alex likes Chris.

The meaning of a complex expression is a function of the meaning of its component parts and the way in which they combine.

Lexicon: Logic [...] World



~~Alex~~ Penny: penny

~~Chris~~: Leonard

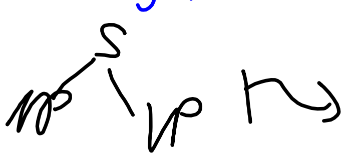
snored:

likes:

Penny $\rightsquigarrow$ penny

Leonard $\rightsquigarrow$ leonard

Snored $\rightsquigarrow$ snore1



Penny $\rightsquigarrow$ penny

snored $\rightsquigarrow$ $snore_1$

P. snored $\rightsquigarrow$ $snore_1(penny)$

$[_S\ NP\ VP]$ $\rightsquigarrow$ $VP^*(NP^*)$

VP^*: the translation of VP

NP^*: " of NP

C. likes P.

L. $\rightsquigarrow$ leonard $like_2(leonard, penny)$

P. $\rightsquigarrow$ penny

likes $\rightsquigarrow$ $like_2$

$[_S\ NP\ VP]$ $\rightsquigarrow$ $VP^*(NP^*)$

$[_{VP}\ V\ NP]$ $\rightsquigarrow$ $V^*(NP^*)$

Funktions and lambdas

snore

like

λx.snore1(x)

λ

[~~λx~~.snore1(x)](alex)

$= snore_1(alex)$

$\lambda x.\ snore_1(x)$

Function that ~~for~~ for each value of x it returns $snore_1(x)$

Formula $snore_1(x)$

Lambda abstraction

$\lambda x.snore_1(x)$

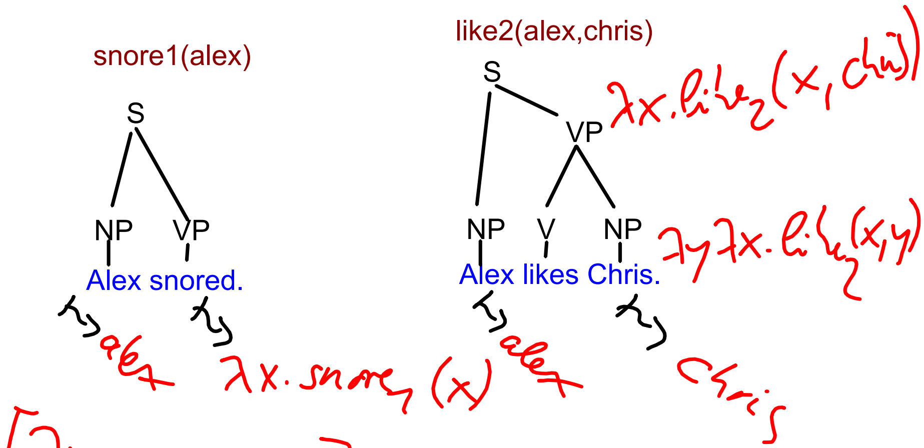
~~Apply~~ Apply a function to an argument:

$[\lambda x.snore_1(x)](penny)$

λ-reduction / λ-conversion

$snore_1(penny)$

Syntax-semantics interface



$[\lambda x.\, snore_1(x)](alex)$

$= snore_1(alex)$

$VP^*: [\cancel{\lambda y}.\lambda x.\, like_2(x,y)](chris)$

$= \lambda x.\, like_2(x, chris)$

$S^* = VP^*(NP^*)$

$[\lambda x.\, like_2(x, chris)](alex)$

$= like_2(alex, chris)$

Magic λ

[λz.z](alex) $=$ alex

Identity function

[λy.y(alex)](λx.snore(x)) $= [\lambda \cancel{y}.y(alex)](\lambda x.snore_1(x))$

$= [\lambda \cancel{x}.snore_1(x)](alex)$

$= snore_1(alex)$

[λyλz.y(z)](λx.snore(x))(alex)

$= [\lambda z.[\lambda x.snore_1(x)](z)](alex)$

$= [\lambda x.snore_1(x)](alex)$

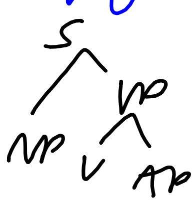
$= snore_1(alex)$

Copula and other "meaningless" elements

Alex $\mapsto$ alex

happy $\mapsto \lambda x.happy_1(x)$

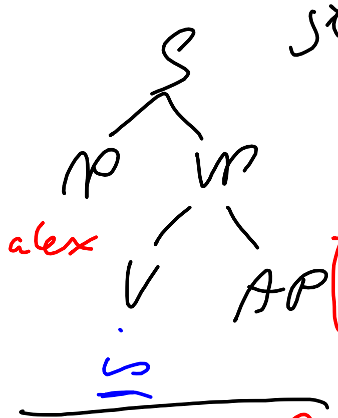
is $\mapsto \lambda y.y$



Alex is happy.

Alex waited for Chris.

$happy_1(alex)$



S^*: $happy_1(alex)$

$VP^*(NP^*)$

$VP^*(alex) = happy_1(alex)$

$\underline{[\lambda x.happy_1(x)](alex)}$

$\lambda x.happy_1(x)$

$VP^* = V^*(AP^*)$

$V^*(\lambda x.happy_1(x)) = \lambda x.happy_1(x)$

$\lambda x.happy_1(x)$

$V^*(\lambda x.happy_1(x))$

$= \lambda x.happy_1(x)$

$\lambda z.z$

is happy:

$[\lambda y.y](\lambda x.happy_1(x))$

$= \lambda x.happy_1(x)$

Alex is happy:

$[\lambda x.happy_1(x)](alex)$

$= happy_1(alex)$

Alex $\mapsto$ $alex$

Chris $\mapsto$ $chris$

waited $\mapsto$ $\lambda y.\lambda x.wait_2(x,y)$

for $\mapsto$ $\lambda y.y$

$wait_2(alex, chris)$

for Chris: $[\lambda y.y](chris) = chris$

waited for Chris: $[\lambda y.\lambda x.w_2(x,y)](chris)$

$= \lambda x.wait_2(x, chris)$

A. waited for Chris:

$[\lambda x.wait_2(x, chris)](alex)$

$= wait_2(alex, chris)$

Logical connectives

and: $\varphi \wedge \psi$

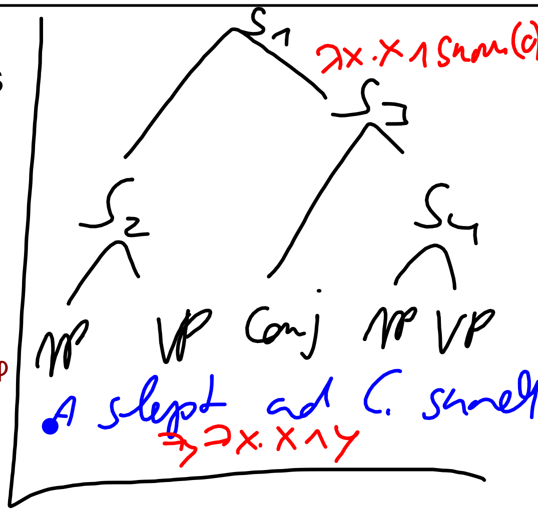
or: $\varphi \vee \psi$

if...then: $\varphi \supset \psi$

not/it is not the case that: $\neg\varphi$

Alex slept and Chris snored.

sleep1(alex) $\wedge$ snore1(chris)



Translation of and?

S_1^*: $sleep_1(alex) \wedge snore_1(chris)$

$= S_3^*(S_2^*)$

$= S_3^*(sleep_1(alex))$

$\lambda x. x \wedge snore_1(chris)$

$S_3^* = Conj^*(S_4^*)$

$\lambda x. x \wedge snore_1(chris)$

$= Conj^*(snore_1(chris))$

$\lambda y. \lambda x. x \wedge y$

λ and the syntax-semantics interface

- λ allows for a very straightforward interface!
- λ-prefix mimics the syntactic structure

Homework

- For some more background/uses: read Chapter 3 of the Krifka script [on olat]
- Give the syntactic structure and the logical form of the following sentence.
- Provide the semantic translation for the words in the sentences.
- Provide the semantic translation of all nodes in the trees for the sentences.

(i) Chris partied.

(ii) Chris invited Alex.

(iii) Chris invited Alex and ~~partied.~~ Chris partied.

(iv) Chris didn't invite Pat. $\neg$ invite$_2$(chris, pat)