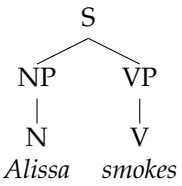
F08: Intro to Composition

Semantics - Ling 331/731
University of Kansas

1 Verbs as functions

(1) *Alissa smokes.*
Here is a woefully simplified sentence structure (we'll get more modern soon enough):



There are two lexical items that feed this structure:

(2) $[\![\textit{Alissa}]\!]$ = Alissa

(3) $[\![\textit{smokes}]\!]$ = ?? *smokes* needs a subject— it is unsaturated.

Question 1. What type of formal objects do unsaturated expressions correspond to?

We can primitively define a verb by the set of objects that do the action it describes.

(4) *Assume a domain:* Let D be the set
{ Alissa, Barack Obama, Prof McKenzie, Simon Cowell, Oprah }

(5) $[\![\textit{smokes}]\!]$ = f : D $\rightarrow$ {1, 0}
for all x $\in$ D, f(x) = 1 if and only if **x smokes**

(6) $[\![\textit{smokes}]\!] = \begin{bmatrix} \text{Alissa} & \rightarrow & 1 \\ \text{Barack Obama} & \rightarrow & 1 \\ \text{Prof. McKenzie} & \rightarrow & 0 \\ \text{Simon Cowell} & \rightarrow & 1 \\ \text{Oprah} & \rightarrow & 0 \end{bmatrix}$

(7) $[\![\textit{smokes}]\!]$ = { x $\in$ D | x smokes }

2 Inserting lexical items

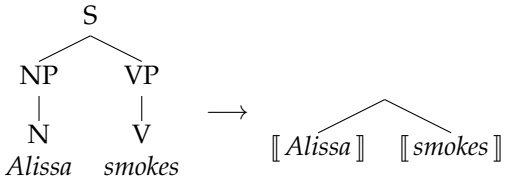
The words *Alissa* and *smokes* are items in the **lexicon**.[1] We can specify the lexicon as containing all items placed at terminal nodes.

Rule 1:

> **Terminal Nodes (TN):**
> If α is a terminal node (at LF), $[\![\alpha]\!]$ is specified in the lexicon.[2]

Rule 1 allows us to interpret the lexical items before any composition is done.

(8)



[1] Technically, *smoke* and *-s* are in the lexicon. However, *-s* is agreement, which has no semantic value. We'll abstract away from that distinction, and treat *smokes* as one morpheme.

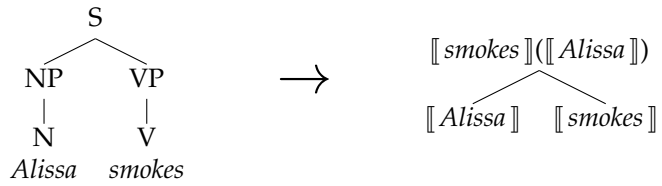
[2] Our semantic rules use Greek letters as variables.

3 Watch out for syntax

Now that we have defined our lexical items, we can compose them. I will write both the LF (on the left) and the semantic composition (on the right).

The semantics always composes from the bottom up. So, we start with the terminal nodes. In this case the N *Alissa* and V *smokes*. In the semantics, we don't have categories or labels; we just use the denotations. Still, it's helpful to talk about the semantic constituents by their syntactic names. We combine the N and the V, and voilà, we interpret the sentence. Right?

(9)



NO!!!! The N and the V do not compose. In fact, they cannot compose.

Question 2. Why not?

> **Syntax-semantics correspondence:**
> A node that has at least one daughter is interpreted as the output of the composition of its daughters at LF.

This correspondence clear for sisters— sisters share a mother in the syntax (hence at LF). You can draw a correspondence between Merge in the syntax and composition in the semantics.

In essence, N and V cannot compose because they do not combine in the syntax. Instead, the N projects to NP, and V projects to VP.[3] The NP and the VP combine.

But the correspondence in this condition is broadly written so as to cover non-branching nodes, which are defined as a node with a single daughter.

[3] We'll ignore the $\overline{\text{X}}$ levels throughout this course.

4 Non-branching nodes

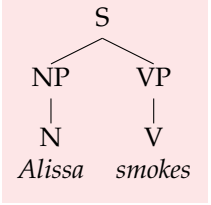
We can observe that a non-branching node has the exact same meaning as its daughter, and capture this observation with our second compositional rule.

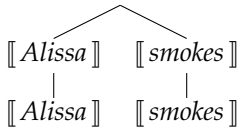
Rule 2:

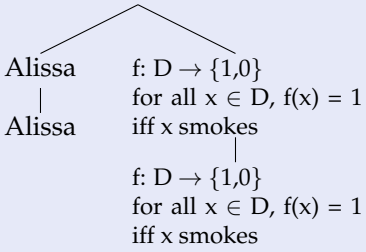
Non-branching nodes (NN):
If α is a non-branching node at LF, and β is its daughter, then $[\![\alpha]\!] = [\![\beta]\!]$

Thus, the VP has the same denotation as V, and the NP has the same denotation as N.

(10)







Note the third tree— it is possible because we can write a semantic tree using the denotations, or their equivalents. The two semantic trees are identical, truth-conditionally.

Now we can combine them. How?

5 Functional Application

We need a new rule of composition, that plugs an individual into a function.

Rule 3:

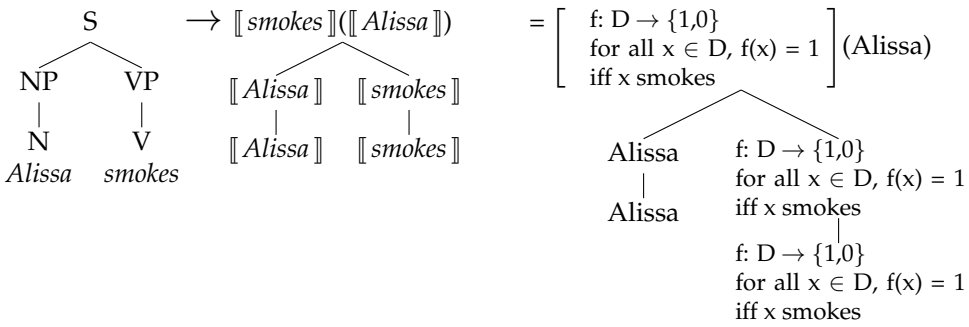
Functional Application (FA):
If α is a branching node whose daughters are $\{\beta, \gamma\}$,
β is a function, and γ is in the domain of β,
then $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$

In English: $[\![\alpha]\!] = [\![\beta]\!]([\![\gamma]\!])$ is read as:
"the denotation of *alpha* is the result of applying the denotation of the function *beta* to the denotation of *gamma*."

Note for later: The definition of Rule 3 is written in a way that it applies to any function and any argument in its domain. As we'll see, we won't always be plugging in individuals.

Functional Application (FA) involves taking an argument plugging it in to a function. That is, we apply the function to it. With our example, the function is $[\![$ *smokes* $]\!]$ and the argument is $[\![$ *Alissa* $]\!]$.

(11)



- The fully saturated sentence denotes a truth-value (1 or 0).
- Functional application does not care what syntactic order the function and argument are in.
- Functional application does not care which component is a head and which is a phrase.
- Functional application does not care which component is the *syntactic* argument of the other.
- In the notation, functions are always written as : function(argument) and typically read as "function of argument"