Semantic Types and Function Application

Ling324
Reading: *Meaning and Grammar*, pg. 87-98
Semantic Types we have specified so far for the fragment of English F1

<table>
<thead>
<tr>
<th>Syntactic Category</th>
<th>Semantic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Truth values (0 or 1)</td>
</tr>
<tr>
<td>N</td>
<td>Individuals</td>
</tr>
<tr>
<td>V_i, VP</td>
<td>Sets of individuals</td>
</tr>
<tr>
<td>V_t</td>
<td>Sets of ordered pairs of individuals</td>
</tr>
<tr>
<td>Conj</td>
<td>Function from pairs of truth values to truth values</td>
</tr>
<tr>
<td>Neg</td>
<td>Function from truth values to truth values</td>
</tr>
</tbody>
</table>
Specifying Semantic Rules in terms of Function Application

• A function takes an input argument from some specified domain and yields an output value.

Applying a function \( f \) to an argument \( x \) yields the value for that argument, which can be written as \( f(x) \).

The mode of combining a function and its argument is called **FUNCTION APPLICATION**.

• The way we have defined the semantics of \( \text{Neg} \) makes use of function application.

\[
\llbracket \text{Neg} \rrbracket^V = \begin{bmatrix} 1 & \rightarrow & 0 \\ 0 & \rightarrow & 1 \end{bmatrix} = \text{the function } f \text{ from truth values to truth values such that: } f(1) = 0 \text{ and } f(0) = 1
\]

• In fact, function application could be used to interpret any syntactic structure with two branches: one branch is interpreted as a function, and the other branch is interpreted as a possible argument of the function.

\[
\llbracket \text{A} \rrbracket^V = \llbracket B \rrbracket^V (\llbracket C \rrbracket^V)
\]

\[
\overline{\text{B}} \; \text{C}
\]
Intransitive Verb

- $\llbracket \text{is cute} \rrbracket^V = \{ x : x \text{ is cute in } V \}$

  For example, let Universe = \{Fiona, Patsy, Jenny, John\}
  $\llbracket \text{is cute} \rrbracket^V = \{ \text{Fiona, Jenny} \}$

- $\llbracket \text{is cute} \rrbracket^V$ = the function $f$ from individuals to truth values such that:
  $f(x) = 1$ if $x \in \{ x : x \text{ is cute in } V \}$, and $f(x) = 0$ otherwise.

  $\llbracket \text{is cute} \rrbracket^V = \begin{bmatrix}
    \text{Fiona} & \rightarrow & 1 \\
    \text{Patsy} & \rightarrow & 0 \\
    \text{Jenny} & \rightarrow & 1 \\
    \text{John} & \rightarrow & 0
  \end{bmatrix}

  (= the characteristic function of $\{ x : x \text{ is cute in } V \}$)

- Characteristic function

Any function that assigns one of two distinct values (0 or 1) to the members of a domain is called **characteristic function**.

Each subset of the domain defines such a function uniquely, and any such function corresponds to a unique subset of the domain.

This means that we can use sets and characteristic function of that set interchangeably when defining the semantic value of intransitive verbs.
Intransitive Verb (cont.)

- Semantic types

  \( e \) (entity): the type of individuals.

  \( t \) (truth value): the type of truth values.

  \( < e, t > \): the type of functions from individuals into truth values.

- Intransitive verb combines with the subject, by function application, and returns a truth value.

  \( [[\text{is cute}]]^V([[\text{John}]]^V) = 1 \text{ or } 0 \) (depending on the situation \( V \))

- **QUESTION**: Provide the semantic value for *is hungry* and *is boring* in terms of set notation, and functional notation.
Transitive Verb

• $\llbracket \text{likes} \rrbracket^V = \{ < x, y > : x \text{ likes } y \text{ in } V \}$

For example, let Universe = \{ Fiona, Patsy, Jenny \}
$\llbracket \text{likes} \rrbracket^V = \{ < \text{Fiona, Patsy }>, < \text{Patsy, Jenny }>, < \text{Jenny, Jenny }> \}$

• The characteristic function of $\llbracket \text{likes} \rrbracket^V$

\[
\begin{bmatrix}
< \text{Fiona, Fiona } > & \rightarrow & 0 \\
< \text{Fiona, Patsy } > & \rightarrow & 1 \\
< \text{Fiona, Jenny } > & \rightarrow & 0 \\
< \text{Patsy, Fiona } > & \rightarrow & 0 \\
< \text{Patsy, Patsy } > & \rightarrow & 0 \\
< \text{Patsy, Jenny } > & \rightarrow & 1 \\
< \text{Jenny, Fiona } > & \rightarrow & 0 \\
< \text{Jenny, Patsy } > & \rightarrow & 0 \\
< \text{Jenny, Jenny } > & \rightarrow & 1 \\
\end{bmatrix}
\]
Transitive Verb (cont.)

• Schönfinkelization: Turning n-ary functions into multiple embedded unary functions.

Left-to-right

- Fiona $\rightarrow$ Patsy $\rightarrow$ 0
- Fiona $\rightarrow$ Jenny $\rightarrow$ 0
- Fiona $\rightarrow$ 0
- Patsy $\rightarrow$ 0
- Patsy $\rightarrow$ 0
- Patsy $\rightarrow$ 0
- Jenny $\rightarrow$ 0
- Jenny $\rightarrow$ 1
- Jenny $\rightarrow$ 1

Right-to-left

- Fiona $\rightarrow$ Patsy $\rightarrow$ 0
- Fiona $\rightarrow$ Jenny $\rightarrow$ 0
- Fiona $\rightarrow$ 1
- Patsy $\rightarrow$ 0
- Patsy $\rightarrow$ 0
- Patsy $\rightarrow$ 0
- Jenny $\rightarrow$ 0
- Jenny $\rightarrow$ 1
- Jenny $\rightarrow$ 1

• Which Schönfinkelization is consistent with the principle of compositional semantics? Left-to-right or Right-to-left?
Transitive Verb (cont.)

- \([\text{likes}]^V = \text{the function } f \text{ from individuals to characteristic functions such that: } f(y) = g_y, \text{ the characteristic function of } \{x : x \text{ likes } y \text{ in } V\}\). 

- Type of functions from individuals to characteristic functions
  
  \(\langle e, \langle e, t \rangle \rangle\)

- Transitive verb combines with a direct object, by function application, and returns a characteristic function of a set.

  \([\text{likes}]^V(\text{[Vivian]}^V) = \text{the function } f \text{ from individuals to truth values such that: } f(x) = 1 \text{ if } x \in \{x : x \text{ likes Vivian in } V\}, \text{ and } f(x) = 0 \text{ otherwise.}\)
Logical Connectives: \textit{and}

- Binary function
  \[
  \begin{bmatrix}
    1, 1 & \rightarrow & 1 \\
    1, 0 & \rightarrow & 0 \\
    0, 1 & \rightarrow & 0 \\
    0, 0 & \rightarrow & 0 \\
  \end{bmatrix}
  \]

- Schönfinkelization
  
  Assume the following two syntactic rules:

  \begin{enumerate}
  \item \( S \rightarrow S \text{ conjP} \)
  \item \( \text{conjP} \rightarrow \text{conj S} \)
  \end{enumerate}

- Unary function
  \[
  \begin{bmatrix}
    1 & \rightarrow & \begin{bmatrix}
      1 & \rightarrow & 1 \\
      0 & \rightarrow & 0 \\
    \end{bmatrix} \\
    0 & \rightarrow & \begin{bmatrix}
      1 & \rightarrow & 0 \\
      0 & \rightarrow & 0 \\
    \end{bmatrix} \\
  \end{bmatrix}
  \]

- Type of functions from truth values to functions from truth values to truth values
  \[
  < t, < t, t > >
  \]
Calculating Truth Conditions using Functional Approach

• Semantic Rules

(2) a. Pass-up:
   If $\Delta$ is a nonbranching node that dominates $a$,
   then $[\Delta]^V = [a]^V$

b. Function Application
   If $\Delta$ is a branching node with daughters $a$ and $b$,
   and $[a]^V$ is a function whose domain contains $[b]^V$,
   then $[\Delta]^V = [a]^V([b]^V)$.

• EXERCISE: For the following examples, calculate their truth conditions compositionally using the semantic rules above.

(3) a. Bob is hungry.
   b. Kitty likes Vivian.
### Specifying Semantic Types in terms of Functional Types

<table>
<thead>
<tr>
<th>Syntactic Category</th>
<th>Semantic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>$t$</td>
</tr>
<tr>
<td>N</td>
<td>$e$</td>
</tr>
<tr>
<td>$V_i$, VP</td>
<td>$&lt; e, t &gt;$</td>
</tr>
<tr>
<td>$V_t$</td>
<td>$&lt; e, &lt; e, t &gt;&gt;$</td>
</tr>
<tr>
<td>Conj</td>
<td>$&lt; t, &lt; t, t &gt;&gt;$</td>
</tr>
<tr>
<td>Neg</td>
<td>$&lt; t, t &gt;$</td>
</tr>
</tbody>
</table>