PART 1: Syntax-Semantics Interface

1. Syntax

(1.1) If $\Delta_1, \ldots, \Delta_n$ are deep structures of the corresponding syntactic categories $\kappa_1, \ldots, \kappa_n$, then the result of applying the ($n$-place) syntactic construction $C$ to $(\Delta_1, \ldots, \Delta_n)$ will be a deep structure of category $\kappa_{n+1}$.

Notation: $(C, \kappa_1, \ldots, \kappa_n, \kappa_{n+1})$

(1.2) old men and women

(1.3)

```
  Adj
  /   \\
old
  \   \\
N
```

(USC) Uniqueness of Structure Constraint on an algebra $(\Sigma, (C_i)_{i \in I})$:

If $C_i(\Delta_1, \ldots, \Delta_n) = C_j(\Delta'_1, \ldots, \Delta'_m)$, then $i = j$, and $(\Delta_1, \ldots, \Delta_n) = (\Delta'_1, \ldots, \Delta'_m)$

(and hence $C_i = C_j$, $m = n$, $\Delta_1 = \Delta'_1$, $\ldots$, $\Delta_n = \Delta'_m$).

(1.4) $(N \text{ Adj old Adj N (N men Conj and Conj N women)N})_N$

(1.5) $\exists \text{ Jones ( seeks a horse such } v_7 \text{ that } \exists \text{ it } v_7 \text{ speaks } \exists \exists ) \exists$

(1.6) John seeks a unicorn, 4

```
John
    seek a unicorn, 5
        seek a unicorn, 2
            unicorn
```

Definition

A (deep) syntax is a quintuple $(\Sigma, (C_i)_{i \in I}, (L_k)_{k \in K}, R, S)$, where

- $(\Sigma, (C_i)_{i \in I})$ satisfies (USC);
- the Lexicon $\bigcup_{k \in K} L_k$ generates the set $\Sigma$ of (syntactic) structures;
  (i.e. no lexical item is a value of an operation $C_i$ and $\Sigma$ is the smallest set that contains the lexicon and is closed under all $C_i$);
- the elements of $R[ules]$ are as in (1.1) – where $C$ is one of the $C_i$ and all $\kappa_j \in K[ategories]$;
- $S[entence] \in K$.

If $\Delta \in \Sigma$ and $k^* \in K$, then $\Delta$ is of category $k^*$ if either $\Delta \in L_{k^*}$ (in which case $\Delta$’s rank $\rho(\Delta)$ is 0), or $\Delta = C_i(\Delta_1, \ldots, \Delta_n)$ (and thus $\rho(\Delta) = \max(\rho(\Delta_1), \ldots, \rho(\Delta_1)) + 1$), $\Delta_1, \ldots, \Delta_n$ are of categories $k_1, \ldots, k_n$, respectively, and $(C_i, k_1, \ldots, k_n, k^*) \in R$. 
2. Compositionality

(2.1) The meaning of a complex expression can be determined from the meanings of its parts.

(2.2) If \( S_1 \) and \( S_2 \) are sentences (of some fixed language), then so is ‘\((S_1 \text{ and } S_2)\)’.

(2.3) \( C(\Delta, \Delta') = '(\Delta \text{ and } \Delta')', \) whenever \( \Delta \) and \( \Delta' \) are structures.

(2.4) If \( C(\Delta_1, \ldots, \Delta_n) \) is a structure, then its meaning is uniquely determined by the meanings of \( \Delta_1, \ldots, \Delta_n \) and \( C \).

(2.5) For each \((n\text{-place})\) syntactic construction \( C \) there is a corresponding \((n\text{-place})\) meaning combination \( M \) such that the meaning of any structure \( C(\Delta_1, \ldots, \Delta_n) \) is \( M(b_1, \ldots, b_n) \), where \( b_1 \) is \( \Delta_1 \)'s meaning, etc.

(2.6) \( \mu(C(\Delta_1, \ldots, \Delta_n)) = M(\mu(\Delta_1), \ldots, \mu(\Delta_n)) \)

(2.7) \( \text{Jones knows that } S, C_0 \)

\( \text{Jones knows that } S, C_1 \)

\( \text{knows that } S \)

(2.8) \( M_0(\mu(\text{Jones}), M_1(\mu(\text{love}), \mu(\text{Smith}))) \)

(2.9) \( \mu(\text{Jones knows that } S), M_0 \)

\( \mu(\text{Jones}) \)

\( \mu(\text{knows that } S), M_1 \)

\( \mu(\text{knows}) \)

\( \mu(\text{that } S) \)

(2.10) \( \mu(\text{Jones knows that } S'), M_0 \)

\( \mu(\text{Jones}) \)

\( \mu(\text{knows that } S'), M_1 \)

\( \mu(\text{knows}) \)

\( \mu(\text{that } S') \)

(2.11) \( \mu(\text{that } S) = \mu(\text{that } S') \Rightarrow \mu(\text{Jones knows that } S) = \mu(\text{Jones knows that } S') \)

(2.12) \( L_1 = \{0, \ldots, 9\}; \) \( L_n = \emptyset \ (n \neq 1) ; \) \( C_n(\Delta, \Delta') = '[\Delta_1 \Delta]') \( R = \{(C_n, 1, n, n+1) \mid n \geq 1\} . \)

(2.13) \( M_n(x, y) = 10^nx + y \)

(2.14) \( \mu(C_2(7, C_1(1,2))) \)

\( = M_2(\mu(7), M_1(\mu(1), \mu(2))) \)

\( = M_2(7, M_1(1, 2)) \)

\( = 10^2 \times 7 + 10^1 \times 1 + 2 \)

\( = 712 \)
Definitions
Given a syntax \((\Sigma, (C_i)_{i \in I}, (L_k)_{k \in K}, R, S)\), a corresponding semantics is a triple \((B, \mu_0, (M_i)_{i \in I})\), where \(B\) is some non-empty set; \(\mu_0: \bigcup_{k \in K} L_k \rightarrow B\); and \((M_i)_{i \in I}\) is similar to \((C_i)_{i \in I}\). Given any \(\Delta \in \Sigma\), then either \(\Delta \in L_k^*\) and its meaning (according to \((B, \mu_0, (M_i)_{i \in I})\)) is \(\mu_0(\Delta)\); or else \(\Delta = C_i(\Delta_1, ..., \Delta_n)\) and its meaning […] is \(\mu(C(\Delta_1, ..., \Delta_n)) = M(\mu(\Delta_1), ..., \mu(\Delta_n))\), where \(\mu(\Delta_1), ..., \mu(\Delta_n)\) are the respective meanings […] of \(\Delta_1, ..., \Delta_n\).

\[(2.15) \quad (\exists x) \ P(x)\]

\[(2.16) \quad P(a)\]

\[(2.17) \quad \text{The meaning of a complex expression is determined by the meanings of (certain) less complex expressions.}\]

\[(2.18) \quad (\forall x) \ P(x)\]

\[(2.19) \quad \text{There exists a function } f \text{ which can be applied to pairs consisting of syntactic constructions and sets of meanings such that the meaning of a complex structure } \Delta \text{ of the form } C(\Delta_1, ..., \Delta_n) \text{ equals the value } f(C, b[X]), \text{ where } X \text{ is some set of structures of ranks less than } \Delta.\]

**PART 2: Meaning and Reference**

3. Local Perspective

(3.1)

<table>
<thead>
<tr>
<th>Syntactic Category</th>
<th>Type of extension</th>
<th>Example</th>
<th>Extension of example</th>
</tr>
</thead>
<tbody>
<tr>
<td>proper name</td>
<td>individual (bearer)</td>
<td>Fritz</td>
<td>Fritz Hamm</td>
</tr>
<tr>
<td>definite description</td>
<td>individual (described)</td>
<td>the fifth-biggest city of France</td>
<td>Nice</td>
</tr>
<tr>
<td>count nouns</td>
<td>set (of individuals)</td>
<td>table</td>
<td>set of tables</td>
</tr>
<tr>
<td>intransitive verb</td>
<td>set (of individuals)</td>
<td>sleep</td>
<td>set of sleepers</td>
</tr>
<tr>
<td>transitive verb</td>
<td>set of pairs (of individuals)</td>
<td>eat</td>
<td>set of pairs (eater,food)</td>
</tr>
<tr>
<td>ditransitive verb</td>
<td>set of triples (of individuals)</td>
<td>give</td>
<td>set of triples (giver,recipient, gift)</td>
</tr>
<tr>
<td>sentence</td>
<td>truth value (Ø or {Ø})</td>
<td>snow is white</td>
<td>1</td>
</tr>
</tbody>
</table>
(3.2) Jones knows that snow is white.
(3.3) Jones knows that grass is green.

(i) \( \{ t, e \} \subseteq \text{IT} \), and \( \{ (a,b), (s, b) \} \subseteq \text{IT} \) if \( (a, b) \subseteq \text{IT} \).

(ii) \( E_t = \{ 0, 1 \} \), \( E_e = \cup_k E_{(a,b)} = E_{(s,a)} = E_t \).

(iii) \( I_a = E_{(s,a)} \), \( M_a = I_a^c \).

(iv) A point of reference is an element of \( W \times C \).

(v) A property is an element of \( I_e \).

(vi) \( P \) is a subproperty of \( Q \) iff \( P(w) \subseteq Q(w) \), for any \( w \in W \).

Definition

A local (Fregean) language is a quintuple \( (\Sigma, f, (M_i)_{i \in I}, \mu_0, \Delta) \), where

- \( \Sigma = (\Sigma, (C_i)_{i \in I}, (L_k)_{k \in K}, R, S) \) is a syntax;
- \( f: K \rightarrow \text{IT} \) is a function (type assignment) such that \( f(S) = t \);
- \( \mu_0: \cup_k L_k \rightarrow \cup_{a \in \text{IT}} M_a \) is a function (lexical meaning assignment) such that \( \mu_0(\delta) \in M_a \) whenever \( \delta \in L_k \) and \( f(k) = a \);
- \( (M_i)_{i \in I} \) is a family of meaning operations (similar to \( (C_i)_{i \in I} \) ) such that \( M_i(b_1, \ldots, b_n) \in M_{(i,k_i)} \) whenever \( b_1 \in M_{(i,k_1)}, \ldots, b_n \in M_{(i,k_n)} \) and \( (M_i, k_1, \ldots, k_n, k_0) \in R \).
- \( \Delta \) is the diagonal, i.e. the set of \( (w, c) \in W \times C \) such that \( w = w_c \).

\( \Rightarrow \) If \( \Delta \) is of category \( k \), then its meaning is in \( M_{(i,k)} \).

(3.15) \( M(b_1, \ldots, b_n) (c) (w) = M'(b_1(c), \ldots, b_n(c)) (c) (w) \)

(3.16) \( M(b_1, \ldots, b_n) (c) = M'(b_1(c), \ldots, b_n(c)) (c) \)

(3.17) \( \delta \) is deictic iff \( \mu(\phi)(c)(w) = 1 \), for all \( w, w' \in W \).

(3.18) \( \delta \) is a hyponym of \( \delta' \) (in a local language \( L \)) iff \( \mu_0(\delta)(c)(w) \) is a subproperty of \( \mu_0(\delta')(c)(w) \) whenever \( (c, w) \) is a point of \( \{ \text{reference} \} \times \{ \text{utterance} \} \).

(3.19) \( \phi \) of category \( S \) is an a priori truth iff \( \mu(\phi)(c)(w) = 1 \), for all \( (w, c) \in \Delta \).
4. Global Perspective

(4.1) The meaning of **pebble** is that (local) meaning $b$ of type $<e,t>$ such that for any point of reference $(w,c)$ and any individual $x$ the following holds:

$$b(c)(w)(x) = \begin{cases} 
1, & \text{if } x \text{ is a pebble with respect to the relevant parameters of } <w,c>; \\
0, & \text{otherwise.}
\end{cases}$$

(4.2) The meaning of **stone** is that closed meaning $b$ of type $(e,t)$ such that for any point of reference $(w,c)$ and any individual $x$ the following holds:

$$b(c)(w)(x) = \begin{cases} 
1, & \text{if } x \text{ is a stone with respect to the relevant parameters of } <w,c>; \\
0, & \text{otherwise.}
\end{cases}$$

**Definitions**

(i) An **ontology** is a pair $(E,C)$ where $C \neq \emptyset$ and there are non-empty sets $D$ and $W$ such that $E = (E_a)_{a \in \text{IT}}$ satisfies the equations (3.14)(ii).

(ii) An **ersatz (Fregean) language** based on an ontology $(E,C)$ is a quintuple that is like a local language except that $E$ and $C$ play the respective rôles of extensions and contexts.

(iii) A **global (Fregean) language** is a class of ersatz local languages that share the same syntax and type assignment.

(iv) $\delta$ is a **deictic a hyponym of $\delta'$** in a global language iff $\delta$ is a hyponym of $\delta'$ in an a priori truth each of its members.

(4.3) $\phi$ of category $S$ is **contingent** (in a given ersatz) local language $L$ iff there are points of reference $(w,c)$ and $(w',c')$ such that $\mu(\phi)(c)(w) = 1$ and $\mu(\phi)(c')(w') = 0$, where $\mu_L(\phi)$ is the meaning of $\mu_L(\phi)$ according to $L$.

(4.4) $\phi$ of category $S$ is **independent of $\psi$ of category $S$** (in a given ersatz) local language) iff $\{(\mu_L(\phi)(c)(w), \mu_L(\psi)(c)(w)) \mid (w,c) \text{ is a point of reference of } L\}$ has 4 members.
**PART 3: Indirect Interpretation**

5. Translation

(5.1) English

<table>
<thead>
<tr>
<th>structure</th>
<th>category</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>N</td>
<td>B</td>
</tr>
<tr>
<td>cheap</td>
<td>Adj</td>
<td>C</td>
</tr>
</tbody>
</table>

\[
\text{book} \eta \text{cheap} = \text{cheap} \text{book} = \lambda x [\mathbf{C}(x) \land \mathbf{B}(x)]
\]

(5.2) \[ F_{\text{mod}}(\Delta, \Delta') = F(x, F_{\text{app}}(\Delta, x), F_{\text{app}}(\Delta', x)) ] \] where \( x \) is a fixed variable

(5.3) A syntactic polynomial (over a given syntax) is a term of the form \( F(X_1, \ldots, X_n) \), where \( F \) is the (unique) name of an \( n \)-place syntactic construction (of that syntax) and each \( X_i \) is either itself a syntactic polynomial (…), or a meta-variable (standing in for an arbitrary structure), or the (unique) name of a particular structure (…).

A derived construction (on a syntax) if is an operation on syntactic structures (…) that is denoted by some syntactic polynomial (…) – in the more or less obvious sense.

(5.4) A translation from a syntax \( \Xi = (\Sigma, (C_i)_{i \in I}, (L_k)_{k \in K}, R, S) \) to a syntax \( \Xi' = (\Sigma', (C'_i)_{i \in I'}, (L'_k)_{k \in K'}, R', S') \) is a triple \( (g, t, (T_i)_{i \in I}) \) such that:

- \( g : K \to K' \) is a function (category assignment) such that \( g(S) = S' \);
- \( t : \bigcup_{k \in K} L_k \to \Sigma' \) is a function (lexical translation) such that \( t(\delta) \) is a structure of category \( g(k) \) in \( \Xi' \) whenever \( \delta \in L_k \);
- \( (T_i)_{i \in I} \) is a family of derived constructions on \( \Xi' \) that is similar to \( (C_i)_{i \in I} \);
- if \( (C_i, k_1, \ldots, k_n, k^*) \in R \), and \( \Delta_1, \ldots, \Delta_n \) are structures of the respective categories \( k_1, \ldots, k_n \), then \( T_i(\Delta_1, \ldots, \Delta_n) \) is of category \( g(k^*) \).

Given any \( \Delta \in \Sigma \), then either \( \Delta \in L_{k^*} \) and its translation \( |\Delta| \) (according to \( (g, t, (T_i)_{i \in I}) \)) is \( t(\Delta) \); or else \( \Delta = C_i(\Delta_1, \ldots, \Delta_n) \) and its translation \( [\ldots] \) is \( |C_i(\Delta_1, \ldots, \Delta_n)| = T_i(|\Delta_1|, \ldots, |\Delta_n|) \), where \( |\Delta_1|, \ldots, |\Delta_n| \) are the respective translations \( [\ldots] \) of \( \Delta_1, \ldots, \Delta_n \).
6. Intensional Type Logic (ITL)

(6.1) The variables of ITL form a family \( \{ \text{Var}_a \}_{a \in \text{IT}} \) of pairwise disjoint, infinite sets; the constants of ITL form a family \( \{ \text{Con}_a \}_{a \in \text{IT}} \) of pairwise disjoint sets; the syncategorematic expressions form the set \( \{ \lambda, (=, \Delta, \&\&\&\&) \} \). No variable is a constant or a syncategorematic expression, etc.

The syntax of ITL is a quintuple \( (\Sigma, (C_i)_{i \in I}, (L_k)_{k \in K}, R, t) \), where:

- \( \Sigma \) consists of (finite) strings over \( \bigcup_{a \in \text{IT}} \text{Con}_a \cup \bigcup_{a \in \text{IT}} \text{Var}_a \cup \{ \lambda, (=, \Delta, \&\&\&\&) \} \);
- \( I = \{ \text{app}, \text{abs}, \text{id}, \text{cup}, \text{cap} \} \);
- \( C_{\text{app}}(\Delta, \Delta') = \Delta(\Delta') \); \( C_{\text{abs}}(\Delta, \Delta') = (\lambda \Delta \Delta') \); \( C_{\text{id}}(\Delta, \Delta') = (\Delta = \Delta') \); \( C_{\text{cup}}(\Delta) = \Delta; C_{\text{cap}}(\Delta) = \&\&\&\& \Delta \);
- \( K = \text{IT} \cup \{ (\text{VAR}, a) \mid a \in \text{IT} \} \);
- \( L_k = \text{Var}_k \cup \text{Con}_k \) if \( k \in \text{IT} \); \( L_k = \text{Var}_a \) if \( k = (\text{VAR}, a) \);
- \( R = \{(C_{\text{app}}(a, b), a, b) \mid a, b \in \text{IT} \}, \{(C_{\text{abs}}((\text{VAR}, a), b, a, b) \mid a, b \in \text{IT} \}, \{(C_{\text{id}}(a, a, t) \mid a \in \text{IT} \}, \{(C_{\text{cup}}(s, a), a \mid a \in \text{IT} \}, \{(C_{\text{cap}}(s, a) \mid a \in \text{IT} \}. \)

An ITL-ontology is a pair \( (E, C) \), where \( E = \{E_a\}_{a \in \text{IT}} \) satisfies the equations (3.14)(ii) and \( C \) is the set of variable assignments, i.e. the set of functions \( h: \bigcup_{a \in \text{IT}} \text{Var}_a \rightarrow \bigcup_{a \in \text{IT}} E_a \) such that \( h(x) \in E_a \) whenever \( x \in \text{Var}_a \).

A local \( (ersatz) \) language of ITL is a Fregean language \( (E, C) (\Xi, f, (M_i)_{i \in I}, \mu_0, \Delta) \) based on an ITL-ontology where

- \( \Xi \) is the syntax of ITL;
- \( f(a) = f((\text{VAR}, a)) = a \), for any \( a \in \text{IT} \);
- for any \( b, b' \in \bigcup_{a \in \text{IT}} E_a \), \( w, w' \in W, h \in C \), and \( u \in U \) the following hold:
  \[
  \begin{align*}
  M_{\text{app}}(b, b')(h)(w) &= b(h)(w) (b'(h)(w)) \text{ whenever } b \in M_{a,b} \text{ and } b' \in M_a; \\
  M_{\text{abs}}(\mu_0(x), b)(h)(w) &= b(h(x))(w) \text{ whenever } x \in \text{Var}_a \text{ and } b \in M_{a,b} \text{ and } h(x) \in \bigcup_{(x, h(x))} (C_{\text{cup}}(s, a)) \cup (C_{\text{cap}}(s, a)) ; \\
  M_{\text{id}}(b, b')(h)(w) &= \{ \varnothing \mid b(h)(w) = b'(h)(w) \} \text{ whenever } b, b' \in M_a; \\
  M_{\text{cup}}(b)(h)(w) &= b(h)(w) \text{ whenever } b \in M_{sa} ; \\
  M_{\text{cap}}(b)(h)(w)(w') &= b(h)(w') .
  \end{align*}
\]

- \( \mu_0(c)(h)(w) = \mu_0(c)(h')(w) \text{ whenever } w \in W, h, h' \in C \text{ and } c \in \bigcup_{a \in \text{IT}} \text{Con}_a; \\
  \mu_0(x)(h)(w) = h(x) \text{ whenever } w \in W, h \in C \text{ and } x \in \bigcup_{a \in \text{IT}} \text{Var}_a; \\
  \Delta = W \times C .
\]

If \( M \) is a local language of ITL, \( \alpha \) is an ITL formula (structure), \( \mu(\alpha) \) is \( \alpha \)’s meaning according to \( M, h \in C, w \in W, [\alpha]^{M,h,w} = \mu(\alpha)(h)(w). \)
Given a local ITL-language \( M \), there exists a function \( F : \bigcup_{a \in \IT} \Con_a \to \bigcup_{a \in \IT} \I_a \) such that
\( F(c) \in \I_a \) whenever \( c \in \Con_a \) and such that the following hold:

(i) \( \sem{\alpha}^{M,g,w} = F(\alpha)(w) \), if \( \alpha \in \Con_a \);
(ii) \( \sem{\alpha}^{M,g,w} = g(\alpha) \), if \( \alpha \in \Var_a \);
(iii) \( \sem{\alpha \beta}^{M,g,w} = \sem{\alpha}^{M,g,w} \sem{\beta}^{M,g,w} \), if \( \alpha = \alpha_1 (\alpha_2) \);
(iv) \( \sem{\lambda x \alpha}^{M,g,w} = \{ (u, \sem{\alpha}^{M,g,w}) \mid u \in D_b \} \), if \( \alpha = (\lambda x \alpha_1) \) and \( x \in \Var_b \);
(v) \( \sem{\alpha \lor \beta}^{M,g,w} = \{ u \mid [u = 0 \text{ and } \sem{\alpha}^{M,g,w} = \sem{\beta}^{M,g,w}] \} \), if \( \alpha = (\alpha_1 \lor \alpha_2) \);
(vi) \( \sem{\alpha}^{M,g,w} = \sem{\alpha}^{M,g,w} \), if \( \alpha = \forall \alpha_1 \);
(vii) \( \sem{\alpha}^{M,g,w} = \{ (w', \sem{\alpha}^{M,g,w}) \mid w' \in W \} \), if \( \alpha = \exists \alpha_1 \).

(6.2) If \( \alpha \) and \( \alpha' \) are ITL-formulae of the same category, then \( \alpha \) and \( \alpha' \) are logically equivalent if \( \sem{\alpha}^{M,g,w} = \sem{\alpha'}^{M,g,w} \) for any local ITL-languages \( M \), worlds \( w \) and assignments \( g \). Notation: \( \alpha \equiv \alpha' \).

(6.3) An ITL-formula \( \alpha \) is modally closed if (i-a) \( \alpha \in \bigcup_{a \in \IT} \Var_a \); or (i-b) \( \alpha = \exists \alpha \) (for some \( \beta \)), or (ii) there are modally closed \( \alpha_1 \) and \( \alpha_2 \) such that (ii-a) \( \alpha = \alpha_1 (\alpha_2) \), or (ii-b) \( \alpha = (\lambda \alpha_1 \alpha_2) \), or (ii-c) \( \alpha = (\alpha_1 = \alpha_2) \).

**Down-Up Cancellation**
\( ^\exists \alpha \equiv \alpha \), for all ITL-formulae \( \alpha \).

**Up-Down Cancellation**
\( ^\forall \alpha \equiv \alpha \), if \( \alpha \) is modally closed (and of a category \( (s,a) \)).

**Two-sorted Type Theory**
\( 2T \) contains \( \iota, \varepsilon, \) and \( \sigma \) and all pairs \( (a,b) \) such that \( a, b \in 2T \).
\( \Var_a, \forall a \in 2T \) and \( \Con_a, \forall a \in 2T \) are analogous to ITL, but the only syntactic constructions are \( C_{\text{app}}, C_{\text{abs}} \), and \( C_{\text{id}} \).

**Gallin’s translation** (\( i \) is a fixed variable in \( \Var_s \)).

(i) \( c^* = 'c (i)' \), if \( c \in \Con_a \);
(ii) \( x^* = x \), if \( x \in \Var_a \);
(iii) \( \alpha(\beta)^* = \alpha^*(\beta^*) \);
(iv) \( (\lambda x \alpha)^* = (\lambda x \alpha^*) \);
(v) \( (\alpha = \beta)^* = (\alpha^* = \beta^*) \);
(vi) \( \forall \alpha^* = \alpha^*(i) \);
(vii) \( \exists \alpha^* = (\lambda i \alpha^*) \).

**Restricted \( \beta \)-conversion (ITL)**
\((\lambda x \alpha) (\beta)) \equiv \alpha[\beta[x]] \), if (i) \( \beta \) does not contain a free variable that would get bound when \( x \) in \( \alpha \) is replaced by \( \beta \) and either (ii-a) no occurrence of \( x \) in \( \alpha \) lies within the scope of \( \exists \), or (ii-b) \( \beta \) is modally closed.
\(\beta\)-conversion (Ty2)

\((\lambda x \alpha)(\beta)) \equiv \alpha^{[\beta]} x \beta\), \(\beta\) does not contain a free variable that would get bound when \(x\) in \(\alpha\) is replaced by \(\beta\).

[Notation: \((\lambda x \alpha)(\beta)) \beta \equiv \alpha^{[\beta]} x \beta\); transitive closure: \(\beta\)].

NB1: \((\lambda x \alpha)(x)) \beta \equiv \alpha^{[\beta]} x \beta\);
NB2: \(\beta\)-contraction may increase length; e.g., if \(x \in \text{Var}\), \(R \in \text{Con} \), \(e \in \text{Var} e\), 
\[
(\lambda x R(x)(x)) (f(c)(c)(c)) \beta \equiv (f(c)(c)(c)) (f(c)(c)(c))
\]

\(\eta\)-conversion (Ty2 & ITL)

\((\lambda x \beta(x)) \equiv \beta\), if \(x \in \text{Fr} \).

\(\alpha\)-conversion (Ty2 & ITL)

\((\lambda x \alpha) \equiv (\lambda y \alpha(x))\) iff no occurrence of \(x\) in \(\alpha\) lies within the scope of (some) \(\lambda y\) and \(y \not\in \text{Fr} \).

**Definition**

(a) \(\alpha\) is immediately reducible to \(\beta\) iff \(\alpha \beta\) or \(\alpha \eta \beta\) or \(\alpha \alpha \beta\).
[Notation: \(\alpha \beta\); transitive closure: \(\alpha \beta\)]
(b) \(\alpha\) is normal iff \(\alpha \beta\) implies \(\alpha \alpha \beta\).
(c) \(\alpha\) is a normal form of \(\beta\) iff \(\alpha \beta\) and \(\beta\) is normal.

**Normal Form Theorem (Ty2)**

Every Ty2-formula has a normal form.

**Church-Rosser Theorem (Ty2)**

If \(\beta\) and \(\beta'\) are normal forms of \(\alpha\), then \(\beta \alpha \beta'\).

(6.4a) \((\lambda x P((\lambda y \gamma)(x)))\beta(c)\) (where \(x \in \text{Var} e\), \(y \in \text{Var} e\), \(c \in \text{Con} e\), \(P \in \text{Con} e\))

(b) \(P((\lambda y \gamma)(c))\)

(c) \((\lambda x P(\gamma(x)))\beta(c)\)

**Four observations on \(*\):**

- \((\wedge \alpha)^* \beta\)
- An ITL-formula \(\alpha\) is modally closed iff \(i \notin \text{Fr}(\alpha)\).
- If \(\alpha\) is modally closed, \((\wedge \alpha)^* \eta \alpha\).
- If all constants and free variables of a Ty2-formula \(\alpha\) of a type in \(2T \setminus \text{IT}\) are of types in \(2T \setminus \text{IT}\), then \(\alpha\) is logically equivalent to the \(\ast\)-image of some ITL-formula.

(6.5a) \((\lambda i (\lambda j (i = j)))\) (where \(i, j \in \text{Var} e\))

(b) \((\lambda i (\lambda F (\lambda i (F = (\lambda p p(i))))))) (\lambda p p(i))\) (where \(F \in \text{Var} \), \(p \in \text{Var} \))

(c) \((\lambda F \gamma (F = (\lambda p \gamma p))) (\lambda p \gamma p)\)
### Abbreviations in ITL and Ty2:

<table>
<thead>
<tr>
<th>Notation</th>
<th>where</th>
<th>is short for</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha(\beta,\gamma))</td>
<td>(\alpha: a(ab); \beta,\gamma: a)</td>
<td>(\alpha(\gamma)(\beta))</td>
</tr>
<tr>
<td>(T)</td>
<td>((\lambda x, x) = (\lambda x x))</td>
<td></td>
</tr>
<tr>
<td>(\perp)</td>
<td>((\lambda x T) = (\lambda x x))</td>
<td></td>
</tr>
<tr>
<td>(\neg \phi)</td>
<td>(\phi: t)</td>
<td>(\phi = \perp)</td>
</tr>
<tr>
<td>((\forall x) \phi)</td>
<td>(x \in \text{Var}; \phi: t)</td>
<td>((\lambda x \phi) = (\lambda x x))</td>
</tr>
<tr>
<td>((\exists x) \phi)</td>
<td>(x \in \text{Var}; \phi: t)</td>
<td>(\neg (\forall x) \neg \phi)</td>
</tr>
<tr>
<td>([\phi \leftrightarrow \psi])</td>
<td>(\phi,\psi: t)</td>
<td>(\phi = \psi)</td>
</tr>
<tr>
<td>([\phi \land \psi])</td>
<td>(\phi,\psi: t)</td>
<td>((\forall R_{(tt)}) [R(\phi,\psi) \leftrightarrow R(T,T)]) [alternatively: ((\forall f_{(t)}) [\phi \leftrightarrow [f(\phi) \leftrightarrow f(\psi)])]</td>
</tr>
<tr>
<td>([\phi \lor \psi])</td>
<td>(\phi,\psi: t)</td>
<td>(\neg \phi \land \neg \psi)</td>
</tr>
<tr>
<td>([\phi \rightarrow \psi])</td>
<td>(\phi,\psi: t)</td>
<td>(\neg \phi \lor \psi)</td>
</tr>
</tbody>
</table>

etc.

### Special ITL-conventions:

<table>
<thead>
<tr>
<th>Notation</th>
<th>where</th>
<th>is short for</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha{\beta})</td>
<td>(\alpha: s(at); \beta: a)</td>
<td>(\vee \alpha(\beta))</td>
</tr>
<tr>
<td>(\alpha{\beta,\gamma})</td>
<td>(\alpha: s(a(at)); \beta,\gamma: a)</td>
<td>(\vee \alpha(\gamma)(\beta))</td>
</tr>
<tr>
<td>(\square \phi)</td>
<td>(\phi: t)</td>
<td>(\neg \Diamond \neg \phi)</td>
</tr>
<tr>
<td>(\Diamond \phi)</td>
<td>(\phi: t)</td>
<td>((\neg \phi = \neg T))</td>
</tr>
</tbody>
</table>

\(\square\) is short for \(\top\) \(\Diamond\)
### PART 4: Descriptive Montague Grammar

#### 7. Extensional Constructions

##### (7.1) Simple constructions

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Rule(s)</th>
<th>Example</th>
</tr>
</thead>
</table>
| $F_{pred}(\Delta, \Delta') = \Delta, \Delta', \text{pred}$ | $(F_{pred}, \text{NP, VP, S})$ | $F_{pred}(\text{is happy}, \text{Mary})$
| | | Mary is happy, pred is happy Mary |
| $F_{obj}(\Delta, \Delta') = \Delta, \Delta', \text{obj}$ | $(F_{obj}, \text{TV, NP, VP})$ | $F_{obj}(\text{likes, the girl})$
| | | likes the girl, obj likes the girl... |
| $F_{cop}(\Delta) = \text{is, cop}$ | $(F_{cop}, \text{Adj, VP})$ | $F_{cop}(\text{happy}) = \text{is happy, cop happy}$ |
| $F_{def}(\Delta) = \text{the, def}$ | $(F_{def}, \text{N, NP})$ | $F_{def}(\text{girl}) = \text{the girl, def girl}$ |

##### (7.2) Naive type assignment

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>(Extension) Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Mary is happy; the boy likes the girl</td>
<td>$t$</td>
</tr>
<tr>
<td>NP</td>
<td>Mary; the boy; the girl</td>
<td>$e$</td>
</tr>
<tr>
<td>VP</td>
<td>is happy; likes the girl</td>
<td>$et$</td>
</tr>
<tr>
<td>TV</td>
<td>likes</td>
<td>$e(et)$</td>
</tr>
<tr>
<td>Adj</td>
<td>happy</td>
<td>$et$</td>
</tr>
</tbody>
</table>

##### (7.3) Naive lexical translation

<table>
<thead>
<tr>
<th>Item</th>
<th>ITL</th>
<th>Ty2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>m</td>
<td>$m_i$ [=$m(i)$]</td>
</tr>
<tr>
<td>boy</td>
<td>B</td>
<td>$B_i$</td>
</tr>
<tr>
<td>girl</td>
<td>G</td>
<td>$G_i$</td>
</tr>
<tr>
<td>likes</td>
<td>L</td>
<td>$L_i$</td>
</tr>
<tr>
<td>happy</td>
<td>H</td>
<td>$H_i$</td>
</tr>
</tbody>
</table>
(7.4) Naive meaning combinations

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{pred}$</td>
<td>$G_{pred}(\alpha,\beta) = C_{app}(\alpha,\beta)$ [(\alpha(\beta))]</td>
</tr>
<tr>
<td>$F_{obj}$</td>
<td>$G_{pred}(\alpha,\beta) = C_{app}(\beta,\alpha)$ [(\beta(\alpha))]</td>
</tr>
<tr>
<td>$F_{cop}$</td>
<td>$G_{cop}(\alpha) = \alpha$</td>
</tr>
<tr>
<td>$F_{def}$</td>
<td>$G_{def}(\alpha) = C_{app}(\iota\iota\iota,\alpha)$ (where $\iota\iota\iota \in Con_{(et)e}$) [(\iota\iota\iota(\alpha))]</td>
</tr>
</tbody>
</table>

(7.5a) Mary is happy, pred

<table>
<thead>
<tr>
<th>is happy, cop</th>
<th>Mary</th>
</tr>
</thead>
<tbody>
<tr>
<td>happy</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{aligned}
| F_{pred}(F_{cop}(\text{happy},\text{Mary})) | = \\
&= G_{pred}(\vert F_{cop}(\text{happy}) \vert, \vert \text{Mary} \vert) \\
&= G_{pred}(G_{cop}(\vert \text{happy} \vert), \vert \text{Mary} \vert) \\
&= G_{pred}(G_{cop}(\text{H}, \text{m})) \\
&= G_{pred}(\text{H}, \text{m}) \\
&= \text{H}(m)
\end{aligned}
\]

(b) the boy likes the girl, pred

<table>
<thead>
<tr>
<th>likes the girl, obj</th>
<th>the boy, def</th>
</tr>
</thead>
<tbody>
<tr>
<td>likes the girl, def</td>
<td>boy</td>
</tr>
</tbody>
</table>

\[
\begin{aligned}
| F_{pred}(F_{def}(\text{boy})), F_{obj}(\text{likes}, F_{def}(\text{girl})) | = \\
&= G_{pred}(\vert F_{def}(\text{boy}) \vert, F_{obj}(\vert \text{likes} \vert, \vert F_{def}(\vert \text{girl} \vert) \vert)) \\
&= G_{pred}(G_{def}(\vert \text{boy} \vert), G_{obj}(\vert \text{likes} \vert, G_{def}(\vert \text{girl} \vert))) \\
&= G_{pred}(G_{def}(\text{B}), G_{obj}(\text{L}, G_{def}(\text{G}))) \\
&= G_{pred}(\text{t}(\text{B}), G_{obj}(\text{L}, \text{t}(\text{G}))) \\
&= \text{L}(\text{t}(\text{G}))(\text{t}(\text{B})) \\
&= \text{L}(\text{t}(\text{B}), \text{t}(\text{G}))
\end{aligned}
\]

(7.6) Every boy likes Mary.

**Context Principle**

The reconstructed extension $\rho_\alpha$ of $\alpha$ (in $F(\alpha, \text{--})$) is a function $f$ that assigns to the extension of any (relevant) $\gamma$ the extension of $F(\alpha, \gamma)$:

- $\rho_\alpha(\mu(\beta)(c)(w)) = \mu(F(\alpha, \beta))(c)(w)$
- $\vert \alpha \vert (\vert \beta \vert) \equiv \vert F(\alpha, \beta) \vert$
- $\vert \alpha \vert = (\lambda x \vert F(\alpha, x) \vert)$

\[\text{direct version}\]
\[\text{indirect version}\]
\[\text{abstract version}\]
Classical Montague Grammar (Zimmermann), NASSLLI 04, UCLA

(7.7a) \( \text{every boy} \ (\text{likes Mary}) \equiv (\forall x) \ [B(x) \rightarrow \text{likes Mary} (x)] \)

\( \text{every boy} \ (\text{is happy}) \equiv (\forall x) \ [B(x) \rightarrow \text{is happy} (x)] \)

\( \ldots \)

\( \text{every boy} \ (\beta) \equiv (\forall x) \ [B(x) \rightarrow \beta (x)] \)

(7.7b) \( \text{every boy} \ | = (\lambda Q_{et} (\forall x) \ [B(x) \rightarrow P(x)]) \)

\( \text{type } (et) \equiv q \)

(7.8a) \( \text{every \ (boy)} \equiv (\lambda Q_{et} (\forall x) \ [\text{boy} (x) \rightarrow Q(x)]) \)

\( \text{every \ (girl)} \equiv (\lambda Q_{et} (\forall x) \ [\text{girl} (x) \rightarrow Q(x)]) \)

\( \ldots \)

\( \text{every \ (\beta)} \equiv (\lambda Q_{et} (\forall x) \ [\beta (x) \rightarrow Q(x)]) \)

(7.8b) \( \text{every} \ = (\lambda P_{et} \ (\lambda Q_{et} (\forall x) \ [P(x) \rightarrow Q(x)]) \)

\( \text{type } (et) \equiv q \)

(7.9a) \( \text{every boy or every girl} \ (\text{likes Mary}) \)

\( \equiv [(\forall x) \ [B(x) \rightarrow \text{likes Mary} (x)] \lor (\forall x) \ [G(x) \rightarrow \text{likes Mary} (x)] \)

\( \text{every boy or every girl} \ (\text{is happy}) \)

\( \equiv [(\forall x) \ [B(x) \rightarrow \text{is happy} (x)] \lor (\forall x) \ [G(x) \rightarrow \text{is happy} (x)] \)

\( \ldots \)

(7.9b) \( \text{every boy or every girl} \ |

\[ = (\lambda Q_{et}[(\forall x) \ [B(x) \rightarrow Q(x)] \lor (\forall x) \ [G(x) \rightarrow Q(x)]) \]

\( \text{type } q \)

(7.10a) \( \text{or} \ (\text{every boy}) \ (\text{every girl}) \)

\( \equiv (\lambda Q_{et}[(\forall x) \ [B(x) \rightarrow Q(x)] \lor (\forall x) \ [G(x) \rightarrow Q(x)]) \)

\( \equiv (\lambda Q_{et}[(\forall x) \ [B(x) \rightarrow Q(x)] \lor (\forall x) \ [G(x) \rightarrow Q(x)]) \)

\( \text{or} \ (\text{every boy}) \ (\text{some girl}) \)

\( \equiv (\lambda Q_{et}[(\forall x) \ [B(x) \rightarrow Q(x)] \lor (\exists x) \ [G(x) \land Q(x)]]) \)

\( \equiv (\lambda Q_{et}[(\forall x) \ [B(x) \rightarrow Q(x)] \lor \text{some girl} \ [Q(x)]) \)

\( \ldots \)

(7.10b) \( \text{or} \ | = (\lambda Q_{et} [(\lambda Q_{et} (\forall x) \ [Q(x)]) \lor (\forall x) \ [Q(x)]) \]

\( \text{type } q(qq) \)

(7.10) \( \text{Mary or every boy} \ |

\( \equiv (\lambda Q_{et} [Q(m)] \lor (\forall x) \ [B(x) \rightarrow Q(x)]) \)

\# (\lambda Q_{et} [\text{Mary} \ [Q(m)] \lor (\forall x) \ [\text{girl} \ [Q(m)]]) \)

\( \text{wrong type } (e \text{ vs. } q) \)

(7.11a) \( \text{Mary} \ (\text{likes Mary}) \equiv \text{likes Mary} \ (m) \)

\( \text{Mary} \ (\text{is happy}) \equiv \text{is happy} \ (m) \)

\( \ldots \)

\( \text{Mary} \ (\beta) = \beta \ (m) \)

(7.11b) \( \text{Mary} \ = (\lambda Q_{et} \ Q(m)) \)

\( \text{type } q \)
### Revised rules and constructions

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Rule(s)</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{\text{pred}}(\Delta,\Delta') = \Delta'_{\text{pred}}\Delta$</td>
<td>$(F_{\text{pred}},\text{NP,VP,S})$</td>
<td>$F_{\text{pred}}(\text{Mary, is happy}) = \text{Mary is happy, pred}$</td>
</tr>
<tr>
<td>$(F_{\text{pred}},\text{Det,N,NP})$</td>
<td></td>
<td>$\Rightarrow \text{Mary is happy}$</td>
</tr>
<tr>
<td>$F_{\text{coord}}(\Delta,\Delta',\Delta'') = \Delta'_{\text{coord}}\Delta\Delta''$</td>
<td>$(F_{\text{coord}},\text{NP,Conj,NP,NP})$</td>
<td>$F_{\text{coord}}(\text{every boy, or, Mary}) = \text{every boy or Mary, coord}$</td>
</tr>
<tr>
<td>$(F_{\text{coord}},\text{VP,Conj,VP,VP})$</td>
<td></td>
<td>$\Rightarrow \text{every boy \ or \ Mary}$</td>
</tr>
<tr>
<td>$F_{\text{cop}}(\Delta) = \text{is}<em>\Delta</em>{\text{cop}}$</td>
<td>$(F_{\text{cop}},\text{Adj,VP})$</td>
<td>$F_{\text{cop}}(\text{happy}) = \text{is happy, cop happy}$</td>
</tr>
<tr>
<td>$F_{\text{obj}}(\Delta,\Delta') = \Delta'_{\text{obj}}\Delta$</td>
<td>$(F_{\text{obj}},\text{TV,NP,VP})$</td>
<td>$F_{\text{obj}}(\text{likes, the girl}) = \text{likes the girl, obj}$</td>
</tr>
</tbody>
</table>

### Revised (and expanded) type assignment

<table>
<thead>
<tr>
<th>Category</th>
<th>(Extension) Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>t</td>
</tr>
<tr>
<td>NP</td>
<td>q</td>
</tr>
<tr>
<td>VP</td>
<td>et</td>
</tr>
<tr>
<td>TV</td>
<td>e(et)</td>
</tr>
<tr>
<td>Adj</td>
<td>et</td>
</tr>
<tr>
<td>Conj</td>
<td>q(qq)</td>
</tr>
</tbody>
</table>
(7.14) Lexical translation: revisions and additions

<table>
<thead>
<tr>
<th>Item</th>
<th>ITL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>((\lambda Q_{et} \ Q(m)))</td>
</tr>
<tr>
<td>or</td>
<td>((\lambda Q_{et}) (\lambda Q_{et}[Q] \lor Q()))</td>
</tr>
<tr>
<td>every</td>
<td>((\lambda P_{et} (\lambda Q_{et}) [P(x) \rightarrow Q(x)]))</td>
</tr>
<tr>
<td>some</td>
<td>((\lambda P_{et} (\lambda Q_{et}) [P(x) \land Q(x)]))</td>
</tr>
<tr>
<td>the</td>
<td>((\lambda P_{et} (\lambda Q_{et}) (\forall x) [(P(y) \leftrightarrow (x = y)) \land Q(x)]))</td>
</tr>
</tbody>
</table>

(7.15) New meaning combinations

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_{pred})</td>
<td>(G_{pred}(\alpha, \beta) = C_{app}(\alpha, \beta)) (= \alpha(\beta))</td>
</tr>
<tr>
<td>(F_{coord})</td>
<td>(G_{coord}(\alpha, \beta, \gamma) = C_{app}(C_{app}(\beta, \gamma, \alpha)) (= \beta(\gamma(\alpha)))</td>
</tr>
<tr>
<td>(F_{cop})</td>
<td>(G_{cop}(\alpha) = \alpha)</td>
</tr>
<tr>
<td>(F_{obj})</td>
<td>(C_{abs}(x, \ C_{app}(\beta, C_{abs}(y, C_{app}(C_{app}(\alpha, y), x)))) (= (\lambda x, \beta(\lambda y \alpha(x, y))))</td>
</tr>
</tbody>
</table>

8. Intensional Constructions

Attitude verbs

(8.1a) John think\(s\) that Mary is happy.
(8.1b) Mary is happy.
(8.1c) Every boy likes Mary.
(8.1d) John think\(s\) that every boy likes Mary.

Opaque verbs

(8.2a) John is looking for a book on Clinton.
(8.2b) Every book on Clinton is a book by Clinton.
(8.2c) John is looking for a book by Clinton.

Core-intensional verbs

(8.3a) The temperature is rising.
(8.3b) The temperature is ninety.
(8.3c) Ninety is rising.

(8.4) Attitude reports: type assignment

<table>
<thead>
<tr>
<th>Category</th>
<th>(Extension) Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttV</td>
<td>((st)(et))</td>
</tr>
<tr>
<td>Prop</td>
<td>(st)</td>
</tr>
</tbody>
</table>
(8.5) Attitude reports: additional rules and constructions

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Rule(s)</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{pred}(\Delta, \Delta') = \Delta', \Delta$</td>
<td>$(F_{pred}, AttV, Prop, VP)$</td>
<td>$F_{pred}(\text{thinks, that Mary is happy}) = \text{thinks that Mary is happy, att}$</td>
</tr>
<tr>
<td>$F_{that}(\Delta) = \text{that } \Delta$</td>
<td>$(F_{that}, S, Prop)$</td>
<td>$F_{that}(\text{Mary is happy}) = \text{that } \Delta$</td>
</tr>
</tbody>
</table>

(8.6) Attitude verbs: lexical translation

<table>
<thead>
<tr>
<th>Item</th>
<th>ITL</th>
<th>Ty2</th>
</tr>
</thead>
<tbody>
<tr>
<td>thinks</td>
<td>$T$</td>
<td>$T_i$</td>
</tr>
<tr>
<td>believes</td>
<td>$(\lambda p \ (\lambda x \ (\lambda q \ [\forall q \rightarrow \forall p])(B(x)))$</td>
<td>$(\lambda p \ (\lambda x \ (\forall j) <a href="j">B_{ij}(x)</a> \rightarrow p_j]$</td>
</tr>
</tbody>
</table>

(8.7) Attitude verbs: additional meaning combination

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding ITL-Polynomial</th>
<th>Corresponding Ty2-Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{that}$</td>
<td>$G_{pred}(\alpha) = C_{cap}(\alpha) = \land \alpha$</td>
<td>$G_{pred}(\alpha) = C_{abs}(i, \alpha) = (\lambda i \ alpha)$</td>
</tr>
</tbody>
</table>

(8.8) John is-trying-for-it-to-be-the-case that John finds a book on Clinton.

(8.9a) | seeks | (a book) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\equiv$</td>
<td>$(\lambda x \ [tries] (x, \land y [B(y) \land [finds] (x, y)])$</td>
<td>$\equiv (\lambda x \ [tries] (x, \land [a book] [\lambda y \ [finds] (x, y)])$</td>
</tr>
</tbody>
</table>

... | seeks | (beta) $\equiv (\lambda x \ [tries] (x, \land \beta [\lambda y \ [finds] (x, y)])$ |

(8.9b) | seeks | $\equiv (\lambda \gamma [s(et)] (\lambda x \ [tries] (x, \land \gamma [\lambda y \ [finds] (x, y)])$ |

(8.10) Opaque verbs: revised type assignment

<table>
<thead>
<tr>
<th>Category</th>
<th>(Extension) Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>(sq)(et)</td>
</tr>
</tbody>
</table>
(8.11) Opaque verbs: revised meaning combination

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Polynomial (ITL)</th>
<th>Corresponding Polynomial (Ty2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{obj}$</td>
<td>$G_{obj}(\alpha, \beta) = C_{app}(\alpha, C_{cap}(\beta))$ $[= \alpha(\lambda \beta)]$</td>
<td>$G_{obj}(\alpha, \beta) = C_{app}(\alpha, C_{abs}(\lambda \beta))$ $[= \alpha(\lambda \beta)]$</td>
</tr>
</tbody>
</table>

(8.11) Transitive verbs: revised lexical translation

<table>
<thead>
<tr>
<th>Item</th>
<th>ITL</th>
<th>Ty2</th>
</tr>
</thead>
</table>
| seeks | $\lambda x. (\lambda F(x,y))\left(\left(\begin{array}{c} x \\
|       | \mbox{tries} \end{array}\right)\right)$                     | $\lambda x. (\lambda F(x,y))\left(\left(\begin{array}{c} x \\
|       | \mbox{tries} \end{array}\right)\right)$                     |
| finds | $\lambda x. (\lambda y F(x,y))$                            | $\lambda x. (\lambda y F(x,y))$                            |
| likes | $\lambda x. (\lambda y L(x,y))$                            | $\lambda x. (\lambda y L(x,y))$                            |
| is    | $\lambda x. (\lambda y (x = y))$                           | $\lambda x. (\lambda y (x = y))$                           |

(8.9a) Mary is looking for a [certain] book on Clinton.

(b) $\lambda x. (\lambda y (F(x,y)) \otimes (x \wedge \mbox{tries} \otimes (m,^\wedge F(m,x))))$

(8.10) Scope construction

<table>
<thead>
<tr>
<th>Construction</th>
<th>Rule(s)</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{scope,x}$</td>
<td>$(\Delta, \Delta') = \Delta[\lambda x, \Delta, \lambda x] \subseteq \Delta'$</td>
<td>$F_{scope,x}(\text{a book, Mary seeks } x)$ Mary seeks a book, scope $x$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>ITL</th>
<th>Ty2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$(\lambda P P(x))$</td>
<td>$(\lambda P P(x))$</td>
</tr>
</tbody>
</table>

(8.11) Scope: meaning combination

<table>
<thead>
<tr>
<th>Construction</th>
<th>Corresponding Polynomial (ITL and Ty2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{scope,x}$</td>
<td>$G_{scope,x}(\alpha, \beta) = C_{app}(\alpha, C_{abs}(\lambda x, \beta))$ $[= \alpha(\lambda x \beta)]$</td>
</tr>
</tbody>
</table>

(8.10) $\lambda x. [\exists j \left( j \triangleleft \wedge \forall k \left( \forall k' \left[ j \lt k \leq k' \lt i \rightarrow \forall k (\lambda x. \forall k (\lambda y (x \leq y))) \right] \right) \right]$

(8.11) The temperature is ninety and it is rising.