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 $(n\text{-place})$ 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
  \n
old

N

N

men

Conj

and

N

women
```

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

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

(1.5) $\not\equiv$ Jones ( seeks a $\not\equiv$ horse such $\not\equiv$ it $\not\equiv$ speaks $\not\equiv\not$ ) $\not\equiv$

(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 ‘($S_1$ 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 ($n$-place) syntactic construction $C$ there is a corresponding ($n$-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) \begin{align*}
&\text{Jones knows that } S, C_0 \\
&\text{Jones knows that } S, C_1 \\
&\text{knows that } S
\end{align*}

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

(2.9) \begin{align*}
&\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)
\end{align*}

(2.10) \begin{align*}
&\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')
\end{align*}

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

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

(2.14) \begin{align*}
&\mu(C_2(7, C_1(1,2))) \\
&= M_2(\mu(7), \mu(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
\end{align*}
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 \(\Delta_1, ..., \Delta_n\).

\(2.15\) \((\exists x)\) \(P(x)\)

\(2.16\) \(P(a)\)

\(2.17\) The meaning of a complex expression is determined by the meanings of (certain) less complex expressions.

\(2.18\) \((\forall x)\) \(P(x)\)

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

PART 2: Meaning and Reference
3. Local Perspective

<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 ((\emptyset \text{ or } {\emptyset}))</td>
<td>snow is white</td>
<td>1</td>
</tr>
</tbody>
</table>
Classical Montague Grammar (Zimmermann), UCI, February 2018

(3.2) Jones knows that snow is white.
(3.3) Jones knows that grass is green.

(3.13) 

sentence

that-clause

extension

intension

extension

truth-value

function from \(k'\) to \(\{0,1\}\)

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

(ii) \(E_t = \{0,1\}, E_e = \cup_i E_{(a,b)} = E_b^{E_a}, E_{(s,b)} = E_b^w\).

(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 \(I_{ct}\).

(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

- \(\Sigma = (\Sigma_i)_{i \in I}, (L_k)_{k \in K}, R, S\) is a syntax;
- \(f: K \to \text{IT}\) is a function (type assignment) such that \(f(S) = t\);
- \(\mu_0: \bigcup_{k \in K} L_k \to \bigcup_{a \in \text{IT}} M_a\) 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 \(M_i(b_1,...,b_n) \in M_{f(k_i)}\) whenever \(b_i \in M_{f(k_i)}\), \(b_n \in M_{f(k_n)}\) and \((M_i,k_1,...,k_n,k_0)\in R\).
- \(\Delta\) 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 \(M_{f(k)}\).

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

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

(3.17) \(\delta\) is deictic iff \(\mu(\phi)(c)(w) = \mu(\phi)(c)(w')\), 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 roles 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 \(\text{deictic a hyponym of } \delta' \) in a global language iff $\delta$ is \(\text{deictic an a priori truth} \) in 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))| (w,c) \text{ is a point of reference of } L) \text{ has 4 members.}$$
### PART 3: Indirect Interpretation

5. Translation

(5.1) | English | Logic |
--- | --- | --- |
structure | category | | type |
book | N | B | et |
cheap | Adj | C | et |

\[(N \text{ (Adj } \text{ cheap}), (N \text{ book})) \]
\[= F_{\text{mod}}(\text{cheap,book}) \]

\[= F_{\lambda x}([C(x) \land B(x)]) \]
\[= F_{\lambda x}(F_{\text{app}}(C,x),F_{\text{app}}(B,x))) \]

(5.2) | \( F_{\text{mod}}(\Delta,\Delta') = F_{\lambda x}(F_{\text{app}}(\Delta,x),F_{\text{app}}(\Delta',x)) \) 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) 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:

- \( g: K \rightarrow K' \) is a function (category assignment) such that \( g(S) = S' \);
- \( t: \bigcup_{k \in K} L_k \rightarrow \Sigma' \) 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 \( […] \) 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 \( […] \) 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 \(\{\lambda, (,),=,\land,\lor,\}\). 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 \(\bigcup_{a \in \text{IT}} \text{Con}_a \cup \bigcup_{a \in \text{IT}} \text{Var}_a \cup \{\lambda, (,),=,\land,\lor,\}\);
- \(I = \{\text{app,abs,id,cup,cap}\}\);
- \(C_{\text{app}}(\Delta, \Delta') = \Delta'(\Delta)\); \(C_{\text{abs}}(\Delta, \Delta') = (\lambda \Delta \Delta')\); \(C_{\text{id}}(\Delta, \Delta') = (\Delta = \Delta')\); \(C_{\text{cup}}(\Delta) = \lor \Delta\); \(C_{\text{cap}}(\Delta) = \land \Delta\);
- \(K = \text{IT} \cup \{(\text{VAR}, a) \mid a \in \text{IT}\}\);
- \(L_k = \text{Var}_k \cup \text{Cond}_k\) 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)) \mid a \in \text{IT}\}, \{(C_{\text{cap}}, (t, a), (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 \(h(\textbf{x}) \in E_a\) whenever \(\textbf{x} \in \text{Var}_a\).

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:
  - \(M_{\text{app}}(b, b') (h)(w) = h(b)(w) (b'(h)(w))\) whenever \(b \in M_{ab}\) and \(b' \in M_b\);
  - \(M_{\text{abs}}(\mu_0(x), b)(h)(u)(w) = b(h(x/u))(w)\) whenever \(x \in \text{Var}_a\) and \(b \in M_{ab}\) and \(h[x/u] = (h \setminus \{(x, h(x))\}) \cup \{(x, u)\}\);
  - \(M_{\text{id}}(b, b')(h)(w) = (\emptyset \setminus (b(h)(w)) = b'(h)(w))\) whenever \(b, b' \in M_a\);
  - \(M_{\text{cup}}(b) (h)(w)' = b(h)(w)\) whenever \(b \in M_{sa}\);
  - \(M_{\text{cap}}(b) (h)(w)' = b(h)(w)\) whenever \(b \in M_{sa}\).
- \(\mu_0(c)(h)(w) = \mu_0(c)(h')(w)\) whenever \(w \in W\), \(h, h' \in C\) and \(c \in \bigcup_{a \in \text{IT}} \text{Con}_a\);
- \(\mu_0(x)(h)(w) = h(x)\) whenever \(w \in W\), \(h \in C\) 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 \(M, h \in C, w \in W\), \(\llbracket \alpha \rrbracket^M_{h, w} = \mu(\alpha)(h)(w)\).
Given a local ITL-language $M$, there exists a function $F: \bigcup_{\alpha \in IT} Con_a \to \bigcup_{\alpha \in IT} I_a$ such that $F(c) \in I_a$ whenever $c \in Con_a$ and such that the following hold:

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

(6.2) If $\alpha$ and $\alpha'$ are ITL-formulae of the same category, then $\alpha$ and $\alpha'$ are logically equivalent if $\llbracket \alpha \rrbracket_{M, g, w}^c = \llbracket \alpha' \rrbracket_{M, g, w}^c$ 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) $\alpha \in \bigcup_{\alpha \in IT} Var_a$; or (i-b) $\alpha = \wedge \alpha$ (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**

$\wedge \alpha \equiv \alpha$, for all ITL-formulae $\alpha$.

**Up-Down Cancellation**

$\vee \alpha \equiv \alpha$, if $\alpha$ is modally closed (and of a category $(s, a)$).

**Two-sorted Type Theory**

$2T$ contains $t$, $e$, and $s$ and all pairs $(a, b)$ such that $a, b \in 2T$.

$(Var_a)_{a \in 2T}$ and $(Con_a)_{a \in 2T}$ are analogous to ITL, but the only syntactic constructions are $C_{app}$, $C_{abs}$, and $C_{id}$.

**Gallin's translation** (i is a fixed variable in $Var_s$).

(i) \[ c^* = \left< c \right>(i), \text{ if } c \in Con_a; \]
(ii) \[ x^* = x, \text{ if } x \in Var_a; \]
(iii) \[ \alpha(\beta)^* = \alpha^*(\beta^*); \]
(iv) \[ (\lambda x \alpha)^* = (\lambda x \alpha^*); \]
(v) \[ (\alpha = \beta)^* = (\alpha^* = \beta^*); \]
(vi) \[ \vee \alpha^* = \alpha^*(i); \]
(vii) \[ \wedge \alpha^* = (\lambda i \alpha^*). \]

**Restricted $\beta$-conversion (ITL)**

$((\lambda x \alpha)(\beta)) \equiv \alpha^*\beta[i^*]$, 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 $\wedge$, or (ii-b) $\beta$ is modally closed.
\(\beta\)-conversion (Ty2)

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

[Notation: \(((\lambda x \alpha) (\beta)) \triangleright_\beta \alpha[[\beta]]\)]

NB1: \(((\lambda x \alpha) (x)) \triangleright_\beta \alpha;\)

NB2: \(\beta\)-contraction may increase length; e.g., if \(x \in \text{Var}_e, R \in \text{Con}_{\text{ele}(e)}, f \in \text{Con}_{\text{ele}(e)}, c \in \text{Con}_e:\)

\((\lambda x \ R(x)(x)(x)) \triangleright_\beta \ R(f(c)(c)(c)) \ (f(c)(c)(c)) \ (f(c)(c)(c)))\)

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

\(((\lambda x \beta(x)) \equiv \beta;\)

if \(x \notin \text{Fr}(\beta)\)

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

\(((\lambda x \alpha) \equiv (\lambda y \alpha[y])\) iff no occurrence of \(x\) in \(\alpha\) lies within the scope of (some) \(\lambda y\) and \(y \notin \text{Fr}(\lambda\alpha)\).

Definition

\((a)\) \(\alpha\) is immediately reducible to \(\beta\) iff \(\alpha = \gamma[\delta], \beta = \gamma[\delta],\) and: \([\delta_1 \triangleright_\gamma \delta_2 \ or \ \delta_1 \triangleright_\beta \delta_2 \ or \ \delta_1 \triangleright_\eta \delta_2]\)

[Notation: \(\alpha \triangleright_\beta\); transitive closure: \(\alpha \triangleright_\beta\) \(\ldots\) for some \(\gamma, \delta, \delta_1\) and variable \(x\) (of the appropriate types),

\((b)\) \(\alpha\) is normal iff \(\alpha \triangleright_\beta\) implies \(\alpha \triangleright_\alpha \beta.\)

\((c)\) \(\beta\) is a normal form of \(\alpha\) iff \(\alpha \triangleright_\beta\) and \(\beta\) is normal.

Normal Form Theorem (Ty 2)

Every Ty2-formula has a normal form.

Church-Rosser Theorem (Ty2)

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

\(4.6a\) \((\lambda x \ P((\lambda y \ \wedge y)(x))) (c)\)

(where \(x \in \text{Var}_e, y \in \text{Var}_e, c \in \text{Con}_e, P \in \text{Con}_{(e)(e)})\)

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

\((c)\) \((\lambda x \ P(\wedge x))(c)\)

Friedman & Wamen (1980)

Gallin 1975; Zimmermann (1989)

Four observations on *:

\(\bullet\) \((\wedge^\alpha)^* \triangleright_\beta \alpha^*\).

\(\bullet\) An ITL-formula \(\alpha\) is modally closed iff \(i \notin \text{Fr}(\alpha^*).\)

If \(\alpha\) is modally closed, \((\wedge^\alpha)^* \triangleright_\eta \alpha^*\).

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 *-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 \(F \in \text{Var}_{(e)(e)}, p \in \text{Var}_{(e)(e)}\))

\((c)\) \((\lambda F \ (\lambda p \ (\wedge p))) \ ((\lambda p \ \wedge p))\)
(6.6) Abbreviations in ITL and Ty2:

<table>
<thead>
<tr>
<th>Notation</th>
<th>where</th>
<th>is short for</th>
</tr>
</thead>
<tbody>
<tr>
<td>α(β,γ)</td>
<td>α: a(ab); β,γ: a</td>
<td>α(γ)(β)</td>
</tr>
<tr>
<td>T</td>
<td>(λx T) = (λx x)</td>
<td></td>
</tr>
<tr>
<td>⊥ φ</td>
<td>φ: t</td>
<td>(φ = ⊥)</td>
</tr>
<tr>
<td>(∀x) φ</td>
<td>x ∈ Var; φ: t</td>
<td>(λx φ) = (λx x)</td>
</tr>
<tr>
<td>(∃x) φ</td>
<td>x ∈ Var; φ: t</td>
<td>¬ (∀x) ¬ φ</td>
</tr>
<tr>
<td>[φ ↔ ψ]</td>
<td>φ,ψ: t</td>
<td>(φ = ψ)</td>
</tr>
<tr>
<td>[φ ∧ ψ]</td>
<td>φ,ψ: t</td>
<td>(∀R_{i(tu)}) [R(φ,ψ) ↔ R(T,T)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[alternatively: (∀f_{i,t}) [φ ↔ f(φ) ↔ f(ψ)]]</td>
</tr>
<tr>
<td>[φ ∨ ψ]</td>
<td>φ,ψ: t</td>
<td>[ ¬φ ∧ ¬ψ]</td>
</tr>
<tr>
<td>[φ → ψ]</td>
<td>φ,ψ: t</td>
<td>[ ¬φ ∨ ψ]</td>
</tr>
<tr>
<td>etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(6.7) Special ITL-conventions:

<table>
<thead>
<tr>
<th>Notation</th>
<th>where</th>
<th>is short for</th>
</tr>
</thead>
<tbody>
<tr>
<td>α{β}</td>
<td>α: s(at); β: a</td>
<td>∀α(β)</td>
</tr>
<tr>
<td>α{β,γ}</td>
<td>α: s(a(at)); β,γ: a</td>
<td>∀α(γ)(β)</td>
</tr>
<tr>
<td>□ φ</td>
<td>φ: t</td>
<td>(⊥φ = ⊥T)</td>
</tr>
<tr>
<td>◇ φ</td>
<td>φ: t</td>
<td>¬ □ ¬ φ</td>
</tr>
</tbody>
</table>
**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>
<tbody>
<tr>
<td>$F_{pred}(\Delta,\Delta')$</td>
<td>$(F_{pred},NP,VP,S)$</td>
<td>$F_{pred}(\text{is happy},\text{Mary})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{Mary is happy}, \text{pred}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{is happy}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{Mary}$</td>
</tr>
<tr>
<td>$F_{obj}(\Delta,\Delta')$</td>
<td>$(F_{obj},TV,NP,VP)$</td>
<td>$F_{obj}(\text{likes}, \text{the girl})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{likes the girl}, \text{obj}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{likes}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{the girl}$</td>
</tr>
<tr>
<td>$F_{cop}(\Delta)$</td>
<td>$(F_{cop},Adj,VP)$</td>
<td>$F_{cop}(\text{happy})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{is happy}, \text{cop happy}$</td>
</tr>
<tr>
<td>$F_{def}(\Delta)$</td>
<td>$(F_{def},N,NP)$</td>
<td>$F_{def}(\text{girl})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{the girl}, \text{def girl}$</td>
</tr>
</tbody>
</table>

(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>$m_i$</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>$F_{\text{pred}}$</td>
<td>$G_{\text{pred}}(\alpha,\beta) = C_{\text{app}}(\alpha,\beta)$</td>
</tr>
<tr>
<td>$F_{\text{obj}}$</td>
<td>$G_{\text{pred}}(\alpha,\beta) = C_{\text{app}}(\beta,\alpha)$</td>
</tr>
<tr>
<td>$F_{\text{cop}}$</td>
<td>$G_{\text{cop}}(\alpha) = \alpha$</td>
</tr>
<tr>
<td>$F_{\text{def}}$</td>
<td>$G_{\text{def}}(\alpha) = C_{\text{app}}(\iota,\alpha)$ (where $\iota \in \text{Con}(\text{et})$)</td>
</tr>
</tbody>
</table>

(7.5a) Mary is happy, pred

is happy, cop

happy

$\alpha$

$\beta$

$\alpha(\beta)$

(7.5b) the boy likes the girl, pred

likes the girl, obj

likes

the girl, def

boy

$\beta$

$\gamma$

$\alpha$}

(7.6) Every boy likes Mary.

Context Principle

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

- $\rho_\alpha(\mu(\beta) (c) (w)) = \mu(F(\alpha,\beta)) (c) (w)$ (direct version)
- $|\alpha| (|\beta|) \equiv |F(\alpha,\beta)|$ (indirect version)
- $|\alpha| = (\lambda x | F(\alpha, x)|)$ (abstract version)
(7.7a) \(\text{every boy} \equiv (\forall x) [B(x) \rightarrow \text{likes Mary} (x)]\)
\(\text{every boy} \equiv (\forall x) [B(x) \rightarrow \text{is happy} (x)]\)
\(\ldots\)
\(\text{every boy} \equiv (\forall x) [B(x) \rightarrow \beta(x)]\)

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

(7.8a) \(\text{every boy} \equiv (\lambda q_{et} (\forall x) [\text{boy}(x) \rightarrow Q(x)])\)
\(\text{every girl} \equiv (\lambda q_{et} (\forall x) [\text{girl}(x) \rightarrow Q(x)])\)
\(\ldots\)
\(\text{every boy} \equiv (\lambda q_{et} (\forall x) [\beta(x) \rightarrow Q(x)])\)

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

(7.9a) \(\text{every boy or every girl} \equiv \text{likes Mary}\)
\(\equiv [(\forall x) [B(x) \rightarrow \text{likes Mary} (x)] \lor (\forall x) [G(x) \rightarrow \text{likes Mary} (x)]]\)
\(\text{every boy or every girl} \equiv (\forall x) [\text{is happy} (x)]\)
\(\equiv [(\forall x) [B(x) \rightarrow \text{is happy} (x)] \lor (\forall x) [G(x) \rightarrow \text{is happy} (x)]]\)
\(\ldots\)
\(\text{every boy} \equiv (\lambda q_{et} (\forall x) [B(x) \rightarrow Q(x)] \lor (\forall x) [G(x) \rightarrow \beta(x)])\)

(7.9b) \(\text{every boy or every girl} =\)
\(= (\lambda q_{et} [(\forall x) [B(x) \rightarrow Q(x)] \lor (\forall x) [G(x) \rightarrow Q(x)])\)

(7.10a) \(\text{or} \equiv (\lambda q_{et} (\forall x) [B(x) \rightarrow Q(x)] \lor (\forall x) [G(x) \rightarrow Q(x)])\)
\(\equiv (\lambda q_{et} (\forall x) [\text{boy}(x) \lor \text{girl}(x)])\)
\(\equiv \text{or} (\forall x) [\text{some girl}]\)
\(\equiv (\lambda q_{et} [(\forall x) [B(x) \rightarrow Q(x)] \lor (\exists x) [G(x) \land Q(x)]]\)
\(\equiv (\lambda q_{et} [(\forall x) [\text{boy}(x) \lor \text{girl}(x)])\)
\(\ldots\)
\(\text{or} (\beta(x)) \equiv (\lambda q_{et} [(\beta(x)] \lor \gamma(x)]\)

(7.10b) \(\text{or} = (\lambda q_{et} (\lambda q_{et} [\beta(x)] \lor \gamma(x)])\)

(7.10) \(\text{Mary or every boy} \equiv\)
\(\equiv (\lambda q_{et} [Q(m)] \lor (\forall x) [B(x) \rightarrow Q(x)])\)

(7.11a) \(\text{Mary} \equiv \text{likes Mary} \equiv \text{Mary} \equiv \text{is happy} \equiv \text{Mary} \equiv \text{is happy}\)
\(\ldots\)
\(\text{Mary} \equiv 13\)

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

Classical Montague Grammar (Zimmermann),
### Revised 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', \text{pred} ) ( \Delta' )</td>
<td>( (F_{pred}, NP, VP, S) )</td>
<td>( F_{pred}(\text{Mary}, \text{is happy}) )</td>
</tr>
<tr>
<td>( F_{coord}(\Delta, \Delta', \Delta'') = \Delta \Delta', \text{coord} \Delta'' )</td>
<td>( (F_{coord}, NP, Conj, NP, NP) )</td>
<td>( F_{pred}(\text{every boy, or, Mary}) )</td>
</tr>
<tr>
<td>( F_{obj}(\Delta, \Delta') = \Delta \Delta', \text{obj} )</td>
<td>( (F_{obj}, TV, NP, VP) )</td>
<td>( F_{obj}(\text{likes Mary, or, is happy}) )</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 V_{(et/it)}(\lambda V_{(et/it)}(\lambda Q_{et}(Q) \lor V((Q)))))))</td>
</tr>
<tr>
<td>every</td>
<td>((\lambda P_{et}(\lambda Q_{et}(\forall x)[P(x) \rightarrow Q(x)])))</td>
</tr>
<tr>
<td>some</td>
<td>((\lambda P_{et}(\lambda Q_{et}(\exists x)[P(x) \land Q(x)])))</td>
</tr>
<tr>
<td>the</td>
<td>((\lambda P_{et}(\lambda Q_{et}(\exists x)(\forall y)[[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 thinks that Mary is happy.*
(8.1b) *Mary is happy.*
(8.1c) *Every boy likes Mary.*
(8.1d) *John thinks 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.*

Montague (1973)

(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>
</table>
| $F_{pred}(\Delta,\Delta') = \Delta'_{\Delta',pred}^{\Delta}$ | $(F_{pred},\text{AttV},\text{Prop},\text{VP})$ | $F_{pred}(\text{thinks,that Mary is happy})$
| | | $= \text{thinks that Mary is happy, att}$
| | | $\text{thinks that Mary is happy}$
| $F_{that}(\Delta) = \text{that } \Delta_{\Delta}$ | $(F_{that},\text{S,Prop})$ | $F_{that}(\text{Mary is happy})$
| | | $= \text{that } \Delta_{\Delta}$

(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>$[\in \text{Con}_{\text{st}(et)}]$</td>
</tr>
<tr>
<td>believes</td>
<td>$(\lambda p (\lambda x (\lambda q \Box [\forall q \rightarrow \forall p])(B(x))))$</td>
<td>$(\lambda p (\lambda x (\forall j)(B_i(x)(j) \rightarrow p_j)]$</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>$G_{pred}(\alpha) = C_{\text{cap}}(\alpha)$</td>
<td>$[= \wedge \alpha]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$G_{pred}(\alpha) = C_{\text{abs}}(i,\alpha)$</td>
</tr>
</tbody>
</table>

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

(8.9a) $\text{seeks} | (| \text{a book} |)$

$\equiv (\lambda x | \text{tries} (x, \wedge (\exists y) \{B(y) \wedge | \text{finds} (x,y)\}))$

$\equiv (\lambda x | \text{tries} (x, \wedge | \text{a book} | (\lambda y | \text{finds} (x,y)))$)

... $\text{seeks} | (| \beta |)$

$\equiv (\lambda x | \text{tries} (x, \wedge | \beta | (\lambda y | \text{finds} (x,y))))$

(8.9b) $\text{seeks} | = (\lambda | (\text{st}(et)) (\lambda x | \text{tries} (x, \wedge | (\lambda y | \text{finds} (x,y))))$

(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>$G_{obj}(\alpha, \beta) = C_{app}(\alpha, C_{cap}(\beta))$</td>
<td>$G_{obj}(\alpha, \beta) = C_{app}(\alpha, C_{abs}(i, \beta))$</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>
<tbody>
<tr>
<td>seeks</td>
<td>$(\lambda \varphi (\lambda x \uparrow \text{tries} \uparrow (x, \lambda y \lambda y F(x, y))))$</td>
<td>$(\lambda \varphi (\lambda x \lambda y F(x, y))))$</td>
</tr>
<tr>
<td>finds</td>
<td>$(\lambda \varphi (\lambda x \lambda y L(x, y)))$</td>
<td>$(\lambda \varphi (\lambda x \lambda y L(x, y)))$</td>
</tr>
<tr>
<td>likes</td>
<td>$(\lambda \varphi (\lambda x \lambda y (x = y)))$</td>
<td>$(\lambda \varphi (\lambda x \lambda y (x = y)))$</td>
</tr>
</tbody>
</table>

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

(b) $(\exists x) ([| \text{book on Clinton} | (x) \land \text{tries} | (m, | F(m, x))])$

(8.10) Scope construction

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

(8.10) Variables in the lexcion

<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>$G_{scope, x}(\alpha, \beta) = C_{app}(\alpha, C_{abs}(x, \beta))$ [$= \alpha(\lambda x \beta)$]</td>
</tr>
</tbody>
</table>

(8.10) $(\lambda \varphi [ (\exists j) [j < i \land (\forall k) (\forall k') [j > k \leq k' \leq i \rightarrow \varphi_k(\lambda x \varphi_k(\lambda y x = y))]] \land (\exists j) [j > i \land (\forall k) (\forall k') [i \leq k \leq k' \leq j \rightarrow \varphi_k(\lambda x \varphi_k(\lambda y x = y))]])$

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