1 Generalized coordination and type-shifting

Coordination combines like categories (N.B.: this is a good first pass but not unproblematic).

- Lexicon: \([\text{[and]}, (X/LX)/R-X, \lambda f_{(a,t)} \lambda g_{(a,t)} \lambda y_{a}. f(y) \land g(y)]\)

Note the use of the type variable \(\alpha\). This is called “type-polymorphism”.

GQs are \((e, t), t\); names are type \(e\). How can we get Mitka and every wolf (howled)?

**Lift-NP**: If \(\alpha = ([\alpha], NP, [\alpha])\), then there is a \(\beta = ([\alpha], S/R(S/LNP), \lambda P_{(e,t)}. P([\alpha]))\).

Lexicon : \([\text{[every]}, (S/R(S/LNP))/R N, \lambda P \lambda Q \forall x. P(x) \rightarrow Q(x)]\)

Lexicon : \([\text{[wolf]}, N, \text{wolf}]\)

R-Appl : \([\text{[every wolf]}, (S/R(S/LNP))/L(S/R(S/LNP)), \lambda g_{(e,t), t} \lambda y_{(e,t)}. [\lambda Q \forall x. \text{wolf}(x) \rightarrow Q(x)](y) \land g(y)]\)

Lexicon : \([\text{[and]}, (X/LX)/R-X, \lambda f_{(a,t)} \lambda g_{(a,t)} \lambda y_{a}. f(y) \land g(y)]\)

R-Appl : \([\text{[and every wolf]}, (S/R(S/LNP))/R(S/R(S/LNP)), \lambda g_{(e,t), t} \lambda y_{(e,t)}. [\lambda Q \forall x. \text{wolf}(x) \rightarrow Q(x)](y) \land g(y)]\)

**β-reduction**: \([\text{[and every wolf]}, (S/R(S/LNP))/L(S/R(S/LNP)), \lambda g_{(e,t), t} \lambda y_{(e,t)}. [\forall x. \text{wolf}(x) \rightarrow Q(x)](y) \land g(y)]\)

Lexicon : \([\text{[Mitka]}, NP, m]\)

**Lift-NP** : \([[\text{Mitka}], S/R(S/LNP), \lambda P_{(e,t)}. P(m)]\).

L-Appl : \([[\text{Mitka and every wolf}], S/R(S/LNP), [\lambda g_{(e,t), t} \lambda y_{(e,t)}. [\forall x. \text{wolf}(x) \rightarrow y(x)] \land g(y)](\lambda P_{(e,t)}. P(m))]\)

**β-reduction**: \([[\text{Mitka and every wolf}], S/R(S/LNP), \lambda y_{(e,t)}. [\forall x. \text{wolf}(x) \rightarrow y(x)] \land g(y)](\lambda P_{(e,t)}. P(m))]\)

**β-reduction**: \([[\text{Mitka and every wolf}}, S/R(S/LNP), \lambda y_{(e,t)}. [\forall x. \text{wolf}(x) \rightarrow y(x)] \land g(y)](\lambda P_{(e,t)}. P(m))]\)

Lexicon : \([[\text{howled}], S/LNP, \text{howled}]\)

R-Appl : \([[\text{Mitka and every wolf howled}}, S, \lambda y_{(e,t)}. [\forall x. \text{wolf}(x) \rightarrow y(x)] \land g(y)](\lambda P_{(e,t)}. P(m))]\)

**β-reduction**: \([[\text{Mitka and every wolf howled}}, S, [\forall x. \text{wolf}(x) \rightarrow \text{howled}(x)] \land \text{howled}(m)]\)
What about coordinations of names? (*Bill and Mary*)

- \[ \lambda f \alpha, \lambda g \alpha, \lambda y \alpha. f(y) \land g(y) ((\lambda P(e,t). P(b))((\lambda Q(e,t). Q(m))) \]

\[ \Rightarrow \lambda y(e,t). (\lambda P. P(b))(y) \land (\lambda Q. Q(m))(y) \]

\[ \Rightarrow \lambda y(e,t). y(b) \land y(m) \]

This gives a reasonable interpretation for *Bill and Mary laughed*:

- \[ \lambda y(e,t). y(b) \land y(m) \](laughed)

\[ \Rightarrow \text{laughed}(b) \land \text{laughed}(m) \]

This is just the GQ that is the intersection of the GQs denoted by lifted Bill and lifted Mary. But, this kind of conjunction can only give us distributive interpretations:

- *Barack and Michelle like carrots* ⇔ *Barack likes carrots and Michelle likes carrots*
- *Barack and Michelle met in 1989* ⇔ *Barack met in 1989 and Michelle met in 1989*

There’s something missing in this semantics for plurals, then!

Generalizing Lift:

- **Lift-α-Left**: If \( \alpha = ([\alpha], A, [\alpha]) \), then, for any category \( B \), there is a \( \beta = ([\alpha], B/_{e,t}((B/_{L,A}) \lambda P_{(e,t)}. P([\alpha]))). \)

- **Lift-α-Right**: If \( \alpha = ([\alpha], A, [\alpha]) \), then, for any category \( B \), there is a \( \beta = ([\alpha], B/_{L}((B/_{R,A}) \lambda P_{(e,t)}. P([\alpha]))). \)

Basically, the idea is to choose the version of Lift that anticipates the slash-direction of the thing that *would have* taken \( \alpha \) as an argument, and turn \( \alpha \) into something that can take *it* as an argument without affecting other aspects of the word order. So, if VP is looking for a NP to its left, a lifted NP needs to be looking for a VP to its right.

Generalized lift predicts a lot of new readings, e.g., those generated by lifting a VP (type \( (e, t) \)) to a property of GQs (type \( ((e, t), t, t) \). This allows a conjunction or disjunction to take “wide scope” over GQs, as you’ll explore in your next homework.

### 2 Intensionality

In the last homework we encountered a problem: intensional functional application (IFA) does not do a good job of dealing with intensional contexts in general, because it applies a world-variable to the embedded clause’s denotation, yielding a truth-value. This has the effect of making the grammar unable to distinguish among sentences that are true at a given world, and among those that are false at a world. This is a problem, because the following is clearly a coherent scenario:

- It’s raining at \( w \). \( \Rightarrow \lambda w. \text{rain}(w) \)
• It’s Tuesday at \( w \). \( \rightarrow \lambda w.\text{Tuesday}(w) \)

• Bill believes at \( w \) that it’s raining.

• Bill doesn’t believe at \( w \) that it’s Tuesday.

The meanings that we want, vs. those we derive:

• Bill believes at \( w \) that it’s raining.
  – What it should denote: \( \lambda w'.\text{believe}(w')(\lambda w.\text{rain}(w))(\text{Bill}) \)
    “Bill believes that he inhabits a world in the rain-set”
  – What an IFA-only grammar predicts it should denote: \( \lambda w'.\text{believe}(w')(\text{rain}(w'))(\text{Bill}) \)
    “If it is raining, Bill believes TRUE; if isn’t, he believes FALSE”

• Bill doesn’t believe at \( w \) that it’s Tuesday.
  – What it should denote: \( \lambda w'.\text{believe}(w')(\lambda w.\text{Tuesday}(w))(\text{Bill}) \)
    “Bill believes that he inhabits a world in the Tuesday-set”
  – What an IFA-only grammar predicts it should denote: \( \lambda w'.\text{believe}(w')(\text{Tuesday}(w'))(\text{Bill}) \)
    “If it is Tuesday, Bill believes TRUE; if isn’t, he believes FALSE”

Jacobson (2014: §19) discusses two solutions:

1. push application of world-variables into the lexicon, as in Montague 1973
2. modify the application rules to make them sensitive to the semantic types of the things being combined (Klein & Sag 1985)

Let’s try them both out.

• Solution 1: both extensional and intensional expressions take intensions as arguments; they just do different things with them. To implement we have to change the application rules to the following. (The difference from before is that we don’t apply the \( w \) that’s \( \lambda \)-abstracted to the argument.)

  **Right application:** If \( \alpha = ([\alpha], A\mid R B, [\alpha]) \) and \( \beta = ([\beta], B, [\beta]) \), then there is a \( \gamma = ([\alpha-\beta], A, \lambda w.[\alpha](w)([\beta])) \).

  **Left application:** If \( \alpha = ([\alpha], A, [\alpha]) \) and \( \beta = ([\beta], B\mid L A, [\beta]) \), then there is a \( \gamma = ([\alpha-\beta], B, \lambda w.[\beta](w)([\alpha])) \).

We then have to rewrite the semantic part of our lexicon: extensional items are now distinguished by the fact that they apply the \( w \) to their argument (thus, in effect, taking its extension as an argument). For example:

\[
[saw] = \lambda w\lambda x\lambda y.\text{saw}(w)(x(w))(y(w)) \quad (\text{compare: } \lambda w\lambda x\lambda y.\text{saw}(w)(x)(y))
\]
We’d need to do some serious lexical semantics to figure out where to go from here; but
sentences with extensional verbs like “Jim ate a fairy (poor guy, there aren’t any)”,
examples like “Jim looked for a fairy (poor guy, there aren’t any)”, as opposed to nonsensical
doesn’t exist at that world. This seems like a good result in light of quite intelligible

\[ \text{looked for} = \lambda w \alpha.\text{lookedFor}(w)(x)(y(w)) \]
\[ \text{looked for Hulk} = \lambda w_1, [\lambda w \alpha \lambda y.\text{lookedFor}(w_1)(x)(y(w_1))](\lambda w.\text{HH})(y(w_1)) \]

We’d need to do some serious lexical semantics to figure out where to go from here; but
the key observation is that \( x \) looked for Hulk could be true at a world, for some \( x \), even if HH
doesn’t exist at that world. This seems like a good result in light of quite intelligible examples like “Jim looked for a fairy (poor guy, there aren’t any)”, as opposed to nonsensical sentences with extensional verbs like “?? Jim ate a fairy (poor guy, there aren’t any)”.

- Solution 2: Type-driven translation.
  - see: type \( \langle s, (e, (e, t)) \rangle \)
  - look for: type \( \langle s, (\langle s, e \rangle, (e, t)) \rangle \)

<table>
<thead>
<tr>
<th>Right application-Extensional: If</th>
<th>Left application-Extensional: If</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha ) is of type ( \langle s, (\delta, \epsilon) \rangle ),</td>
<td>( \alpha ) is of type ( \langle s, \delta \rangle ),</td>
</tr>
<tr>
<td>( \alpha = ([\alpha], A/R B, [\alpha]) ),</td>
<td>( \alpha = ([\alpha], A, [\alpha]) ),</td>
</tr>
<tr>
<td>( \beta ) is of type ( \langle s, \delta \rangle ), and</td>
<td>( \beta ) is of type ( \langle s, (\delta, \epsilon) \rangle ), and</td>
</tr>
<tr>
<td>( \beta = ([\beta], B, [\beta]) ),</td>
<td>( \beta = ([\beta], B/L A, [\beta]) ),</td>
</tr>
<tr>
<td>then there’s a ( \gamma = ([\alpha - \beta], A, \lambda w.(<a href="w">\alpha</a>(\beta(w)))) ).</td>
<td>then there’s a ( \gamma = ([\alpha - \beta], A, \lambda w.<a href="w">\beta</a>(<a href="w">\alpha</a>)) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Right application-Intensional: If</th>
<th>Left application-Intensional: If</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha ) is of type ( \langle s, (\langle s, \delta \rangle, \epsilon) \rangle ),</td>
<td>( \alpha ) is of type ( \langle s, \delta \rangle ),</td>
</tr>
<tr>
<td>( \alpha = ([\alpha], A/R B, [\alpha]) ),</td>
<td>( \alpha = ([\alpha], A, [\alpha]) ),</td>
</tr>
<tr>
<td>( \beta ) is of type ( \langle s, \delta \rangle ), and</td>
<td>( \beta ) is of type ( \langle s, (\langle s, \delta \rangle, \epsilon) \rangle ), and</td>
</tr>
<tr>
<td>( \beta = ([\beta], B, [\beta]) ),</td>
<td>( \beta = ([\beta], B/L A, [\beta]) ),</td>
</tr>
<tr>
<td>then there’s a ( \gamma = ([\alpha - \beta], A, \lambda w.(<a href="w">\alpha</a>(\beta(w)))) ).</td>
<td>then there’s a ( \gamma = ([\alpha - \beta], B, \lambda w.<a href="w">\beta</a>([\alpha])) ).</td>
</tr>
</tbody>
</table>
The effect is of course the same: the difference is that we can get away without manipulating \( w \)'s individually for all extensional items. Maybe this is a savings, if one or the other is the default option.

References

