APPENDIX B: SOLUTIONS TO THE PROBLEMS

PRESESSION COURSE

Logic for Linguists

Christopher Potts
UMass Amherst

LSA Institute 2007, Stanford, July 1–3

\[ \lambda w \ldots \]
This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
# Problems and their solutions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Relative truth</td>
<td>94</td>
</tr>
<tr>
<td>B.2</td>
<td>Tarski’s hierarchy</td>
<td>95</td>
</tr>
<tr>
<td>B.3</td>
<td>Idioms</td>
<td>95</td>
</tr>
<tr>
<td>B.4</td>
<td>Nondeterministic translation</td>
<td>96</td>
</tr>
<tr>
<td>B.5</td>
<td>A subtlety of predicate notation</td>
<td>97</td>
</tr>
<tr>
<td>B.6</td>
<td>Exclusive union</td>
<td>98</td>
</tr>
<tr>
<td>B.7</td>
<td>Is it a function?</td>
<td>98</td>
</tr>
<tr>
<td>B.8</td>
<td>Characteristic sets and functions</td>
<td>100</td>
</tr>
<tr>
<td>B.9</td>
<td>Some counting</td>
<td>102</td>
</tr>
<tr>
<td>B.10</td>
<td>Schönfikelization and implications</td>
<td>103</td>
</tr>
<tr>
<td>B.11</td>
<td>nor</td>
<td>104</td>
</tr>
<tr>
<td>B.12</td>
<td>The type definition for ( \text{PL}_f )</td>
<td>105</td>
</tr>
<tr>
<td>B.13</td>
<td>A more readable ( \text{PL}_f )</td>
<td>105</td>
</tr>
<tr>
<td>B.14</td>
<td>Interdefinability</td>
<td>106</td>
</tr>
<tr>
<td>B.15</td>
<td>Relating ( \text{PL} ) and ( \text{PL}_f )</td>
<td>107</td>
</tr>
<tr>
<td>B.16</td>
<td>( \text{PL}_f ) and negation</td>
<td>108</td>
</tr>
<tr>
<td>B.17</td>
<td>( \text{PL}_f ) and implication</td>
<td>109</td>
</tr>
<tr>
<td>B.18</td>
<td>Exclusive disjunction</td>
<td>109</td>
</tr>
<tr>
<td>B.19</td>
<td>( \text{PL}_f ) and compositionality</td>
<td>111</td>
</tr>
<tr>
<td>B.20</td>
<td>Conjunctions and constituency</td>
<td>111</td>
</tr>
<tr>
<td>B.21</td>
<td>Coordination and function composition</td>
<td>113</td>
</tr>
<tr>
<td>B.22</td>
<td>( \text{PL} ) intensions</td>
<td>113</td>
</tr>
<tr>
<td>B.23</td>
<td>Alternative type definition</td>
<td>115</td>
</tr>
<tr>
<td>B.24</td>
<td>Possible types given assumptions</td>
<td>116</td>
</tr>
<tr>
<td>B.25</td>
<td>Vacuous abstraction</td>
<td>116</td>
</tr>
<tr>
<td>B.26</td>
<td>Partiality</td>
<td>117</td>
</tr>
<tr>
<td>B.27</td>
<td>Novel types and meanings</td>
<td>118</td>
</tr>
<tr>
<td>B.28</td>
<td>Types, expressions, and domains</td>
<td>118</td>
</tr>
<tr>
<td>B.29</td>
<td>Recursive interpretation</td>
<td>119</td>
</tr>
<tr>
<td>B.30</td>
<td>An alternative mode of composition</td>
<td>120</td>
</tr>
<tr>
<td>B.31</td>
<td>Assignments</td>
<td>122</td>
</tr>
<tr>
<td>B.32</td>
<td>Variable names</td>
<td>123</td>
</tr>
<tr>
<td>B.33</td>
<td>Cross-categorial and</td>
<td>123</td>
</tr>
<tr>
<td>B.34</td>
<td>A relational reinterpretation</td>
<td>124</td>
</tr>
<tr>
<td>B.35</td>
<td>What’s the source of the ill-formedness?</td>
<td>125</td>
</tr>
<tr>
<td>B.36</td>
<td>Building a fragment</td>
<td>127</td>
</tr>
<tr>
<td>B.37</td>
<td>Substitution</td>
<td>129</td>
</tr>
<tr>
<td>B.38</td>
<td>Beta reductions</td>
<td>129</td>
</tr>
<tr>
<td>B.39</td>
<td>Eta conversion and distinguishable meanings</td>
<td>131</td>
</tr>
</tbody>
</table>
B.40 Extensional beliefs? ...................................................... 131
B.41 Modals ................................................................. 132
B.42 Hintikka’s believe ..................................................... 133
B.43 Individual concepts .................................................. 133
B.44 How many worlds are there? ..................................... 134
B.45 Finding common ground ............................................ 134
B.46 Definites and semantic composition ............................. 135
B.47 Contradictory beliefs .................................................. 136
B.48 Degree constructions ................................................ 137
B.49 Singular and plural .................................................... 138
B.50 Exactly 1 in first-order logic? ..................................... 138
B.51 A closer look at the universal ..................................... 139
B.52 All and only Lisa’s properties .................................... 140
B.53 Intensional quantifiers ............................................... 140
B.54 Nonconservative determiners? ................................. 141
B.55 Coordination and monotonicity ................................. 142
B.56 Indexicals as proper names? ........................................ 143
B.57 Indexicals and constants: A crucial difference ............. 143
B.58 Denotations as sets of assignments ............................ 144
B.59 Dynamic indefinites .................................................... 144
B.60 Probabilities and sets .................................................. 146
Handout B: Problems and their solutions

This appendix repeats the exercises from appendix A and provides my (generally quite complete) solutions to them. Of course, for problems marked OPEN, I can’t wrap things up neatly. For them, my answers probably just give a sense for why the problems remain open.

B.1 Relative truth

Background  Section 2.1.1 of handout 2 introduced the truth values and a highly qualified perspective on truth. In introductory texts, one often gets the sense that truth is presumed to be Truth in some absolute sense — truth about our reality. But advanced texts and the primary literature tend to evaluate for truth in less absolute terms.

Your task  Find two situations in which truth is evaluated, not with respect to our reality, but rather with respect to a (potentially) different one. Might the people in these situations show some awareness that their reality isn’t the only (or true) one? Might your situations have an impact on how we design our semantic theory? If so, how? If not, why not?

My answer  Here are a few such situations:

- **Fiction (film, books, plays, etc.)**  The characters in these are unlikely to be aware that they are in a fictional world, though many artists toy with this boundary. Fictional worlds are blends of the real world and imaginary ones. There is no doubt that truth is defined in terms of these alternative realities.

- **Suppositions**  If I say, *Suppose it were . . .* We might reason for an indefinite period of time in terms of the hypothetical reality I set up, depending throughout on the nature of the suppositions to define truth. How different this world is from the actual world is a highly variable affair. We’re likely to hang on to our awareness that the actual world might not work the way our suppositions do.

- **What might have been . . .**  We are expert at imagining alternative courses of events than the ones that actually transpired.

- **What people believe, claim, dream, . . .**  Our beliefs (and related cognitive objects) can be totally out of whack with reality, but they are likely to have their own internal deductive structure, build up from relativized notions of truth. When we talk about others’ belief states, we
often invoke a contrast between that believed reality and the actual one as we perceive it. Speakers can do this as well, which is the start of an understanding for why I think that $S$ is generally weaker than $S$ alone.

**B.2 Tarski’s hierarchy**

**Background** In section 2.1.3 of handout 2, I distinguished object languages from metalanguages, and I recruited some notation from set theory (handout 3, section 3.1) to play the role of rigorous metalanguage.

**Your task** How would you respond to the skeptic who claimed that he needed a semantics for set theory, to feel confident that its interpretation was well defined? Would giving a semantic theory of $\in$, $\subseteq$ and the like satisfy the skeptic? (Why not?)

**My answer** It is difficult to respond to the skeptic. He is unlikely to be satisfied by attempts to give a rigorous semantics for set theory. Suppose you do that using some metalanguage $M$. The skeptic, if he is worth is salt, will then demand a metalanguage for $M$. Use $M'$ for that. He’ll ask for a semantics for $M'$. And so forth, ad infinitum. We are climbing the Tarski hierarchy, named for the logician Alfred Tarski, who is one of the main developers of truth-definitions of the sort we use in this course. The bottom line: at some point, you have to stop asking questions and appeal instead to our intuitive understanding of the concepts involved. I think this is the only reply to the skeptic within this general semantic framework.

**B.3 Idioms**

**Background** Complex idioms are obvious challenges for compositionality in any of its forms. It seems that, on their idiomatic uses, none of the expressions in (A.1) has a meaning that is predictable from the meanings of its parts:

(B.1) a. Ed kicked the bucket. (Ed died.)
    b. Ed bought the farm. (Ed died.)
    c. Ed kept tabs on Joe. (Ed tracked Joe’s actions.)

One strategy would be to say that phrases like kick the bucket are lexical items. Thus, they are where compositionality bottoms out: atomic items with primitive (non-derived) meanings. But the following examples seem to reveal that at least some idioms are syntactically complex:
Solutions

(B.2)  

a. "The bucket was kicked by Ed." (slightly odd on the idiomatic reading).

b. Tabs were kept on Joe (by Ed).

Your task  Articulate why these facts are challenging for compositionality, and outline a possible resolution (or a few of them).

My answer  Compositionality demands that we give a meaning to each lexical item and a principled means of combining those lexical meanings into ever more complex meanings. Thus, we know what *kick* means, and we know what *the* and *bucket* mean, so we put them together to form the verb phrase *kick the bucket*, and our semantics gives us the set of all entities $d$ such that $d$ kicked the (contextually salient) bucket . . . .

We certainly want to be able to do that with *kick the bucket*. The problem comes when we consider the idiomatic reading. It seems that there will be no principled way of combining these lexical items into something that means (roughly) what *die* means. If there were such a principled way, then the meaning wouldn’t be idiomatic!

We could try to say that idioms are complex lexical items. But then we run afoul of the fact that many clearly have their own internal structure, as we see from movements like those in (B.2).

The only resolution I see is to relax compositionality in the places where it fails even at an intuitive level, namely, for idioms. Doing this might actually be freeing. Proponents of Construction Grammar embrace the idea that many or most phrases have some degree of idiomaticity. It doesn’t worry them. Rather, it is evidence that we often don’t fully decompose meanings in the way that compositionally suggests.

B.4 Nondeterministic translation

Background  Throughout section 2.4 of handout 2, I tacitly assumed that translation into the logical language was functional: though two expressions might map to the same piece of logic, no single expression mapped to more than one piece of logic. That is, I assumed translation was functional (handout 3, section 3.4).

Your task  Suppose that the translation process can be one-to-many. That is, suppose a single expression $E$ translates to distinct logical symbols $L$ and $L'$, and suppose that $L$ and $L'$ denote different model-theoretic objects. What would this mean for the status of translation? How could a motivated argument for this nondeterministic translation inform the debate about whether interpretation is direct or indirect?
My answer  The problem asks us to imagine a situation like the following (where $L$ and $L'$ are different logical symbols and $M$ and $M'$ are different meanings):

If such situations exist, then the logical symbols are essential. By assumption, there is just one piece of syntax/morphology $E$. But it has two distinct translations, and if it is true that $E$ can ultimately be perceived as either $M$ and $M'$, then we need to look to the logic to make that distinction. Thus, the meaning language is elevated from convenience to theoretical necessity.

An important qualification: there are many situations that look, from the outside, like the above. They are cases of ambiguity. The usual response is to say that there are multiple $E$s sharing the same phonology. This is entirely reasonable, of course. I mention it here only to emphasize that it could be very hard to establish that situations like the above are attested.

B.5 A subtlety of predicate notation

Background  The predicate notation for sets (section 3.1, handout 3) is mostly straightforward. But it hides some subtleties. Let’s tease out one of them.

Your task  Describe the following sets in a way that is less obscure:

i. \{n \mid n \text{ is a natural number and } 3 < 4\}

ii. \{n \mid n \text{ is a natural number and } 3 > 4\}

What role does the second conjunct play in each case?

My answer  The first set picks out the set of all natural numbers. The second picks out the empty set:

- In the first, the second conjunct is always true, no matter what. So we look to the first conjunct, and we find that it specifies the natural numbers.
- In the second, the second conjunct is false, and hence the entire statement is false for any choice of $n$. Even if we pick a natural number, thereby satisfying the first conjunct, the second prevents our selection from being included.
B.6  Exclusive union

**Background**  The union $A \cup B$ of two sets $A$ and $B$ contains the members of $A \cap B$ (handout 3, section 3.1.4.2). This can be slightly counterintuitive.

**Your task**

1. Use $\subseteq$ to specify the relation that lawfully holds between $A \cap B$ and $A \cup B$.
   
   **My answer**  For all $A$ and $B$, $(A \cap B) \subseteq (A \cup B)$. Thus, there is an entailment relation: intersection of two sets entails (is a subset of) the union of those two sets.

2. Define an exclusive union operator — symbol of your choosing — that excludes $A \cap B$.

   **My answer**  Exclusive union can be defined in a few equivalent ways. Here is a succinct version:
   
   $$A \overline{\cup} B \overset{\text{def}}{=} (A \cup B) - (A \cap B)$$

   Now it holds that $(A \cap B)$ and $(A \overline{\cup} B)$ are always disjoint.

B.7  Is it a function?

**Background**  Functions are defined in section 3.4 of handout 3. They are everywhere in linguistics, so it is essential that you be able to spot them in the wild.

**Your task**  For each of (B.3)–(B.9), say whether or not it is a function. If it is a function, say also whether it is an *onto* function and whether it is a *total* function.

(B.3)  

1  

0

**My answer**  It’s not a function; Lisa has multiple values.
My answer  It’s a function. It is total. It is not onto: nothing maps to 0.

My answer  It’s a function, but it is neither total nor onto.

My answer  It’s a function — total, onto, and one-to-one, so it is a bijection.
My answer  It's a function. Not a total one (Homer has no value), but an onto one. It's also a one-to-one correspondence (but not a bijection, due to Homer's lack of a value).

(B.8) the relation $R$ from nodes to nodes in tree structures that maps each node to its daughter(s)

My answer  This is not a function for any tree with branching. In such trees, at least some nodes have more than one daughter. We can, however, define a (partial) set-valued function that maps a node to the set of all its daughters.

(B.9) the relation $R^{-1}$ from nodes to nodes in tree structures that maps each node to its mother(s)

My answer  For trees without multiple mothers, this is a function. It is partial, though, because the root has no mother. It is not onto because terminals have no daughters.

B.8  Characteristic sets and functions

Background  Semanticists are apt to switch freely back and forth between talking in terms of sets and talking in terms of functions. So it's smart to get used to making such mental remappings.
**Problems**

**Your task**  Specify the characteristic set for the function depicted here:

My answer  The characteristic set of the above is 

\[ \{ T, A \} \]

**Your task**  Assume that the domain of inquiry is 

Specify the characteristic function of this set: 

My answer  The characteristic function of the above is
B.9 Some counting

Background Handout 3 defines lots of different relations on sets. It can be illuminating to count the number of objects in a given domain (of functions, of sets, of tuples, etc.). It is revealing of the relationships between domains, and it also gives us a sense of just how large our models are.

The notation $D^D_r$ is often used to describe total functions from the domain $D_{\sigma}$ into the domain $D_r$. So the range is inline, and the domain is a superscript. This is presumably because the number of such functions is

$$|D_r|^{|D_{\sigma}|}$$

where $|A|$ is the cardinality of the set $A$ and the superscript is an exponent.

Your task Let $A = \{a, b, c\}$ and let $B = \{T, F\}$.

i. How many total functions are there from $A$ into $B$?

My answer There are $|B|^{|A|} = 2^3 = 8$ total functions from $A$ into $B$.

ii. How many objects are in the powerset of $A$? How does your result help us understand why the powerset of $A$ is often given as $2^A$?

My answer There are $2^{|A|} = 2^3 = 8$ sets in the powerset of $A$, so the $2^A$ notation is just intentionally blurring the distinction between the set and its cardinality.

The number is a clue as to why we have a correspondence between sets and total functions (handout 3, section 3.4.3). For every set in the powerset, there is a corresponding function, and the reverse.

iii. How many objects are in $A \times B$. And what is the general method for calculating the number of $n$-tuples in $X_1 \times \cdots \times X_n$?

My answer There are $|A| \cdot |B| = 3 \cdot 2 = 6$ objects in this cross-product. The general method is just to get the cardinality of each set and multiply those numbers together. The idea is that we form these pairs by picking one thing from each set.

iv. How many objects are in the set of all functions from $A \times B$ into $A$? In general, how many objects are in the set of all functions from $X$ into the set of all functions from $Y$ into $X$ (i.e., $X \mapsto (Y \mapsto Z)$?)
Problems

**My answer** Let’s look at the numbers for two specific cases:

- \((A \times B) \mapsto A\) \(|A|^{A|B|} = 2^6 = 64\)
- \((B \mapsto A) \mapsto A\) \(|A|^{A|B|} = 2^3 = 64\)

So, in general, we just keep taking exponents on the domain of the function to get the cardinalities for complex functions.

This is also a look at why the Schönfinkel trick works. Since the counts match, we can set up a bijection between the two sets of functions.

## B.10 Schönfikelization and implications

**Background** Section 3.4.5 of handout 3 discusses schönfikelization with only functions in mind. But the idea is in evidence in natural language as well:

(S)  
- a. If I take out the trash, then if I mow the lawn, I will get my allowance.
- b. If I take out the trash and mow the lawn, I will get my allowance.
- c. If I mow the lawn and take out the trash, I will get my allowance.
- d. If I mow the lawn, then if I take out the trash, I will get my allowance.

These statements have identical truth conditions. If you write (a) or (d), you do well to rewrite it as (b) or (c), which are easier to understand. But your choice doesn’t affect the claim you are making about the world.

**Your task** Articulate the intuitive connection between the data in (S) and Schönfinkel’s trick.

**My answer** If we think of functions as implications, then the implications in (S) have the same basic pattern as the unary and binary functions that we can relate by Schönfikelization. Thus, we are seeing, at an intuitive level, why Schönfikelization preserves the structure of the functions involved.

**Your task** Create PL or PL₉ translations of the sentences in (S) and give their truth table. What do you see?

**My answer** The following supports our intuitions about the English examples, and it helps us see why the Schönfinkel equivalences hold, because the
three formulae that correspond roughly to binary and unary functions all have the same PL truth conditions.

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>(p ∧ q)</th>
<th>(p → r)</th>
<th>(q → r)</th>
<th>((p ∧ q) → r)</th>
<th>(p → (q → r))</th>
<th>(q → (p → r))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

B.11  *nor*

**Background**  There are many more definable connectives than appear on handout 4. A couple of them have the magical property of being truth functionally complete, and one might even be a reasonable translation of an English word. Let’s look.

**Your task**  Define a PL connective that seems suitable for the English expression *neither* . . . *nor*. (You can imagine that it’s just *nor* you’re defining, so that you have a binary operator akin to ∨.)

i. State the translation hypothesis in a form comparable to that of hypothesis (4.7b).

(You can make up your own symbol.)

**My answer**  The hypothesis: *nor* \( \sim \dagger \)

ii. Give the type for your connective, using the system of section 4.3.1.1.

**My answer**  The type of this connective is \( \langle t, \langle t, t \rangle \rangle \).

iii. Provide the interpretation for your connective, in the manner of section 4.3.2.2.

**My answer**  The interpretation of this connective:

\[
\begin{bmatrix}
T & \mapsto & T \\
F & \mapsto & F \\
F & \mapsto & T \\
T & \mapsto & F \\
\end{bmatrix}
\]

That is, this is the function that take us to T iff both arguments are F.
B.12  The type definition for PLₜ

Background  The type definition given in section 4.3.1.1 of handout 4 is restrictive in two ways. First, all types have \( t \) inputs (where the input is the left member). The input is never anything more complex like \( \langle t, t \rangle \). Second, it is finite: only \( t \), \( \langle t, t \rangle \), and \( \langle \langle t, t \rangle \rangle \) are types.

Your task  Generalize the type definition so that it specifies infinitely many types, but maintain the restriction that inputs are always \( t \).

My answer  Here’s the requisite definition:

i.  \( t \) is a type.

ii.  If \( \sigma \) is a type, then \( \langle t, \sigma \rangle \) is a type.

iii.  Nothing else is a type.

The intuition: the second clause allows these functions to grow in terms of the size of their outputs, but they always have \( t \) inputs. So there is no provision for forming types like \( \langle \langle t, t \rangle, t \rangle \). But we can have types like \( \langle t, \langle t, \langle t \rangle \rangle \rangle \). That is, we could define 3-place, 4-place, \ldots, connectives if we wanted to.

B.13  A more readable PLₜ

Background  Section 4.3.1.2 of handout 4 defines the syntax of PLₜ. It is an efficient definition, and I proposed it because it means that we can fit this system into the ones that come later with no fuss. But the results quickly become hard to parse. Some examples, with their more intuitive versions at right:

i.  \( (((\land(p))(q)))(q \land p) \)

ii.  \( (\lor((\land(p))(q))(p))(p \lor (q \land p)) \)

iii.  \( (((\land(\neg(p)))(p))(p \land \neg p) \)

The expressions are just tools for helping people understand what is happening with the denotations (functions). At present, they are not particularly illuminating.

Your task  Devise some new syntactic rules, replacements for those in (4.3.1.2), that determine expressions of the sort at right but maintain the virtue of the current system that \( \land, \land p \), and the like are well formed.
My answer  The following does a pretty good job, though there is still an excess of bracketing compared to standard PL:

i. \( p, q, p', q', \ldots \) are formulae of PL\(_t\), type \( t \).

ii. \( \neg, \top, \bot \) are formulae of PL\(_t\) (one-place connectives), type \( \langle t, t \rangle \).

iii. \( \land, \lor, \rightarrow, \leftrightarrow \) are formulae of PL\(_t\) (two-place connectives), type \( \langle t, \langle t, t \rangle \rangle \).

iv. If \( \dagger \) is of type \( \langle t, t \rangle \) and \( \varphi \) is of type \( t \), then
   a. if \( \dagger \) is defined in clause (ii), then \( \dagger \varphi \) is a formula of PL\(_t\), type \( t \)
   b. else \( \varphi \dagger \) is a formula of PL\(_t\), type \( t \)

v. If \( \ddagger \) is of type \( \langle t, \langle t, t \rangle \rangle \) and \( \varphi \) is of type \( t \), then \( \ddagger \varphi \) is of type \( \langle t, t \rangle \).

The clause for one-place connectives is a bit complex, because we want lexical type \( \langle t, t \rangle \) operators like \( \neg \) to take their arguments to the right, but we want derived \( \langle t, t \rangle \) operators like \( \land p \) to take their arguments to the left.

B.14 Interdefinability

Background  It is possible to make due entirely with just one binary connective and a negation. All others are definable in terms of combinations of them. For instance, it is common to treat \( \varphi \rightarrow \psi \) as an abbreviation for \( \neg \varphi \lor \psi \).

Your task  Using truth tables, show that treating \( \varphi \rightarrow \psi \) as an abbreviation for \( \neg \varphi \lor \psi \) gives us the arrow defined on handout 4. Then show how this definition works for the PL\(_t\) functions as well. If you want to push still further by combining this answer with exercise A.11, then see how much mileage you can get out of a nor-like connective.

My answer  Here are the truth-tables for the two formulae in question:

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \neg p )</th>
<th>( p \lor q )</th>
<th>( \neg p \lor q )</th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The most efficient way to see the equivalence in terms of functions is to consider the functional composition of \( \|\neg\|^M \) with \( \|\lor\|^M \):

\[
\begin{bmatrix}
T \\
F \\
F
\end{bmatrix} \iff \begin{bmatrix}
T \\
F \\
F
\end{bmatrix} \circ \begin{bmatrix}
T \iff T \\
T \iff F \\
F \iff T
\end{bmatrix} = \begin{bmatrix}
T \\
F \\
F
\end{bmatrix} \iff \begin{bmatrix}
T \iff T \\
F \iff F \\
F \iff T
\end{bmatrix}
\]
And here are some equivalences for $\dagger$ as defined in (A.11), i.e., some ways of defining the other connectives in terms of $\dagger$:

i. $\neg \varphi \overset{\text{def}}{=} (\varphi \dagger \varphi)$

ii. $(\varphi \lor \psi) \overset{\text{def}}{=} \neg (\varphi \dagger \psi)$

iii. $(\varphi \land \psi) \overset{\text{def}}{=} \neg (\neg \varphi \lor \neg \psi)$

**B.15 Relating PL and PL$_t$**

**Background** It would be useful to establish that PL and PL$_t$ are logically equivalent, so that we can be confident that our use of PL$_t$ doesn’t affect our predicted truth conditions.

**Your task**

i. Establish a translation function from the expressions of PL to those of PL$_t$.

ii. Show that this translation function preserves truth.

(The reverse direction is harder because not all PL$_t$ expressions correspond to well-formed formulae of PL. One must concentrate on the truth-valued expressions.)

**My answer** Here is the translation procedure, which make use of a function $\text{Trans}[\cdot]$ that maps PL formulae into PL$_t$ formulae:

i. $\text{Trans}[p] = p$ if $p$ is a propositional letter

ii. $\text{Trans}[\neg \varphi] = (\neg (\text{Trans}[\varphi]))$

iii. $\text{Trans}[(\varphi \land \psi)] = ((\land (\text{Trans}[\psi]))(\text{Trans}[\varphi]))$

(and so forth for the other binary connectives)

Here is a complex example illustrating the recursive nature of this definition:

$$\text{Trans}[\neg (p \land q)] = (\neg (\text{Trans}[(p \land q)]))$$

$$= (\neg ((\land (\text{Trans}[p]))(\text{Trans}[q])))$$

$$= (\neg ((\land (p))(q)))$$

The second part of this task is to show that the translation is truth preserving. This calls for a proof, and the best way to proceed with the proof is to go clause-by-clause, starting with first one. Throughout, we assume that we are interpreting the formulae in a single model $M$.

Clause (i) is clearly truth preserving: letters are the same in both logical languages.
Problems

For clause (ii), we need to make an *inductive hypothesis*: on the grounds of our results for clause (i), we suppose that the truth-preservation hypothesis holds for \( \varphi \) and we move to showing that \( \neg \) doesn’t falsify the hypothesis.

So suppose we are looking at the PL formula \( \neg \varphi \), and assume that the hypothesis holds for \( \varphi \), i.e., that \( \varphi \) has the same interpretation for both PL and \( \text{PL}_f \). Suppose that interpretation is \( \$ \), which is either \( T \) of \( F \), of course. Then the overall value of \( \neg \varphi \) is \( T \) if \( \$ = F \), else it is interpreted as \( T \) — we read that off the truth table. Similarly, \( \neg(\varphi) \) is interpreted as \( T \) if \( \$ = F \), else it interpreted as \( F \) — we read that off the denotation of \( \neg \) and the rule of composition.

The proof works similarly for the binary connectives. For them, we just assume the inductive hypothesis holds for the two arguments and then reason based on the fact that \( \land \) has the same basic semantics in both PL and \( \text{PL}_f \).

B.16 \text{PL}_f \text{ and negation}

**Background** Suppose we hypothesize that *not/n’t \( \sim \neg \), where \( \neg \) is the \( \text{PL}_f \) operator whose semantics is given in section 4.3.2.2 of handout 4.

**Your task** How does this hypothesis fare in light of the natural language data you know about? (The best way to answer this is to create a list of properties of \( \neg \) and check them against the linguistic facts.)

**My answer** This question, like the next one, is open-ended, and a lot of facts from a lot of languages could be relevant to evaluating it. So let me instead just highlight some properties of \( \neg \) that might or might not be shared by natural language negation:

- \( \neg \) applies recursively: \( \neg \neg \varphi \) is well-formed if \( \varphi \) is, and so on for more negations. Stack away!
- Multiple negations flip truth values: \( \neg \neg \varphi \) has the same meaning as \( \neg \varphi \), and \( \neg \neg \varphi \) has the same meaning as \( \neg \varphi \). In general, if two negations occur in a row (in the same clause), you can remove them without changing the semantic value.
- \( \text{PL}_f \) negation can take only truth-valued arguments.
- \( \text{PL}_f \) negation appears at the periphery of clauses.
### B.17 PLₓ and implication

**Background**  Suppose we hypothesize that if... then →, where → is the PL₃ operator whose semantics is given in section 4.3.2.2 of handout 4.

**Your task**  How does this hypothesis fare in light of the natural language data you know about? (The best way to answer this is to create a list of properties of → and check them against the linguistic facts.)

**My answer**  As with exercise A.16 above, this question is extremely open in terms of relevant data and considerations. So I’ll again just highlight important features of → and leave the linguistic evaluation to you:

- (ϕ → ψ) is true whenever its left argument (antecedent) is true, no matter what the meaning of the consequent is.
- (ϕ → ψ) does not entail (ψ → ϕ).
- (ϕ → ψ) does entail (¬ψ → ¬ϕ). (This inference is called *modus tollens*.)
- → is binary.
- → sits between its arguments.
- PL₃ gets its arguments one at a time.

### B.18 Exclusive disjunction

**Background**  Linguists and laymen alike are constantly tempted to assume that sentences involving or entail that exactly one of the disjuncts is true. It is therefore tempting to define an exclusive disjunction with those truth conditions and hypothesize that it is the basis for the semantics of or. This exercise is likely lead you to the conclusion that this approach won’t work.

**Your task**

i. The PL₃ operator ∨ is defined so that \[ [p \lor q]^M = T \text{ if } [p]^M = [q]^M = T \]. Define a corresponding exclusive disjunction operator that excludes this case (symbol of your choosing).
My answer

Here is a picture of a PL$_e$ exclusive disjunction:

\[ ||\lor||^M = \begin{bmatrix}
T & T & F \\
F & T & T \\
F & F & F
\end{bmatrix}\]

It is exclusive because it “excludes” the case where both conjuncts are true. It entails, but is not entailed by, inclusive disjunction.

ii. Draw a truth table for a formula consisting of two exclusive disjunctions. What is odd about the values this turns up?

My answer

Here’s a look at the truth-table for a ternary exclusive disjunction:

<table>
<thead>
<tr>
<th>$\varphi$</th>
<th>$\psi$</th>
<th>$\omega$</th>
<th>$(\varphi \lor \psi)$</th>
<th>$((\varphi \lor \psi) \lor \omega)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

The surprising lines are the ones like the first. All three disjuncts are true in that line, but the ternary exclusive disjunction comes out true there, because one of its disjuncts is false! We were hoping for an operator that would deliver truth iff exactly one of the disjuncts involved (however many there may be) was true. But no such operator can be built up from binary formulae in this way.

iii. Suppose that your exclusive disjunction provides the meaning for English or. What prediction would this make about the sentence *Sam is at the store, or Sam is on his cell phone, or Sam is inspecting broccoli*?

My answer

If we hypothesized that English or translates as $\lor$, then we would predict that *Sam is at the store, or Sam is on his cell phone, or Sam is inspecting broccoli* would be judged true if all three of its disjuncts were true. We would also incidentally, predict very strange things about negated disjunctions. (Think about the truth table for $\neg(\varphi \lor \psi)$: truth iff neither or both disjuncts are true.)
Problems

B.19  PL_f and compositionality

**Background**  Our hypothesis is that every declarative sentence $S$ translates as some propositional letter $p$.

**Your task**  Amass as many arguments as you can think of for why this is a hopelessly bad hypothesis.

**My answer**  I think the following (some covered in handout 4) are the central factual and conceptual reasons for rejecting this hypothesis:

i. We would have to list all the monoclausal declarative sentences.

ii. We would be unable to relate even obviously very closely related sentences (e.g., Chris runs and Chris smiles and Kathryn smiles).

iii. Though we could match each distinct sentence with a unique letter, it would be a sort of pointless proliferation of logical symbols: we care about the meanings, and there are only two meanings for letters: T or F.

iv. We would wrongly predict that speakers do not have any intuitions at all about the well-formedness of subparts of sentences.

B.20  Conjunctions and constituency

**Background**  The conjunction operator of PL_f takes first its right argument and then its left argument. This strongly suggests that the syntax is asymmetric in a parallel way. Suppose we deny this. That is, suppose that the syntactic structure of coordinations is more like

\[
\begin{array}{c}
S \\
/ \\
S \quad \text{and} \quad S
\end{array}
\]

**Your task**  How might we devise a semantics that does justice to these structures and predicts the same truth conditions as our binary version? (It might be useful to think about currying in this case (section 3.4.5 of handout 3).

If you’re looking for an additional challenge: what would it take to generalize your definition of and to $n$-ary conjunction, for any finite $n$? You might try to write down a lambda term. Be sure to confront, in prose if not in symbols, the fact that we can’t fix $n$ ahead of time.
My answer  The central tool here is the Schönfinkel correspondence, which tells us that the following relationship is just one instance of a general correspondence:

\[
\begin{array}{cccccccc}
T & \mapsto & T \\
F & \mapsto & F \\
\end{array}
\begin{array}{cccccccc}
\langle T, T \rangle & \mapsto & T \\
\langle T, F \rangle & \mapsto & F \\
\langle F, T \rangle & \mapsto & F \\
\langle F, F \rangle & \mapsto & F \\
\end{array}
\]

If we use the function that maps pairs of truth values into truth values, then we need a few additional ingredients:

i. **Product types**  A product type is of the form \(\sigma \times \tau\), where \(\sigma\) and \(\tau\) are types.

ii. **Domains**  The domain for a product type \(\sigma \times \tau\) is \(D_\sigma \times D_\tau\). As a result of these assumptions, product types associate with ordered pairs. The type we need for the above is \(t \times \bar{t}\).

iii. **Composition**  Our old rule of composition won’t work for these cases. Here is a new one:

- If \(\varphi\) and \(\psi\) are of type \(\sigma\) and \(\mathfrak{d}\) is of type \(\langle\sigma \times \sigma, \tau\rangle\), then \((\varphi \upharpoonright \psi)\) is of type \(\tau\).

This rule determines flat parsetrees like the following:

\[ S \]
\[ \varphi \]
\[ \wedge \]
\[ \psi \]

These resemble the parsetrees for PL, but with the crucial gain that \(\wedge\) is a terminal node, i.e., it is introduced categorematically (handout 4, section 4.4.2).

iv. **Interpretation**  The interpretation rule gathers the two conjuncts into a pair and applies them to the meaning of the connective:

\[ \llbracket (\varphi \upharpoonright \psi) \rrbracket^M = \llbracket \mathfrak{d} \rrbracket^M (\llbracket \varphi \rrbracket^M, \llbracket \psi \rrbracket^M) \]

An added challenge comes from trying to generalize this to \(n\)-ary conjunction, so that we can analyze coordinations like Bob, (and) Mary, (and) Ted, and Alice. Intuitively, we just want to gather all the arguments into an \(n\)-tuple. For this, we need a schematic definition, one that holds for any \(n\). Here are the steps required:

i. Let \(\sigma_1 \times \cdots \times \sigma_n\) abbreviate a product type of product types with \(n\) members.
ii. Assume there are infinitely many $\wedge$ operators, one for each natural number $i$, defined as follows:
   a. $\wedge_i : (t_1 \times \cdots \times t_i, t)$ where $i \geq 2$
   b. $||\wedge_i||^M$ is the function that maps any $i$-tuple of truth-values $A$ to $T$ iff every member of $A$ is $T$.

B.21 Coordination and function composition

**Background**  Function composition is defined in handout 3, section 3.4.4. It’s also helpful to see what it looks like when expressed in lambda terms:

$$\text{Where } \varphi : \langle \sigma, \rho \rangle \text{ and } \psi : \langle \rho, \tau \rangle, \psi \circ \varphi \overset{\text{def}}{=} \lambda \chi. \psi(\varphi(\chi))$$

**Your task**  Is function composition commutative? If it is, then prove that claim. If it is not, then find two functions $f$ and $g$ for which $(f \circ g)$ and $(g \circ f)$ are well defined, but $(f \circ g) \neq (g \circ f)$. What would happen if we modeled coordination in these terms?

**My answer**  Function composition is not commutative. (Order matters.) Here is a simple example involving just two simple unary truth functions:

$$\begin{align*}
\begin{bmatrix} T & \mapsto & F \\
F & \mapsto & T
\end{bmatrix} & \circ \begin{bmatrix} T & \mapsto & T \\
F & \mapsto & T
\end{bmatrix} = \begin{bmatrix} T & \mapsto & F \\
F & \mapsto & F
\end{bmatrix} \\
\begin{bmatrix} T & \mapsto & T \\
F & \mapsto & T
\end{bmatrix} & \circ \begin{bmatrix} T & \mapsto & F \\
F & \mapsto & F
\end{bmatrix} = \begin{bmatrix} T & \mapsto & T \\
F & \mapsto & T
\end{bmatrix}
\end{align*}$$

In dynamic semantics, and, as well as the operator that joins independent sentences, is often analyzed as function composition. This goes a long way to capturing the order dependency we see in discourse when we look at anaphoric constructions like ellipsis, anaphora, and presupposition.

B.22 PL intensions

**Background**  This course is building quickly to an intensional perspective on meanings. It is important to see that the roots of this idea are present in PL and PL$^F$ as well. This exercise asks you to draw that perspective out, by redefining the logical constants so that they are less about truth than about possibilities.
**Your task**  Suppose we wanted to take more seriously the metalogical observations about intensionality in PL, as summarized in section 4.7. Suppose we wanted to do interpretation in terms of the sets of indices at the bottom of that truth table.

i. What would be an appropriate domain for the type \( \tau \) in that case?

**My answer**  \( D_\tau \) should be the set of all interpretation functions.

ii. What would be appropriate denotations for the following connectives in light of your reformulation of \( D_\tau \)?

a. \( \neg \)

**My answer**  \( \neg \) is set complementation: the meaning of \( \neg \phi \) is the set of all interpretation functions \( \llbracket \cdot \rrbracket \) such that \( \llbracket \phi \rrbracket = \mathbb{F} \).

b. \( \land \)

**My answer**  \( \land \) is set intersection: the meaning is as follows:

\[
\llbracket \phi \land \psi \rrbracket = \llbracket \phi \rrbracket \cap \llbracket \psi \rrbracket
\]

c. \( \lor \)

**My answer**  \( \lor \) is set union: the meaning is as follows:

\[
\llbracket \phi \lor \psi \rrbracket = \llbracket \phi \rrbracket \cup \llbracket \psi \rrbracket
\]

d. \( \to \)

**My answer**

\[
\llbracket \phi \to \psi \rrbracket = \{ \llbracket \cdot \rrbracket \mid \llbracket \phi \rrbracket = \mathbb{F} \text{ or } \llbracket \psi \rrbracket = \mathbb{T} \}
\]

We could also define a stronger notion of entailment that delivers truth values:

\[
\llbracket \phi \leftrightarrow \psi \rrbracket = \mathbb{T} \iff \llbracket \phi \rrbracket \subseteq \llbracket \psi \rrbracket
\]

e. \( \leftrightarrow \)

**My answer**

\[
\llbracket \phi \leftrightarrow \psi \rrbracket = \{ \llbracket \cdot \rrbracket \mid \llbracket \phi \rrbracket = \llbracket \psi \rrbracket \}
\]

We could also define a stronger notion of equality that delivers truth values:

\[
\llbracket \phi \leftrightarrow \psi \rrbracket = \mathbb{T} \iff \llbracket \phi \rrbracket = \llbracket \psi \rrbracket
\]
Problems

iii. What has happened to truth in this reformulation?

My answer  Truth remains fundamental, in the sense that specific interpretation functions are included or excluded in the meanings based on whether they map the (sub)formulae in question to truth or falsity. But formulae themselves no longer denote truth values (unless we go with the strong versions \(\rightarrow\) and \(\leftrightarrow\)). They denote sets, objects to which we cannot assign the value true or the value false.

B.23 Alternative type definition

Background  The type definition of handout 5 is so common that one starts passing over the types without a glance. But they can have an interesting logic in their own right, often delimiting the space of meanings in important ways. So it is worth getting accustomed to thinking about their freedoms and limitations.

Your task  Consider the following type definition:

i. \(\circ\) and \(\bullet\) are types

ii. If \(\sigma\) is a type, then \(\langle \top, \sigma \rangle\) is a type.

iii. Nothing else is a type.

For each of the following, say whether it is in the above type space:

i. \(\langle \top, \circ \rangle\)

My answer  Yes, this is a type. Its output is a type by clause (i), and clause (ii) thus defines it as well-formed.

ii. \(\langle \circ, \top \rangle\)

My answer  No, this is not a type. The definition ensures that \(\top\) is never an output. It is introduced only in clause (ii), and there it is only an input.

iii. \(\langle \circ, \bullet \rangle\)

My answer  This is not a type. Though the input and the output are both types, we have no clause for putting them together in this way.

iv. \(\langle \top, \langle \top, \circ \rangle \rangle\)
Problems

My answer  Yes, this is a type. Clause (ii) says that \( \langle \dagger, \tau \rangle \) is a type if \( \tau \) is. Our \( \tau \) is \( \langle \dagger, \langle \dagger, \circ \rangle \rangle \). Clause (ii) instructs us to break this down yet again, into \( \langle \dagger, \rho \rangle \), which is a type if \( \rho \) is. Here, \( \rho = \circ \), which is a type by clause (i).

B.24 Possible types given assumptions

Background  Semantic types have a very strict logic, often forcing decisions upon us, for good or for ill. It is important to see this — important for problem solving, and important for thinking about the models (which are arguably too big and complicated to reason about directly).

Your task  What are the two possible types for \( \alpha \) if the mode of composition is functional application? What is the model-theoretic reason for this limitation?

\[
\gamma : \langle e, t \rangle \\
\alpha : \beta : \langle e, \langle e, t \rangle \rangle \\
\]

My answer  \( \alpha : \langle \langle e, \langle e, t \rangle \rangle, \langle e, t \rangle \rangle \) or \( \alpha : e \). The more general idea: \( \alpha \) can be the functor, or else it can be the argument. (Those are the only two possibilities on the assumption that we can use only functional application to put meanings together.)

B.25 Vacuous abstraction

Background  The lambda calculus allow vacuous abstraction. Should we allow our linguistic theory to inherit this freedom?

Your task  Try to articulate what aspects of the system allow vacuous abstraction, and try also to find evidence for or against allowing it in our linguistic theory as well. The following examples might be useful in this regard.

(V1) People would call all the time and ask for Ali and we would say,”it’s not our company,” she said. Usually, the calls were complaints, Ms. Tams said, adding: “It’s been one of those things where we were going to go to him and talk to him about having him change his fictitious name, but it’s something we never got around to doing. And I wish we did.”

Problems

(V2) “Johnny, believe me. We may be dealing with something neither one of us should get near, something way up in the clouds that we — I — don’t have the knowledge to make a proper decision.”

My answer The definition of lambda abstraction (see 5.2.2 of handout 5) allows us to take any formula $\alpha$ and abstract over it. There is no check to make sure that the variable next to the lambda occurs at all after the dot (i.e., in the body of the lambda abstraction).

Is this a good thing, or should we, as linguists, filter off such vacuous abstractions? The answer to this is complicated, to be sure. The examples above seem at first to indicate that vacuous quantification of a sort is possible, but one could easily argue that this is an incorrect analysis of the examples. After all, they seem to involve some kind of abstraction over situations or events.

B.26 Partiality

Background It is very common to find that an author is implicitly or explicitly depending on some functions in the functional domains being partial, rather than total (handout 3, section 3.4). Such functions can provide an elegant formal basis for a theory of presuppositions (Beaver 1997). But it has logical consequences that we should be aware of (Muskens 1989, 1995).

Your task Suppose that the denotes a partial function from properties into entities (type $\langle \langle e, t \rangle, e \rangle$, one that is defined for $f$ iff $f$ is true of just one entity in $D_e$ (uniqueness). What are the consequences of this for expressions like the(dog), where we assume that dog is of type $\langle e, t \rangle$? What consequences does this have for our link between the types of expressions and their domains (see exercise A.28)?

My answer If the is of type $\langle \langle e, t \rangle, e \rangle$ but $||\text{the}||^M$ is defined for only some $\langle e, t \rangle$ functions, then some expressions that are well-formed syntactically will fail to have meanings, and we will, in a sense, lose the very tight connection between expressions, types, and denotations that is currently reflected in principles like (4.2) and (5.1).

---

### B.27 Novel types and meanings

**Background** This exercise involves both ingredients required to build a new logical constant: a type and a meaning. (It is important to keep sight of the fact that a type is not a meaning. A type only delimits the space of potential meanings for the expression.)

**Your task** Fill out the following semantic analysis with types and logical expressions, and then provide a denotation for bet that makes it a reasonable hypothesis for the translation of English bet.

*Chris bet Ali $500 that the earth is flat.*

**My answer** I’ve boxed the expressions and types that form this part of the answer.

\[
\begin{align*}
\text{bet}(\text{ali})(500)(\text{flat}(\text{the}(\text{earth})))(\text{chris}) : t \\
\text{chris} : e \\
\text{bet}(\text{ali})(500)(\text{flat}(\text{the}(\text{earth}))) : \langle e, t \rangle \\
\text{bet}(\text{ali})(500) : \langle t, \langle e, t \rangle \rangle \\
\text{flat}(\text{the}(\text{earth})) : t \\
\text{bet} : \langle e, \langle t, \langle e, t \rangle \rangle \rangle \\
\text{ali} : e \\
\end{align*}
\]

And here’s a basic meaning for bet:

- \(|\text{bet}|^M = \text{the function } f \text{ such that } f(\odot) = \text{the function } g \text{ such that } g(\$) = \text{the function } h \text{ such that } h(\pi) = \text{the function } k \text{ such that } k(\odot) = T \text{ iff } \odot \text{ bet } \odot \$ \text{ that } \pi.\]

### B.28 Types, expressions, and domains

**Background** In the logic we work with, the types appear in the definition of well-formed expressions and in the definitions for the hierarchy of semantic domains. There is, therefore, a sense in which they organize both the expressions and the models. Let’s consider again definition (4.2) from section 4.3.2.2 of handout 4:

\[\text{(4.2) A } \text{PL}_T \text{ expression } \varphi \text{ is of type } \sigma \text{ iff } |\varphi|^M \in D_\sigma.\]
Your task  Try to articulate the nature of the connection established by (4.2). If we write down \((\varphi(\psi))\) where \(\varphi\) is of type \(\langle \sigma, \tau \rangle\) and \(\psi\) is of type \(\rho\), where \(\rho \neq \sigma\), what happens when we try to interpret that formula? In what sense is our type-theoretic problem also a semantic (model-theoretic) problem?

My answer  One beautiful aspect of these systems is that ill-formed expressions are ill-formed because something is wrong with the meanings involved, and the reverse. This is what (4.2) is all about.

More specifically, if \((\varphi(\psi))\) is ill-formed because the type of \(\psi\) is not the input type of \(\varphi\), then attempts to interpret these formulae will be attempts to feed a function something that is not in its domain. The move is akin to trying to cram Euros into a vending machine that accepts only U.S. currency as input. The bottom line is that ill-formedness in this system is no mere stipulative quirk. Rather, it reflects something fundamental about the meanings.

B.29  Recursive interpretation

Background  In section 5.4.2, we looked briefly at some pseudo computer code for recursively interpreting lambdas terms. The code is meant to bring out the idea that interpretation is a recursion, with very complex things flowing from a few simple operations.

Your task  Provide the missing clause for interpreting lambda abstracts, i.e., the case in which \(\varphi\) is of the form \((\lambda x. \psi)\). Please feel free also to write an actual program for parsing and interpreting lambda terms!

My answer  Below is the full pseudo-code, with the clause for lambda expressions filled in at the end. To keep the relevant functions clearly in view, I defined an auxiliary routine BUILDFUNCTION which does the work of building up the lambda term.

\[
\text{INTERPRET}(\varphi, M, g) \\
1\quad \text{if } \varphi \text{ is a constant} \\
2\quad \quad \text{then return } \|\varphi\|^M \\
3\quad \text{elseif } \varphi \text{ is a variable} \\
4\quad \quad \text{then return } g(\varphi) \\
5\quad \text{elseif } \varphi \text{ is of the form } (\alpha(\beta)) \\
6\quad \quad \text{then return } \text{INTERPRET}(\alpha, M, g)(\text{INTERPRET}(\beta, M, g)) \\
7\quad \text{elseif } \varphi \text{ is of the form } (\lambda x. \alpha) \\
8\quad \quad \text{then return } \text{BUILDFUNCTION}(x, \alpha, M, g)
\]
\begin{verbatim}
    BuildFunction(\chi, \alpha, M, g)
    1   newFunction f
    2   for d \in D_{Type(\chi)}
    3       do f(d) = Interpret(\alpha, M, g[\chi \mapsto d])
    4   return f
\end{verbatim}

A slightly more detailed presentation would define a third routine for assignment updating — that is, it would unpack \(g[\chi \mapsto d]\) into a piece of computation. This would simply say “reset \(g(\chi)\) so that it is \(d\) and then return this modified \(g\)”. We would invoke this for every \(d\) in the domain, at line 3 of BuildFunction.

### B.30 An alternative mode of composition

**Background** One sometimes encounters a rule of *predicate modification* (Heim and Kratzer 1998). Here is a general formulation of the rule and its semantics:

\begin{align}
    & (B.10) \quad \\
    & \text{a. If } \alpha : \langle \sigma, t \rangle \text{ and } \beta : \langle \sigma, t \rangle, \text{ then } \alpha \sqcap \beta : \langle \sigma, t \rangle \\
    & \text{b. } \llbracket \alpha \sqcap \beta \rrbracket^M = \text{the function that maps any } \odot \in D_\sigma \text{ to } T \text{ iff } \llbracket \alpha \rrbracket^M(\odot) = \llbracket \beta \rrbracket^M(\odot) = T
\end{align}

**Your task**

i. In what sense, if any, does predicate modification expand what we can do with the logic? (Could we define this rule using just functional application?)

**My answer** In a sense, this does not expand the power of the logic. We can duplicate the above results with just functional application, if we introduce a new (class of) functions of this form:

\((\lambda f. \lambda g. \lambda x. f(x) \land g(x))\)

where \(f\) and \(g\) are functions of any type \(\langle \sigma, t \rangle\) and \(x\) is of type \(\sigma\).

ii. What predictions does this rule make about the dependencies between \(\alpha\) and \(\beta\)? Does it allow that the interpretation of one might be conditioned by the interpretation of the other. (This aspect of the problem is ‘HARD’.)
Problems

**My answer** The question requires a bit of background in the way that adjectival modification works. For so-called *intersective modifiers* like *Swedish* and *linguist*, we can get the meanings simply by intersection: *Swedish linguist* picks out the intersection of the set of Swedish things with the set of linguists, i.e., the objects of which both properties are true. Since Predicate Modification is just an intersective operator (as is evident especially from its lambda formulation just above), it is appropriate for intersective modification. Importantly, intersective modification is completely symmetric, because intersection itself is symmetric.

But not all modifiers are symmetric. Consider *fake*. To get the meaning of *fake computer*, we don’t take the intersection of the set of computers with the set of fake things. In fact, this is sort of nonsense: there is no set of fake things. A fake computer might be a real toy, for example. For these cases, the modifiers are not in a symmetric relationship. Clearly, *fake* is the mover and shaker here. For such instances, Predicate Modification is inappropriate.

iii. The rule is stated so as to allow any type ending in \(t\). How might we characterize this class of domains?

**My answer** This is the domain of truth functions, very broadly construed. This is not an accident. Since we are feeding everything to a \(\land\), we need to get everything into the \(t\) domain.

iv. Could we generalize this rule to any type \(\langle\sigma, \tau\rangle\)?

**My answer** We can generalize the rule almost completely. As long none of the output types contains a function into \(e\), we are all set. Here is the recursive definition:

a. If \(\alpha\) and \(\beta\) are of type \(t\), then \((\alpha \& \beta) = (\alpha \land \beta)\).

b. If \(\alpha\) and \(\beta\) are of type \(\langle\sigma, \tau\rangle\), then \((\alpha \& \beta) = \lambda\chi. (\alpha(\chi) \& \beta(\chi))\), where \(\chi : \sigma\).

An example involving the \(\langle\langle e, t\rangle, \langle e, t\rangle\rangle\) expressions *sadly* and *slowly*:

\[
(sadly \& slowly) = \lambda f. (sadly(f) \& slowly(f)) \\
= \lambda f. \lambda x. (sadly(f)(x) \& slowly(f)(x)) \\
= \lambda f. \lambda x. (sadly(f)(x) \land slowly(f)(x))
\]
B.31 Assignments

**Background**  The act of changing an assignment according to an instruction can seem complicated, but in fact it is quite minimal and easy to visualize with a little practice.

**Your task**  Fill out these equality expressions (I’ve not relativized to a model to keep things simple):

**My answer**  My answers are filled in.

1. \[
\begin{align*}
\[x\] & \quad y \mapsto \quad = \\
\end{align*}
\]

2. \[
\begin{align*}
\[y\] & \quad y \mapsto \quad = \\
\end{align*}
\]

3. \[
\begin{align*}
\[x\] & \quad y \mapsto \quad = \\
\end{align*}
\]

4. \[
\begin{align*}
\[\text{happy}(y)\] & \quad y \mapsto \quad = \[\text{happy}\]( ) \\
\end{align*}
\]

5. \[
\begin{align*}
\[\lambda y. \text{happy}(y)\] & \quad y \mapsto \quad = \[\text{happy}\]
\end{align*}
\]
B.32 Variable names

**Background**  The axioms of the lambda calculus (handout 6) have some important consequences for what we can and cannot do with variables by way of making meaning distinctions. This exercise is a glimpse at that realm (see also handout 10, section 10.2).

**Your task**  For each pair, say whether its members can differ model-theoretically. If they can, exemplify the difference. If they can’t, try to articulate why they can’t.

- **X**  
  a. `happy(x)`  
  b. `happy(y)`

  **My answer**  These two can differ model-theoretically. This is in virtue of the fact that a given assignment $g$ might map $x$ to a happy entity and $y$ to an unhappy entity.

- **Y**  
  a. $\lambda x. \text{happy}(x)$  
  b. $\lambda y. \text{happy}(y)$

  **My answer**  These two are alphabetic variants, and, moreover, each reduces to `happy` by $\eta$-conversion (handout 6). So they cannot differ model-theoretically.

B.33 Cross-categorial and

**Background**  With the lambda calculus, we can improve on a major shortcoming of the hypothesis that `and` translates as $\land$. Recall that, in the context of PL$_f$, this meant that we could have only sentence-level coordination. In a sense, it is easy to show that this prediction is false. Any major category can be conjoined. It is much closer to the truth to say that `and` can conjoin two things as long as they have the same type. This generalization is now within reach in an extensional lambda calculus.

**Your task**  Define a cross-categorial `and` that is build up from our `\land` from PL$_f$ but can take any pair of arguments in $\langle \sigma, t \rangle$.

**My answer**  Here’s the cross-categorial `and` we’re after:

- $\text{and} \rightarrow \lambda X \lambda Y \lambda \chi. (Y(\chi) (\land X(\chi)))$

  where $X$ and $Y$ are of type $\langle \sigma, t \rangle$ and $\chi$ is of type $\sigma$.
See also the answer to problem A.30 above, which provides an even more general conjunction operator.

**B.34 A relational reinterpretation**

**Background** Muskens (1989, 1995) argues at length for a relational perspective on denotations, rather than a functional one. One of his arguments appeals to the desire to construct theories that are easy to grasp. He writes:

This is all very well, until we realise that we have coded binary relations between ternary relations as functions from functions from individuals to functions from individuals to functions from individuals to truth values to functions from functions from individuals to functions from individuals to truth values. In other words, we have replaced objects that we have some intuitive grasp on by monsters that we can reason about only in an abstract way. (Muskens 1995:12)

It's a point well taken. One might object that these monsters are necessary if we want a theory that assigns a meaning to each syntactic phrase. But Muskens answers that objection. See if you can do so as well.

**Your task** Formulate an operation on relational meanings that essentially abstracts over one of the coordinates in the tuples it contains, so that we can maintain our usual theory of composition with these (arguably) simpler relational objects.

**My answer** A relational meaning, in the present sense, is a set of \( n \)-tuples. We can get at the individual coordinates of the members of these sets via the following operation, which is from Muskens (1995:14):

- Let \( R \) be an \( n \)-ary relation \((n > 0)\) and let \( 0 < k \leq n \). Define the \( k \)-th slice function of \( R \) by:

\[
F^k_R(d) = \{ \langle d_1, \ldots, d_{k-1}, d_{k+1}, \ldots, d_n \rangle \mid \langle d_1, \ldots, d_{k-1}, d, d_{k+1}, \ldots, d_n \rangle \in R \}
\]

Thus, \( F^1_{\text{love}^M}(d) \) is the set of all 1-tuples (which are just entities) \( d' \) such that \( \langle d, d' \rangle \in \llbracket \text{love} \rrbracket^M \), and \( F^2_{\text{love}^M}(d') \) is the set of all 1-tuples \( d \) such that \( \langle d, d' \rangle \in \llbracket \text{love} \rrbracket^M \).
B.35 What’s the source of the ill-formedness?

**Background** A linguistic theory might offer many different potential ways of explaining the deviance of some example. It can be very difficult to settle on one of the options, and it can be even more difficult to figure out how, or whether, to remove redundancies in one’s account. This exercise gives you a glimpse of such challenges.

**Your task** The task is to provide explanations for the deviance of each of the examples in (a)–(e).

Two things to keep in mind:

- We are not (necessarily) after a unified theory of the deviance.
- If you’re unsure of how to analyze a constituent semantically and it isn’t important to your argument how it is analyzed, then translate it into a single predicate. For example, *The A-train ↞ the-train*.

(Da) *Ed devoured. (but what about *Ed ate?*)

**My answer** If *devour* denotes a two-place function, then the example has a well-defined meaning — unfortunately, that meaning is the set of things that devour Ed!

- *devour*(ed) : ⟨e, t⟩
  
  ed : e
  
  devour : ⟨e, ⟨e, t⟩⟩

- **⟦devour(ed)⟧**
  
  M = the function in D⟨e,t⟩ that takes an ⊗ ∈ De to T iff ⊗ devour Ed.

We can move past this problem by imposing the following general condition:

- A parsetree is well-formed only if its root has a type t term on it.

The tree for *Ed devoured* doesn’t satisfy this condition. This leaves only the well-formedness of *Ed ate*. Since this is intuitively truth-valued, we need to fill in its object position (first semantic argument):

- **Existential closure**
  
  i. ∃ : ⟨⟨e, t⟩, t⟩
  
  ii. **⟦∃⟧** = the function that takes any *f* ∈ D⟨e,t⟩ to T iff there is a ⊗ ∈ De such that *f*(⊗) = T
This kind of existential closure should be lexically restricted. But it is hard to see how that could be done compositionally.

(Db) * I saw Sue and that it was raining.

**My answer** If coordination demands two things with identical types, then this makes sense on the motivated assumption that *Sue and that it was raining* translate as expressions with different types. Thus, there is an easy semantic explanation available.

(Dc) * Ed glimpsed the dog the printer.

**My answer** This makes sense semantically, because there is no way to incorporate all the pieces into the meaning, due to the fact that the verb denotes a two-place (not a three-place) relation:

\[
\begin{array}{c}
glimpse(\text{the(dog)})(\text{ed}) : t \\
\text{the(printer)} : e \\
ed : e \\
glimpse(\text{the(dog)}) : \langle e, t \rangle \\
glimpse : \langle e, \langle e, t \rangle \rangle \\
\text{the} : \langle \langle e, t \rangle, e \rangle \\
\text{dog} : \langle e, t \rangle
\end{array}
\]

Two things to keep in mind. First, it is very likely that there are good syntactic reasons for this ungrammaticality. And, second, the semantic explanation is complicated by the availability of lambda abstraction, which could apply to \(\text{glimpse(}\text{the(dog)})(\text{ed})\) to turn it into an \(\langle e, t \rangle\) expression. We could then feed in \(\text{the(printer)}\). Since the abstraction would be vacuous, \(\text{the(printer)}\) would disappear, and we would predict that the sentence simply meant that Ed glimpsed the dog.

(Dd) # It’s not raining, but Sue realizes it’s raining.

**My answer** The verb realize is a presupposition trigger: it is defined for a propositional argument \(p\) only if \(p\) is true.

- **realize** : \(\langle t, \langle e, t \rangle \rangle\)

- \(\|\text{realize}\|^{M} =\) is defined for a truth value \(\$\) only if \(\$ = T\)
  
  where defined, \(\|\text{realize}\|^{M}\) is the the partial function \(f\) such that \(f(\$) = \text{the function } g\) such that \(g(\ominus) = T\) iff \(\ominus\) is aware of \(\$\)
This analysis says that there is nothing necessarily wrong with the composition of *Sue realizes it’s raining*. The markedness arises only because the discourse places contradictory demands on the semantic value of *raining*.

This analysis is something of a fiction — we have to pretend that it makes sense to say that someone is aware of a truth value. But this weirdness can be fixed by moving to an intensional semantics.

(De)  
#The A-train suffered an existential crisis.

(cf. *I dreamed that the A-train suffered an existential crisis.*)

**My answer**  
We might be initially tempted to treat this as a case of semantic-type mismatch. We could say that the verb phrase *suffer an existential crisis* denotes a function that is defined only for thinking entities. But the *dream* example suggests that this would be too rigid. To understand it, we would have to say that the *A-train* can shift domains, moving from the thinking to the non-thinking realm, presumably only in certain contexts.

I suggest that we analyze this as an example that is nearly guaranteed to be false, in virtue of the fact that, in our world (in our epistemic space), only cognitive agents can have existential crises. When we move to non-actual words (out of our epistemic space) and into, say, dream worlds, we find that the markedness disappears. So (e) is semantically impeccable but pragmatically anomalous — one shouldn’t go around saying things one knows to be false.

### B.36 Building a fragment

**Background**  
In linguistic semantics, a fragment is a complete theory of a (very) small chunk of a natural language.

**Your task**  
Your goal for this part is to construct a fragment that handles all the intransitive verb constructions in (F). (Ignore all issues relating to tense.)

(F)  
a. Bart burps.
b. Maggie giggles.
c. Lisa muses.

Your fragment should have the following parts:

i. A specification of the class of well-formed expressions of your logical language.
ii. A type theory for organizing the logical expressions.

iii. A specification of the domains for each of the types.

iv. An interpretation function that takes the logical expressions to objects in the domains for the types (in a way that respects typing).

v. A translation procedure for mapping English phrases to expressions of the logical language.

To show readers how your fragment works, you should provide a derivation of some kind for one of the sentences in (F).

Strive for generality. If your fragment works for the examples in (F), it will also work for lots of other intransitive sentences. Either sketch how your fragment could be generalized to new intransitive sentences with proper-name subjects or (better) define your fragment so that it has this level of generality built into it.

My answer  My answer is a complete theory of intransitive (one-place predications), with an English-like syntax:

i. Types: $e$, $t$, and $\langle e, t \rangle$ are types.

ii. Domains
   a. $D_e$ is a set of entities.
   b. $D_t = \{T, F\}$
   c. $D_{\langle e, t \rangle}$ is the set of all functions from $D_e$ into $D_t$

iii. Well-formed expressions
   a. bart, maggie, lisa, . . . are well-formed expressions of type $e$.
   b. muse, burp, giggle, . . . are well-formed expressions of type $\langle e, t \rangle$.
   c. If $\alpha$ is a well-formed expression of type $e$ and $\beta$ is well-formed expression of type $\langle e, t \rangle$, then $(\alpha \beta)$ is a well-formed expression of type $t$.
   d. Nothing else is a well-formed expression.

iv. Interpretation
   a. $\|\alpha\|^M \in D_e$ if $\alpha$ is a constant of type $e$.
   b. $\|\alpha\|^M \in D_{\langle e, t \rangle}$ if $\alpha$ is a constant of type $\langle e, t \rangle$.
   c. $\|\alpha \beta\|^M = ||\beta||^M(\|\alpha\|^M)$

v. Translation: Each constant of the language is the interpretation of the corresponding English word (in the expected way).
vi. Example

\[
\text{bart burps} : t
\]
\[
\text{bart} : e \quad \text{burps} : \langle e, t \rangle
\]

\[
[[\text{bart burps}]]^M = ||\text{burps}||^M(||\text{bart}||^M)
\]

B.37 Substitution

Background Substitution is an apparently simple operation on formulae that is nonetheless complicated by the conditions geared towards ensuring that no accidental binding takes place. It’s worth practicing a bit.

Your task For each of the following perform the substitution operation if it is permitted, else indicate what blocks the substitution.

i. \(y[y\rangle(\text{cyclist}(x))\]

My answer Permitted. The output is \((\text{cyclist}(x))\).

ii. \((\text{cyclist}(x))[x\rangle y\]

My answer Permitted. The output is \((\text{cyclist}(y))\).

iii. \((\lambda x. (\lambda y. ((f(y))(y))))[y\rangle x\]

My answer Not permitted. It would result in \(\lambda x\) binding two variable instead of none.

B.38 Beta reductions

Your task Beta reduction (lambda conversion) is the workhorse of the proof system when it comes to linguistic analysis. We often build up big lambda terms and need to reduce them to fully understand what meanings they pick out. This exercise gives you some practice with such reductions.

Background Reduce each of the following expressions as far as possible, indicating each step.

i. \((\lambda x. \text{like}(x))(y)\)
My answer

\[(\lambda x. \text{like}(x))(y) \Rightarrow \text{like}(y)\]

ii. \((\lambda x. \text{like}(y))(x)\)

My answer

\[(\lambda x. \text{like}(y))(x) \Rightarrow \text{like}(y)\]

iii. \((\lambda x. \text{run})(x)\)

My answer

\[(\lambda x. \text{run})(x) \Rightarrow \text{run}\]

iv. \((\lambda f. f(\text{ali}))(\lambda y. \text{fast}(y))\)

My answer

\[(\lambda f. f(\text{ali}))(\lambda y. \text{fast}(y)) \Rightarrow \]

\[(\lambda y. \text{fast}(y))(\text{ali}) \Rightarrow \]

\[\text{fast} (\text{ali})\]

v. \((\lambda P. \lambda Q. Q(\lambda x. P(\lambda y. \text{admire}(y)(x)))(\lambda f. f(\text{ali}))(\text{chris}))\)

My answer

\[
\left(\lambda P. \lambda x. P(\lambda y. \text{admire}(y)(x))\right)\left(\lambda f. f(\text{ali})\right)\left(\text{chris}\right) \Rightarrow \\
\left(\lambda x. \left(\lambda f. f(\text{ali})\right)(\lambda y. \text{admire}(y)(x))\right)\left(\text{chris}\right) \Rightarrow \\
\left(\lambda f. f(\text{ali})\right)(\lambda y. \text{admire}(y)(\text{chris})) \Rightarrow \\
\left(\lambda y. \text{admire}(y)(\text{chris})\right)(\text{ali}) \Rightarrow \\
\text{admire} (\text{ali})(\text{chris})
\]

vi. \((\lambda P. \lambda Q. Q(\lambda x. P(\lambda y. \text{admire}(y)(x)))(\lambda f. f(\text{ali}))(\lambda g. g(\text{chris})))\)
My answer

\[
(\lambda P. \lambda Q. Q(\lambda x. P(\lambda y. \text{admire}(y)(x))))(\lambda f. \text{ali})(\lambda g. \text{chris}) \Rightarrow \\
(\lambda Q. (\lambda x. (\lambda f. \text{ali})(\lambda y. \text{admire}(y)(x))))(\lambda g. \text{chris}) \Rightarrow \\
(\lambda g. \text{chris})(\lambda x. ((\lambda y. \text{admire}(y)(x))(\lambda f. \text{ali}))) \Rightarrow \\
(\lambda g. \text{chris})(\lambda x. (\lambda y. \text{admire}(y)(x))(\lambda f. \text{ali})) \Rightarrow \\
(\lambda g. \text{chris})(\lambda x. \text{admire}(\text{ali})(x)) \Rightarrow \\
(\lambda x. \text{admire}(\text{ali})(x))(\lambda g. \text{chris}) \Rightarrow \\
\text{admire}(\text{ali})(\lambda g. \text{chris})
\]

B.39  Eta conversion and distinguishable meanings

Background  Lambdas are so closely associated with meaning analysis that it can be hard for people to see that they are not constitutive of a proposal about a particular denotation.

Your task  Try to articulate the model-theoretic grounding for \(\eta\)-conversion. Why is it guaranteed to work, given the formulation of functional application and functional abstraction. (It might help to think about what happens when you abstract over \(x\) in an expression like \text{happy}(x).)

My answer  Eta-conversion is valid in such simple cases because of the way application and abstraction are paired. If I apply \(x\) to a function, it takes me to some value for that function. If I then abstract over \(x\), I simply build up the original function again.

B.40  Extensional beliefs?

Background  Section 7.1 of handout 7 begins building the case that we cannot have an extensional theory of predicates like \text{believe}.

Your task  See if you can make matters worse for “extensional \text{believe}”, by perhaps deriving meanings that run directly counter to our intuitions. Can you make Lisa both believe and disbelieve every truth?
My answer For this answer, it is useful to refashion our functional meanings as set of ordered pairs. Thus, whereas our extensional believe officially denote a function from entities into a function from truth values into truth values, we will think of it as a set of ordered pairs consisting of an entity and a truth value.

Let’s suppose that Lisa believes Bart burps, and suppose also that it is true that Bart burps. Then

$$\langle A_0, T \rangle \in \|\text{believe}\|^M$$

Now suppose that Lisa believes that Burns does not like to read, but suppose that he does. Then:

$$\langle A_0, T \rangle \notin \|\text{believe}\|^M$$

This is our only way of expressing Lisa’s lack of belief. But it contradicts our first assumption. We’re forced to the conclusion that Lisa’s belief that Bart burps entails that she also believes that Burns likes to read.

B.41 Modals

Background Modal verbs seem to say something about propositional content. But, within an extensional model, our ‘propositions’ are just truth values.

Your task Why can’t we just use one of the functions from $D_t$ into $D_t$ to analyze modals?

My answer The bottom line is that there are just four functions from $D_t$ into $D_t$, and all of them fail to come even close to capturing what modals mean:

i. The identity function maps truth values to themselves. This would render the modal vacuous. But speakers assign different truth conditions to modal statements and their unmodalized counterparts.

ii. The universal function maps both truth values to $T$. This would make every modal sentence true, wrongly predicting that objections to them are incoherent.

iii. The contradictory function maps both truth values to $F$. This would make every modal sentence false, wrongly predicting that speakers are always wrong to say them.

iv. The negation function flips truth values. While this makes some sense for not (see exercise B.16 above), it is plainly wrong for modals.
B.42 Hintikka’s *believe*

**Background** The proposed meaning of *believe* in handout 7 is essentially Hintikka’s (1969). Hintikka proposed to associate with every individual *a* a subset of the set of possible worlds representing *a*’s belief state. This function has since come to be called Dox, for ‘doxastic’. Extensionally, Dox maps entities to sets of possible worlds.

**Your task** Formulate Dox precisely, but make sure that you take into account, somehow, that an individual’s beliefs can vary from world to world. Rework the definition of \([\text{believe}]^{M,g}\) using this new Dox, and explain your decisions about how to handle the world arguments throughout.

**My answer** In extensional terms, Dox is a function that maps entities into propositions. It maps an entity *d* into the proposition that has as its entailments all and only the things *d* believes. This is not enough, though. Our beliefs can vary from world to world. We see this in phrases like, *If I believed that . . .*, which take us to potentially counterfactual worlds in which my beliefs are (or at least can be) different than they are in our reality. Thus, we need to rethink Dox. It should be a function that maps an entity *d* into a function that maps a world *w* to the proposition that has as its entailments all and only the things *d* believes in *w*. That is, \(\text{Dox}(d)(w)\) is the belief state of *d* in *w*.

Here is a definition of the semantics of \([\text{believe}]^{M,g}\) in these terms:

- \([\text{believe}]^{M,g}\) = the function \(\Phi\) such that \(\Phi(\pi) = \text{the function } \Psi\) such that \(\Psi(\odot) = \text{the function } \pi'\) such that \(\pi'(\odot) = \top\) iff, for all \(\odot'\), if \(\text{Dox}(\odot)(\odot)(\odot') = \top\), then \(\pi(\odot') = \top\)

B.43 Individual concepts

**Background** The linguistic hypotheses of handout 7 (section 7.3) leave proper names out of the intensional sphere, by assigning them to a single domain \(D_e\) that is independent of specific possible worlds. But many have argued that the domain of entities is relativized to specific possible worlds.

**Your task** Where do you come down on this issue? Why? What impact, if any, does your position have on the treatment of proper names? (You might widen your scope enough to include fictional names and the like — it depends on how daring your are.)
Problems

My answer  In my view, there is something right about the idea that proper names are rigid (in the sense of Saul Kripke’s work on reference), and thus that we should treat that as type e expressions even when we are working with intensional models.

There is a further question of whether different worlds have different domains associated with them. Could you exist in one world but not in another?

And there is a further question still: does it make sense to say that one and the same entity exists in more than one possible world. Modal realists — researchers who believe, as Lewis did, that possible worlds are more than theoretical constructs — generally say that it is not possible for a single entity to exist in more than one world, and they generally invoke counterparts for situations in which we seem to say talk about an entity of one world existing in another.

B.44 How many worlds are there?

Background  It is worth trying to figure out how big our intensional models are. The answer could impact our view of how semantics and cognitive science link up, and it might push us towards a proof-theoretic semantics, rather than a model-theoretic one.

Your task  Suppose there are $n$ individuals. How many different properties can there be? How many worlds should we have to ensure that we can make all the distinctions among meaning that we want to be able to make?

My answer  If there are $n$ individuals, then there are $2^n$ different properties we can form in a given extensional model. If we perceive that there are more semantically-distinct properties than that, then we are in trouble — without expanding the domain of entities, this space of available properties will not get bigger.

Here’s a slightly different perspective. Suppose we want to define $n$ intensionally distinct properties. That is, $n$ distinct functions from entities into functions from worlds into truth values. We know from exercise A.9 that there are $2^{||W||^{|P|}}$ functions of this sort, and hence we want to ensure that

$$2^{||W||^{|P|}} \geq n$$

B.45 Finding common ground

Background  Handout 7, section 7.4.4, suggests that we can get back our old notion of truth for sentences by having a designated constant for the real world, @. The idea is that
when speakers assert something, they assert that it holds in @.

But the move is somewhat puzzling. We don’t know which is the actual world. Knowing that would involve knowing absolutely everything about our reality. However, as soon as we admit that we don’t know, for example, who is standing at the northmost corner of Prospect Park right now, we admit that we don’t quite know which world we occupy.

It is more accurate to say that we have a set of worlds that we consider contenders for the actual world. Let’s call that set the common ground.

Your task Define a notion of truth relative to a common ground, so that we can still have truth/assertion after giving up on @.

My answer If our sentence meanings denote propositions, then we can define truth relative to a common ground C in terms of entailment:

- \( p \) is true in C iff for all \( w \), if \( C(w) = T \), then \( p(w) = T \).

Here, C is of course a proposition — the proposition that has as its entailments all and only the things we consider to be in our common ground (our mutual public beliefs).

B.46 Definites and semantic composition

Background Section 7.5.3 of handout 7 briefly motivates the idea that definite descriptions are of type \( \langle s, e \rangle \). But this leaves us with a problem: it is not clear how they combine with our predicates’ meanings.

Your task Propose a solution to this problem. You might change our assumptions about predicates. You might sneak in a free variable over worlds. You might define a new rule of semantic composition. Feel free to think freely, but do try to motivate the choice you make.

My answer Let’s suppose first that we keep our properties in type \( \langle e, \langle s, t \rangle \rangle \) and our transitive verbs in type \( \langle e, \langle e, \langle s, t \rangle \rangle \rangle \). Then we predict entity level arguments to combine with neither fuss nor muss. But if definites are of type \( \langle s, e \rangle \), then they cannot combine directly with these functions. We have to feed definites a world argument so that we get them down to the e type. Thus, we won’t deal semantically with the(cyclist) in argument position. Rather, we will deal with the(cyclist)(w), where w is a free variable over worlds. Semantically, this will be the dog at the index \( \llbracket w \rrbracket^M,a \).
This free variable might get bound later, or it might be allowed to remain free. We will need carefully crafted scenarios that get at people’s judgments about intensionality to decide the matter.

Suppose the free variables in definites turn out to be problematic. We might then change our verb meanings. For instance, properties could be of type \(\langle\langle s, e\rangle, \langle s, t\rangle\rangle\). The most conservative sort of meanings would look like this:

- \(\lambda x \lambda w. \text{happy}(x(w))(w)\)

This would map an individual concept to the set of worlds \(\Box\) in which \([ [x]_M^\theta(\Box)\) was happy in \(\Box\). In contrast to the free variable approach, this allows us very little freedom in how we evaluate the individual concepts intensionally. Everything is left up to the verb, making it a very local affair (whereas solutions involving free variables can be highly nonlocal if we want them to be).

### B.47 Contradictory beliefs

**Practice**

**Background**  This question continues section 7.5.4’s critical examination of our theory of belief predications.

**Your task**  Suppose I believe something impossible. What does my belief state look like then? Is this realistic? If no, how might we do better?

**My answer**  If I believe something that is impossible, then my belief state is the empty set of propositions. In terms of exercise B.42, if I believe, in \(\Box\), an impossibility, then \(\text{Dox}(\llbracket \text{chris}_M^\theta\rrbracket^\Box) = \emptyset\).

Is this realistic? No. It predicts that believing an impossibility (even if I do not know that I do) will force me to believe every proposition. This will represent a complete collapse of my epistemic state.

There is relatively little work in semantics that really pushes past this problem. I regard it as an open question. But it is clear to me that we would make progress if we could let inconsistencies in speakers’ belief states go unnoticed until there was reason to access information that is relevant to them, at which point we would identify the chain of events that leads to the resolution of the problem. If I accidentally schedule two meetings for the same time in different locations, my belief state does not collapse. I might not even notice right away. If and when I do notice, I will set about making minimal modifications to correct the impossibility.
B.48 Degree constructions

Background  Handout 8 emphasizes that the systems defined here are by no means the end of the story. It is routine to find constructions that demand extensions of these basic systems. Degree constructions are an excellent case in point. Here are some basic cases.

(B.11) a. That mouse is tall.
b. That elephant is tall.
c. That elephant is ten-feet tall.
d. Sam is taller than Sue.

It is extremely hard to see how we would fit these into our current type system and associated domains. We seem to need new objects, ones that can help us get at notions like Sam’s degree of height.

Your task  Extend an extensional or intensional lambda calculus with a type for degrees and an associated domain, then use this enriched set of tools to give a meaning for tall. Strive for a meaning that will work in a broad range of cases. (It might seem easiest to start with examples like That mouse is tall, but in fact it is easier to start with comparative data.)

My answer  The accepted wisdom at this point is that degree constructions call for new primitives. There have been attempts to make due with the usual kinds of properties, but they seem not to have the generality of theories that are willing to identify degree constructions as a natural class.

Here are the basic ingredients:

i. A type $d$ for degrees, with the domain of $d$ some kind of scale (some segment of the real numbers on their usual ordering).

ii. Gradable adjectives denote (perhaps when combined with some functional material) functions from degrees into functions from entities to propositions. Thus, for example tall$(d)$(sam) might pick out the set of worlds in which Sam is tall to degree $d$.

iii. Comparative morphology operates on gradable adjectives. For instance, $\llbracket\text{-er}\rrbracket^M$ takes gradable adjectives and maps them into two place relations: taller denotes a function that takes two arguments and maps them to the set of worlds in which the second argument is tall to some degree that is taller than the maximal degree to which that argument is tall.
B.49   Singular and plural

**Background** It is easy to find predicates that care about whether their arguments are intuitively singular or plural (independently of issues concerning morphological agreement).

i. The crowd gathered in the town square.

ii. *Sam gathered in the town square.

**Your task** How should we account for this pattern? What changes to the logical system does your answer require?

**My answer** There is a very good case to be made for the idea that singular and plural are semantically distinct categories. This makes intuitive sense, and it is supported by the fact that, as we see in the above examples, some predicates select for plural arguments. This kind of treatment of plurals is often paired with the view that there are plural entities in the model, usually formed according to the axioms of the calculus of individuals (mereology).

One sensible move is to introduce a new type for plural entities, perhaps defining it is a subtype (sort) of the $e$ type, since many predicates do not care about the count value of their arguments (they are looking for type $e$ of any sort). Another a sensible move is to treat plurals as denoting sets of entities, equivalently, functions in $\langle e, t \rangle$, and in turn saying that predicates like gather translate as expressions of type $\langle \langle e, t \rangle, t \rangle$, rather than the singular property type $\langle e, t \rangle$.

B.50   Exactly 1 in first-order logic?

**Background** The usual first-order quantifiers are $\exists$ and $\forall$. While we can’t define all the natural language determiners in their terms, the pair of them can be remarkably expressive.

**Your task** Give a semantics for expressions like $\exists!x \varphi$ that makes them true iff there is exactly one entity with the property $\varphi$. And then describe how you would generalize this to exactly $n$, for any natural number $n$.

**My answer** *Exactly 1* in first-order logic:

$$\llbracket \exists!1 x \varphi \rrbracket^M_g = T \text{ iff there is exactly one } \odot \text{ such that } \llbracket \varphi \rrbracket^M_{g[\odot \leftarrow \odot]} = T$$

*Exactly 1* in lambdas:

$$\lambda f \lambda g. \exists x[f(x) \land g(x) \land [\forall y[(f(y) \land g(y)) \rightarrow x = y]]]$$
Similar definitions are available for any $n$. Here is a look at Exactly 2 in lambdas:

$$\lambda f. \lambda g. \exists x [f(x) \land g(x) \land \exists z [f(z) \land y(z) \land x \neq z \land \forall y [(f(y) \land g(y)) \rightarrow (x = y \lor z = y)]]]$$

B.51 A closer look at the universal

Background The universal quantifier has been around, in one form or another, for centuries, and it has always been tied to natural language words like every. But it has properties that its presumed natural language exponents do not obviously have, if they have them at all. We should be somewhat skeptical of the connection.

Your task

• What if the restriction to every is empty (maps all elements to $F$)?

  **My answer** In the case where the restriction to the quantificational determiner meaning every is empty, the result is clear: the overall expression is true no matter what VP comes in, in virtue of the fact that the empty set is a subset of every set and $\forall M$ is built up from the subset relation:

  $$\emptyset \subseteq A$$ is true for any set $A$

  What about the English determiner every? Here, the results are much less clear. There is certainly an expectation that we will not quantify over the empty set, as witnessed by the fact that it is very strange for me to say *All the giraffes in my apartment play bridge*. But are such sentences false? Or are they merely pragmatically anomalous in virtue of the fact that they are highly uninformative in virtue of being true no matter what the circumstances?

• What if the restriction and nuclear scope have the same extension?

  **My answer** Once again, the logic gives us a clear answer: if the restriction and the nuclear scope are the same, then the result is true. This is probably matched by our intuitions as well. However, speakers are likely to regard such sentences as marked in virtue of the fact that, like the empty restriction case, they are always true, because every set is an improper subset of itself:

  $$A \subseteq A$$ is true for any set $A$
B.52  All and only Lisa’s properties

**Background**  It can be a bit surprising at first that proper names can denote in the generalized quantifier domain. So let’s look a bit more closely at what we do when we view Lisa from the perspective of the set of all her properties.

**Your task**  Suppose that Lisa is young, intelligent, and literate. She is not angry, and she is not tall. Assume that there are no other properties besides these. Using these facts, draw a picture of the function specified in (A.12).

(B.12)  \( \lambda f. f(\text{lisa}) \)

**My answer**  Meet the generalized quantifier Lisa:

\[
\begin{align*}
\llbracket \text{young} \rrbracket^M & \mapsto T \\
\llbracket \text{intelligent} \rrbracket^M & \mapsto T \\
\llbracket \text{literate} \rrbracket^M & \mapsto T \\
\llbracket \text{angry} \rrbracket^M & \mapsto F \\
\llbracket \text{tall} \rrbracket^M & \mapsto F
\end{align*}
\]

B.53  Intensional quantifiers

**Background**  The quantifiers given throughout handout 9 are purely extensional. Thus, they won’t fit easily into the intensional setting of handout 7. We should fix this.

**Your task**  Provide a type for intensionalized quantificational determiners, and provide the meanings for *every* and *most* in these new terms. What did you decide to do with the world arguments? Why?

**My answer**  By and large, quantification is an extension affair. When I say to you, *Most elephants have trunks*, I clearly do not mean to quantify over elephants in worlds in which they are small animals that wear suits and do people’s tax returns. Thus, the following intensional types and meanings for *every* and *most* seek to locate the quantifications in the world of evaluation:

- i.  *every* : \( \langle\langle e, \langle s, t \rangle\rangle, \langle s, t \rangle\rangle \)
  
  ii.  *every* \( = \lambda f. \lambda g. \lambda w. \forall x f(x)(w) \rightarrow g(x)(w) \)

- i.  *most* : \( \langle\langle e, \langle s, t \rangle\rangle, \langle s, t \rangle\rangle \)
  
  ii.  *most* \( = \lambda f. \lambda g. \lambda w. \{ x \mid f(x)(w) \land g(x)(w) \} > \{ x \mid f(x)(w) \land \neg g(x)(w) \} \)
In each, the world variable is involved in every predication and is bound by the lambda abstraction that determines the output meaning as propositional. This ensures that, if we evaluate for truth by feeding in a world argument, everything will be about that world.

I should add that some quantifiers, particularly generics (Dogs have tails) and free choice items (Anyone can learn semantics) have important intensional aspects to their meanings.

B.54 Nonconservative determiners?

Background Are there nonconservative determiners in natural language? The proposed universal that all natural language determiners are conservative is surprising, and surprisingly robust. Does it hold up?

“With at most a few exceptions English Dets denote conservative functions.” (Keenan 1996:55)

“All putative counterexamples to Conservativity in the literature are ones in which a sentence of the form Det A’s are B’s is interpreted as D(B)(A), where D is conservative. So the problem is not that Det fails to be conservative, rather it lies with matching the Noun and Predicate properties with the arguments of the Det denotation.”

Keenan has in mind examples like the following:

(C)  
   a. Only dogs bark
   b. Many Scandinavians have won the Nobel Prize.

Your task

i. Run the conservativity test on each sentence in (C), and indicate which if any of the entailments go through and which don’t.

   My answer Here are the tests, with the arrows and negated arrows indicating the results:

   • Only dogs bark $\iff$ Only dogs are dogs that bark.
     (Just suppose that seals bark. Then the left side is false but the right side is true. In fact, the right side is trivially true.)
   • Many Scandinavians have won the Nobel Prize. $\iff$ Many Scandinavians are Scandinavians who have won the Nobel Prize.

ii. Articulate why your results seem problematic for the conservativity generalization.
Problems

My answer  The results falsify the conservativity universal if only and many are both determiners. . .

iii. Propose a resolution (reject the generalization, follow Keenan’s advice in the quotation, something else entirely).

My answer  The case of only is not especially worrisome because it seems not to be a determiner, but rather a highly flexible adverbial and adnominal modifier (Sam only skis; Sam said only that he was sick). In addition, it precedes the determiner in many cases (e.g., only the boys), which is normally impossible (∗every the boys; but note your every move).

The case of many is trickier. The expected semantics for our central case is as follows:

\[
\{x \mid \text{scandinavian}(x) \land \text{nobelist}(x)\} \quad \{x \mid \text{nobelist}(x)\}
\]

where \(k\) is a contextually-set percentage

Suppose that there are 20 million Scandinavians, and suppose that there are 1000 Nobel Prize winners, 600 of whom are Scandinavian. Then the expected semantics is wildly false.

However, if we reverse the restriction and nuclear scope, the result is arguably the intended reading:

\[
\{x \mid \text{scandinavian}(x) \land \text{nobelist}(x)\} \quad \{x \mid \text{nobelist}(x)\}
\]

where \(k\) is a contextually-set percentage

In the situation sketched above, the value of the fraction is \(\frac{3}{5}\), which will suffice for most reasonable choices of the contextual variable \(k\). Thus, the puzzle is arguably not that many is nonconservative, but rather how its restriction and nuclear scope can (apparently) swap places when we do semantic interpretation. If we allow them to switch, then the meanings are in line with compositionality.

B.55  Coordination and monotonicity

Background  Barwise and Cooper (1981) propose to link preferences concerning the choice of and and but in DP coordinations with the monotonicity properties of the DP arguments. I started to provide relevant data in section 9.4.2.1 of handout 9, but I didn’t go far enough. We must check systematically both right and left monotonicity properties before we can be sure of the generalization’s proper formulation.
Problems

Your task  Provide the data needed to obtain a generalization concerning the choice of coordinating element and formulate that generalization, and then try to formulate a suitable generalization.

My answer  This is hard work! A quantificational determiner defines monotonicity properties for both its restriction (left) and nuclear scope (right) argument, so mixed cases are extremely important. In this sense every is important, since it is left downward and right upward. Let’s see how it fares with a quantifier that is downward in restriction and scope:

• i. every student {but/?and} no professor
  ii. no student {but/?and} every professor

We should also check nonmonotone quantifiers like exactly 3 NP:

• i. Exactly three students {but/?and} no professors
  ii. No students {but/?and} but exactly three professors.

These data suggest that the generalization is “Use but if the quantifiers differ in their nuclear scope monotonicity properties, else and”.

B.56   Indexicals as proper names?

Background  The Kaplanian context (the c parameter) is an enrichment of our theory of interpretation. We should try to be sure that the move is justified.

Your task  Suppose, then, that we tried to analyze indexicals as proper names. What (if anything) would this analysis get right, and what (if anything) would it get wrong?

My answer  Indexicals change all the time, as the context changes, whereas proper names are rigid. Now, we find some context-dependency with names — Armstrong is likely to be heard as referring to an astronaut in some contexts and a cyclist in others. But this seems like a different sort of context-dependency, with the names ambiguous but the indexical simply unspecified as to whom they refer to.

B.57   Indexicals and constants: A crucial difference

Background  Kaplan (1989:506) writes, “Indexicals have a context-sensitive character. […] Nonindexical have a fixed character.”
Your task  Explain what this means in the context of the theory described in section 10.1 of handout 10.

My answer  If names are truly rigid, then their value is the same at every context: their Kaplanian character is fixed in the sense that every context c delivers us to the same semantic content. In contrasts, indexicals vary widely across contexts.

B.58  Denotations as sets of assignments

Background  In the text, I suggested how we can assign rich denotations to expressions that contain free variables. This is an important part of this new perspective on interpretation, but it doesn’t cover all the cases.

Your task  Provide denotations of the following in terms of sets of assignments, keeping in mind that an assignment g is in the denotation of an expression φ iff interpreting φ relative to g produces T.

i. \[[\text{happy}(\text{sam}) \lor \neg\text{happy}(\text{sam})]\]^M

My answer  It’s the set of all assignments g: there are no free variables in the formula, and the formula itself is a tautology, so every assignment makes it true.

ii. \[[\text{happy}(\text{sam}) \land \neg\text{happy}(\text{sam})]\]^M

My answer  This is the empty set (of assignments). Since the formula is always false, no choice of assignment can make it true.

B.59  Dynamic indefinites

Background  The discussion in section 10.2 of handout 10 provides a theory of free pronouns that depends on conceiving of denotations as sets of assignment functions. This increases the meaning distinctions we can make, and it is also the first step towards a dynamic theory. But my discussion leaves one wondering how new discourse referents are introduced into the discourse, so that pronouns can pick up on them.

The linguistic answer is that indefinites are the prototypical vehicles for new discourse referents. You say, “I saw a movie last night”, and we now have a new discourse referent to talk about.
Your task  Devise an assignment-based theory of indefinites that captures the fact that they introduce new information. A first step might be to assume that assignments are partial functions on infinite lists of variables. When one uses an indefinite, one adds new variable–entity mappings in some uniform fashion.

My answer  The theory sketched in section 10.2 of handout 10 is not quite dynamic, but it’s getting there. In order to answer this question, we need to make it dynamic.

Familiarity  We now assume that interpretation happens relative to a set of partial assignment functions $G$, with the following representative of the general mode of interpretation:

- $G + \{ \text{happy}(x_{\text{familiar}}) \}^M$ is defined iff every $g \in G$ is such that $g(x)$ is defined.
- Where defined,

$$G + \{ \text{happy}(x_{\text{familiar}}) \}^M = \{ g \in G \mid \text{happy}(x)^Mg = T \}$$

Thus, the effect of a sentence like *He is happy* is to get hold of the subset of assignments $g$ in the input state $G$ that map the formula in question to truth. The *familiar* marker on the variable indicates that we need to have seen the corresponding pronoun before. Formally, this is just to say that all the assignments in our information state supply some value for $x$.

Novelty  We now propose that indefinites have a different sort of effect on the input state. Like pronouns, they pick out free variables, but their new marking corresponds to a different semantics:

- $G + \{ \text{happy}(x_{\text{novel}}) \}^M$ is defined iff $g(x)$ is undefined for every $g \in G$.
- Where defined,

$$G + \{ \text{happy}(x_{\text{novel}}) \}^M = \{ g' \mid g \sim_x g' \text{ for some } g \in G \text{ and happy}(x)^{Mg'} = T \}$$

The relation $g \sim_x g'$ holds just in case $\text{Domain}(g) = \text{Domain}(g') \cup \{ x \}$ and $g$ and $g'$ agree on all the values that they assign to the variables in $\text{Domain}(g)$.

Though the definition is somewhat complicated, its interpretation is relatively straightforward: using an indefinite with variable $x$ brings us to an information state in which every assignment supplies a value for $x$. The definedness condition ensures that the variable is not one we have seen before. This corresponds to the natural language fact that indefinites cannot function anaphorically. It is a sort of converse of the conditions on pronouns, which requires them to be anaphoric.
B.60 Probabilities and sets

Background The second clause of the definition of probability distributions in (10.7) of handout 10 connects set union with additivity. How deep do the connections between sets and probabilities go?

Your task What are the analogues of set intersection, set difference, and subset in the realm of probabilities? Can you find important ways in which the pairs you propose differ?

My answer Where the sets in question are independent, we have the following correspondences between set and probabilities:

1. \( P(A) \cdot P(B) \quad A \cap B \)
2. \( 1 - P(A) \quad U - A \) where \( U \) is the universe of discourse
3. \( P(B|A) = 1 \quad A \subseteq B \)