Appendix to Lecture 7:
Montague’s intensional logic, with lambdas and types.

A.1 Introduction

Tools like Montague’s Intensional Logic are important in making a more satisfactory compositional analysis of natural language semantics possible. What are the differences between Montague’s IL and PC? Here are some of the most important:

(i) The rich type structure of IL.

(ii) The central role played by function-denoting expressions. All of the types except the basic types $e$ and $t$ are functional types, and all of the expressions of IL except those of types $e$ and $t$ are expressions which denote functions. Functions may serve as the arguments and as the values of other functions. In particular, all relations are also represented as functions.

(iii) The inclusion of the operation of “functional application” or “function-argument application”, the application of a function to its argument.

(iv) The use of lambda-expressions. The lambda-operator is the basic tool for building expressions which denote functions.

(v) In place of the one “world” of PC (where there is in effect no distinction between a “world” and a model), the models of IL include a set of possible worlds. Possible worlds are crucially connected with intension/extension distinction and with intensional types. Possible worlds, in particular, underlie the interpretation of modal operators and referential opacity.

(vi) The models of IL also include, in one way or another, a structure of time, used among other things in the interpretation of tense operators like $\text{PAST}$ in the fragment below.

A.2. Intensional Logic (IL).

A.2.1. Types and model structures.

A.2.1.1. Types

Montague’s IL is a typed intensional language; unlike the predicate calculus, which has variables of only one type (the type of entities or individuals), and expressions only of the types of individuals, truth-values, and n-ary relations over individuals, IL has a rich system of types which makes it much easier to achieve a (relatively) close fit between expressions of various categories of a natural language and expressions of IL. The types serve as syntactic categories for the expressions of IL; because of the role of IL as an intermediate language in the semantic interpretation of natural language, the same types are referred to as semantic types for expressions of natural language.

The types of Montague’s IL are as follows:

**Basic types:** $e$ (entities), $t$ (truth values)

**Functional types:** If $a,b$ are types, then $<a,b>$ is a type (the type of functions from $a$-type things to $b$-type things.) Note: We use interchangeably the two notations $<a,b>$ and $a \rightarrow b$, both of which are common in the literature.

**Intensional types:** If $a$ is a type, then $<s,a>$ is a type (the type of functions from possible worlds to things (extensions) of type $a$.)

(In some systems, the basic type $t$ is taken as intensional, interpreted as the type of propositions rather than of truth-values. In general, we will mostly ignore intensionality in these lectures, working most of the time with extensional versions of our fragments and
mentioning intensionality only where directly relevant. But that is only for simplicity of exposition; in general, a thoroughly intensional semantics is presupposed.)

A.2.1.2. Model structures.
In the second lecture, we introduced the simple model structure \( M_1 \) for interpreting the predicate calculus. A model \( M \) for the typed intensional logic IL has much more structure, but that structure is built up recursively from a small set of primitives.

**Model structure for IL:** \( M = \langle D, W, \leq, I \rangle \). Each model must contain:
- A domain \( D \) of entities (individuals)
- A set \( W \) of possible worlds (or possible world-time pairs, or possible situations)
- \( \leq \): an ordering (understood as temporal order) on \( W \)
- \( I \): Interpretation function which assigns semantic values to all constants.

The domains of possible denotations for expressions of type \( a \) (relative to \( D, W \)) are defined recursively as follows:
\[
D_e = D \\
D_t = \{0,1\} \\
D_{<a,b>} = \{ f \mid f : D_a \rightarrow D_b \} \text{ (i.e. the set of all functions } f \text{ from } D_a \text{ to } D_b \text{)} \\
D_{<s,a>} = \{ f \mid f : W \rightarrow D_a \} \text{ (i.e. the set of all functions } f \text{ from } W \text{ to } D_a \text{)}
\]

The semantic interpretation of IL also makes use of a set \( G \) of assignment functions \( g \), functions from variables of all types to values in the corresponding domains.

Each expression of IL has an *intension* and, at each \( w \) in \( W \), an *extension*. The intension is relative to \( M \) and \( g \); the extension is relative to \( M, w, \) and \( g \). But we will not discuss intensions and extensions in this lecture.

A.2.2. Atomic expressions (“lexicon”), notation, and interpretation.
The atomic expressions of IL are constants and variables; there are infinitely many constants and infinitely many variables in each type. Montague introduced a general nomenclature for constants and variables of a given type, using \( c \) and \( v \) with complex subscripts indicating type and an index. In practice, including Montague’s, more mnemonic names are used. Our conventions will be as follows:

Constants of IL will be written in **non-italic boldface**, and their names will usually reflect the English expressions of which they are translations: *man, love*, etc. Their types will be specified. Variables of IL will be written in **italic boldface**, usually observing the following conventions as to types:
- Type \( e: w,x,y,z \), with and without subscripts or primes (this modifier holds for all types.)
- Type \( <e,t>: P, Q \)
- Various relational types such as \( <e,<e,t>>: R \)
- The type of generalized quantifiers: \( T \)

The interpretation of constants is given by the interpretation function \( I \) of the model, and the interpretation of variables by an assignment \( g \), as specified in Rule 1 below.
A.2.3. Syntactic rules and their model-theoretic semantic interpretation

The syntax of IL takes the form of a recursive definition of the set of “meaningful expressions of type a”, ME_a, for all types a. The semantics gives an interpretation rule for each syntactic rule.

Note: when giving syntactic and semantic rules for IL, as for predicate logic, we use a metalanguage which is very similar to IL; but we are not boldfacing the constants and variables of the metalanguage. The metalanguage variables over variables are most often chosen as u or v.

The first rule is a rule for atomic expressions, and the first semantic rule is its interpretation:

**Syntactic Rule 1:** Every constant and variable of type a is in ME_a.

**Semantic Rule 1:**
(a) If " is a constant, then \( \|\alpha\|_{M,w,g} = I(\alpha)(w) \).
(b) If " is a variable, then \( \|\alpha\|_{M,w,g} = g(\alpha) \).

Note: The recursive semantic rules give extensions relative to model, world, and assignment. Read \( \|\alpha\|_{M,w,g} \) as “the semantic value (extension) of alpha relative to M, w, and g.” The interpretation function I assigns to each constant an intension, i.e. a function from possible worlds to extensions; applying that function to a given world \( w \) gives the extension.

**Syntactic Rule 2.** (logical connectives and operators that apply to formulas, mostly from propositional and predicate logic, plus some modal and tense operators.) If \( \phi, \psi \in ME_t \), and \( u \) is a variable of any type, then \( \neg \phi, \phi \& \psi, \phi \lor \psi, \phi \rightarrow \psi \) (also written as \( \phi \equiv \psi \)), \( \exists u \phi, \forall u \phi, \square \phi, \text{PAST} \phi \in ME_t \). Note: “\( \square \phi \)” is read as “Necessarily phi”.

**Semantic Rule 2:**
(a) \( \|\neg \phi\|_{M,w,g} = 1 \) iff \( \|\phi\|_{M,w,g} = 1 \) for all \( w' \) in \( W \).
(b) \( \|\phi \& \psi\|_{M,w,g} = 1 \) iff \( \|\phi\|_{M,w',g} = 1 \) for all \( w' \) in \( W \).
(c) \( \|\text{PAST} \phi\|_{M,w,g} = 1 \) iff \( \|\phi\|_{M,w',g} = 1 \) for some \( w' \leq w \). (This is a simplification; here we are treating each \( w \) as a combined “world/time index”, possibly a situation index; \( w' \leq w \) if \( w' \) is a temporally earlier slice of the same world as \( w \).)

**Syntactic Rule 3: (=):** If \( \alpha, \beta \in ME_a \), then \( \alpha = \beta \in ME_t \).

**Semantic Rule 3:** \( \|\alpha = \beta\|_{M,w,g} = 1 \) iff \( \|\alpha\|_{M,w,g} = \|\beta\|_{M,w,g} \).

(The next two pairs of rules, concerning the “up” and “down” operators, are crucial for intensionality, but we will not discuss them and will not use them.)

**Syntactic Rule 4: (“up”)-operator.)** If \( \alpha \in ME_a \), then \( \langle \alpha \rangle \in ME_{<s,a>} \).

**Semantic Rule 4:** \( \|\langle \alpha \rangle\|_{M,w,g} \) is that function \( h \) of type \( <s,a> \) such that for any \( w' \) in \( W \), \( h(w') = \|\alpha\|_{M,w',g} \).

**Syntactic Rule 5: (“down”)-operator.)** If \( \alpha \in ME_{<s,a>} \), then \( \langle \alpha \rangle \rangle \in ME_a \).

**Semantic Rule 5:** \( \|\langle \alpha \rangle\|_{M,w,g} \) is \( \|\alpha\|_{M,w,g}(w) \).

The next two pairs of rules, function-argument application and lambda-abstraction, are among the most important devices of IL, and we will make repeated use of them.

**Function-argument application:**
Syntactic Rule 6: If $\alpha \in ME_{<a,b>}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.

Semantic Rule 6: $||\alpha(\beta)||_{M,w,g}^{M,w,g} = ||\alpha||_{M,w,g}^{M,w,g} (||\beta||_{M,w,g}^{M,w,g})$

Lambda-abstraction:

Syntactic Rule 7: If $\alpha \in ME_a$ and $u$ is a variable of type $b$, then $\lambda u[\alpha] \in ME_{<b,a>}$. 

Semantic Rule 7: $||\lambda u[\alpha]||_{M,w,g}^{M,w,g}$ is that function $f$ of type $b \rightarrow a$ such that for any object $d$ of type $b$, $f(d) = ||\alpha||_{M,w,g}^{M,w,g}[d/u]$. 