
1. English Fragment 1.

1.0 Introduction

In this handout we present a small sample English grammar (a “fragment”, in MG terminology), that is, an explicit description of the syntax and semantics of a small part of English. This fragment is intended to serve several purposes: making certain aspects of formal semantics more explicit, including (and illustrating) more of the basics of the lambda-calculus. The fragment is of interest in its own right and will also serve as background for the next lecture. The fragment, with its very minimal lexicon, also illustrates the typically minimal treatment of the lexicon in classical Montague grammar.

The semantics of the fragment will be given via translation into Montague’s Intensional Logic (IL) (the alternatives would be to give a direct model-theoretic interpretation, or an interpretation via translation into some other model-theoretically interpreted intermediate language). In Lecture 2 we presented Montague’s IL: Its type structure and the model structures in which it is interpreted, and its syntax and model-theoretic semantics.

Now we introduce the fragment of English: first the syntactic categories and the category-type correspondence, then the basic syntactic rules and the principles of semantic interpretation, and then a small lexicon and some meaning postulates. In Section 2 we present some examples. Certain rules of the fragment are postponed to Section 3 where they receive separate discussion; these are rules that go beyond the simple phrase structure rule schemata of Section 1.

1.1. Syntactic categories and their semantic types.

<table>
<thead>
<tr>
<th>Syntactic category</th>
<th>Semantic type</th>
<th>Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProperN</td>
<td>e</td>
<td>names (John)</td>
</tr>
<tr>
<td>S</td>
<td>t</td>
<td>sentences</td>
</tr>
<tr>
<td>CN(P)</td>
<td>e → t</td>
<td>common noun phrases (cat)</td>
</tr>
<tr>
<td>NP</td>
<td>(i) e</td>
<td>“e-type” or “referential” NPs (John, the king)</td>
</tr>
<tr>
<td></td>
<td>(ii)(e→t) t</td>
<td>noun phrases as generalized quantifiers (every man, the king, a man, John)</td>
</tr>
<tr>
<td>ADJ(P)</td>
<td>(iii) e → t</td>
<td>NPs as predicates (a man, the king)</td>
</tr>
<tr>
<td></td>
<td>(i) e → t</td>
<td>predicative adjectives (carnivorous, happy)</td>
</tr>
<tr>
<td></td>
<td>(ii)(e→t)(e→t)</td>
<td>adjectives as predicate modifiers (skillful)</td>
</tr>
<tr>
<td>REL</td>
<td>e → t</td>
<td>relative clauses (who(m) Mary loves)</td>
</tr>
<tr>
<td>VP, IV</td>
<td>e → t</td>
<td>verb phrases, intransitive verbs (loves Mary, is tall, walks)</td>
</tr>
<tr>
<td>TV(P)</td>
<td>type(NP) → type(VP)</td>
<td>transitive verb (phrase) (loves)</td>
</tr>
<tr>
<td>is</td>
<td>(e → t) → (e → t)</td>
<td>is</td>
</tr>
<tr>
<td>DET</td>
<td>type(CN) → type(NP)</td>
<td>a, some, the, every, no</td>
</tr>
</tbody>
</table>

1.2. Syntactic Rules and Semantic Rules.

Two different approaches to semantic interpretation of natural language syntax (both compositional, both formalized, and illustrated, by Montague):

A. Direct Model-theoretic interpretation: Semantic values of natural language expressions (or their “underlying structure” counterparts) are given directly in model-theoretic terms; no intermediate language like Montague’s intensional logic (but for some linguists there is a syntactic level of “logical form” to which this model-theoretic interpretation applies, so the distinction between the two strategies is not always sharp.) This is the direct “English as a formal language” strategy. For illustration, see Heim and Kratzer (1998). Also see the discussion in Larson’s chapter 12.

B. Interpretation via translation: Stage 1: compositional translation from natural language to a language of semantic representation, such as Montague’s intensional logic. For an expression of category C formed from expressions of category A and of category B, determine TR(φ) as a function of TR(ψ) and TR(β). Stage 2: Apply the compositional model-theoretic interpretation rules to the intermediate language.

We will follow the strategy of interpretation via translation, using Montague’s IL as the intermediate language. But everything we do could also be done by direct interpretation.

Some abbreviations and notational conventions:

We will sometimes write φ’ as a shorthand for TR(φ). And sometimes we use the category name in place of a variable over expressions of that category, writing TR(A, or A’, in place of TR(φ) when φ is an expression of category A. And we will write some of our
syntactic rules like simple phrase structure rules. Here is an example of a syntactic rule and corresponding translation rule, and their abbreviations as they will appear below.

**Official Syntactic Rule:** If $\alpha$ is an expression of category DET and $\beta$ is an expression of category CNP, then $F_\alpha(\alpha, \beta)$ is an expression of category NP, where $F_\alpha(\alpha, \beta) = \alpha \delta \beta$.

**Official Semantic Rule:** If $\text{TR}(\alpha) = \alpha'$ and $\text{TR}(\beta) = \beta'$, then $\text{TR}(F_\alpha(\alpha, \beta)) = \alpha'(\beta')$.

**Abbreviated Syntactic Rule:** $NP \rightarrow DET \ CNP$

**Abbreviated Semantic Rule:** $NP' = DET'(CNP')$

### 1.2.1. Basic syntactic rules

#### Basic rules, phrasal:
- $S \rightarrow NP \ VP$
- $NP \rightarrow DET \ CNP$
- $CNP \rightarrow ADJP \ CNP$
- $CNP \rightarrow CNP \ REL$
- $VP \rightarrow TVP \ NP$
- $VP \rightarrow \text{is ADJP}$
- $VP \rightarrow \text{is NP}$

**Basic rules, non-branching rules introducing lexical categories:**
- $NP \rightarrow \text{ProperN}$
- $CNP \rightarrow \text{CN}$
- $TVP \rightarrow TV$
- $ADJP \rightarrow ADJ$
- $VP \rightarrow IV$

### 1.2.2. Semantic interpretation of the basic rules.

The basic principle for all semantic interpretation in formal semantics is the principle of compositionality; the meaning of the whole must be a function of the meanings of the parts. In the most “stipulative” case, we write a semantic interpretation rule (translation or direct model-theoretic interpretation) for each syntactic formation rule, as in classical MG. In more contemporary approaches, we look for general principles governing the form of the rules and their correspondence (possibly mediated by some syntactic level of “Logical Form”). Here we are using an artificially simple fragment, and we have presented the syntactic rules in a form which is explicit but not particularly general; but we have the tools to illustrate a few basic generalizations concerning syntax-semantics correspondence.

#### 1.2.2.1. Type-driven translation.


To a great extent, possibly completely, we can formulate general principles for the interpretation of the basic syntactic constructions based on the semantic types of the constituent parts.

So suppose we are given a rule $A \rightarrow B \ C$, and we want to know how to determine $A'$ as a function of $B'$ and $C'$ (equivalently, $\text{TR}(A)$ as a function of $\text{TR}(B)$ and $\text{TR}(C)$; and ultimately, $||A||$ as a function of $||B||$ and $||C||$). Similarly for non-branching rules $A \rightarrow B$.

**General principles:** function-argument application, predicate conjunction, identity.

The following versions of general type-driven interpretation principles are taken from Heim and Kratzer (1995). In the original they are written for direct model-theoretic interpretation.

1. **Terminal Nodes (TN):** If $\alpha$ is a terminal node, then $||\alpha||$ is specified in the lexicon.
2. **Non-Branching Nodes (NN):** If $\alpha$ is a non-branching node, and $\beta$ is its daughter node, then $||\alpha|| = [||\beta||].$
3. **Functional Application (FA):** If $\alpha$ is a branching node, $||\beta, \gamma||$ is the set of $\alpha$'s daughters, and $||\beta||$ is a function whose domain contains $||\gamma||$, then $||\alpha|| = [||\beta||][||\gamma||]$. (4) **Predicate Modification (PM):** If $\alpha$ is a branching node, $||\beta, \gamma||$ is the set of $\alpha$’s daughters, and $||\beta||$ and $||\gamma||$ are both in $D_{e,t}$, then $||\alpha|| = \lambda x \epsilon D_e. [||\beta|| (x) = 1]$ and $||\gamma|| (x) = 1$.

(A further principle is needed for intensional functional application, which we will mention only later.)

Exactly analogous principles can be written for type-driven translation:

1. **Terminal Nodes (TN):** If $\alpha$ is a terminal node, then $\text{TR}(A)$ is specified in the lexicon.
2. **Non-Branching Nodes (NN):** If $A \rightarrow B$ is a unary rule and $A, B$ are of the same type, then $\text{TR}(A) = \text{TR}(B)$.
3. **Functional Application (FA):** If $A$ is a branching node, $|||B, C|||$ is the set of $A$’s daughters, and $B'$ is of a functional type $a$ and $C'$ is of type $a$, then $\text{TR}(A) = \text{TR}(B) \text{TR}(C)$. (4) **Predicate Modification (PM):** If $A$ is a branching node, $|||B, C|||$ is the set of $A$’s daughters, and if $B'$ and $C'$ are of (same) predicative type $a \rightarrow t$, and the syntactic category $A$ can correspond to type $a \rightarrow t$, then $\text{TR}(A) = \lambda x \epsilon D_{e}. [\text{TR}(B)(x) \& \text{TR}(C)(x)].$ (i.e. $||A|| = [||B||] \cap [||C||]$.)

**1.2.2.2. Result of those principles for the translation of the basic rules.**

**Function-argument application:** $S \rightarrow NP \ VP$, $NP \rightarrow DET \ CNP$, $VP \rightarrow TVP \ NP$, $VP \rightarrow \text{is ADJP}$, $VP \rightarrow \text{is NP}$, and those instances of CNP $\rightarrow ADJP \ CNP$ in which ADJP is of type $t \rightarrow e \rightarrow (e \rightarrow t)$.

Example: Consider the rule $S \rightarrow NP \ VP$. If $NP$ is of type $(e \rightarrow t) \rightarrow t$ and $VP$ is of type $e \rightarrow t$, then the translation of $S$ will be $NP'(VP')$ (e.g., *Every man walks*). If $NP$ is of type $e$ and $VP$ is of type $e \rightarrow t$, then the translation of $S$ will be $VP'(NP')$ (e.g., *John walks*).

**Predicate modification:** $CNP \rightarrow CNP \ REL$, and those instances of CNP $\rightarrow ADJP \ CNP$ in which ADJP is of type $e \rightarrow t$.

**Non-branching nodes:** $NP \rightarrow \text{ProperN}$, $CNP \rightarrow \text{CN}$, $TVP \rightarrow TV$, $ADJP \rightarrow ADJ$.

### 1.2.3. Rules of Relative clauses, Quantification, Phrasal Negation. See Section 3.
1.2.4. Type multiplicity and type shifting.

We noted in Lecture 1 that classical model-theoretic semantics in the Montague tradition requires that there be a single semantic type for each syntactic category. But in Fragment 1, several syntactic types have more than one corresponding semantic type. The possibility of type multiplicity and type shifting has been increasingly recognized in the last decade or so, and there are a variety of formal approaches that accommodate type multiplicity without giving up compositionality. We will not go into details about formal issues here, but we do want to include a number of categories with multiple semantic types; several were introduced in Fragment 1, and more will be introduced in later lectures.

Montague tradition: uniform treatment of NPs as generalized quantifiers, type (e \rightarrow i) \rightarrow t.

\[
\lambda P [P(John)]
\]

\[
\lambda P x [\text{fool}(x) \& P(x)]
\]

\[
\lambda P x [\text{man}(x) \rightarrow P(x)]
\]

Intuitive type multiplicity of NPs (and see Heim 1982, Kamp 1981):

\[
\text{John}
\]

"referential use": \text{John} type \(e\)

\[
\text{a fool}
\]

"predicative use": \text{fool} type \(e \rightarrow t\)

\[
\text{every man}
\]

"quantifier use": (above) type \((e \rightarrow t) \rightarrow t\)

Resolution: All NPs have meaning of type \((e \rightarrow t) \rightarrow t\); some also have meanings of types \(e\) and/or \(e \rightarrow t\). General principles for predicting (Partee 1986). Predicates may semantically take arguments of type \(e\), \(e \rightarrow t\), or \((e \rightarrow t) \rightarrow t\), among others. (More on type-shifting in Lecture 6 or see RGGU 2005 Lecture 8 on my website).

Type choice determined by a combination of factors including coercion by demands of predicates, "try simplest types first" strategy, and default preferences of particular determiners.

Note the effects of this type multiplicity on type-driven translation. The \(S \rightarrow NP\) VP rule, for instance, will have two different translations. The VP, we have assumed, is always of type \(e \rightarrow t\). If the NP is of type \(e\), the translation will be VP('NP'), whereas if the NP is of type \((e \rightarrow t) \rightarrow t\), the translation will be NP('VP'), as noted above in Section 1.2.2.2. [See Homework problem #3 of Homework 1.]

1.3. Lexicon.

Here we illustrate the treatment of the lexicon in Montague (1973) ("PTQ"). Montague, not unreasonably, saw a great difference between the study of the principles of compositional semantics, which are very similar to the principles of compositional semantics for logical languages as studied in logic and model theory, and the study of lexical semantics, which he perceived as much more "empirical". For Montague, it was important to figure out the difference in logical type between \text{easy} and \text{eager}, or between \text{seem} and \text{try}, but he did not try to say anything about the difference in meaning between two elements with the same "structural" or type-theoretic behavior, such as \text{easy} and \text{difficult} or \text{run} and \text{walk}. For Montague, most lexical items were considered atomic expressions of a given type, and simply translated into constants of IL of the given type.

First we simply list some lexical items of various syntactic categories; aside from the category DET, these are all open classes. Then we discuss their semantics.

In later lectures we will be concerned with how best to enrich the semantic information associated with the lexicon in ways compatible with a compositional semantics.

ProperN: John, Mary, Bill, ...  
DET: some, a, the, every  
ADJ: carnivorous, happy, skillful, tall, former, alleged, old, ...  
CN: man, king, violinist, surgeon, fish, senator, ...  
TV: sees, loves, catches, eats, ...  
IV: walks, talks, runs, ...

Semantics of Lexicon (MGU):

Open class lexical items (nouns, adjectives, verbs) translated into constants of appropriate type (notation: English expressions man, tall translated into IL constants man, tall, etc.). Interpretation of these constants a central task of lexical semantics. A few open class words (e.g. be, entity, former) sometimes treated as part of the "logical vocabulary".

Closed class lexical items: some treated like open class items (e.g. most prepositions), others (esp. "logical" words) given explicit interpretations, as illustrated below.

Determiners:

We have three types of NPs and correspondingly three types of DETs. Not all DETs occur in all types; the is one of the few that does. For DETs that occur in more than one type, we will subscript the "homonyms" with mnemonic subscripts: \(\varepsilon\) for those that combine with a CNP to form an e-type NP, \(\text{pred}\) for those that form predicate nominals, and \(\text{GQ}\) for those that form generalized quantifier-type NPs. (Note that these are not the types of the DETs themselves, but their own types have unpleasantly long and hard-to-read names.) There are systematic relations among these "homonyms" (Partee 1987), but we are not discussing them here.

(i) \(e\)-forming DETs.

For the translation of the \(\varepsilon\), we need to add the \text{ iota-operator} \(\iota\) to IL.

Syntax:  If \(\varepsilon \in \text{ME}\) and \(\varepsilon\) is a variable of type \(e\), then \(\iota\varepsilon \in \text{ME}\).

Semantics:  \[\iota\varepsilon\varepsilon^M = d\] if there is one and only one \(d \in D\) such that \[\varepsilon^M[d] = 1\]

So \(\iota\varepsilon\varepsilon^M = d\) is defined only if \(d\) exists.

(ii) predicate-forming DETs.

DETs as functors forming predicate nominals are of type \((e \rightarrow t) \rightarrow (e \rightarrow t)\).

\[\text{TR}(\text{the} \varepsilon) = \lambda x (\text{man}(x) \& \forall y (\text{man}(y) \rightarrow (x = y)))\]
3. Rules of Relative clause formation, Quantifying In, Phrasal Negation.

3.1. (Restrictive) Relative clause formation.

We begin with an illustration of what the rule does before stating it (in a sketchy form).

Consider the CNP man who Mary loves:

Syntactic derivation (very sketchy):

| CNP | REL: who Mary loves [e]
| CN  | S: Mary loves him
| man |

The types for CN, CNP, and REL are all e → t; so the principle for combining CNP and REL gives: λy[CNP(y) & REL(y)] (Predicate conjunction)

The relative clause itself is a predicate formed by λ-abstraction on the variable corresponding to the WH-word. (Partee 1976 suggests a general principle that all “unbounded movement rules” are interpreted as involving variable-binding; and λ-abstraction can be taken as the most basic variable-binding operation.)

A syntactically very crude and informal version of the relative clause rule, with its semantic interpretation, can be stated as follows:

Rel Clause Rule, syntax: If φ is an S and φ contains an indexed pronoun he / him, in relativizable position, then the result of adjoining who(φ) to S and leaving a trace ei in place of he / him, is a REL.

Rel Clause Rule, semantics: If φ translates as φ’, then REL translates as λx[φ’].

Semantic derivation corresponding to the syntactic derivation above; compositional translation into IL: (read bottom-to-top) (and see Homework problem 5a)

\[ \lambda y[\text{man}(y) \land \lambda x[\text{love}(\text{Mary}, x)](y)] \]

\[ \text{man} \quad \lambda x[\text{love}(\text{Mary}, x)] \]

\[ \text{love}(\text{Mary}, x) \]

By λ-conversion, the top line is equivalent to: \( \lambda y[\text{man}(y) \land \text{love}(\text{Mary}, y)] \)

3.2. Quantifying In.

This is (an informal statement of) Montague’s Quantifying In rule; it is similar to the Quantifier-Lowering rule of Generative Semantics and Quantifier Raising (QR) of May (1977); various alternative treatments of quantifier scope ambiguity exist, including Cooper-
storage (Cooper 1975) and Herman Hendriks’s flexible typing approach (Hendriks 1988, 1993).

**Quantifying In Rule, Syntax:** (informally stated): An NP combines with a sentence with respect to a choice of variable (“he,” in MG). Substitute the NP for the first occurrence of the variable; change any further occurrences of the variable into pronouns of the appropriate number and gender.

**Semantic rule:** NP’(λx[S’])  (The set of properties denoted by the NP includes the property denoted by the λ-expression derived from the sentence.)

We illustrate with two derivations for the ambiguous sentence *Every student read a book.*

**Syntactic derivation (i) (rough sketch; read from bottom to top. Bold is used here to show which variables are substituted for at each step.)**

\[
\begin{align*}
S: & \text{ every student read a book} \\
NP: & \text{ every student} \quad S: \text{ he3 read him2} \\
NP: & \text{ a book} \quad S: \text{ he3 read him2}
\end{align*}
\]

Compositional Translation: *(every student)’(λx₂[[(a book’)(λx₂ [read (x₁, x₂))])]*)

Rough paraphrase: Every student has the property that there is a book that he read.

If you write out the interpretations of the NPs and apply Lambda-Conversion as many times as possible, the result will be (some alphabetic variant of) the first-order PC formula \( \forall x (\text{student}(x) \rightarrow \exists y (\text{book}(y) \land \text{read}(x,y))) \).

**Syntactic derivation (ii)**

\[
\begin{align*}
S: & \text{ every student read a book} \\
NP: & \text{ a book} \quad S: \text{ he3 read him2} \\
NP: & \text{ every student} \quad S: \text{ he3 read him2}
\end{align*}
\]

Compositional Translation: *(a book’)(λx₂[every student’)(λx₂ [read (x₁, x₂))])]*

**Homework problem 6:**

Underlying Structure (Generative Semantics), Czech Dependency Grammar (Č), GPSG (Prague), Deep Syntactic Structure (Mel’čuk), Underlying Structure (Generative Semantics), LFG, HPSG, and various contemporary versions of Categorial Grammar are attempts to represent all the necessary syntactic information directly in a single “level” of syntax.

### 3.3. Conjunction.

One simple and elegant application of lambda abstraction which Montague used in PTQ is its use in defining the interpretation of “Boolean” phrasal conjunction, disjunction, and negation in terms of the corresponding sentential operations.

“Boolean” phrasal conjunction, illustrated in all the examples below, is distinguished from “part-whole” or “group” conjunction, illustrated by “John and Mary are a happy couple” and “The flag is red and white”, which are not equivalent, respectively, to “John is a happy couple and Mary is a happy couple” and “The flag is red and the flag is white”.

To illustrate this application, we add a few syntactic and semantic rules to our fragment. Note: in the semantic rules, we use \( S_1 \) and \( S_2 \), etc., to refer to the first and second S in the syntactic rule.

**Syntactic rules for conjunction:**

**Corresponding semantic rules:**

\[
\begin{align*}
S & \rightarrow S \land S' \\
S & \rightarrow S \lor S' \\
NP & \rightarrow NP \lor NP' \\
VP & \rightarrow VP \lor VP'
\end{align*}
\]

The NP conjunction and disjunction rules presuppose that the NPs are interpreted as generalized quantifiers, type \(<e,t>\times\). \( P \) is a variable of type \(<e,t>\). (Conjoined NPs of type \( e \) can be interpreted as groups, but not as conjoined by Boolean conjunction.)

**Examples:**

*Some animals swim and some animals fly.* (S-conjunction)

*Some animals swim and fly.* (VP-conjunction)

*Every fish and some birds swim.* (NP-conjunction)

*Every painting and every statue was photographed or (was) videotaped.* (NP-conjunction and VP-conjunction (or conjunction of participles, if we omit the second ‘was’, but it’s equivalent to VP conjunction). The rules do correctly “predict” which conjunction has wider scope. (Optional unlisted extra homework question: work out the last example. Treat “was photographed” and “was videotaped” as simple 1-place predicates for this exercise.)

We could extend the rules above, and generalize them (as is done in Partee and Rooth 1983), so as to include further types of phrasal conjunction such as the following:

*John bought and read a new book.* (TV conjunction)

*No number is even and odd.* (Predicate ADJP conjunction)

*Mary saw an old and interesting manuscript.* (Pre-nominal ADJP conjunction.)
In fact, we do not have to “stipulate” the rules one-by-one as we have done above; it is possible to predict them in a general way from the types of the expressions being conjoined. But that goes beyond the scope of these lectures; see Partee and Rooth 1983.

3.4. Phrasal and lexical negation.

As an additional augmentation of our grammar which adds further illustration of the application of the lambda calculus, let us consider the relations among sentence negation, phrasal negation, and lexical (prefixal) negation.

The syntax of sentential negation in English is slightly complicated because of its interaction with the system of verbal auxiliaries, which we have not included in our simple grammar. Let us ignore the syntactic complexities here and work with a “Logical Form” grammar in which we have a simple phrase structure rule S → NEG S. And let us add to the lexicon an element “NOT” of syntactic category NEG, of semantic type t → t, whose translation into IL is simply ¬. Then the interpretation of NEG S, by function-argument application, will just be ¬S.

Now what about phrasal negation, like not every boy, not today, Mary but not John, not very intelligent, not love Bill? Negation, like conjunction, is a “Boolean” operation, and it is easy to define its interpretation with expressions of various types on the basis of its interpretation with sentences. For example:

Syntax: VP → NEGVP VP
Semantics: Type: Since the type of VP is e→t, the type of NEGVP must be (e→t)→(e→t).

\[ \text{TR}(\text{NEGVP}) = \lambda x[\text{TR}(\text{VP})]\]

Similar rules for negation of other phrasal categories can be derived in a uniform way.

(See Homework Problem 3b, which asks for NEGNP and optionally NEGNP.)

Both negation and conjunction (and disjunction) thus have natural extensions to a wide range of types (see Partee and Rooth 1983), with meanings systematically derivable from their basic meanings, which apply to sentences. Lexical negation, as in unhappy, impossible, unbroken, inedible, can also be defined with the use of lambdas:

\[ \text{TR}(\text{unhappy}) = \lambda x[\text{TR}(\text{happy})(x)] = \lambda x[\text{happy}(x)] \]

**REFERENCES.**

***Note: See website for links to some of these in downloadable form.***


