1. Syntactic categories and their semantic types.

<table>
<thead>
<tr>
<th>Syntactic category</th>
<th>Semantic type (extensionalized)</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, he.)</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>REL</td>
<td>(i) (e → t) (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>REL</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>DET</td>
<td>type(CN) → type(NP)</td>
<td>is</td>
</tr>
<tr>
<td></td>
<td>a, some, the, every, no</td>
<td></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(α) 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 ϕα as a shorthand for TR(α). 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(α) when α 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 $T_\alpha(\alpha, \beta)$ is an expression of category NP, where $T_\alpha(\alpha, \beta) = \alpha \beta$. 

**Official Semantic Rule:** If $TR(\alpha) = \alpha'$ and $TR(\beta) = \beta'$, then $TR(T_\alpha(\alpha, \beta)) = \alpha'(\beta')$.

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

**Abbreviated Semantic Rule:** NP$^*$ $\rightarrow$ 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 $\rightarrow$ is ADJP
- VP $\rightarrow$ is NP
- Basic rules, non-branching rules introducing lexical categories:
  - NP $\rightarrow$ ProperN
  - NP $\rightarrow$ Pronoun
  - CNP $\rightarrow$ 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. (Partee 1976, Partee and Rooth 1983, Klein and Sag 1985)

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, $TR(A)$ as a function of $TR(B)$ and $TR(C)$; 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, $[[\alpha, \gamma]]$ 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, $[[\alpha, \gamma]]$ is the set of $\alpha$’s daughters, and $[[\beta]]$ and $[[\gamma]]$ are both in $D_{x\in\gamma}$, then $[[\alpha]] = \lambda x \in D_{\gamma}[[\beta]](x) = 1$ and $[[\gamma]] (\alpha) = 1$.

(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 $TR(\alpha)$ 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 $TR(A) = TR(B)$.
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 $a \rightarrow b$ and $C'$ is of type $a$, then $TR(A) = TR(B) \circ TR(C)$.
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 also correspond to type $a \rightarrow t$, then $TR(A) = \lambda x [TR(B)(x) \& TR(C)(x)]$ (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, $VP \rightarrow is$ ADJP, $VP \rightarrow is$ NP, and those instances of CNP $\rightarrow$ ADJP CNP in which ADJP is of type $(e \rightarrow t) \rightarrow (e \rightarrow t)$.

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 $NP'(VP'(x))$ (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 $NP'(VP)$ (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 $\rightarrow$ ProperN, CNP $\rightarrow$ 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 → t) → t.

<table>
<thead>
<tr>
<th>Simplicity</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>[ ϕ[\text{John}] ]</td>
</tr>
<tr>
<td>he</td>
<td>[ ϕ[x] ]</td>
</tr>
<tr>
<td>a fool</td>
<td>[ ϕ[\text{foo} \times \text{P}(x)] ]</td>
</tr>
<tr>
<td>every man</td>
<td>[ ϕ[\text{man} \times \text{P}(x)] ]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Simplicity</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>&quot;referential use&quot;: [ ϕ[\text{John}] ]</td>
</tr>
<tr>
<td>he</td>
<td>&quot;e-type variable&quot;: [ x_1 ]</td>
</tr>
<tr>
<td>a fool</td>
<td>&quot;predicative use&quot;: [ ϕ ]</td>
</tr>
<tr>
<td>every man</td>
<td>&quot;quantifier use&quot;: [ \lambda (\text{man} \times \text{P}(x)) ]</td>
</tr>
</tbody>
</table>

Resolution: All NPs have meaning of type (e→t); some also have meanings of types e and/or t→t. General principles for predicting (Partee 1986). Predicates may semantically take arguments of type e, e→t, or (e→t)→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→NP VP rule, for instance, will have two different translations. The VP, we have assumed, is always of type e→t. If the NP is of type e, the translation will be VP(NP), whereas if the NP is of type (e→t)→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 easy and eager, or between seem and 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 easy and difficult or run and 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 categories DET and Pronoun, 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, ...

Pronoun: he1, he2, ...,

DET: some, a, the, every

ADJ: carniverous, 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 (MG):

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.

Pronouns:

We will treat the pronouns he, she, it as expressions of type e (like proper nouns). In certain circumstances they may get other types, but we ignore that for now. Initially, we will follow Montague's practice of introducing them lexically as indexed "dummy pronouns", (he1, he2, ...), translated as x1, x2, .... More details about how they get their features of case, gender, and number will be discussed later. Both pronouns and proper nouns can be thought of as lexical members of the category NP. (Some would argue that proper nouns are lexical members of the same class as common nouns, CN, but with a property of combining with a null form of the definite article, and that pronouns are underlyingly the class DET, with the property of combining with a null empty common noun (Postal argued for this, citing examples like we Americans:].

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: e for those that combine with a CNP to form an e-type NP, pred for those that form predicate nominals, and 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, we need to add the "ata-operator" ∈ IL.

Syntax: If \( \varphi \in \text{ME} \) and \( a \) is a variable of type e, then \( \text{ua}[\varphi] \in \text{ME} \).

Semantics: \[ \text{ua}[\varphi] \text{M}^e, x = d \text{ if and only one } d \in D \] such that \( \text{ua}[\varphi] \text{M}^e, x \) is undefined otherwise.
So $\textit{t}(\text{king}(x))$ denotes the unique individual who is king, if there is one, and is undefined if there is either no king or more than one king.

The type for determiners as functors forming e-type ("referential") terms is $(e \rightarrow t) \rightarrow e$; the only determiner of this type we will introduce is $\textit{the}$. For ease of reading, we will give the translations of representative NPs, rather than for the DET itself; the DET translation can be formed in each case by $\lambda$-abstraction on the CNP (see below, $\text{TR}(\alpha_{\text{pred}})$).

$\text{TR}(\text{the king}) = \textit{t}(\text{king}(x))$

(ii) predicate-forming DETs.

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

$\text{TR}(\alpha_{\text{pred} \text{ man}}) = \textit{man}$
$\text{TR}(\alpha_{\text{pred} \text{ violinist}}) = \lambda x[(\text{man}(x) \land \forall y(\text{man}(y) \rightarrow y=x))]
$

We illustrate the translation of the DET itself with the translation of $\alpha_{\text{pred} \text{ man}}$.

$\text{TR}(\alpha_{\text{pred} \text{ man}}) = \lambda [P]$

(iii) generalized quantifier-forming DETs.

DETs as functors forming generalized quantifiers are of type $(e \rightarrow t) \rightarrow (e \rightarrow t)$.

$\text{TR}(\alpha_{\text{GQ} \text{ man}}) = \lambda \exists x[(\text{man}(x) \land \forall y(\text{man}(y) \rightarrow y=x)) \land \forall P(P[x])])$

The copula be:

$\text{TR}(\text{is}) = \lambda P\lambda x(P(x))$ ("Predicate")

Results (see also Section 5, and Homework 2, problem 2):

$\text{TR}(\text{is green}) = \textit{green}$
$\text{TR}(\text{is $\alpha_{\text{GQ} \text{ man}}$}) = \textit{man}$
$\text{TR}(\text{is the $\alpha_{\text{GQ} \text{ king}}$}) = \lambda x[(\text{king}(x) \land \forall y(\text{king}(y) \rightarrow y=x))]]$

2. Examples

(1) $\text{is happy}$

$\text{TR}(\text{is happy}) = \lambda P\lambda x(P(x))$

$\text{TR}(\text{happy}) = \textit{happy}$

$\text{TR}(\text{is happy}) = \lambda P\lambda x(P(x))$ (happy)

$\lambda x[\textit{happy}(x)] = \textit{happy}$

(2) The violinist is happy (with e-type interpretation of subject)

$\text{TR}(\text{the violinist}) = \lambda x[\text{violinist}(x)]$

$\text{TR}(\text{is the violinist}) = \textit{happy}$

$\text{TR}(\text{is the violinist}) = \lambda x[\text{violinist}(x)]$

$\text{TR}(\text{the violinist is happy}) = \lambda x[\text{violinist}(x)]$ (happy)

(3) Every violinist is happy (with GQ-type subject)

$\text{TR}(\text{every violinist}) = \lambda P\lambda x[\text{violinist}(x) \rightarrow P(x)]$

$\text{TR}(\text{is happy}) = \textit{happy}$

$\text{TR}(\text{is the violinist}) = \lambda x[\text{violinist}(x)]$

$\text{TR}(\text{every violinist is happy}) = \lambda P\lambda x[\text{violinist}(x) \rightarrow P(x)] (\text{happy})$

$\lambda x[\text{violinist}(x) \rightarrow \text{happy}(x)]$

(4) Every surgeon is a skillful violinist

(The type of every surgeon must be $(e \rightarrow t) \rightarrow t$; the type of a skillful violinist must be $e \rightarrow t$. Assume the type of skillful is $(e \rightarrow t) \rightarrow (e \rightarrow t)$.)

TR$($every surgeon is a skillful violinist$) = \forall x[\text{surgeon}(x) \rightarrow \text{skillful(violinist)(x))]

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):

$$\text{CNP}
\text{REL: who Mary loves [e]}
\text{CN}
\text{S: Mary loves him1}
\text{man}
$$

The types for CN, CNP, and REL are all $e \rightarrow t$; so the principle for combining CNP and REL gives: $\lambda [P](\text{CNP}([P]y) \land \text{REL}([P]y))$ (Predicate conjunction).

The relative clause itself is a predicate formed by $\lambda$-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 $\lambda$-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 $\phi$ is an S and $\phi$ contains an indexed pronoun he, his, in relativizer position, then the result of adjoining who(m) to S and leaving a trace e in place of he, his, is a REL.

Rel Clause Rule, semantics: If $\phi$ translates as $\phi^*$, then REL translates as $\lambda x [\phi^*]$. 

Semantic derivation corresponding to the syntactic derivation above; compositional translation into IL: (read bottom-to-top) (and see Homework 1, Problem 5)

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

By $\lambda$-conversion, the top line is equivalent to: $\lambda x [\text{man}(y) \land \text{love (Mary, y)]}$
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, 1995).

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.

Semantical 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.)

S: every student read a book
NP: every student S: he3 read a book
NP: a book S: he3 read him2

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 ∀x(student(x) →∃y(book(y) & read(x,y))).

Syntactic derivation (ii)

S: every student read a book
NP: a book S: every student read him2
NP: every student S: he3 read him2

Compositional Translation: (a book)’(λx2[(every student)’(λx2 [read (x1, x2)])]) [See Homework 2, problem 1 (next week).]

Paraphrase: Some book has the property that every student read it. After applying Lambda-Conversion as many times as possible, the result will be (some alphabetic variant of) the first-order PC formula ∃y(book(y) & ∀x(student(x) → read(x,y))).

Observation: Compositional semantics requires that every ambiguous sentence be explainable on the basis of ambiguous lexical items and/or multiple syntactic derivations. Semantic structure mirrors syntactic part-whole structure, which in Montague Grammar is represented by syntactic derivational structure, not surface structure. There are different theories of the semantically relevant syntactic structure: “Derivation trees” or “analysis trees” (MG), LF (Chomskian GB or Minimalist theory), Tectogrammatical Dependency Trees (Prague), Deep Syntactic Structure (Mel’euik) Underlying Structure (Generative Semantics), ... GPSG, HPFG, 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 S1 and S2, etc., to refer to the first and second S in the syntactic rule.

Syntactic rules for conjunction:  Corresponding semantic rules:

S → S and S
S → S or S
VP → VP and VP
VP → VP or VP
NP → NP and NP
NP → NP or NP

NP conjunction and disjunction rules presuppose that the NPs are interpreted as generalized quantifiers, type <<e,t>,t>; 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 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.

(Original 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. Bound variable anaphora – preview (New 2008)

The Relative Clause rule and the Quantifying In rule are the only rules we’ve seen so far that mention the pronouns he, she, and he. Those rules create variable binding.

(i) who, which in relative clauses: expressions of lambda abstraction. (See Homework 1, Problem 5) who loves Mary: λx[love(x, m)], derived from he, loves Mary.

whom Mary loves: λx[love(m, x)], derived from Mary loves him.

(ii) Relative clauses may contain bound variable pronouns:
a man [who loves a woman who loves him]: the bracketed relative clause is derived from a sentence such as he loves a woman who loves him, (or any other choice of variable).
The relative clause formation rule “abstracts on” x, adding who and deleting the first occurrence of he, and replaces the other occurrence of he by a pronoun of suitable gender. (Puzzle: at what stage of the derivation is the gender determinable?)

(iii) The Quantifying In rule causes an underlying he to be bound by a lambda operator. See Syntactic Derivations (i) and (ii) above: Exercise 1 of Homework 2 involves filling in the missing steps in Derivation (ii).

(iv) The Quantifying In rule can also create sentences that contain bound variable pronouns, for instance if we combine every professor with the (open) sentence He knows a student who admires him, the result will be Every professor knows a student who admires him. (See Homework 2, Exercises 6a and 7.)

“Free variable pronouns”: Some pronouns are like free variables with values assigned by context.

What should we say about “sentences” like He knows a student who admires him? One option is to call any “sentence” containing pseudo-pronouns like he, grammatical. Another option is to let such sentences be the sources for sentences with non-bound-variable pronouns like He knows a student who admires him or She knows a student who admires her. On this option, we assume that the discourse context must include an assignment function g indicating the “intended referent” of he or she, and we also assume that gender-marking reflects a presupposition about the gender of this intended referent. We will assume this latter hypothesis, at least for now. The truth conditions of such a sentence may very well depend on the choice of assignment function. Later we’ll discuss the difference between “demonstrative” uses of pronouns and what is sometimes called “discourse anaphora”, both of which are different from “bound variable anaphora” (Partee 1978)