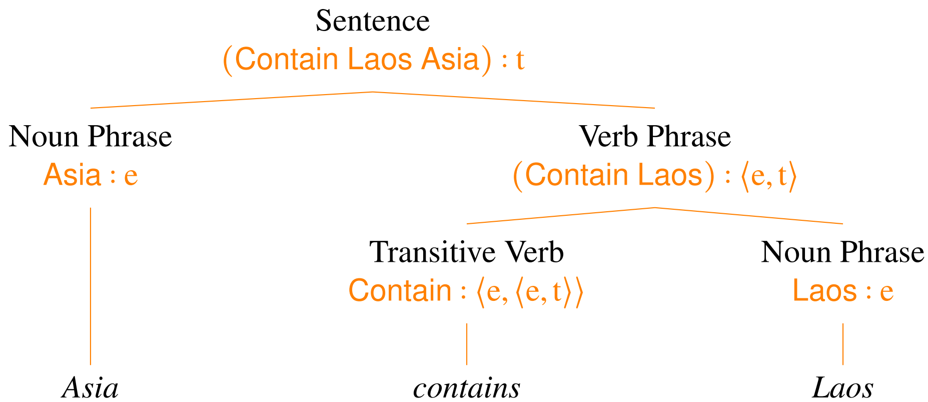
LING4400: Lecture Notes 10
Composition and Schematized Functions

Contents

Earlier we drew translation trees that were isomorphic to syntax, which translate directly to logic:



Today we preserve this isomorphism in conjunction and negation of incomplete propositions.

10.1 Schematized conjunction and disjunction [Partee & Rooth, 1983]

We often encounter conjunctions of incomplete propositions:

(1) a. *Etna erupts or is dormant.*
b. (entail and entailed by 1a:) *Etna erupts or Etna is dormant.*

You might think we could just copy the subject into the conjuncts (called **conjunction reduction**).

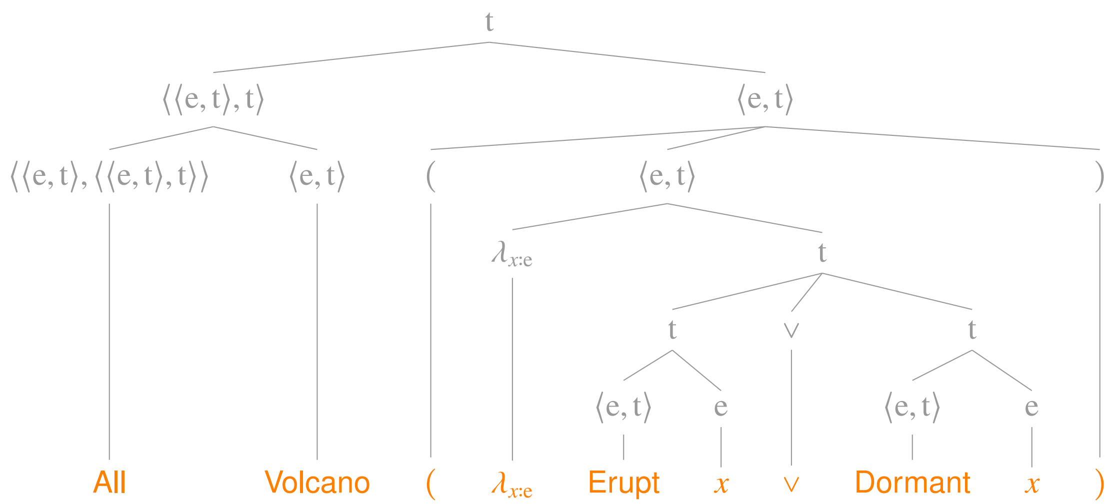
But that doesn't work with quantified subjects:

(2) a. *All volcanoes erupt or are dormant.*
b. (**not** entailed by 2a:) *All volcanoes erupt or all volcanoes are dormant.*

Sentence 2a is true if some volcanoes erupt and the rest are dormant; sentence 2b is not.

To model 2a, we need to get the disjunction inside the nuclear scope of *All*.

This can be done in a derivation tree:



but not in a translation tree, since no words in that sentence translate as lambda.

To do this we generalize across these types using **schemas**, defined with meta-variables γ and δ.
Specifically, we can allow functions to take any type γ_n with any number n of arguments $\delta_1, ..., \delta_n$:

$$\begin{aligned} \gamma_0 &= \mathsf{t} \\ \gamma_n &= \langle \delta_n, \gamma_{n-1} \rangle \end{aligned}$$

For example, using the second rule to substitute γ_2 and γ_1 and then the first rule to substitute γ_0:

$$\gamma_2 = \langle \delta_2, \gamma_1 \rangle = \langle \delta_2, \langle \delta_1, \gamma_0 \rangle \rangle = \langle \mathrm{e}, \langle \mathrm{e}, \mathrm{t} \rangle \rangle \quad (\text{if } \delta_1 = \mathrm{e} \text{ and } \delta_2 = \mathrm{e})$$

(These schematized types are also called **polymorphic**, because they have several forms.)

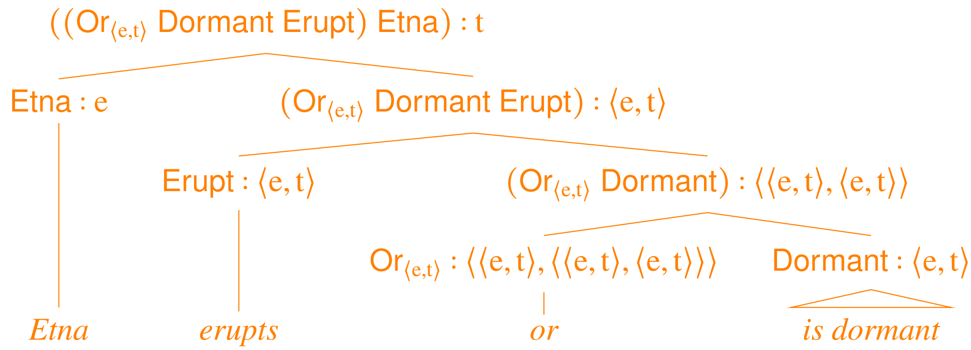
We can now define **schematized conjunction and disjunction** of type $\langle \gamma_n, \langle \gamma_n, \gamma_n \rangle \rangle$:

$$\begin{aligned} [\![\mathsf{And}_{\gamma_n}]\!]^M &= [\![\lambda_{f:\gamma_n}\ \lambda_{g:\gamma_n}\ \lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1}\ f\ x_n \ldots x_1 \wedge g\ x_n \ldots x_1]\!]^M \\ [\![\mathsf{Or}_{\gamma_n}]\!]^M &= [\![\lambda_{f:\gamma_n}\ \lambda_{g:\gamma_n}\ \lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1}\ f\ x_n \ldots x_1 \vee g\ x_n \ldots x_1]\!]^M \end{aligned}$$

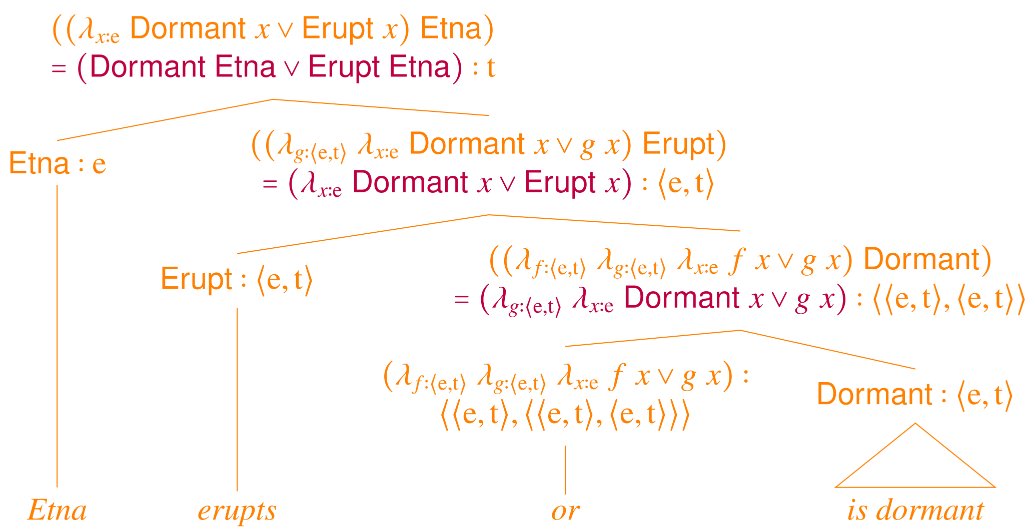
For example, if $n = 1$ and $\gamma_1 = \langle \mathrm{e}, \mathrm{t} \rangle$, we have a disjunction over an intransitive verb or verb phrase:

$$[\![\mathsf{Or}_{\langle \mathrm{e},\mathrm{t} \rangle}]\!]^M = [\![\lambda_{f:\langle \mathrm{e},\mathrm{t} \rangle}\ \lambda_{g:\langle \mathrm{e},\mathrm{t} \rangle}\ \lambda_{x_1:\mathrm{e}}\ f\ x_1 \vee g\ x_1]\!]^M$$

Here is an example translation, again isomorphic to syntax (read the translation off the top):

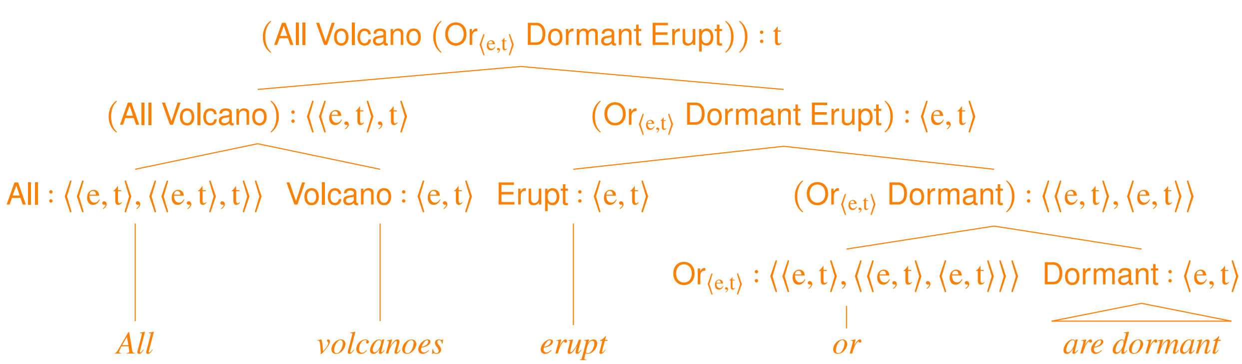


Example translation with variables (requires beta reduction):

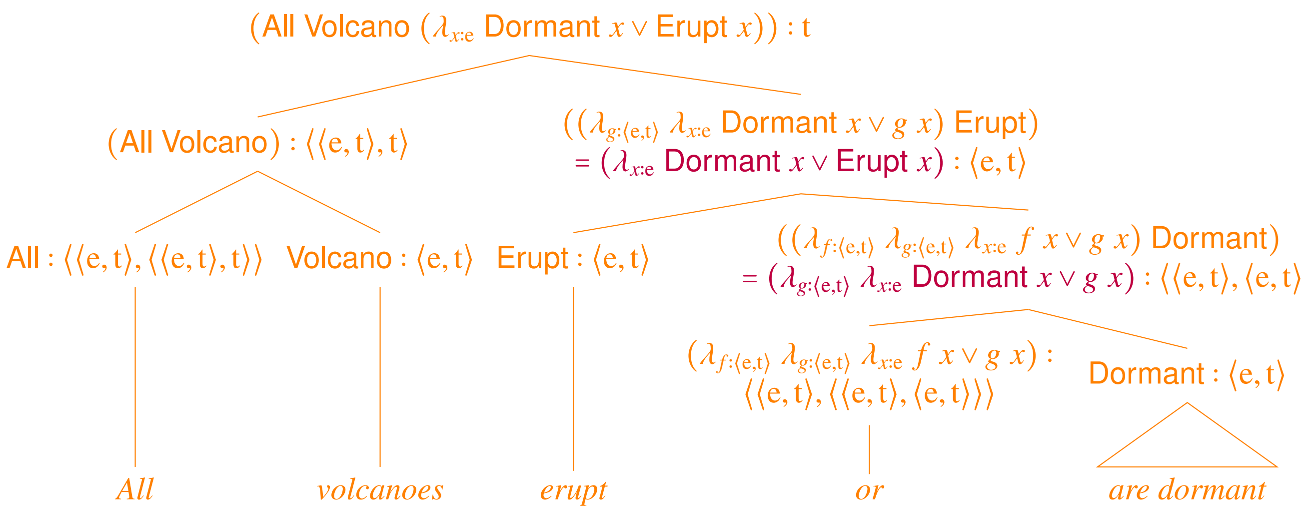


The translation $\mathsf{Dormant\ Etna} \vee \mathsf{Erupt\ Etna}$ is the same as for *Etna erupts or Etna is dormant*.

Here's the analysis for the quantified noun phrase:



And here's the example with variables (requires beta reduction):

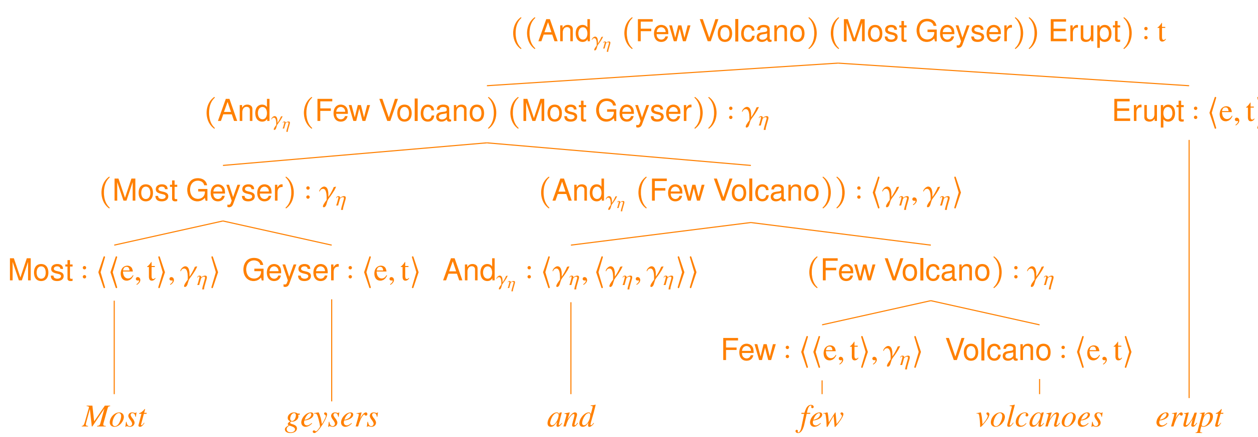


Note this is not All Volcano Erupt $\vee$ All Volcano Dormant – the disjunction is for each volcano.

Schematization also works for conjunctions of quantified noun phrases:

(3) a. *Most geysers and few volcanoes erupt.*
  b. (entail and entailed by 3a:) *Most geysers erupt and few volcanoes erupt.*

Derivation using schematized conjunction (where $\gamma_\eta = \langle\langle e, t\rangle, t\rangle$):



Practice 10.1: schematized function

Define a schematized And function for conjoining transitive verbs like *peel* and *eat* of type $\langle e, \langle e, t\rangle\rangle$.

10.2 Schematized negation

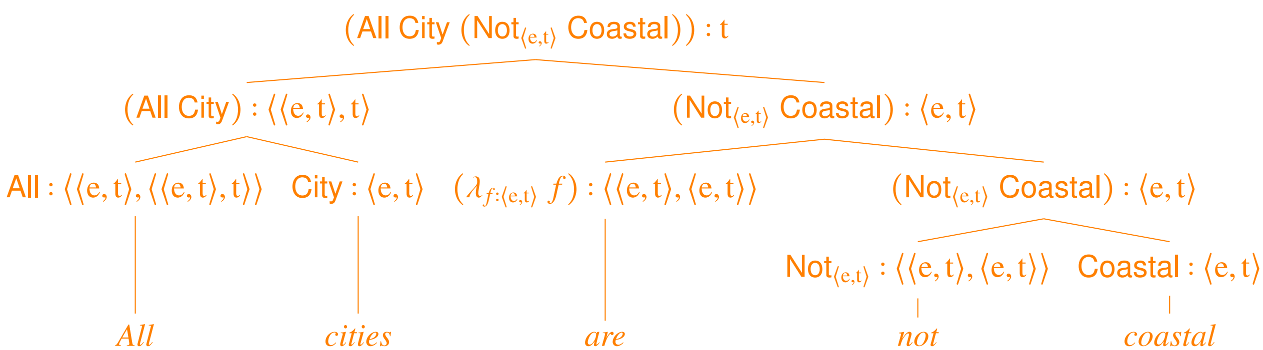
We also need schemas of type $\langle \gamma_n, \gamma_n \rangle$ in order to negate phrases which are not type t:

$$[\![\textsf{Not}_{\gamma_n}]\!]^M = [\![\lambda_{f:\gamma_n}\ \lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1} \neg\ (f\ x_n \ldots x_1)]\!]^M$$

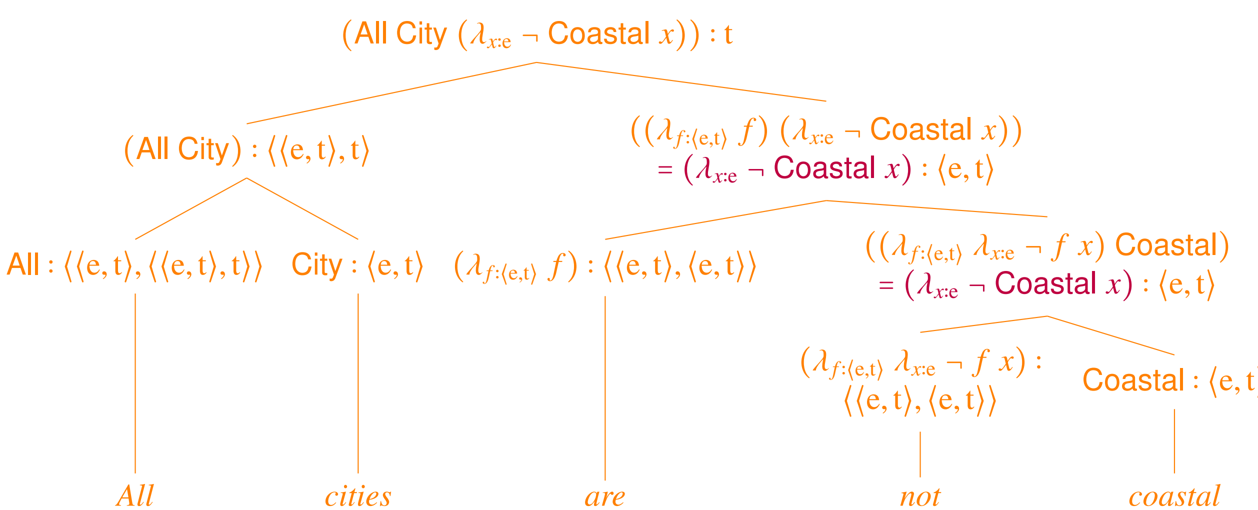
Here's an example schema definition:

$$[\![\textsf{Not}_{\langle e,t \rangle}]\!]^M = [\![\lambda_{f:\langle e,t\rangle}\ \lambda_{x_1:e} \neg\ (f\ x_1)]\!]^M$$

And here's the full translation, again isomorphic to syntax (read the translation off the top):



Example translation with variables (requires beta reduction):



Practice 10.2: schematized function

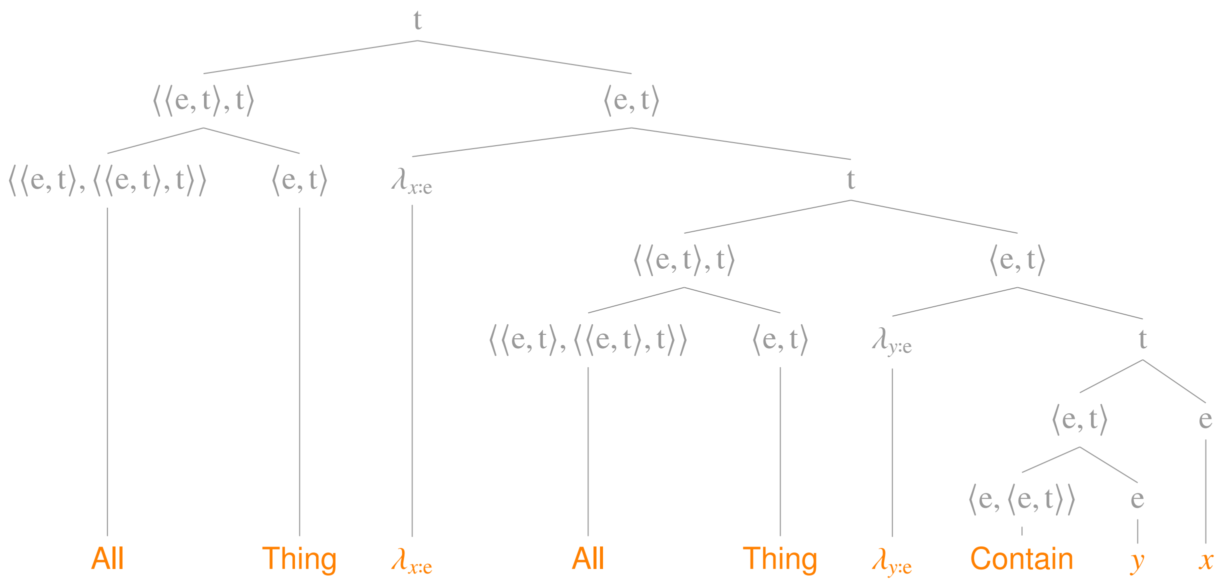
Define a schematized $\textsf{Not}_{\gamma_n}$ function that can combine with All.

Practice 10.3: tree drawing

Draw a translation tree for *Not all countries are coastal* using the above function.

10.3 Schematized quantifiers

We have a similar problem with quantifiers as syntactic objects – we can do this in derivations:



But we can't mark up a syntax tree this way and read off the translation – lambdas aren't words!

To translate by making derivations isomorphic to phrase structure trees, we can't use abstraction.

Subject/object quantifiers, which take intransitive/transitive arguments, must have different types.

A schematized quantifier of type $\langle\langle e,t\rangle,\langle\langle e,\gamma_n\rangle,\gamma_n\rangle\rangle$, can then be defined for any type γ_n:

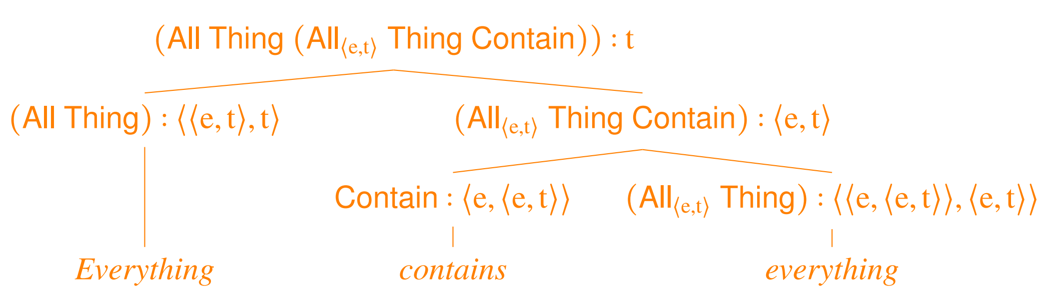
$$[\![\mathsf{All}_{\gamma_n}]\!]^M = [\![\lambda_{r:\langle e,t\rangle}\ \lambda_{s:\langle e,\gamma_n\rangle}\ \lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1}\ \mathsf{All}\ r\ (\lambda_{x_{n+1}:e}\ s\ x_{n+1} \ldots x_1)]\!]^M$$

It takes restriction r, nuclear scope s and 'extra' arguments $x_{1..n}$ and passes the extras along to s.

For example, if $n = 1$ and $\gamma_1 = \langle e,t\rangle$ we have a quantifier over a second argument (direct object):

$$[\![\mathsf{All}_{\langle e,t\rangle}]\!]^M = [\![\lambda_{r:\langle e,t\rangle}\ \lambda_{s:\langle e,\langle e,t\rangle\rangle}\ \lambda_{x_1:e}\ \mathsf{All}\ r\ (\lambda_{x_2:e}\ s\ x_2\ x_1)]\!]^M$$

Here it is in a translation, which is now isomorphic to syntax (read the translation off the top):



Practice 10.4: tree drawing

Draw a translation tree for *Everyone sent everyone everything,* using type $\langle e, \langle e, \langle e, t\rangle\rangle\rangle$ for *sent*.

Practice 10.5: translate English to logic

Translate the following into logic by drawing a tree with a logical expression at each branch:

Few people see a volcano.

References

[Partee & Rooth, 1983] Partee, B. & Rooth, M. (1983). Generalized conjunction and type ambiguity. In R. Bauerle, C. Schwarze, & A. von Stechow (Eds.), *Meaning, Use and Interpretation of Language* (pp. 361–383). Berlin: Walter de Gruyter.