LING4400: Lecture Notes 11
Modifiers

Contents

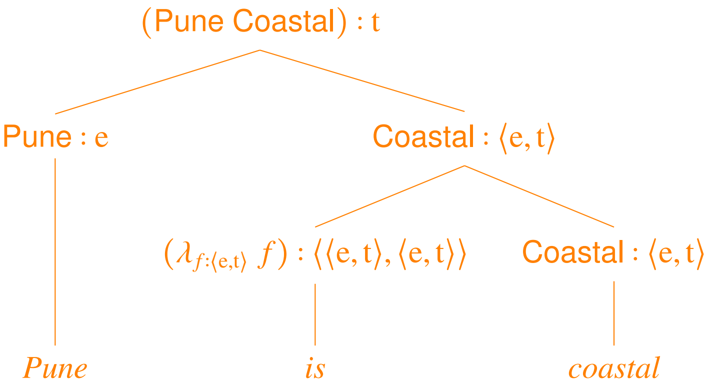
11.1 Modifiers

We need to model modifiers too: *Pune is a coastal city.*

Usually modifiers entail intersectively:

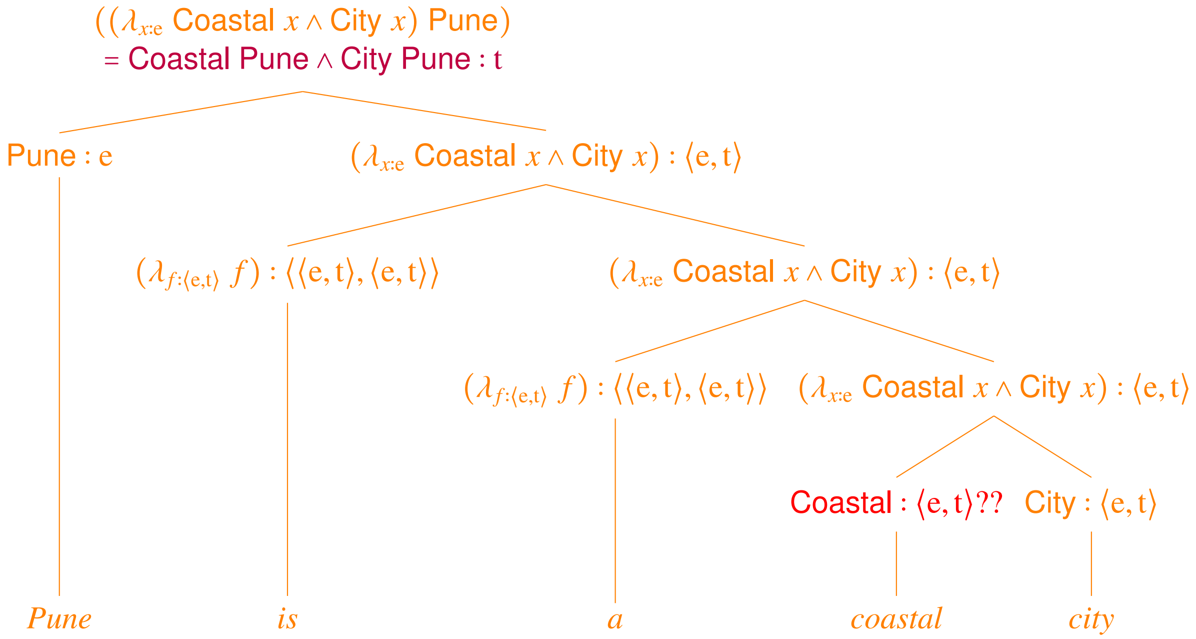
(1) a. *Pune is a coastal city.*
b. (entailed by 1a:) *Pune is a city.*
c. (entailed by 1a:) *Pune is coastal.*

In 1c the word *coastal* (which has the same meaning as in 1a) needs type $\langle \mathrm{e}, \mathrm{t} \rangle$ to fit with *is*:



(Pune Coastal) : t
Pune : e
Coastal : $\langle e, t\rangle$
$(\lambda_{f:\langle e,t\rangle}\ f) : \langle\langle e,t\rangle,\langle e,t\rangle\rangle$
Coastal : $\langle e, t\rangle$
Pune
is
coastal

But that type won't work for *coastal* when used as a modifier:



$((\lambda_{x:e}\ \mathsf{Coastal}\ x \wedge \mathsf{City}\ x)\ \mathsf{Pune})$
$= \mathsf{Coastal\ Pune} \wedge \mathsf{City\ Pune} : t$
$\mathsf{Pune} : e$
$(\lambda_{x:e}\ \mathsf{Coastal}\ x \wedge \mathsf{City}\ x) : \langle e, t \rangle$
$(\lambda_{f:\langle e,t \rangle}\ f) : \langle \langle e, t \rangle, \langle e, t \rangle \rangle$
$(\lambda_{x:e}\ \mathsf{Coastal}\ x \wedge \mathsf{City}\ x) : \langle e, t \rangle$
$(\lambda_{f:\langle e,t \rangle}\ f) : \langle \langle e, t \rangle, \langle e, t \rangle \rangle$
$(\lambda_{x:e}\ \mathsf{Coastal}\ x \wedge \mathsf{City}\ x) : \langle e, t \rangle$
$\mathsf{Coastal} : \langle e, t \rangle ??$
$\mathsf{City} : \langle e, t \rangle$
Pune
is
a
coastal
city

Here we have three options:

1. We could give it a function like this, which takes a set as input and returns a set as output:

$$(\lambda_{g:\langle e,t \rangle}\ \lambda_{x:e}\ \mathsf{Coastal}\ x \wedge g\ x) : \langle e, t \rangle$$

   But entailments like 1c are **productive** – they hold for many adjectives.

   That means it's unlikely language learners memorize this separate meaning for each word.

2. We can avoid this with a **lexical rule** defining modifier functions for (predicative) adjectives:

$$f : \langle e, t \rangle \quad \Rightarrow \quad (\lambda_{g:\langle e,t \rangle}\ \lambda_{x:e}\ f\ x \wedge g\ x) : \langle \langle e, t \rangle, \langle e, t \rangle \rangle$$

   But this increases the number of rules we need in any translation.

3. Alternatively we can define new (schematized) **translation rules** for modifier attachment:

$$f : \langle e, \gamma_n \rangle \quad g : \langle e, t \rangle \quad \Rightarrow \quad (\lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1}\ f\ x_n \ldots x_1 \wedge g\ x_1) : \langle e, \gamma_n \rangle \ \text{(Backward Modification)}$$
$$f : \langle e, t \rangle \quad g : \langle e, \gamma_n \rangle \quad \Rightarrow \quad (\lambda_{x_n:\delta_n} \ldots \lambda_{x_1:\delta_1}\ f\ x_1 \wedge g\ x_n \ldots x_1) : \langle e, \gamma_n \rangle \quad \text{(Forward Modification)}$$

   This replaces function application in translation without increasing the number of rules used.

In some sense the last solution is simpler, but all three of these produce the correct entailment.

Practice 11.1: trees with rules

Label each branch in the translation tree for the sentence *Pune is a coastal city* with a rule name (forward function application, backward function application, forward modification, backward modification).

11.2 Relative clauses

Relative clauses have the same intersective entailment as modifiers:

(2) a. *Pune is city that Asia contains.*
b. (entailed by 2a:) *Pune is a city.*
c. (entailed by 2a:) *Asia contains Pune.*

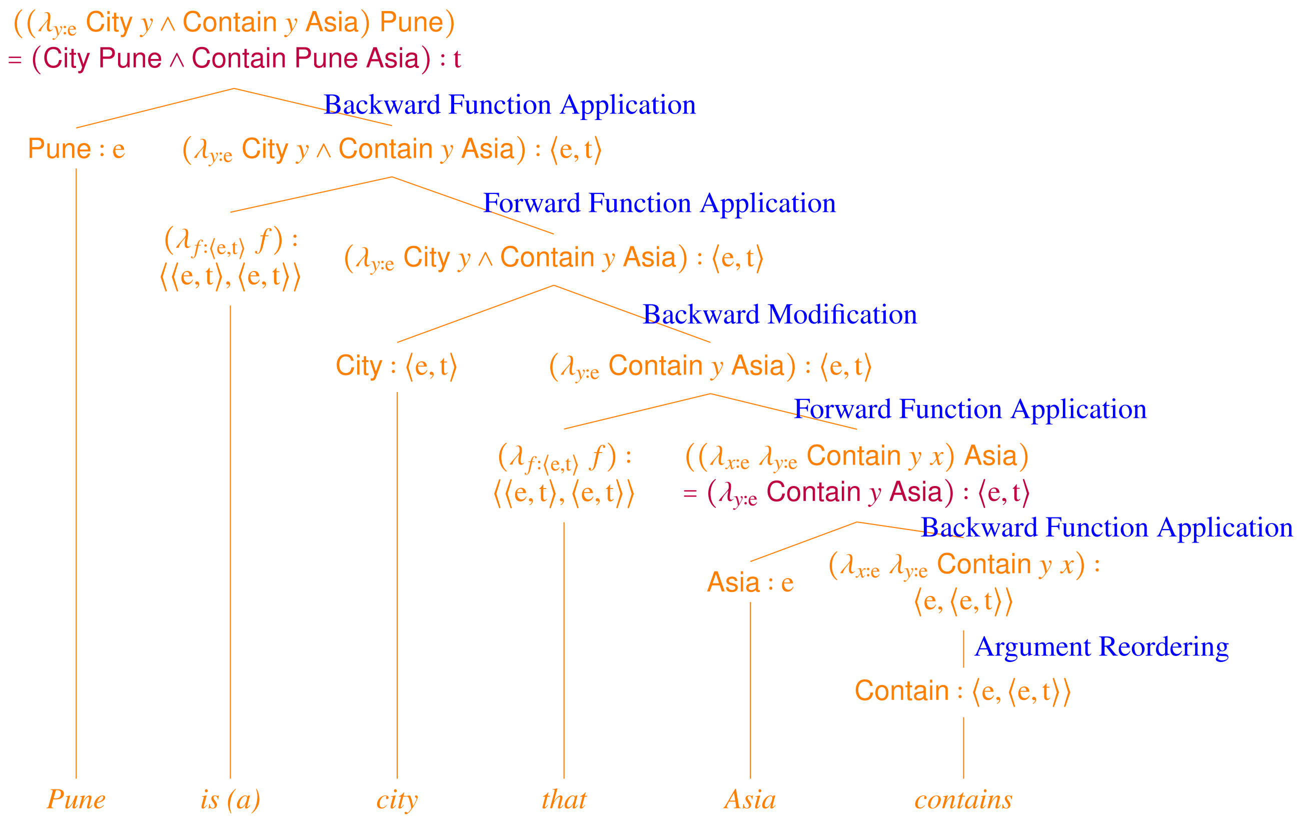
We can model (restrictive) relative clauses using this same composition rule.

But we'll also need a rule to account for the direct object in the relative clause.

We can model this as a change in the order of the arguments:

$$(\lambda_{y:\mathrm{e}}\ \lambda_{x:\mathrm{e}}\ f\ y\ x) : \langle \mathrm{e}, \langle \mathrm{e}, \mathrm{t}\rangle\rangle \quad \Rightarrow \quad (\lambda_{x:\mathrm{e}}\ \lambda_{y:\mathrm{e}}\ f\ y\ x) : \langle \mathrm{e}, \langle \mathrm{e}, \mathrm{t}\rangle\rangle \qquad \text{(Argument Re-ordering)}$$

Here's the translation, with translation rules labeled in blue:


$((\lambda_{y:\mathrm{e}}$ City $y \wedge$ Contain y Asia$)$ Pune$)$
$=$ (City Pune $\wedge$ Contain Pune Asia$)$: t
Backward Function Application
Pune : e
$(\lambda_{y:\mathrm{e}}$ City $y \wedge$ Contain y Asia$) : \langle \mathrm{e}, \mathrm{t}\rangle$
Forward Function Application
$(\lambda_{f:\langle \mathrm{e},\mathrm{t}\rangle}\ f) :$
$\langle\langle \mathrm{e}, \mathrm{t}\rangle, \langle \mathrm{e}, \mathrm{t}\rangle\rangle$
$(\lambda_{y:\mathrm{e}}$ City $y \wedge$ Contain y Asia$) : \langle \mathrm{e}, \mathrm{t}\rangle$
Backward Modification
City : $\langle \mathrm{e}, \mathrm{t}\rangle$
$(\lambda_{y:\mathrm{e}}$ Contain y Asia$) : \langle \mathrm{e}, \mathrm{t}\rangle$
Forward Function Application
$(\lambda_{f:\langle \mathrm{e},\mathrm{t}\rangle}\ f) :$
$\langle\langle \mathrm{e}, \mathrm{t}\rangle, \langle \mathrm{e}, \mathrm{t}\rangle\rangle$
$((\lambda_{x:\mathrm{e}}\ \lambda_{y:\mathrm{e}}$ Contain $y\ x)$ Asia$)$
$= (\lambda_{y:\mathrm{e}}$ Contain y Asia$) : \langle \mathrm{e}, \mathrm{t}\rangle$
Backward Function Application
Asia : e
$(\lambda_{x:\mathrm{e}}\ \lambda_{y:\mathrm{e}}$ Contain $y\ x) :$
$\langle \mathrm{e}, \langle \mathrm{e}, \mathrm{t}\rangle\rangle$
Argument Reordering
Contain : $\langle \mathrm{e}, \langle \mathrm{e}, \mathrm{t}\rangle\rangle$
Pune
is (a)
city
that
Asia
contains

Practice 11.2: trees with rules

(a) Draw a translation tree for the sentence *all cities that Peru built are coastal*.

(b) Label each branch in this translation tree with a rule name (forward function application, backward function application, forward modification, backward modification, argument re-ordering).

Practice 11.3: trees with rules

(a) Not all relative clauses need argument re-ordering. Draw a translation tree for the sentence *all countries that border Haiti are coastal*.

(b) Label each branch in this translation tree with a rule name (forward function application, backward function application, forward modification, backward modification, argument re-ordering).