LING4400: Study Guide 2

Practice 10.1: schematized function

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

$$\llbracket \text{And}_{\langle e, \langle e, t \rangle \rangle} \rrbracket^M = \llbracket \lambda_{f:\langle e, \langle e, t \rangle \rangle} \; \lambda_{g:\langle e, \langle e, t \rangle \rangle} \; \lambda_{x_2:e} \; \lambda_{x_1:e} \; f \; x_2 \; x_1 \vee g \; x_2 \; x_1 \rrbracket^M$$
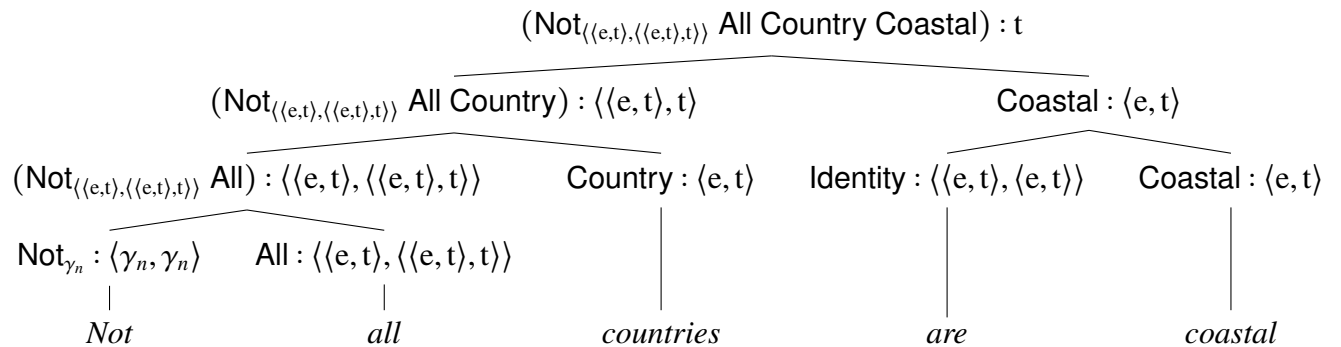
Practice 10.2: schematized function

Define a schematized Not_{γ_n} function that can combine with All.

$$\llbracket \text{Not}_{\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle} \rrbracket^M = \llbracket \lambda_{f:\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle} \; \lambda_{x_2:\langle e, t \rangle} \; \lambda_{x_1:\langle e, t \rangle} \; \neg \; f \; x_2 \; x_1 \rrbracket^M$$
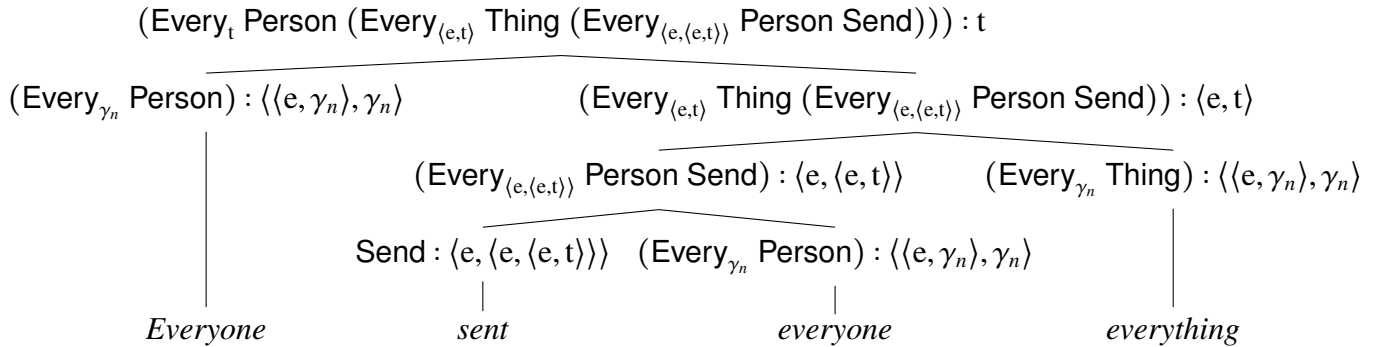
Practice 10.3: tree drawing

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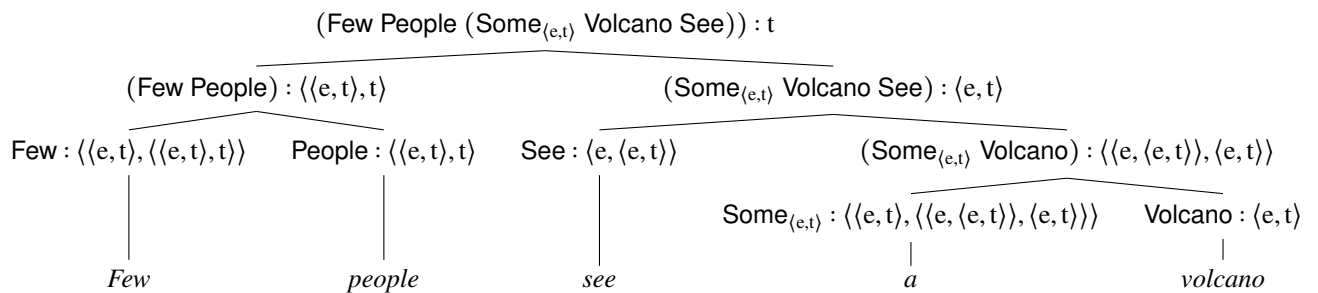

Practice 10.4: tree drawing

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

$(\text{Every}_t \text{ Person } (\text{Every}_{\langle e,t\rangle} \text{ Thing } (\text{Every}_{\langle e,\langle e,t\rangle\rangle} \text{ Person Send}))) : t$

$(\text{Every}_{\gamma_n} \text{ Person}) : \langle\langle e, \gamma_n\rangle, \gamma_n\rangle$ $\qquad$ $(\text{Every}_{\langle e,t\rangle} \text{ Thing } (\text{Every}_{\langle e,\langle e,t\rangle\rangle} \text{ Person Send})) : \langle e, t\rangle$

$(\text{Every}_{\langle e,\langle e,t\rangle\rangle} \text{ Person Send}) : \langle e, \langle e, t\rangle\rangle$ $\qquad$ $(\text{Every}_{\gamma_n} \text{ Thing}) : \langle\langle e, \gamma_n\rangle, \gamma_n\rangle$

$\text{Send} : \langle e, \langle e, \langle e, t\rangle\rangle\rangle$ $\quad$ $(\text{Every}_{\gamma_n} \text{ Person}) : \langle\langle e, \gamma_n\rangle, \gamma_n\rangle$

Everyone $\qquad$ *sent* $\qquad$ *everyone* $\qquad$ *everything*

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.

$(\text{Few People } (\text{Some}_{\langle e,t\rangle} \text{ Volcano See})) : t$

$(\text{Few People}) : \langle\langle e, t\rangle, t\rangle$ $\qquad$ $(\text{Some}_{\langle e,t\rangle} \text{ Volcano See}) : \langle e, t\rangle$

$\text{Few} : \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$ $\quad$ $\text{People} : \langle\langle e, t\rangle, t\rangle$ $\quad$ $\text{See} : \langle e, \langle e, t\rangle\rangle$ $\qquad$ $(\text{Some}_{\langle e,t\rangle} \text{ Volcano}) : \langle\langle e, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$

$\text{Some}_{\langle e,t\rangle} : \langle\langle e, t\rangle, \langle\langle e, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$ $\quad$ $\text{Volcano} : \langle e, t\rangle$

Few $\qquad$ *people* $\qquad$ *see* $\qquad$ *a* $\qquad$ *volcano*

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).

$$((\lambda_{x:e} \text{ Coastal } x \wedge \text{City } x) \text{ Pune})$$
$$= \text{Coastal Pune} \wedge \text{City Pune} : t$$

Backward Function Application

$\text{Pune} : e$ $(\lambda_{x:e} \text{ Coastal } x \wedge \text{City } x) : \langle e, t \rangle$
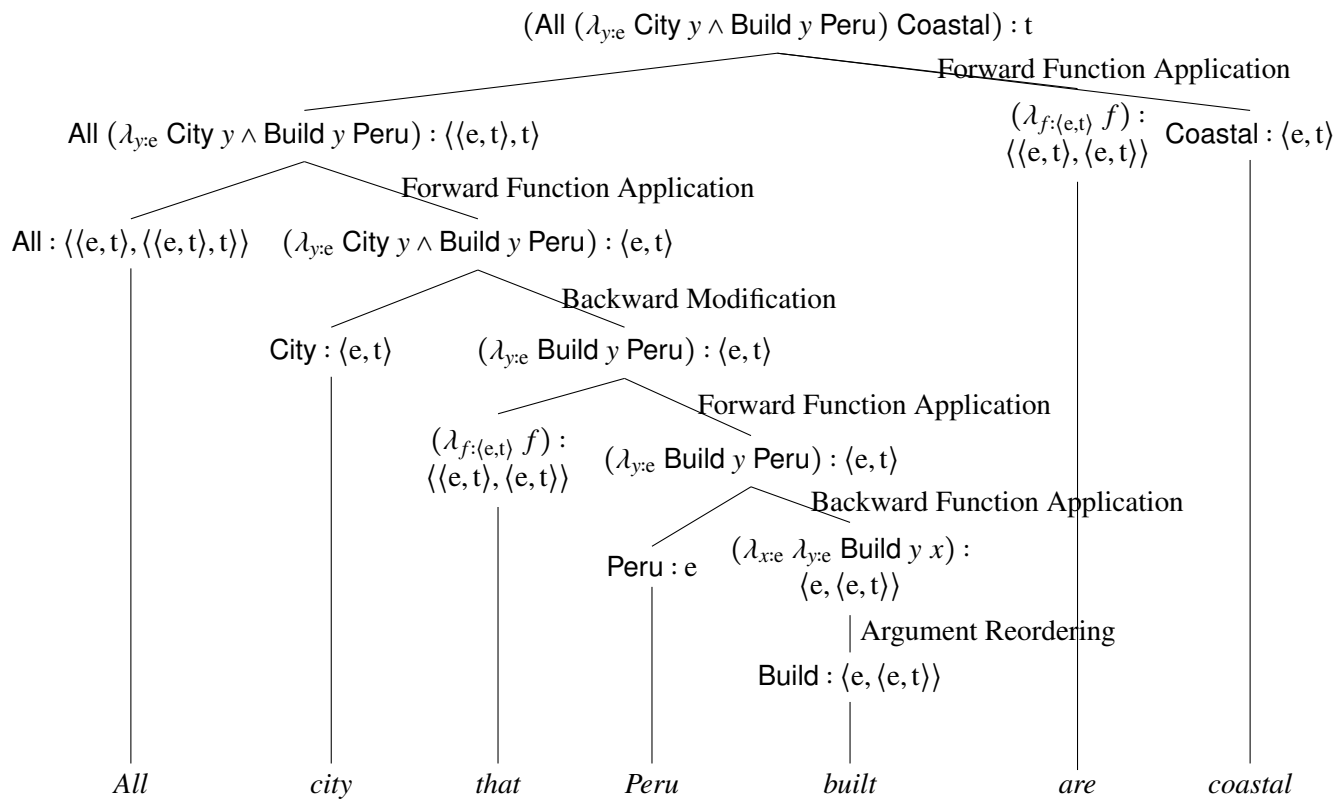
Forward Function Application

$(\lambda_{f:\langle e,t \rangle} f) : \langle \langle e, t \rangle, \langle e, t \rangle \rangle$ $(\lambda_{x:e} \text{ Coastal } x \wedge \text{City } x) : \langle e, t \rangle$

Forward Function Application

$(\lambda_{f:\langle e,t \rangle} f) : \langle \langle e, t \rangle, \langle e, t \rangle \rangle$ $(\lambda_{x:e} \text{ Coastal } x \wedge \text{City } x) : \langle e, t \rangle$

Forward Modification

$\text{Coastal} : \langle e, t \rangle$ $\text{City} : \langle e, t \rangle$

*Pune*   *is*   *a*   *coastal*   *city*
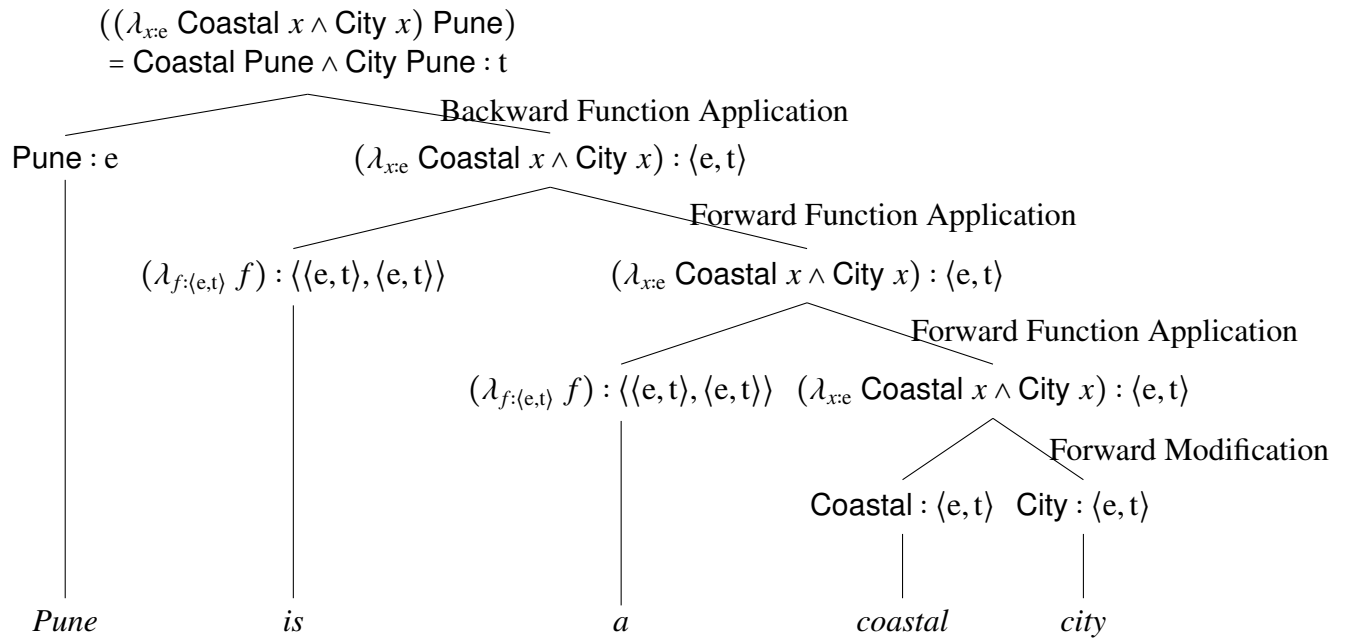
Practice 11.2: trees with rules

(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).

$(\text{All } (\lambda_{y:e} \text{ City } y \wedge \text{Build } y \text{ Peru}) \text{ Coastal}) : t$

Forward Function Application

$\text{All } (\lambda_{y:e} \text{ City } y \wedge \text{Build } y \text{ Peru}) : \langle\langle e, t\rangle, t\rangle$

$(\lambda_{f:\langle e,t\rangle} f) : \langle\langle e, t\rangle, \langle e, t\rangle\rangle$ $\text{Coastal} : \langle e, t\rangle$

Forward Function Application

$\text{All} : \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$ $(\lambda_{y:e} \text{ City } y \wedge \text{Build } y \text{ Peru}) : \langle e, t\rangle$

Backward Modification

$\text{City} : \langle e, t\rangle$ $(\lambda_{y:e} \text{ Build } y \text{ Peru}) : \langle e, t\rangle$

Forward Function Application

$(\lambda_{f:\langle e,t\rangle} f) : \langle\langle e, t\rangle, \langle e, t\rangle\rangle$ $(\lambda_{y:e} \text{ Build } y \text{ Peru}) : \langle e, t\rangle$

Backward Function Application

$\text{Peru} : e$ $(\lambda_{x:e} \lambda_{y:e} \text{ Build } y \text{ } x) : \langle e, \langle e, t\rangle\rangle$

Argument Reordering

$\text{Build} : \langle e, \langle e, t\rangle\rangle$

All   *city*   *that*   *Peru*   *built*   *are*   *coastal*

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).
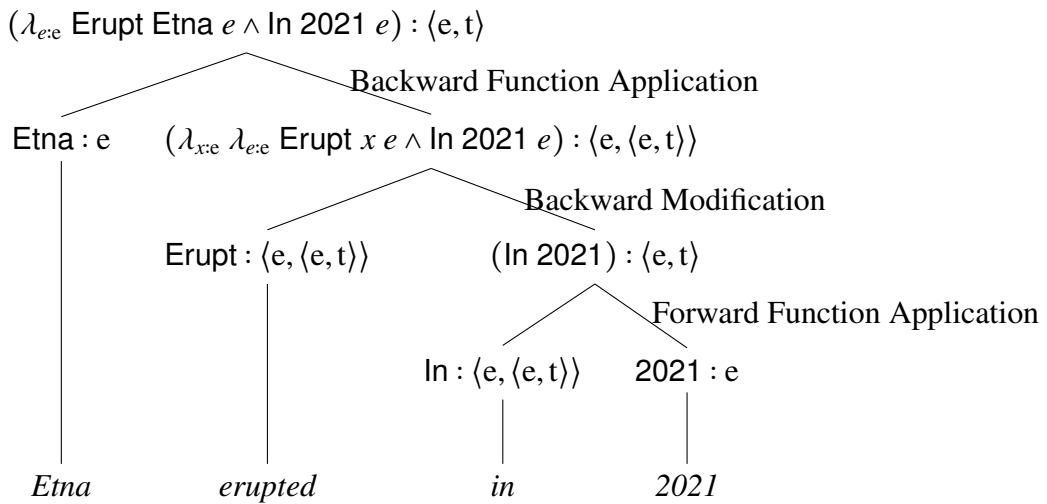
4

(All ($\lambda_{y:e}$ Country $y \wedge$ Border Haiti y) Coastal) : t

Forward Function Application

All ($\lambda_{y:e}$ Country $y \wedge$ Border Haiti y) : $\langle\langle e,t\rangle, t\rangle$

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

Forward Function Application

All : $\langle\langle e,t\rangle, \langle\langle e,t\rangle, t\rangle\rangle$ ($\lambda_{y:e}$ Country $y \wedge$ Border Haiti y) : $\langle e,t\rangle$

Backward Modification

Country : $\langle e,t\rangle$ (Border Haiti) : $\langle e,t\rangle$

Forward Function Application

($\lambda_{f:\langle e,t\rangle}$ f) : $\langle\langle e,t\rangle, \langle e,t\rangle\rangle$ (Border Haiti) : $\langle e,t\rangle$

Forward Function Application

Border : $\langle e, \langle e,t\rangle\rangle$ Haiti : e

All   *countries*   *that*   *border*   *Haiti*   *are*   *coastal*
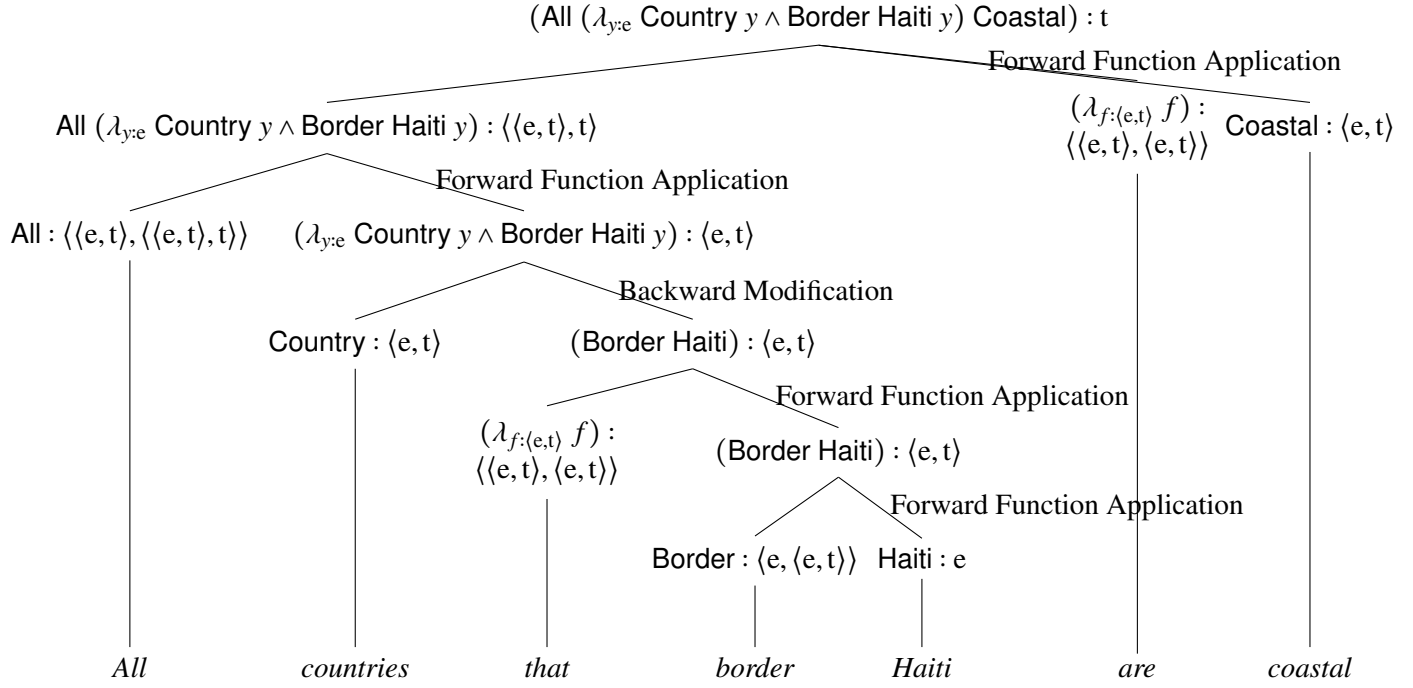
Practice 12.1: trees with rules

Label the initial tree for *Etna erupted in 2021* with rules.

($\lambda_{e:e}$ Erupt Etna $e \wedge$ In 2021 e) : $\langle e,t\rangle$

Backward Function Application

Etna : e   ($\lambda_{x:e}$ $\lambda_{e:e}$ Erupt x $e \wedge$ In 2021 e) : $\langle e, \langle e,t\rangle\rangle$

Backward Modification

Erupt : $\langle e, \langle e,t\rangle\rangle$ (In 2021) : $\langle e,t\rangle$

Forward Function Application

In : $\langle e, \langle e,t\rangle\rangle$ 2021 : e

Etna   *erupted*   *in*   *2021*

Practice 12.2: trees with rules

Label the complete tree for *Etna erupted in 2021* with rules.

(Some ($\lambda_{e:e}$ Erupt Etna $e \wedge$ In 2021 e) ($\lambda_{e:e}$ True)) : t

　　|　Nuclear Scope Closure

(Some ($\lambda_{e:e}$ Erupt Etna $e \wedge$ In 2021 e)) : $\langle\langle e, t\rangle, t\rangle$

　　|　Existential Closure

($\lambda_{e:e}$ Erupt Etna $e \wedge$ In 2021 e) : $\langle e, t\rangle$

　　　　　Backward Function Application

Etna : e　　($\lambda_{x:e}\ \lambda_{e:e}$ Erupt $x\ e \wedge$ In 2021 e) : $\langle e, \langle e, t\rangle\rangle$

　　　　　　Backward Modification

Erupt : $\langle e, \langle e, t\rangle\rangle$　　(In 2021) : $\langle e, t\rangle$

　　　　　　　Forward Function Application

In : $\langle e, \langle e, t\rangle\rangle$　　2021 : e

Etna　　*erupted*　　*in*　　*2021*
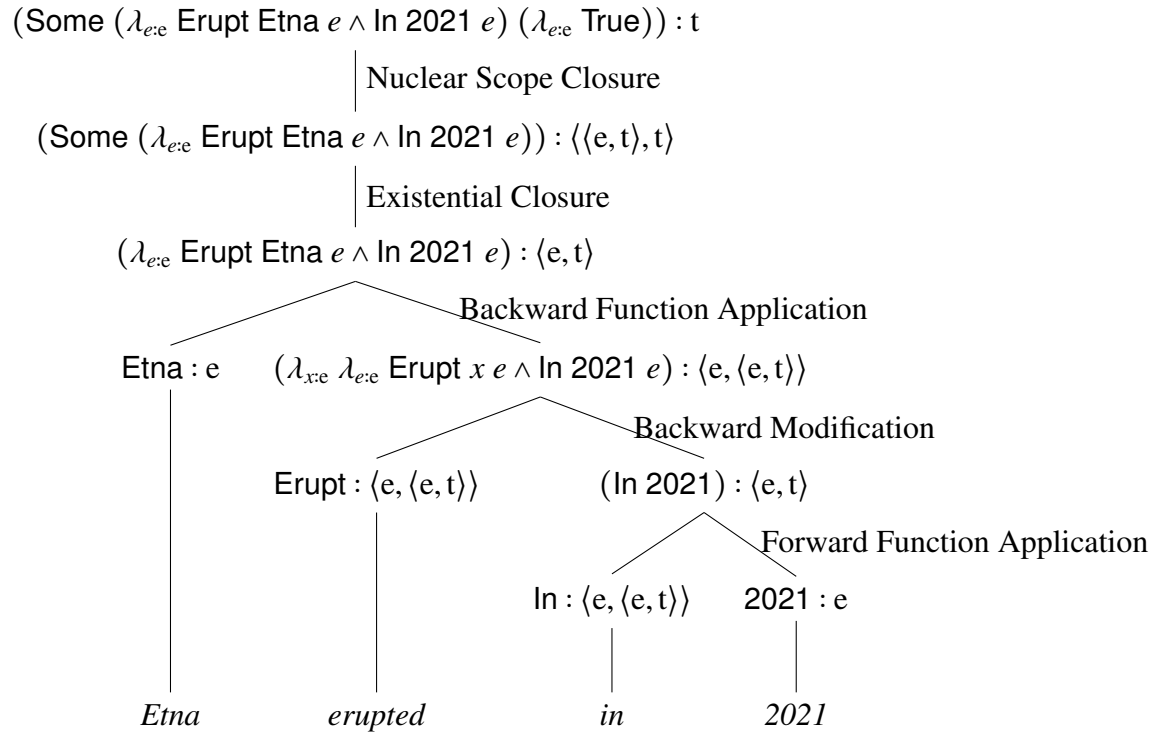
Practice 12.3: trees with rules

Label the tree for *Etna erupted after Wolf erupted* with rules.

$(\text{Some } (\lambda_{e:e} \text{ Erupt Etna } e \wedge (\text{Some}_{\langle e,t\rangle} (\text{Erupt Wolf}) \text{ After}) \, e) \, (\lambda_{e:e} \text{ True})) : t$

　Nuclear Scope Closure

$(\text{Some } (\lambda_{e:e} \text{ Erupt Etna } e \wedge (\text{Some}_{\langle e,t\rangle} (\text{Erupt Wolf}) \text{ After}) \, e)) : \langle\langle e,t\rangle, t\rangle$

　Existential Closure

$(\lambda_{e:e} \text{ Erupt Etna } e \wedge (\text{Some}_{\langle e,t\rangle} (\text{Erupt Wolf}) \text{ After}) \, e) : \langle e,t\rangle$

　Backward Modification

$(\text{Erupt Etna}) : \langle e,t\rangle$　　　$(\text{Some}_{\langle e,t\rangle} (\text{Erupt Wolf}) \text{ After}) : \langle e,t\rangle$

BFA　　　　　　　　　　　　BFA

$\text{Etna} : e$　$\text{Erupt} : \langle e, \langle e,t\rangle\rangle$　$\text{After} : \langle e, \langle e,t\rangle\rangle$　$(\text{Some}_{\langle e,t\rangle} (\text{Erupt Wolf})) : \langle\langle e, \langle e,t\rangle\rangle, \langle e,t\rangle\rangle$

　Existential closure

$(\text{Erupt Wolf}) : \langle e,t\rangle$

BFA

$\text{Wolf} : e$　$\text{Erupt} : \langle e, \langle e,t\rangle\rangle$

Etna　　　*erupted*　　　*after*　　　*Wolf*　　　*erupted*

Practice 13.1:

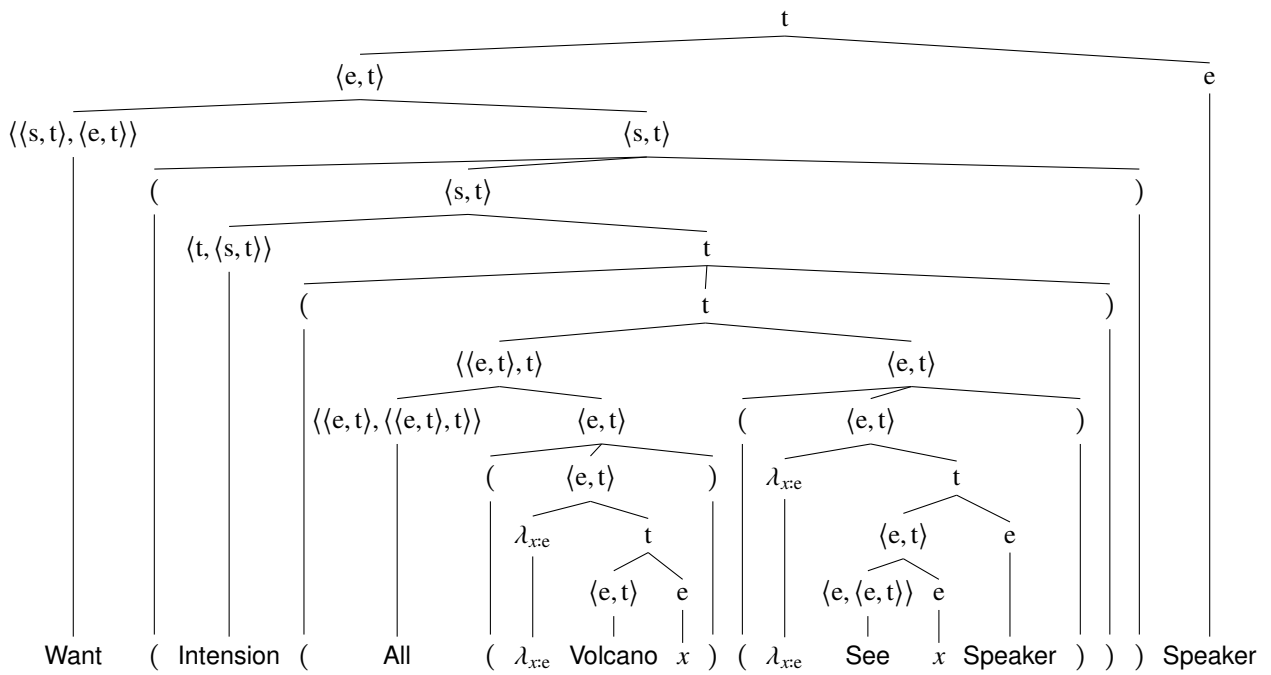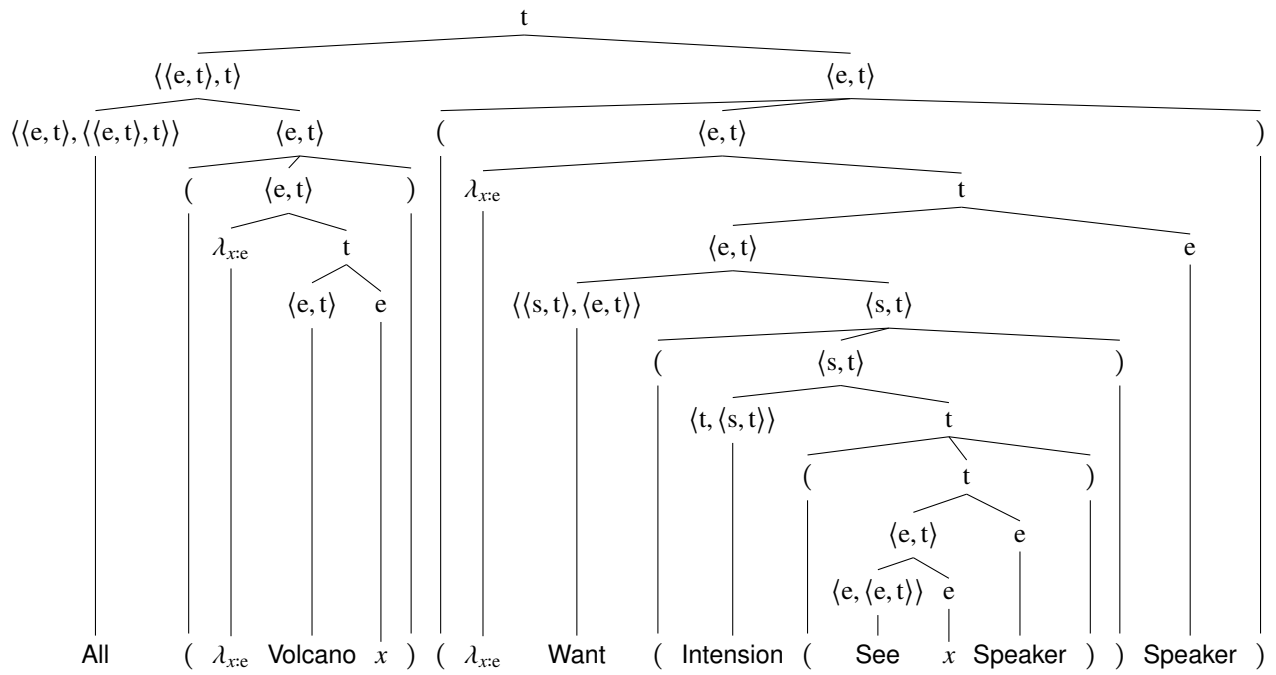Write **logical translations** that distinguish the following sentences:

1. *France believes it's not true that Etna erupted twice.*

2. *It's not true that France believes Etna erupted twice.*

---

1. $\text{Some } (\text{Believe } (\text{Intension } (\text{Not } (\text{Two } (\text{Erupt Etna}) \, (\lambda_{e:e} \text{ True})))))\, \text{France}) \, (\lambda_{e:e} \text{ True})$

2. $\text{Not } (\text{Some } (\text{Believe } (\text{Intension } (\text{Two } (\text{Erupt Etna}) \, (\lambda_{e:e} \text{ True})))\, \text{France}) \, (\lambda_{e:e} \text{ True}))$

Practice 13.2:

Draw **derivation trees** (with just types at each branch) for the above expressions:

1. $\text{All } (\lambda_{x:e} \text{ Volcano } x) \, (\lambda_{x:e} \text{ Want } (\text{Intension } (\text{See } x \text{ Speaker})) \text{ Speaker})$

2. $\text{Want } (\text{Intension } (\text{All } (\lambda_{x:e} \text{ Volcano } x) \, (\lambda_{x:e} \text{ See } x \text{ Speaker}))) \text{ Speaker}$

Tree 1 (top):

t
⟨⟨e, t⟩, t⟩   ⟨e, t⟩
⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩⟩   ⟨e, t⟩   (⟨e, t⟩   )
(⟨e, t⟩   )   λ_{x:e}   t
λ_{x:e}   t   ⟨e, t⟩   e
⟨e, t⟩   e   ⟨⟨s, t⟩, ⟨e, t⟩⟩   ⟨s, t⟩
(⟨s, t⟩   )
⟨t, ⟨s, t⟩⟩   t
(t   )
⟨e, t⟩   e
⟨e, ⟨e, t⟩⟩   e

Terminals: All   (λ_{x:e}   Volcano   x   )   (λ_{x:e}   Want   (Intension   (See   x   Speaker   )   )   Speaker   )

Tree 2 (bottom):

t
⟨e, t⟩   e
⟨⟨s, t⟩, ⟨e, t⟩⟩   ⟨s, t⟩
(⟨s, t⟩   )
⟨t, ⟨s, t⟩⟩   t
(t   )
⟨⟨e, t⟩, t⟩   ⟨e, t⟩
⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩⟩   ⟨e, t⟩   (⟨e, t⟩   )
(⟨e, t⟩   )   λ_{x:e}   t
λ_{x:e}   t   ⟨e, t⟩   e
⟨e, t⟩   e   ⟨e, ⟨e, t⟩⟩   e

Terminals: Want   (Intension   (All   (λ_{x:e}   Volcano   x   )   (λ_{x:e}   See   x   Speaker   )   )   )   Speaker
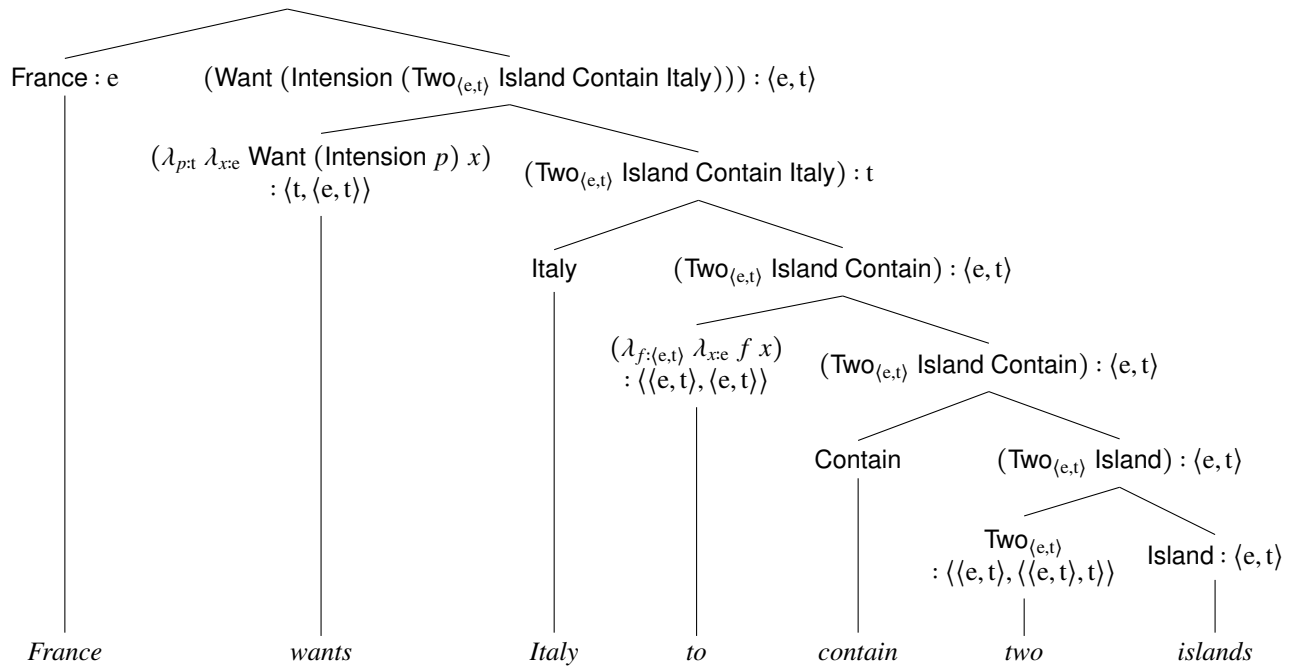
8

Practice 13.3:

Write an **English translation** of the following logical form (with no eventualities) and draw a **derivation tree with a logical form at each branch** for your translation:

$$\text{Want (Intension (Two}_{\langle e,t\rangle} \text{ Island Contain Italy)) France}$$

You may assume the following expression for the word *want*:

$$\lambda_{p:t}\, \lambda_{x:e}\, \text{Want (Intension } p)\, x$$

(Want (Intension (Two$_{\langle e,t\rangle}$ Island Contain Italy)) France) : t

France : e   (Want (Intension (Two$_{\langle e,t\rangle}$ Island Contain Italy))) : $\langle e,t\rangle$

$(\lambda_{p:t}\, \lambda_{x:e}\, \text{Want (Intension } p)\, x)$: $\langle t, \langle e,t\rangle\rangle$ (Two$_{\langle e,t\rangle}$ Island Contain Italy) : t

Italy   (Two$_{\langle e,t\rangle}$ Island Contain) : $\langle e,t\rangle$

$(\lambda_{f:\langle e,t\rangle}\, \lambda_{x:e}\, f\, x)$: $\langle\langle e,t\rangle, \langle e,t\rangle\rangle$ (Two$_{\langle e,t\rangle}$ Island Contain) : $\langle e,t\rangle$

Contain   (Two$_{\langle e,t\rangle}$ Island) : $\langle e,t\rangle$

Two$_{\langle e,t\rangle}$: $\langle\langle e,t\rangle, \langle\langle e,t\rangle, t\rangle\rangle$ Island : $\langle e,t\rangle$

France   *wants*   *Italy*   *to*   *contain*   *two*   *islands*

Practice 14.1:

Which of the following are valid entailments?

1. *Etna erupts and Wolf erupts*, so *Etna erupts*.

2. *Etna erupts or Wolf erupts*, so *Etna erupts*.

3. *If Etna erupts then Wolf erupts*, and *Etna erupts* so *Wolf erupts*.

1. Valid (because of conjunction elimination)

2. Not valid (disjunction may be satisfied by other disjunct)

9

3. Valid (because of modus ponens)

Practice 14.2:

Classify the following as right upward or right downward entailing or neither:

1. *at least seven*
2. *at most seven*
3. *exactly seven*
4. *most*

1. Right upward (if *at least seven volcanoes erupt now* then *at least seven volcanoes erupt*)
2. Right downward (if *at most seven volcanoes erupt* then *at most seven volcanoes erupt now*)
3. Neither (no test entailments hold)
4. Right upward (if *most volcanoes erupt now* then *most volcanoes erupt*)

Practice 14.3:

Classify the following as left upward or left downward entailing or neither:

1. *at least seven*
2. *at most seven*
3. *exactly seven*
4. *most*

1. Left upward (if *at least seven coastal volcanoes erupt* then *at least seven volcanoes erupt*)
2. Left downward (if *at most seven volcanoes erupt* then *at most seven coastal volcanoes erupt*)
3. Neither (no test entailments hold)
4. Neither (no test entailments hold)

Practice 14.4:

Which of the following are valid entailments?

1. *Two volcanoes erupted*, so *Two coastal volcanoes erupted*.
2. *Two coastal volcanoes erupted*, so *Two volcanoes erupted*.

1. Not valid (the volcanoes may be inland)

2. Valid (*[at least] two* is left upward entailing)

Practice 14.5:

Do the following words behave like negative polarity markers?

1. *at all*

2. *usually*

---

1. Yes (*No volcanoes erupt at all* is more grammatical than *Some volcanoes erupt at all*)

2. No (*No volcanoes erupt usually* is not more grammatical than *Some volcanoes erupt usually*)

Practice 15.1: trees with sequents

Draw **a derivation tree with logical sequents at each branch** for the phrase:

each country

in which *each country* undergoes storage.

$$(\vdash (\mathsf{All\ Country}) : \langle\langle e, t\rangle, t\rangle,\ y : e) \vdash y : e$$
$$|$$
$$\vdash (\mathsf{All\ Country}) : \langle\langle e, t\rangle, t\rangle$$
$$\vdash \mathsf{All} : \langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle \quad \vdash \mathsf{Country} : \langle e, t\rangle$$
$$|\qquad\qquad\qquad\qquad |$$
$$each \qquad\qquad\qquad country$$
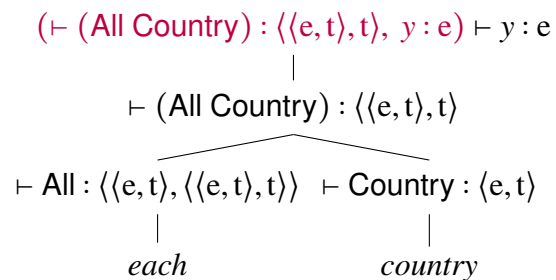
Practice 15.2: trees with sequents

Draw **a derivation tree with logical sequents at each branch** for the following sentence:

A city in each country is coastal.

in which *each country* is scoped high.

$\vdash (\text{All Country } (\lambda_{y:e} \text{ Some } (\lambda_{x:e} \text{ City } x \wedge \text{In } y \, x) \text{ Coastal})) : t$

$(\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle, \ y:e) \vdash (\text{Some } (\lambda_{x:e} \text{ City } x \wedge \text{In } y \, x) \text{ Coastal}) : t$

$(\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle, \ y:e) \vdash (\text{Some } (\lambda_{x:e} \text{ City } x \wedge \text{In } y \, x) : \langle\langle e,t\rangle,t\rangle$

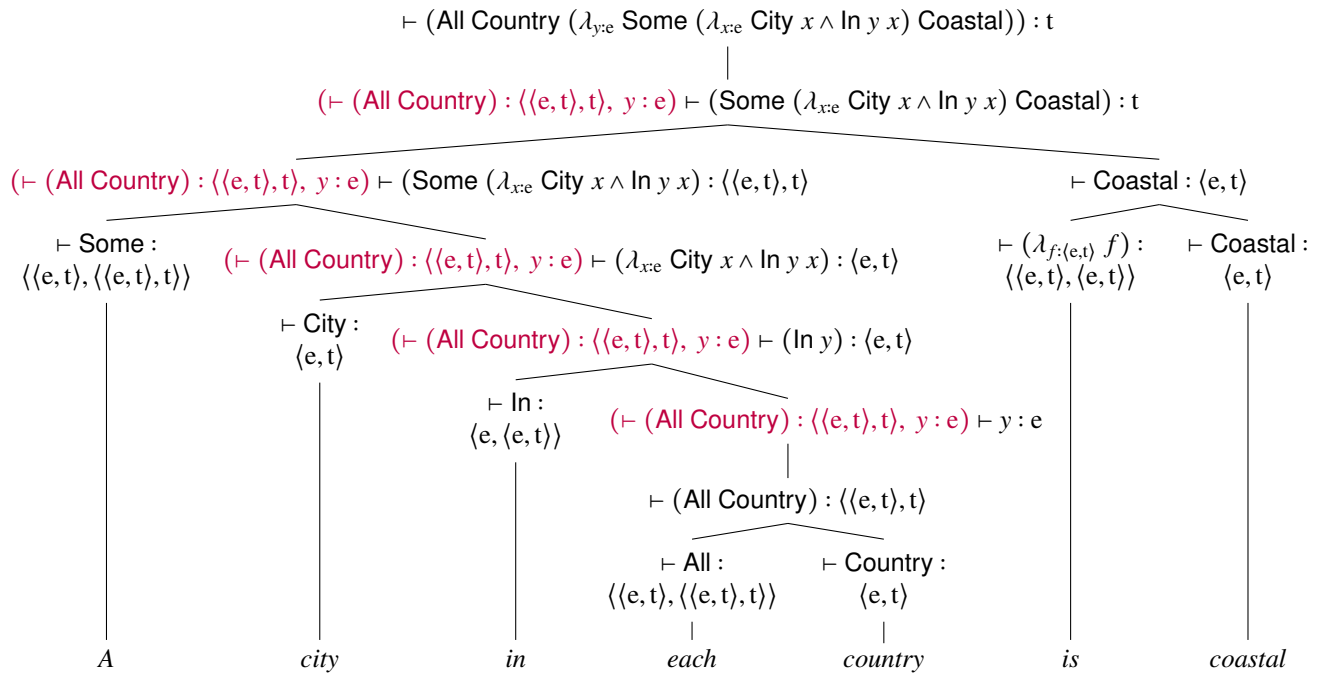$\vdash \text{Coastal} : \langle e,t\rangle$

$\vdash \text{Some} :$
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

$(\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle, \ y:e) \vdash (\lambda_{x:e} \text{ City } x \wedge \text{In } y \, x) : \langle e,t\rangle$

$\vdash (\lambda_{f:\langle e,t\rangle} f) :$
$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

$\vdash \text{Coastal} :$
$\langle e,t\rangle$

$\vdash \text{City} :$
$\langle e,t\rangle$

$(\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle, \ y:e) \vdash (\text{In } y) : \langle e,t\rangle$

$\vdash \text{In} :$
$\langle e,\langle e,t\rangle\rangle$

$(\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle, \ y:e) \vdash y : e$

$\vdash (\text{All Country}) : \langle\langle e,t\rangle,t\rangle$

$\vdash \text{All} :$
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

$\vdash \text{Country} :$
$\langle e,t\rangle$

A   *city*   *in*   *each*   *country*   *is*   *coastal*

Practice 15.3: rule labeling

Label the **rules** in the above tree for *Two volcanoes erupted*.

12

$\vdash (\text{Two Volcano } (\lambda_{x:e} \text{ Some } (\text{Erupt } x) \ (\lambda_{e:e} \text{ True}))) : t$

$\Big|$ Quantifier Retrieval

$(\vdash (\text{Two Volcano}) : \langle\langle e,t\rangle,t\rangle, \ x:e) \vdash (\text{Some } (\text{Erupt } x) \ (\lambda_{e:e} \text{ True})) : t$

$\Big|$ Nuclear Scope Closure

$(\vdash (\text{Two Volcano}) : \langle\langle e,t\rangle,t\rangle, \ x:e) \vdash (\text{Some } (\text{Erupt } x)) : \langle\langle e,t\rangle,t\rangle$

$\Big|$ Existential Closure

$(\vdash (\text{Two Volcano}) : \langle\langle e,t\rangle,t\rangle, \ x:e) \vdash (\text{Erupt } x) : \langle e,t\rangle$

Forward Function Application

$(\vdash (\text{Two Volcano}) : \langle\langle e,t\rangle,t\rangle, \ x:e) \vdash x:e$ $\qquad$ $\vdash \text{Erupt} : \langle e,\langle e,t\rangle\rangle$

$\Big|$ Quantifier Storage

$\vdash (\text{Two Volcano}) : \langle\langle e,t\rangle,t\rangle$

Forward Function Application

$\vdash \text{Two} : \langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ $\qquad$ $\vdash \text{Volcano} : \langle e,t\rangle$

Two $\qquad\qquad$ *volcanoes* $\qquad\qquad\qquad$ *erupted*

Practice 16.1:

Translate the following sentences into **logic** using Antecedent and Anaphor functions:

Two volcanoes erupted. They are in Italy.

13

(Antecedent 1 Two Volcano Erupt) ∧ (All (Anaphor 1) (In Italy)) : t

(Antecedent 1 Two Volcano Erupt) : t ∧ (All (Anaphor 1) (In Italy)) : t

(Antecedent 1 Two Volcano) : ⟨⟨e, t⟩, t⟩   Erupt : ⟨e, ⟨e, t⟩⟩   (All (Anaphor 1)) : ⟨e, ⟨e, t⟩⟩   (In Italy) : ⟨e, t⟩

(Antecedent 1 Two) : ⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩⟩   Volcano : ⟨e, t⟩ $(\lambda_{f:\langle e,t\rangle}\ f)$: ⟨⟨e, t⟩, ⟨e, t⟩⟩   (In Italy) : ⟨e, t⟩

In : ⟨e, ⟨e, t⟩⟩   Italy : e

Two   *volcanoes*   *erupted*   *They*   *are*   *in*   *Italy*
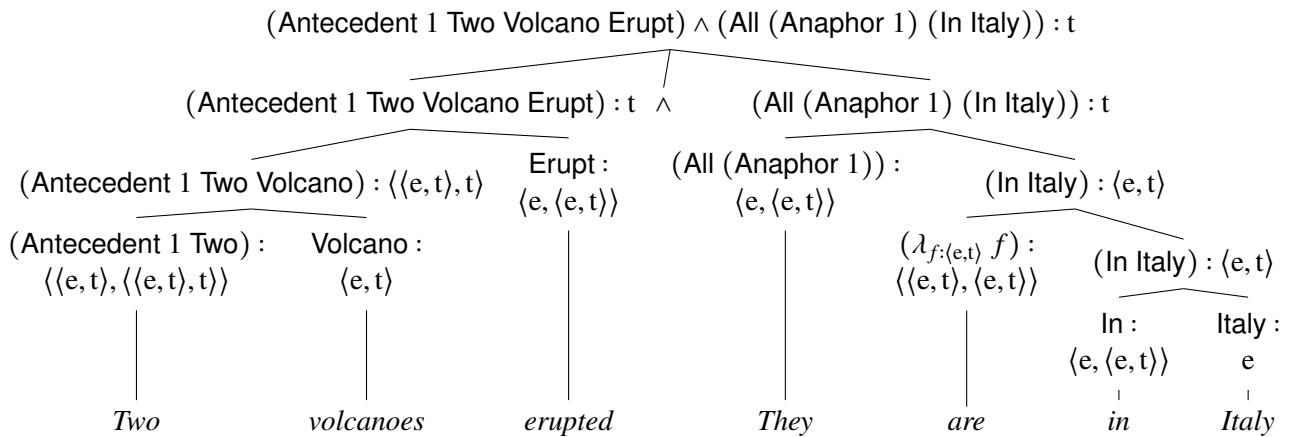
Practice 16.2:

Translate the following sentences into **logic** by **expanding** Antecedent and Anaphor functions:

Two volcanoes erupted. They are in Italy.

---

t

   t   ∧   t

⟨⟨e, t⟩, t⟩   ⟨e, t⟩   ⟨⟨e, t⟩, t⟩   ⟨e, t⟩

...   ⟨e, t⟩   ...   ⟨e, t⟩   ⟨e, ⟨e, t⟩⟩   e

(⟨e, t⟩   )

$\lambda_{x:e}$ t

t   ∧   t

⟨e, t⟩   e   ⟨e, t⟩   e

Two   Volcano   Erupt   ∧   All   ($\lambda_{x:e}$ Volcano x ∧   Erupt x)   In   Italy

14