

Expanding Our Formalism, Part 1¹

1. Review of Our System Thus Far

Thus far, we've built a system that can interpret a very narrow range of English structures: sentences whose subjects are proper names, and whose Vs are intransitive.

(1) Main Components of the System Thus Far

- a. Rules:
- (i) Function Application (FA)
 - (ii) Non-Terminal Nodes (NN)
 - (iii) Terminal Nodes

b. Lexical Entries (Primitives):

- Names*
- (i) [[Barack]] = Barack
 - (ii) [[Joe]] = Joe

Intransitive Verbs

- (iii) [[smokes]] = $f: \{ x : x \text{ is an entity} \} \rightarrow \{ T, F \}$
for every $y \in \{ x : x \text{ is an entity} \}$, $f(y) = T$ iff y smokes
- (iv) [[dances]] = $h: \{ x : x \text{ is an entity} \} \rightarrow \{ T, F \}$
for every $y \in \{ x : x \text{ is an entity} \}$, $h(y) = T$ iff y dances

(2) A Helpful New Shorthand

- a. $D_e = \{ x : x \text{ is an entity} \}$ *the set of entities*
b. $D_t = \{ T, F \}$ *the set of truth-values*

(3) Illustration of the Shorthand

- a. [[smokes]] = $f: D_e \rightarrow D_t$
for every $x \in D_e$, $f(x) = T$ iff x smokes
- b. [[dances]] = $h: D_e \rightarrow D_t$
for every $x \in D_e$, $h(x) = T$ iff x dances

¹ These notes are based on the material in Heim & Kratzer (1998: 26-29).

2. The System of Semantic Types

Besides being a helpful shorthand, the notation in (2) gives us an easy way to talk about the kinds – or *types* – of extensions that natural language expressions can have.

(4) Example: The Semantic Type of Proper Names

a. What We Already Know:

(i) Proper names like “Barack” have **entities** as their extension.

(ii) “Barack” has an **entity** as its extension.

b. Some ‘Fancy’ New Ways of Saying This:

(i) *Proper names like “Barack” are of type **e** (‘e’ for entity)*

(ii) *“Barack” is an expression of type **e***

(iii) $[[\text{Barack}]] \in D_e$

(5) Example: The Semantic Type of Sentences

a. What We Already Know:

(i) Sentences like “Barack smokes” have **truth-values** as their extension.

(ii) “Barack smokes” has a **truth-value** as its extension.

b. Some ‘Fancy’ New Ways of Saying This:

(i) *Sentences like “Barack smokes” are of type **t** (‘t’ for truth-value)*

(ii) *“Barack smokes” is an expression of type **t***

(iii) $[[\text{Barack smokes}]] \in D_t$

Question: What about intransitive verbs like “smoke”? What ‘type’ are they?

Another Helpful New Shorthand

$D_{\langle e,t \rangle}$ = all the possible functions from entities to truth-values
the set of functions from entities to truth-values

(6) **The Semantic Type of Intransitive Verbs**

a. What We Already Know:

- (i) Intransitive verbs like “smokes” have as their extensions **functions from entities to truth-values.**
- (ii) “Smokes” has as its extension **a function from entities to truth-values.**

b. Some ‘Fancy’ New Ways of Saying This:

- (i) *Intransitive verbs like “smokes” are of type $\langle e,t \rangle$*
- (ii) *“Smokes” is an expression of type $\langle e,t \rangle$*
- (iii) $[[\text{smokes}]] \in D_{\langle e,t \rangle}$

(7) **A Schematic of the Notation for ‘Functional’ Types**

$\langle \textit{Semantic-Type-of-Argument} , \textit{Semantic-Type-of-Value} \rangle$

Examples:

- a. $\langle e,t \rangle =$ the type of function whose arguments are of type e, and whose values are of type t (e.g. $[[\text{smokes}]]$)
- b. $\langle t,t \rangle =$ the type of function whose arguments are of type t, and whose values are of type t (e.g. $[[\text{it is not the case that}]]$)
- c. $\langle t,e \rangle =$ the type of function whose arguments are of type t, and whose values are of type e (*doesn't correspond to a meaning in language*)
- d. $\langle e, \langle e,t \rangle \rangle =$ the type of a function whose arguments are of type e, and whose values are of type $\langle e,t \rangle$
(as we will see, this is the extension of a *transitive verb*...)

What is the point of all this?

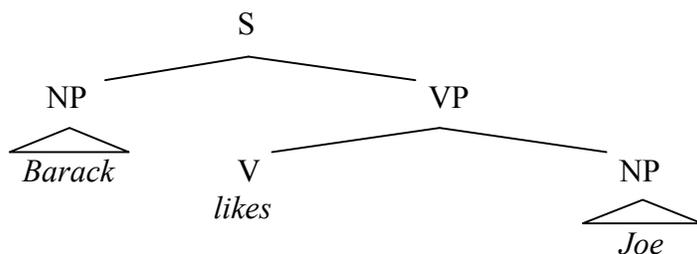
- This notation is useful for working out semantic problems
- It's especially useful for illustrating *or reasoning about* how the semantic derivation for a particular syntactic tree proceeds...

... as we will see right now!

3. The Semantics of Transitive Verbs

Question: Can our system interpret the syntactic structure below, where the main V of the clause is a *transitive V*?

(8) Sentence Containing a Transitive Verb



Problem 1: We don't have a lexical entry for *likes*.

Sub-Question: Which of the following semantic types (the ones currently in our system) is the semantic type of *likes*?

(9) Our Current Semantic Types

- Type e
- Type t
- Type $\langle e, t \rangle$

Problem 2: None of the semantic types in (9) will work as the semantic type for *likes*!

- The extension of *likes* must combine with the extension of *Joe* to give us the extension of the VP *likes Joe*.
- The NP *Joe* is of type e , so the extension of *likes* must be some function that has D_e as its domain.
- In (9), the only such functional semantic type is $\langle e, t \rangle$.
- But, if *likes* were type $\langle e, t \rangle$, then the whole VP *likes Joe* would be of type t , and so it couldn't combine with the meaning of the subject *Barack* (which is type e).

Conclusion:

- The set of types in (9) is not enough to interpret transitive sentences of English.
- We have to introduce a new semantic type in order to write out a lexical entry for *likes*.

OK... But, what 'type' would work?

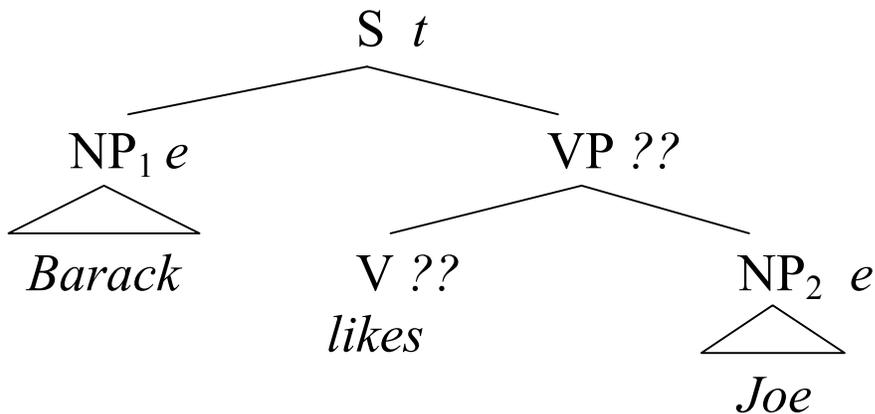
(10) Reasoning Through the Type Assignments

Here are the semantic type assignments we already know

$[[S]] \in D_t$; $[[Barack]] \in D_e$; $[[Joe]] \in D_e$

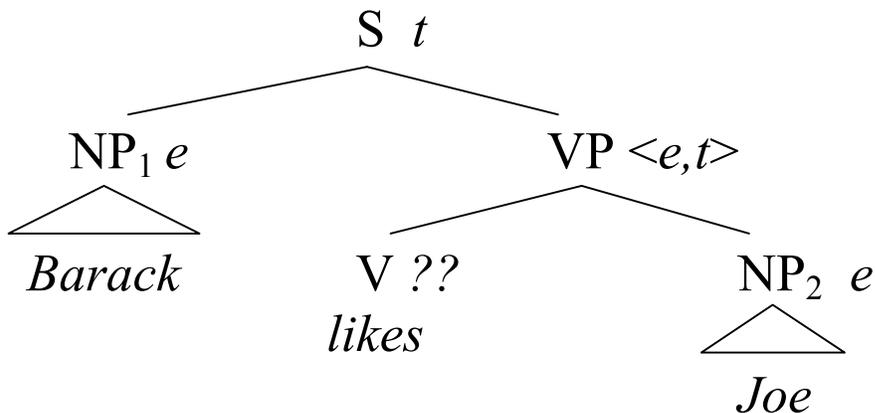
Here are the semantic type assignments we don't yet know

$[[VP]] \in ??$; $[[V]] \in ??$



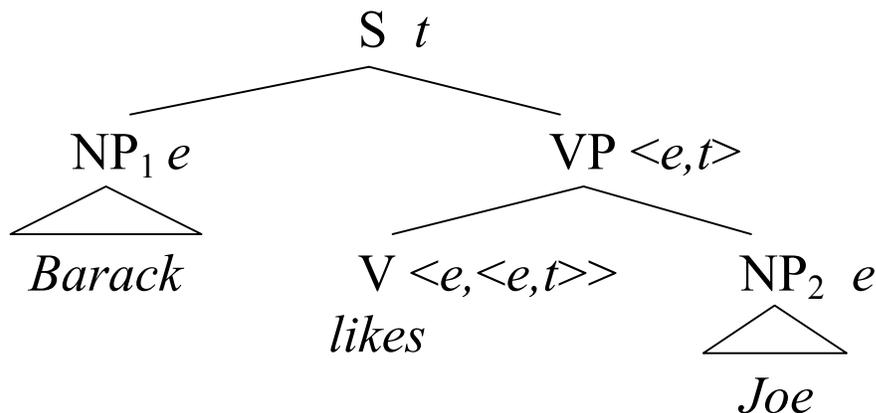
(11) Reasoning Out the Semantic Type of the VP

- The entire sentence is type *t*
- The subject NP *Barack* is type *e*
- Now... the extension of the VP must ‘combine’ with the extension of the subject to give us the extension of the whole sentence...
- The only rule we have for such ‘combination’ is Function Application (FA)...
- So, $[[VP]]$ has to be a function that takes *Barack* and gives back a T-value...
- **THEREFORE, the semantic type of the VP must be $\langle e,t \rangle$ (like an intransitive V)**



(12) Reasoning Out the Type of the V *Likes*

- The entire VP *likes Joe* is of type $\langle e, t \rangle$
- The direct object *Joe* is of type e
- Now, the extension of the V *likes* must ‘combine’ with the extension of the direct object *Joe* to give us the extension of the whole VP...
- The only rule we have for such ‘combination’ is FA...
- So, $[[V]]$ has to be a function which takes *Joe* and gives back $[[VP]]$ (of type $\langle e, t \rangle$)
- **THEREFORE, the semantic type of the transitive V *likes* must be $\langle e \langle e, t \rangle \rangle$**



Conclusion:

We know, abstractly speaking, what *kind* of function the V *likes* must have as its extension:

*a function from entities to
functions from entities to truth-values*

(13) New Goal

Develop a lexical entry for *likes* that does the following:

- Stipulates its extension as a function of type $\langle e, \langle e, t \rangle \rangle$
- Allows our system to derive the following truth-conditional statement...

“Barack likes Joe” is T iff Barack likes Joe.

How do we do this?

Let’s work backwards, step-by-step, starting with the larger VP “likes Joe” ...

(14) Reasoning Out the Extension of the VP *likes Joe*

a. Let's start off by considering the T-conditions of some other sentences containing the VP "*likes Joe*".

- (i) "Mitch likes Joe" is T iff Mitch likes Joe
- (ii) "Seth likes Joe" is T iff Seth likes Joe
- (iii) "Rachael likes Joe" is T iff Rachael likes Joe

b. KEY GENERALIZATION

[[NAME likes Joe]] = T iff NAMED THING likes Joe

c. KEY DEDUCTION

- We've already concluded that [[likes Joe]] is of type $\langle e, t \rangle$
- Consequently, our rule of FA entails that

$$[[\text{NAME likes Joe}]] = [[\text{likes Joe}]]([[\text{NAME}]])$$

- But, given our 'Key Generalization' in (14b) above, it follows that:

$$[[\text{likes Joe}]]([[\text{NAME}]]) = T \text{ iff NAMED THING likes Joe}$$

- Finally, given our assumed semantics for names, where [[NAME]] = NAMED THING, it follows that:

$$[[\text{likes Joe}]](\text{NAMED THING}) = T \text{ iff NAMED THING likes Joe}$$

- **KEY CONCLUSION**

[[likes Joe]] is a function which takes as argument some entity x, and yields T iff x likes Joe.

d. DEDUCED SEMANTICS FOR THE VP

$$[[\text{likes Joe}]] = \begin{array}{l} g: D_e \rightarrow D_t \\ \text{for every } x \in D_e, g(x) = T \text{ iff } x \text{ likes Joe} \end{array}$$

So, we've deduced what the extension of the VP *likes Joe* is...

... but this equation can't just be stipulated in our lexicon.

Instead, we need our system to derive the equation in (14d) from the meaning (extension) of "likes" and the meaning (extension) of "Joe"...

(15) **Reasoning Out the Extension of Extension of the V likes**

a. Let's start off by considering the T-conditions of some other sentences that contain the V "likes"

- (i) "Barack likes Seth" is T iff Barack likes Seth
- (ii) "Joe likes Seth" is T iff Joe likes Seth
- (iii) "Barack likes Nancy" is T iff Barack likes Nancy
- (iv) "Joe likes Nancy" is T iff Joe likes Nancy

b. GENERALIZATIONS

(i) Given exactly the reasoning laid out in (14) earlier, we can deduce from the facts in (15ai) and (15aai) that:

$$[[\text{likes Seth}]] = \quad i: D_e \rightarrow D_t \\ \text{for every } x \in D_e, i(x) = T \text{ iff } x \text{ likes Seth}$$

(ii) Given exactly the reasoning laid out in (14) earlier, we can deduce from the facts in (15aiii) and (15aiv) that:

$$[[\text{likes Nancy}]] = \quad j: D_e \rightarrow D_t \\ \text{for every } x \in D_e, j(x) = T \text{ iff } x \text{ likes Nancy}$$

c. KEY GENERALIZATION

$$[[\text{likes NAME}]] = \quad k: D_e \rightarrow D_t \\ \text{for every } x \in D_e, k(x) = T \text{ iff } x \text{ likes NAMED THING}$$

d. KEY DEDUCTION

- We've already concluded that [[likes]] is of type $\langle e, \langle e, t \rangle \rangle$
- Consequently, FA entails that $[[\text{likes NAME}]] = [[\text{likes}]]([[\text{NAME}]])$
- Thus, given our 'Key Generalization' in (15c), it follows that:

$$[[\text{likes}]]([[\text{NAME}]]) = \quad k: D_e \rightarrow D_t \\ \text{for every } x \in D_e, k(x) = T \text{ iff } x \text{ likes NAMED THING}$$

• **KEY CONCLUSION**

[[likes]] is a function which takes as argument some entity y, and yields the following function:

$$\mathbf{f: D_e \rightarrow D_t} \\ \text{for every } x \in D_e, \mathbf{f(x) = T \text{ iff } x \text{ likes } y}$$

(16) **Deduced Semantics for the Transitive Verb *Likes***

$$[[\text{likes}]] = g: D_e \rightarrow D_{\langle e,t \rangle}$$

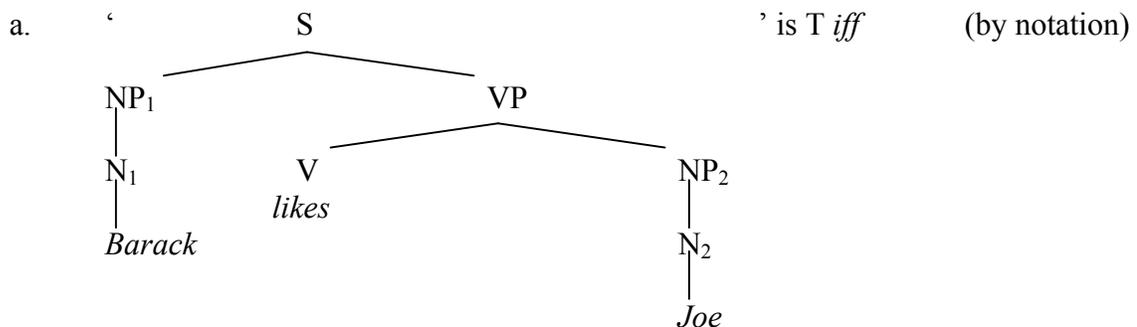
$$\text{for every } x \in D_e, g(x) = i_x: D_e \rightarrow D_t$$

$$\text{for every } y \in D_e, i_x(y) = T \text{ iff } y \text{ likes } x$$

“the function g from entities to functions from entities to T-values, which for every entity x , maps x to the function i_x from entities to T-values, which for every entity y maps y to T iff y likes x ”

With this lexical entry for “likes”, we can now derive the T-conditions of the transitive sentence “*Barack likes Joe*”!

(17) **Derivation of the Truth-Conditions of “*Barack likes Joe*”**



b. $[[S]] = T$

c. **Subproof**

(i) $[[NP_1]] =$ (by NN)

(ii) $[[N_1]] =$ (by NN)

(iii) $[[Barack]] =$ (by TN)

(iv) Barack

d. **Subproof**

(i) $[[NP_2]] =$ (by NN)

(ii) $[[N_2]] =$ (by NN)

- (iii) $[[\text{Joe}]] =$ (by TN)
- (iv) Joe
- e. **Subproof**
- (i) $[[\text{V}]] =$ (by NN)
- (ii) $[[\text{likes}]] =$ (by TN)
- (iii) g
- f. **Subproof**
- (i) $[[\text{VP}]] =$ (by FA, d, e)
- (ii) $[[\text{V}]]([[NP_2]]) =$ (by e)
- (iii) $g([[NP_2]]) =$ (by d)
- (iv) $g(\text{Joe}) =$ (by def. of 'g')
- (v) $i_{\text{Joe}}: D_e \rightarrow D_t$
for every $y \in D_e, i_{\text{Joe}}(y) = T$ iff y likes Joe
- g. $[[\text{S}]] = T$ iff (by FA, c, f)
- h. $[[\text{VP}]]([[NP_1]]) = T$ iff (by c)
- i. $[[\text{VP}]](\text{Barack}) = T$ iff (by f)
- j. $i_{\text{Joe}}(\text{Barack}) = T$ iff (by def. of ' i_{Joe} '))
- k. Barack likes Joe

PROVEN: "Barack likes Joe" is T iff Barack likes Joe

The Upshot:

- In order to capture the semantics of transitive verbs, we must expand our system of types for natural language, so that it also includes functions of type $\langle e, \langle e, t \rangle \rangle$
- That is, we can model the extensions of transitive verbs as functions of type $\langle e, \langle e, t \rangle \rangle$

The extension of a transitive verb is a function that takes an entity (the direct object), and returns a function that takes an entity (the subject) and returns a truth-value!

4. The Semantics of “Or”

(18) Key Feature of Our Semantics for Transitive Verbs

The semantic type of a transitive verb is a function which takes entities as arguments...
*and then delivers **another function** as value (function from entities to T-values)...*

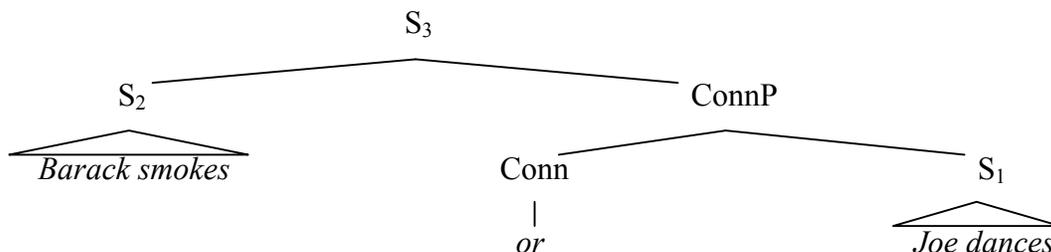
FACT: Such ‘function-valued functions’ will play a *huge* role in our overall system.
Such functions can do a lot of work besides simply model transitive verbs.

(19) Example: The Logical Connective “Or”

We’ll see in this section how the extension of the logical connective “or” can also be modeled as a function-valued function.

(20) Background on “Or”: Syntax

The connective “or” combines with one sentence S_1 and then combines with another sentence S_2 to create a larger sentence S_3



(21) **Background: Truth Conditions**

The truth-conditions of a sentence containing “or” are as follows:

“ [S₂ [or S₁]]” is T *iff* S₂ or S₁

Examples:

- a. “ [Barack smokes [or Joe dances]]” is T *iff* Barack smokes or Joe dances.
- b. “ [Seth bowls [or Peggy curls]]” is T *iff* Seth bowls or Peggy curls.

From these background facts/assumptions, we can start to build a theory of what [[or]] is.

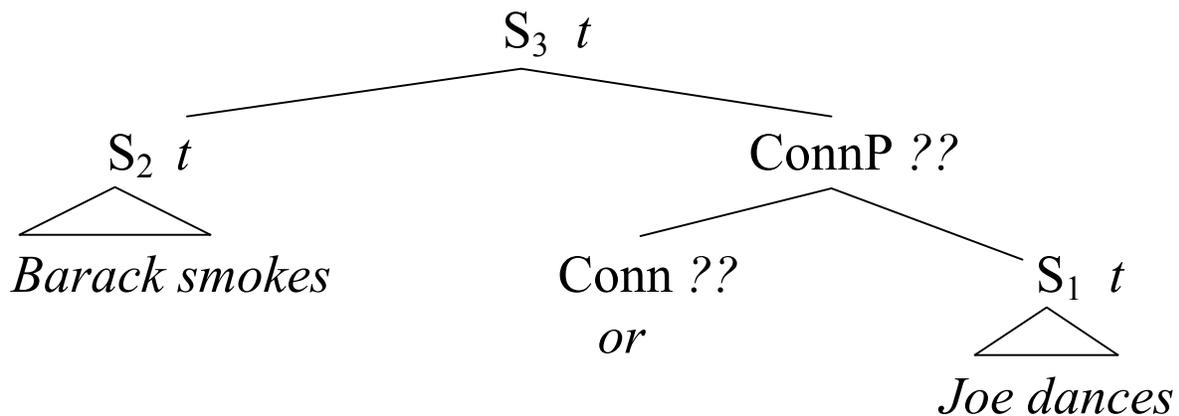
(22) **Reasoning Through the Type Assignments**

Here are the semantic type assignments we already know

[[S₃]] ∈ D_t ; [[S₂]] ∈ D_t ; [[S₁]] ∈ D_t

Here are the semantic type assignments we don't yet know

[[ConnP]] ∈ ?? ; [[Conn]] ∈ ??



(28) Reasoning Out the Extension of the ConnP “or Joe dances”, Part 2

a. KEY DEDUCTION

- We’ve already concluded that $[[\text{ or Joe smokes }]]$ is of type $\langle t, t \rangle$
- Consequently, our rule of FA entails that

$$[[S [\text{ or Joe smokes }]]] = [[\text{ or Joe smokes }]]([[S]])$$

- But, given our ‘Key Generalization’ in (27c) above, it follows that:

$$[[\text{ or Joe smokes }]]([[S]]) = T \text{ iff } [[S]] = T \text{ or Joe dances}$$

- **KEY CONCLUSION**

$[[\text{ or Joe smokes }]]$ is a function which takes as argument a T-value x , and yields T iff *either* $x = T$ *or* Joe dances

b. Deduced Semantics for the ConnP

$$[[\text{ or Joe smokes }]] = k: D_t \rightarrow D_t \\ \text{for every } x \in D_t, k(x) = T \text{ iff } x = T \text{ or Joe dances}$$

So, we’ve deduced what the extension of the ConnP “or Joe smokes” is...
... but this equation can’t just be stipulated in our lexicon.

Instead, we need our system to derive the equation in (28b) from the meaning (extension) of “or” and the meaning (extension) of “Joe dances”...

(29) Reasoning Out the Extension of Extension of the Connective “Or”, Part 1

Let’s start off by considering the T-conditions of some other sentences that contain the connective “or”

- “Barack smokes **or Seth drinks**” is T *iff* Barack smokes **or Seth drinks**.
- “Joe dances **or Seth drinks**” is T *iff* Joe dances **or Seth drinks**.
- “Barack smokes **or Peggy curls**” is T *iff* Barack smokes **or Peggy curls**.
- “Joe dances **or Peggy curls**” is T *iff* Joe dances **or Peggy curls**.

(30) Reasoning Out the Extension of Extension of the Connective “Or”, Part 2

a. GENERALIZATIONS

- (i) Given exactly the reasoning laid out in (27)-(28) earlier, we can deduce from the facts in (29ai) and (29aai) that:

$$\begin{aligned} [[\text{ or Seth drinks }]] &= l: D_t \rightarrow D_t \\ &\text{for every } x \in D_t, l(x) = T \text{ iff } x = T \text{ or Seth drinks} \end{aligned}$$

- (ii) Given exactly the reasoning laid out in (27)-(28) earlier, we can deduce from the facts in (29aiii) and (29aiv) that:

$$\begin{aligned} [[\text{ or Peggy curls }]] &= m: D_t \rightarrow D_t \\ &\text{for every } x \in D_t, m(x) = T \text{ iff } x = T \text{ or Peggy curls} \end{aligned}$$

b. KEY GENERALIZATION

- (i) $[[\text{ or S }]]$ = n: $D_t \rightarrow D_t$
for every $x \in D_t$, $n(x) = T$ iff $x = T$ or **S**

...or equivalently (given (26))...

- (ii) $[[\text{ or S }]]$ = n: $D_t \rightarrow D_t$
for every $x \in D_t$, $n(x) = T$ iff $x = T$ or $[[S]] = T$

c. KEY DEDUCTION

- We’ve already concluded that $[[\text{ or }]]$ is of type $\langle t, \langle t, t \rangle \rangle$
- Consequently, FA entails that $[[\text{ or S }]]$ = $[[\text{ or }]]$ ($[[S]]$)
- Thus, given our ‘Key Generalization’ in (30b), it follows that:

$$\begin{aligned} [[\text{ or }]]([[S]]) &= p: D_t \rightarrow D_t \\ &\text{for every } x \in D_t, p(x) = T \text{ iff } x = T \text{ or } [[S]] = T \end{aligned}$$

- **KEY CONCLUSION**

$[[\text{ or }]]$ is a function which takes as argument a truth-value y , and yields the following function:

$$\begin{aligned} &\mathbf{p: } D_t \rightarrow D_t \\ &\text{for every } x \in D_t, p(x) = T \text{ iff } x = T \text{ or } y = T \end{aligned}$$

(31) **Deduced Semantics for the Logical Connective “Or”**

$$[[\text{or}]] = \quad q: D_t \rightarrow D_{\langle t, t \rangle}$$

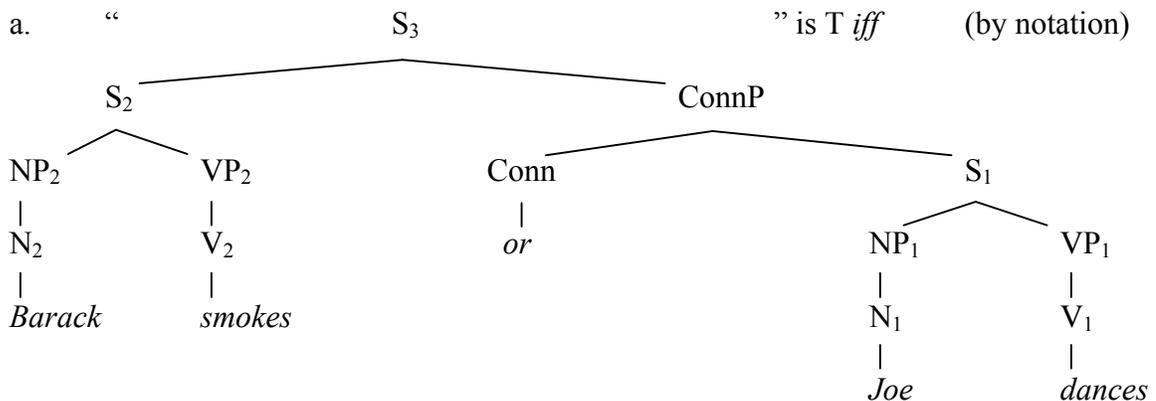
$$\quad \text{for every } x \in D_t, q(x) = p_x : D_t \rightarrow D_t$$

$$\quad \text{for every } y \in D_t, p_x(y) = T \text{ iff } y=T \text{ or } x=T$$

“the function q from T-values to functions from T-values to T-values, which for every T-value x , maps x to the function p_x from T-values to T-values, which for every T-value y maps y to T iff $x=T$ or $y=T$ ”

With this lexical entry for “or”, we can now derive the T-conditions of the complex sentence “Barack smokes or Joe dances”.

(32) **Derivation of the Truth-Conditions of “Barack smokes or Joe dances”**



b. $[[S_3]] = T$

c. **Subproof**

(i) $[[\text{Conn}]] =$ (by NN)

(ii) $[[\text{or}]]$ (by TN, (31))

(iii) q

d. **Subproof**

(i) $[[\text{NP}_1]] =$ (by NN)

(ii) $[[\text{N}_1]] =$ (by NN)

(iii) $[[\text{Joe}]]$ = (by TN)

(iv) Joe

e. **Subproof**

(i) $[[\text{VP}_1]]$ = (by NN)

(ii) $[[\text{V}_1]]$ = (by NN)

(iii) $[[\text{dances}]]$ = (by TN, (3))

(iv) h

f. **Subproof**

(i) $[[\text{S}_1]]$ = (by FA, d,e)

(ii) $[[\text{VP}_1]]([[\text{NP}_1]])$ = (by d, e)

(iii) h(Joe)

g. **Subproof**

(i) $[[\text{ConnP}]]$ = (by FA, c, f)

(ii) $[[\text{Conn}]]([[\text{S}_1]])$ = (by c)

(iii) $q([[\text{S}_1]])$ = (by def. of 'q')

(iv) $p : D_t \rightarrow D_t$
for every $y \in D_t, p(y) = T$ iff $y=T$ or $[[\text{S}_1]] = T$ = (by f)

(v) $p : D_t \rightarrow D_t$
for every $y \in D_t, p(y) = T$ iff $y=T$ or $h(\text{Joe})=T$ = (by def. of 'h')

(vi) $p : D_t \rightarrow D_t$
for every $y \in D_t, p(y) = T$ iff $y=T$ or Joe dances

h. **Subproof**

(i) $[[\text{NP}_2]]$ = (by NN)

(ii) $[[\text{N}_2]]$ = (by NN)

- (iii) $[[\text{Barack}]]$ = (by TN)
- (iv) Barack
- i. **Subproof**
- (i) $[[\text{VP}_2]]$ = (by NN)
- (ii) $[[\text{V}_2]]$ = (by NN)
- (iii) $[[\text{smokes}]]$ = (by TN)
- (iv) f
- j **Subproof**
- (i) $[[\text{S}_2]]$ = (by FA, h, i)
- (ii) $[[\text{VP}_2]]([\text{NP}_2])$ = (by h, i)
- (iii) f(Barack)
- k. $[[\text{S}_3]]$ = T *iff* (by FA, g, j)
- l. $[[\text{ConnP}]]([\text{S}_2])$ = T *iff* (by g)
- m. p($[[\text{S}_2]]$) = T *iff* (by def. of ‘p’)
- n. $[[\text{S}_2]]$ = T or Joe dances *iff* (by j)
- o. f(Barack) = T or Joe dances *iff* (by def. of ‘f’)
- p. Barack smokes or Joe dances

The Upshot:

- In order to capture the semantics of sentential connectives like “or”, we must expand our system of types for natural language, so that it also includes functions of type $\langle t, \langle t, t \rangle \rangle$
- That is, we can model the extensions of logical sentential connectives as functions of type $\langle t, \langle t, t \rangle \rangle$