LING 510: Week 9 review

November 6, 2026

Shaunak Phadnis
sphadnis@umass.edu
Office hours: W 1:30–3:30PM, N421H | Zoom

1 Enriching the system with indices

- We introduced pronouns in our system.
- Pronouns are a shorthand to refer to an entity that is *salient* in the context. we don't like repeating names that have already been mentioned/made salient for some reason.

(1) a. #John went home. John was tired. John played with John's cat.
b. John went home. He was tired. He played with his cat.

- Meaning of a pronoun is context dependent. The sentence (2) is the same in all three cases depicted, but its meaning (and therefore truth value) changes with the context.

(2) She is a nun.



c_1: pointing at Abigail (*She/her*)	c_2: pointing at Aditi (*She/her*)	c_3: pointing at Skye (*they/them*)
Abigail is a nun.	Aditi is a nun.	
true!	*false!*	*bad!*

- The pronoun she also comes with a condition of its own: it can only refer to someone who goes by *she/her*. In c_3 that condition fails, so we get no truth value at all, much like the presupposition failures we have seen before.
- The same linguistic token can be used as shorthand to fix reference to multiple entities. We are pretty good at it but sometimes it can be hard to figure out who is being referred to.

(3) a. John hit Bill. He also shouted at him.
b. ??John and Bill partied all night. He had fun.

- Here's a way of thinking about (3):

1. John (the first entity mentioned in c) hit Bill (the second entity mentioned in c). The first entity also shouted at the second entity.

2. John (the first entity mentioned in c) and Bill (the second entity mentioned in c) partied all night. ???? had fun.

- Notice that in making sense of these sentences, we are tracking entities in order to fix reference of the pronoun. We keep a *list* that helps us track the reference.

(4) John hit Bill. He_1 also shouted at him_2.

Fact

Pronouns are linguistic objects that come with an index. An index is a natural number $n \in \mathbb{N}$.

- Using *indices* is a neat way to track objects– there are infinitely many of them which is useful since there are a lot of people in the world and we talk about people *all the time*.

2 Enriching the system with assignment functions

We need a way to track entities being referred to i.e., we need a way to ensure that an index actually points to the right entity when we compute the denotation. We *also* want to respect the context-dependence of interpretation of pronouns.

Since, we want pronouns to be uniquely picking out entities, and when uniqueness is not satisifed it leads to failure to interpret, we can utilize *functions that map indices to entities in a context*.

Definition

An assignment function[a] is a (partial) function $g : \mathbb{N} \to D_e$ from numerical indices to the domain of entities.

[a] or a variable assignment or simply, an assignment

$$g = \begin{bmatrix} 1 \mapsto \text{Skye} \\ 2 \mapsto \text{Aditi} \\ 3 \mapsto \text{Abigail} \\ \vdots \end{bmatrix} \quad g' = \begin{bmatrix} 1 \mapsto \text{Abigail} \\ 2 \mapsto \text{Aditi} \\ 3 \mapsto \text{Bill} \\ \vdots \end{bmatrix} \quad h = \begin{bmatrix} 1 \mapsto \text{Aditi} \\ 2 \mapsto \text{Aditi} \\ 3 \mapsto \text{Aditi} \\ \vdots \end{bmatrix} \quad \ldots$$

$$g(1) = \text{Skye}, g'(1) = \text{Abigail}, g'(2) = h(3) = \text{Aditi}$$

There are many ways to define the mapping between indices and entities, I have just listed 3 above. Since interpretation of pronouns depends on what entity their index gets mapped to (and there are many different ways that could happen), our denotations are now sensitive to this.

So far, $[\![\cdot]\!]$, the interpretation function mapped linguistic objects (trees) to denotations, Now we will have a **family** of interpretation functions that are *parametrized* to assignments!

- We differentiate interpretation functions with superscripts: $[\![\cdot]\!]^g$, $[\![\cdot]\!]^{g'}$, $[\![\cdot]\!]^{g''}$ ...;
- The notation $[\![\cdot]\!]^g$ indicates the interpretation function that uses the assignment g;
- If there are assignment functions $g \neq g'$, then there are interpretation functions $[\![\cdot]\!]^g \neq [\![\cdot]\!]^{g'}$.

Composition rules, now relative to an assignment g

Non-Branching Nodes (NN). If α is a node with sole daughter β, then, for any assignment g, $[\![\alpha]\!]^g = [\![\beta]\!]^g$.

Function Application (FA). If α is a branching node with daughters β and γ, and $[\![\beta]\!]^g$ is a function whose domain contains $[\![\gamma]\!]^g$, then $[\![\alpha]\!]^g = [\![\beta]\!]^g([\![\gamma]\!]^g)$.

Predicate Modification (PM). If α is a branching node with daughters β and γ, and both $[\![\beta]\!]^g$ and $[\![\gamma]\!]^g$ are of type $\langle e, t\rangle$, then

$$[\![\alpha]\!]^g = \left[\lambda x_e.\ T \text{ iff } [\![\beta]\!]^g(x) = T \text{ and } [\![\gamma]\!]^g(x) = T\right].$$

Pronoun Rule (PR). If α is a pronoun bearing the index i, then

$$[\![\alpha_i]\!]^g = g(i).$$

Why did we relativize all the rules to g instad of just the new pronoun rule? (*hint:* $[\![\cdot]\!]$ is a function)

Practice

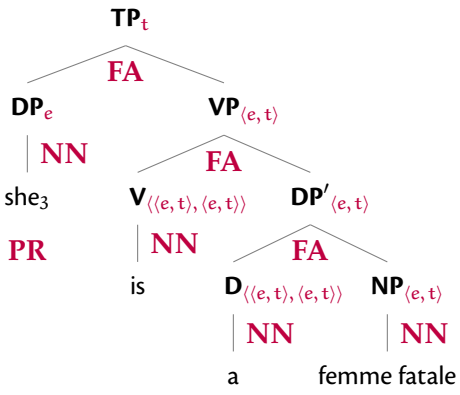
Compute the meaning of the following sentences relative to the two assignment functions.

(5) a. She$_3$ is a femme fatale.
b. She$_2$ likes her$_3$.

$$g = \begin{bmatrix} 1 \mapsto \text{Siouxsie} \\ 2 \mapsto \text{Janis} \\ 3 \mapsto \text{Nico} \\ \vdots \end{bmatrix} \qquad h = \begin{bmatrix} 1 \mapsto \text{Stevie} \\ 2 \mapsto \text{Whitney} \\ 3 \mapsto \text{Nico} \\ \vdots \end{bmatrix}$$

Suppose Janis likes Nico, but Whitney does not like Nico. What truth value does the sentence get relative to g and h?

(a)



Lexical entries (the same under every assignment g)

$$[\![\text{is}]\!]^g = [\lambda f_{\langle e,t\rangle}.\ f]$$
$$[\![\text{a}]\!]^g = [\lambda f_{\langle e,t\rangle}.\ f]$$
$$[\![\text{femme fatale}]\!]^g = [\lambda x_e.\ T \text{ iff } x \text{ is a femme fatale}]$$
$$[\![\text{likes}]\!]^g = [\lambda x_e.[\lambda y_e.\ T \text{ iff } y \text{ likes } x]]$$

Pronoun Rule (PR)

$$[\![\alpha_i]\!]^g = g(i)$$

(a) She_3 is a femme fatale.

Subproof a. DP′

1. $\left[\!\!\left[\begin{array}{c} \text{DP}'_{\langle e,t\rangle} \\ \text{D}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \quad \text{NP}_{\langle e,t\rangle} \\ \text{NN} \quad\quad \text{NN} \\ \text{a} \quad\quad \text{femme fatale} \end{array} \right]\!\!\right]^g = \left[\!\!\left[\begin{array}{c} \text{D}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \\ \text{NN} \\ \text{a} \end{array} \right]\!\!\right]^g \left(\left[\!\!\left[\begin{array}{c} \text{NP}_{\langle e,t\rangle} \\ \text{NN} \\ \text{femme fatale} \end{array} \right]\!\!\right]^g \right)$ (by FA)

2. $= [\![\text{a}]\!]^g\big([\![\text{femme fatale}]\!]^g\big)$ (by NN)

3. $= [\lambda f_{\langle e,t\rangle}.\, f]\big([\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]\big)$ (by lexical entries)

4. $= [\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]$ (by β-reduction)

Subproof b. VP

1. $\left[\!\!\left[\begin{array}{c} \text{VP}_{\langle e,t\rangle} \\ \text{V}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \quad \text{DP}'_{\langle e,t\rangle} \\ \text{NN} \quad\quad \text{a femme fatale} \\ \text{is} \end{array} \right]\!\!\right]^g = \left[\!\!\left[\begin{array}{c} \text{V}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle} \\ \text{NN} \\ \text{is} \end{array} \right]\!\!\right]^g \left(\left[\!\!\left[\begin{array}{c} \text{DP}'_{\langle e,t\rangle} \\ \text{a femme fatale} \end{array} \right]\!\!\right]^g \right)$ (by FA)

2. $= [\![\text{is}]\!]^g\big([\![\text{DP}']\!]^g\big)$ (by NN)

3. $= [\lambda f_{\langle e,t\rangle}.\, f]\big([\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]\big)$ (by lexical entry, a.)

4. $= [\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]$ (by β-reduction)

There is no pronoun inside the VP, so we get the same result under *every* assignment. We can reuse it for both g and h.

Derivation. TP, relative to g

1. $\left[\!\!\left[\begin{array}{c} \text{TP}_t \\ \text{DP}_e \quad\quad \text{VP}_{\langle e,t\rangle} \\ \text{NN} \quad\quad \text{is a femme fatale} \\ \text{she}_3 \end{array} \right]\!\!\right]^g = \left[\!\!\left[\begin{array}{c} \text{VP}_{\langle e,t\rangle} \\ \text{is a femme fatale} \end{array} \right]\!\!\right]^g \left(\left[\!\!\left[\begin{array}{c} \text{DP}_e \\ \text{NN} \\ \text{she}_3 \end{array} \right]\!\!\right]^g \right)$ (by FA)

2. $= [\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]\big([\![\text{she}_3]\!]^g\big)$ (by b., NN)

3. $= [\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}]\big(g(3)\big)$ (by PR)

4. $= [\lambda x_e.\, \top \text{ iff } x \text{ is a femme fatale}](\text{Nico})$ ($g(3) = \text{Nico}$)

5. $= \top \text{ iff Nico is a femme fatale}$ (by β-reduction)

Derivation. TP, relative to h

1. $[\![\text{TP}]\!]^h = [\![\text{VP}]\!]^h([\![\text{DP}]\!]^h)$ (by FA)
2. $= [\lambda x_e.\, \text{T iff } x \text{ is a femme fatale}]([\![\text{she}_3]\!]^h)$ (by b., NN)
3. $= [\lambda x_e.\, \text{T iff } x \text{ is a femme fatale}](h(3))$ (by PR)
4. $= [\lambda x_e.\, \text{T iff } x \text{ is a femme fatale}](\text{Nico})$ (by $h(3)$=Nico)
5. $=$ T iff Nico is a femme fatale (by β-reduction)

$g(3) = h(3) =$ Nico, so (a) has the same truth conditions under g and h.

3 Free and bound pronouns

We have dealt with pronouns whose interpretation is context (or assignment dependent) – they were **free**. But a pronoun's interpretation can also be bound under certain linguistic environments i.e., meaning of a pronoun **co-varies** with another DP in the structure.

(6) a. Every woman likes her job.
$\rightsquigarrow$ Abigail likes Abigail's job, Aditi likes Aditi's job, Stevie likes Stevie's job …
b. Only Shawn likes his cat.
$\rightsquigarrow$ Barry doesn't like Barry's job, Smith doesn't like Smith's job …

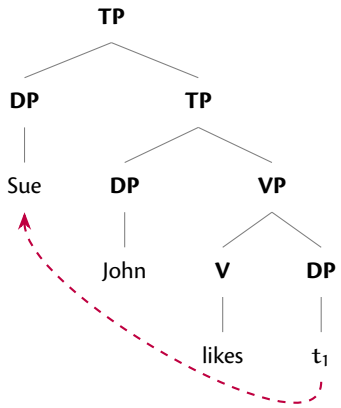
Pronouns that get a bound reading can, in principle, get a free reading as well in the same syntactic context. This is most transparent in cases of gender mismatch.

(7) Every man likes her job. (*Let's say, Abigail's*)

4 Movement, traces, and relative clauses

When a constituent moves, it is pronounced in a place that is not the place where it gets interpreted. For instance, consider topicalization in (8).

(8) Sue, John likes t_1.



The DP Sue is pronounced at the front of the sentence, but it is interpreted as the object of likes. We don't want the interpretation to be *free*! Let's say we had the following assignment.

$$g = \begin{bmatrix} 1 \mapsto \text{Stella} \\ 2 \mapsto \text{Billy} \\ 3 \mapsto \text{John} \\ \vdots \end{bmatrix}$$

Derivation without predicate abstraction. Sue, John likes t_1, relative to the original g (so $g(1) =$ Stella). Treat the trace like a pronoun.

1. $[\![t_1]\!]^g = g(1) = \text{Stella}$ (PR)
2. $[\![\text{likes } t_1]\!]^g = [\lambda x_e.\, [\lambda y_e.\, \text{T iff } y \text{ likes } x]](\text{Stella}) = [\lambda y_e.\, \text{T iff } y \text{ likes Stella}]$ (FA, TN, 1, β)
3. $[\![\text{John likes } t_1]\!]^g = [\lambda y_e.\, \text{T iff } y \text{ likes Stella}](\text{John}) = \text{T iff John likes Stella}$ (FA, TN, 2, β)
4. $[\![\text{Sue, John likes } t_1]\!]^g = ??$ The daughters are $[\![\text{Sue}]\!]^g = \text{Sue}$ (type e) and $[\![\text{John likes } t_1]\!]^g$ (type t). FA needs one daughter to be a function, and neither is. PM needs two $\langle e, t\rangle$s. **No rule applies.**

What went wrong

1. **Type mismatch.** The moved DP (type e) and the lower TP (type t) cannot combine, so the sentence gets no denotation at all.
2. **Wrong value for the trace.** Even if we ignored Sue, the trace took its value from the context: we computed *John likes Stella*. We want *T iff John likes Sue*.

Both problems have one source: nothing in the tree connects Sue to the index on t_1.

What we need is a mechanism that will keep rest of the assignments intact but only changes the assignment for the index appearing in the tree.

$$g = \begin{bmatrix} 1 \mapsto \text{Stella} \\ 2 \mapsto \text{Billy} \\ 3 \mapsto \text{John} \\ \vdots \end{bmatrix} \quad \Longrightarrow \quad \underbrace{g = \begin{bmatrix} 1 \mapsto \text{Sue} \\ 2 \mapsto \text{Billy} \\ 3 \mapsto \text{John} \\ \vdots \end{bmatrix}}_{\text{assignment we want}}$$

Note that the source of our problem lies in the syntax – **not** in how the assignment was defined.

- We want to overwrite the assignment $g(1) =$ Stella to $g(1) =$ Sue.
- We want the moved constituent Sue to be used up to do so.
- We want rest of the things to remain the same.

In a way, we need g to temporarily map 1 to a variable that can then be mapped to Sue in the course of the derivation!

Definition: Modified assignment function

Given any assignment function g, the modified assignment function $g[n \mapsto a]$ is the assignment function just like g except—at most—that it maps n to a.

$$g[n \mapsto a](m) = \begin{cases} a & \text{if } m = n \\ g(m) & \text{if } m \neq n \end{cases}$$

Since, Sue is ultimately going to be used up via function application, it needs to be an argument of some function that inputs entities!

Predicate Abstraction Rule (PA)

If α is a branching node whose daughters are β_i and γ, where β_i is a binder index i (left behind by movement) or a relative pronoun bearing the index i, then for any assignment g,

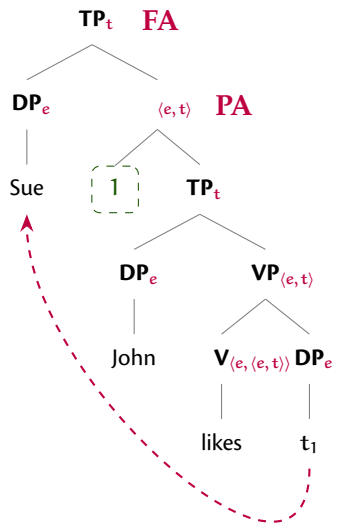
$$\left[\!\!\left[\begin{array}{ccc} & \alpha & \\ \beta_i & & \gamma \end{array} \right]\!\!\right]^g = \lambda x_e.\, [\![\gamma]\!]^{g[i \mapsto x]}$$

where for each object x, $g[i \mapsto x]$ is the modified assignment defined by:

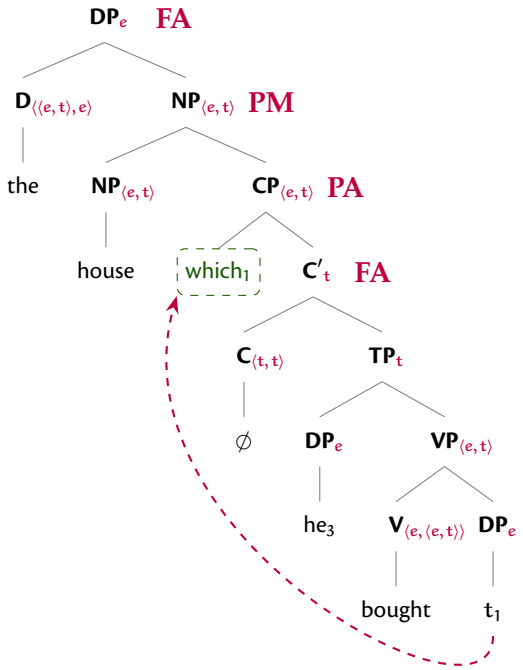
- for each index $j \neq i$, $g[i \mapsto x](j) = g(j)$;
- and $g[i \mapsto x](i) = x$.

When a DP moves, it leaves behind a **binder index** i right below its landing site, and a trace t_i in its original position. PA applies at the node where the binder index is a daughter. It builds a function $\lambda x_e.\, [\![\gamma]\!]^{g[i \mapsto x]}$, in which the trace t_i inside γ picks out whatever x is, no matter what g said about i. The moved DP then combines with that function by FA, so it gets used up. The same rule will handle relative clauses, where the relative pronoun $\text{who}_i/\text{which}_i$ plays the role of the binder. Here is our topicalized sentence (a), next to a relative clause (b) for comparison:

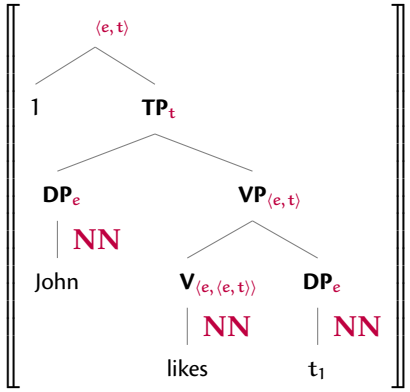
(a) Sue, John likes t_1



(b) the house which$_1$ he$_3$ bought t_1



Subproof. the sister of Sue, relative to g (where $g(1) = \text{Stella}$)

1. $\left[\!\!\left[\right.$  $\left.\right]\!\!\right]^g = \lambda x_e.\, [\![\text{TP}]\!]^{g[1\mapsto x]}$ (by PA)

2. $= \lambda x_e.\, [\![\text{VP}]\!]^{g[1\mapsto x]}([\![\text{John}]\!]^{g[1\mapsto x]})$ (by FA, NN)

3. $= \lambda x_e.\, [\![\text{likes}]\!]^{g[1\mapsto x]}([\![t_1]\!]^{g[1\mapsto x]})(\text{John})$ (by FA, NN, TN)

4. $= \lambda x_e.\, [\![\text{likes}]\!]^{g[1\mapsto x]}(g[1 \mapsto x](1))(\text{John})$ (by PR)

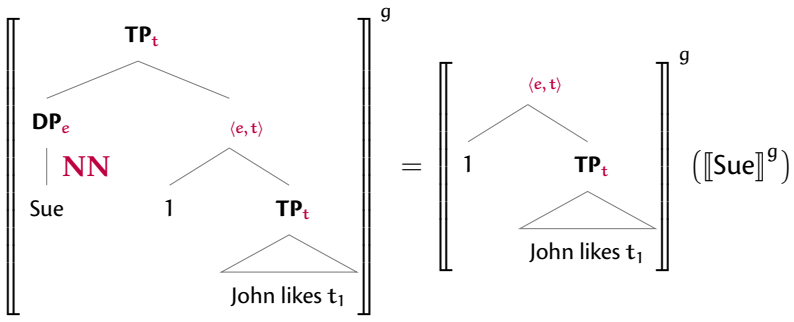
5. $= \lambda x_e.\, [\![\text{likes}]\!]^{g[1\mapsto x]}(x)(\text{John})$ (by def. of $g[1 \mapsto x]$)

6. $= \lambda x_e.\, [\lambda z_e.\, [\lambda y_e.\, \text{T iff } y \text{ likes } z]](x)(\text{John})$ (by TN)

7. $= \lambda x_e$. T iff John likes x (by β-reduction 2×)

PA is where the binding happens: it introduces λx *and* shifts the assignment to $g[1 \mapsto x]$. When the trace t_1 is interpreted, it looks up index 1 and finds x, the very variable that λx binds (dashed arrows).

Derivation. higher TP, relative to g

1. 

(by FA)

2. $= [\lambda x_e$. T iff John likes $x](\llbracket \text{Sue} \rrbracket^g)$ (by Subproof)

3. $= [\lambda x_e$. T iff John likes $x](\text{Sue})$ (by NN, TN)

4. = T iff John likes Sue (by β-reduction)

Practice

Can you derive the meaning of the relative clause the house which$_1$ he$_3$ bought?