Relative Clauses

LING 553

February 18-20, 2008

1. SIMPLE RELATIVE CLAUSES

(1) The college that is in Hanover is famous.[1] ≡ "Dartmouth is famous"

We want "the college that is in Hanover" to denote Dartmouth. So "college that is in Hanover" should denote {Dartmouth}.

This is easy, if not mindless:

- ⟦**college ~~that is~~ in Hanover**⟧ = ⟦**college in Hanover**⟧ = λx_e . x is a college and x is in Hanover

(2) The college that enrolled Robert Frost is in Hanover.

Again, this can be fairly straightforward:

- ⟦**college ~~that~~ graduated Robert Frost**⟧:
  ⟦**college**⟧ = λx_e . x is a college
  ⟦**graduated Robert Frost**⟧ = λx_e . x graduated Robert Frost
  ...once again, use Predicate Modification

2. HARDER RELATIVE CLAUSES

(3) The college [that Robert Frost attended] is in Hanover.

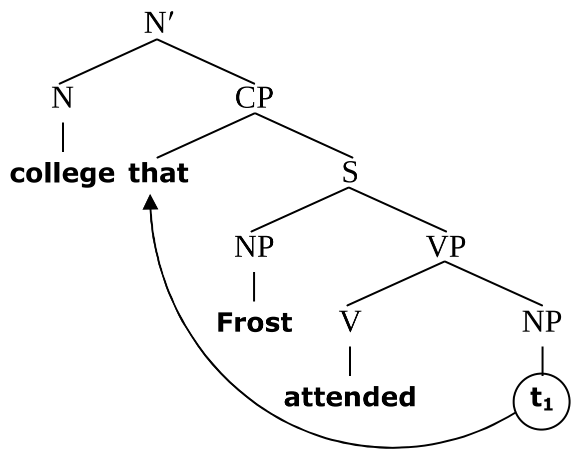
- ⟦**college ~~that~~ Robert Frost attended**⟧:
  ⟦**college**⟧ = λx_e . x is a college
  ⟦**Robert Frost attended**⟧ = λx_e . x attended Robert Frost ← **WRONG!**

What can we do?

[1] We'll take "Hanover" to denote Hanover, New Hampshire. There's only one college there.

2.1. A First Revision of Our System

Answer: *movement* and *traces*.



Because ⟦**attended**⟧ has type $\langle e, \langle e, t\rangle\rangle$, the trace must denote something of type e.

Or, at Aaron's suggestion, something higher. See appendix.

Suppose it's an arbitrary variable over individuals: x. Then working bottom-up:

⟦$\mathbf{t_1}$⟧ = x
⟦**attended**⟧ = $[\lambda z_e \,.\, \lambda y_e \,.\, y$ attended $z]$

⟦**attended** $\mathbf{t_1}$⟧ = $[\lambda z_e \,.\, \lambda y_e \,.\, y$ attended $z](x) = [\lambda y_e \,.\, y$ attended $x]$
⟦**Frost attended** $\mathbf{t_1}$⟧ = $[\lambda y_e \,.\, y$ attended $x]$(Frost) = Frost attended x

And working top-down, we want:

⟦**college that Frost attended** $\mathbf{t_1}$⟧ = $\lambda x_e \,.\, x$ is a college and Frost attended x
⟦**that Frost attended** $\mathbf{t_1}$⟧ = λx . Frost attended x

So it looks like the relative pronoun **that** means "put a λx in front". That's a new rule of composition for us:

(4) **The Rule of Function Abstraction** (or λ-abstraction) – first draft
If we have a node α, and its daughters are β and a relative pronoun, then ⟦α⟧ = [λx . ⟦β⟧].

Except that...how did we know to put an x in there? We could look into the inner workings of β, but it's better to avoid doing that kind of thing.

Moreover: **such that Frost attended it** has roughly the same meaning as **that Frost attended $\mathbf{t_1}$**, so if ⟦**it**⟧ = ⟦$\mathbf{t_1}$⟧ = x, then we're fine...until we have

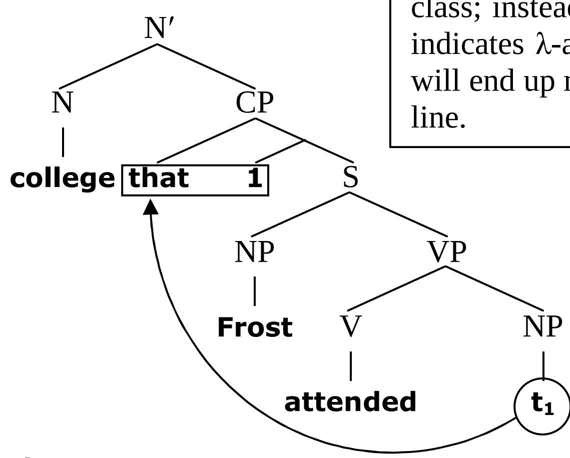
⟦**the college such that I read the book that $\mathbf{t_1}$ is about it**⟧ =

"the unique x where x is a college and I read the unique x where x is a book and x is about x", and boy is that wrong. (The unique thing that's both a book and about itself?)

2.2. A Re-Revision of Our System

What we really need is a way of keeping track of which variable is which. So let's number our traces and pronouns, i.e. "index" them.

> *Note*: this is slightly different from what we did in class; instead of **that$_1$** indicating λ-abstraction, **1** indicates λ-abstraction and **that** is vacuous. This will end up making things easier, farther down the line.



All right, but:
- What do the numbers mean?
- What does *t* mean?

Instead of having $\mathbf{t_1}$ denote an arbitrary individual, we'll have it denote "individual number one". Then:

(5) **The Rule of λ-Abstraction** – second draft
If we have a node α, and its daughters are β and an index n, then ⟦α⟧ = $[\lambda x_n$. ⟦β⟧$]$.

That gives:

⟦$\mathbf{t_1}$⟧ = x_1
⟦**attended**⟧ = $[\lambda z_e . \lambda y_e . y \text{ attended } z]$

⟦**attended $\mathbf{t_1}$**⟧ = $[\lambda z_e . \lambda y_e . y \text{ attended } z](x_1) = [\lambda y_e . y \text{ attended } x_1]$
⟦**Frost attended $\mathbf{t_1}$**⟧ = $[\lambda y_e . y \text{ attended } x_1](\text{Frost}) = \text{Frost attended } x_1$
⟦**1 Frost attended $\mathbf{t_1}$**⟧ = λx_1 . ⟦**Frost attended $\mathbf{t_1}$**⟧ = $\lambda x_1 . \text{Frost attended } x_1$

Which is what we wanted.

2.3. One More Revision of Our System

...sort of. How do we know who "individual number one" is? And what if we'd chosen $\mathbf{t_{23}}$? What we need is a way to assign individuals to numbers. So:

(6) *Assignments*
An assignment function is a function g : {integers} → D_e.

Some sample assignment functions:

$$g = \begin{bmatrix} 1 \to \text{George W. Bush} \\ 2 \to \text{Beatrice Santorini} \\ 3 \to \text{Dartmouth} \\ \vdots \\ 23 \to \text{Michael Jordan} \\ 24 \to \text{Michael Jackson} \\ 25 \to \text{Mary Jane Watson} \\ \vdots \end{bmatrix} \qquad g' = \begin{bmatrix} 1 \to \text{Michael Jackson} \\ 2 \to \text{Michael Jackson} \\ 3 \to \text{Michael Jackson} \\ \vdots \\ 23 \to \text{Michael Jackson} \\ 24 \to \text{Michael Jackson} \\ 25 \to \text{Michael Jackson} \\ \vdots \end{bmatrix}$$

We also need to say that trees are interpreted relative to assignment functions:

(7) *The interpretation function*
A tree α is in the domain of ⟦...⟧ iff, for all assignment functions g, g': $[\![\alpha]\!]^g = [\![\alpha]\!]^{g'}$.
If α is in the domain of ⟦...⟧, then for all assignment functions g, $[\![\alpha]\!] = [\![\alpha]\!]^g$.

...and that the meaning of a trace (or pronoun) is determined by the index, while the meaning of other things is assignment-function-independent:

(8) *The meaning of traces and pronouns*
a. If α is a trace with index i, then $[\![\alpha]\!]^g = g(i)$.
b. If α is a pronoun with index i, then $[\![\alpha]\!]^g = g(i)$.

That'll get us most of the way through a tree. Now we have:

$[\![\mathbf{t_1}]\!]^g = g(1)$
$[\![\textbf{attended}]\!] = [\lambda z_e \,.\, \lambda y_e \,.\, y \text{ attended } z]$

$[\![\textbf{attended } \mathbf{t_1}]\!]^g = [\![\textbf{attended}]\!]^g([\![\mathbf{t_1}]\!]^g) = [\![\textbf{attended}]\!]([\![\mathbf{t_1}]\!]^g)$
$= [\lambda z_e \,.\, \lambda y_e \,.\, y \text{ attended } z](g(1)) = [\lambda y_e \,.\, y \text{ attended } g(1)]$

$[\![\textbf{Frost attended } \mathbf{t_1}]\!]^g = [\lambda y_e \,.\, y \text{ attended } g(1)](\text{Frost}) = \text{Frost attended } g(1)$

Of course, ⟦**Frost attended t_1**⟧ is undefined. Why? Because, if we use the g, g' above:

⟦**Frost attended t_1**⟧^{g} = Frost attended $g(1)$ = Frost attended George W. Bush
⟦**Frost attended t_1**⟧$^{g'}$ = Frost attended $g'(1)$ = Frost attended Michael Jackson

which means that its meaning isn't the same for all assignment functions.

What we want at this point is a way to say, "Whatever assignment function we're using, I'll modify it so that it always maps '1' to the same individual, say, George W. Bush. That way, **Frost attended t_1** will always mean 'Frost attended George W. Bush', independent of the particular assignment function I started with."

(9) *Modified assignments*
If g is an assignment function, then $g[i \rightarrow x]$, also written $g^{x/i}$, is the assignment function h such that:
(i) $h(i) = x$, and
(ii) for all integers $j \neq i$, $h(j) = g(j)$.

How does *that* help? Well:

(10) **The Rule of λ-Abstraction (rule #3)**
If α has daughters β, γ, where β is an index i,
then for any assignment function g, $[\![\alpha]\!]^g = \lambda x : x \in D_e \,.\, [\![\gamma]\!]^{g[i \rightarrow x]}$

Putting this to work:

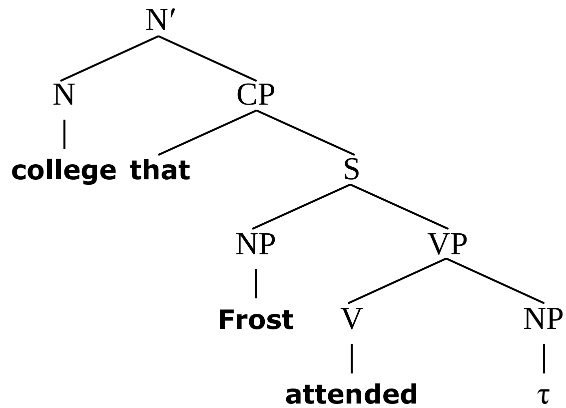
⟦**1 Frost attended t_1**⟧^{g}
$= \lambda x : x \in D_e \,.\, [\![\textbf{Frost attended t}_1]\!]^{g[1 \rightarrow x]}$
$= \lambda x : x \in D_e \,.\, [\![\textbf{attended}]\!]^{g[1 \rightarrow x]}([\![\textbf{t}_1]\!]^{g[1 \rightarrow x]})([\![\textbf{Frost}]\!]^{g[1 \rightarrow x]})$
$= \lambda x : x \in D_e \,.\, [\![\textbf{attended}]\!]([\![\textbf{t}_1]\!]^{g[1 \rightarrow x]})([\![\textbf{Frost}]\!])$
$= \lambda x : x \in D_e \,.\, [\lambda z_e \,.\, \lambda y_e \,.\, y \text{ attended } z](g[1 \rightarrow x](1))(\text{Frost})$
$= \lambda x : x \in D_e \,.\, [\lambda z_e \,.\, \lambda y_e \,.\, y \text{ attended } z](x)(\text{Frost})$
$= \lambda x : x \in D_e \,.\, \text{Frost attended } x$

Which is what we wanted.

3. APPENDIX: AARON'S τ

If there's an invisible τ with a higher meaning, could it take ⟦**attended**⟧ as an argument and return something that can take ⟦**Frost**⟧ and put it into the subject position?

Answer: sure...



$[\![\tau]\!] = \lambda R : R \in D_{\langle e, \langle e, t\rangle\rangle} . \lambda x : x \in D_e . \lambda y : y \in D_e . [R(y)(x)]$. This gives:

$$[\![\textbf{attended}]\!] = [\lambda x_e . \lambda y_e . y \text{ attended } x]$$

$$
\begin{aligned}
& [\![\text{VP}]\!] = [\![\tau]\!]([\![\textbf{attended}]\!]) \\
= & [\lambda R_{\langle e, \langle e, t\rangle\rangle} . \lambda x_e . \lambda y_e . [R(y)(x)]]([\lambda x_e . \lambda y_e . y \text{ attended } x]) \\
= & [\lambda R_{\langle e, \langle e, t\rangle\rangle} . \lambda x_e . \lambda y_e . [R(y)(x)]]([\lambda u_e . \lambda v_e . v \text{ attended } u]) \\
= & \lambda x_e . \lambda y_e . [[\lambda u_e . \lambda v_e . v \text{ attended } u](y)(x)] \\
= & \lambda x_e . \lambda y_e . [[\lambda v_e . v \text{ attended } y](x)] \\
= & \lambda x_e . \lambda y_e . x \text{ attended } y
\end{aligned}
$$

...which is ⟦**attended**⟧ with its arguments reversed. (Exercise: verify for yourself that this gives the right meaning for **college that Frost attended**.)

3.1. Problems

Why not do things this way?

- It's awfully powerful. Can we ensure that τ doesn't show up unexpectedly in trees, so that we actually have **Booth shot** τ **Lincoln**, with the meaning "Lincoln shot Booth"?
- The "gap" that the τ needs to fill happens to be in the object position of a transitive verb; but it can go elsewhere, e.g. indirect object position, subject position.... How many τs do we need?
- Are pronouns also τ? If so, does [the college such that I read a book that τ is about τ] work, or do we end up needing to somehow "index" each τ?