Chapter 5 – Relative Clauses

(Part 1) 5.1 – 5.2.3

Roberto Herrera, Anna Sykorova
May 7, 2013
Definitions

- Different relativization strategies

1. Defining the values

Map 122A shows relativization on subjects:

Values of Map 122A. Relativization on Subjects

- Relative pronoun: 12
- Non-reduction: 24
- Pronoun-retention: 5
- Gap: 125
- Total: 166

Map 123A shows the relativization on oblique:

Values of Map 123A. Relativization on Oblique

- Relative Pronoun Strategy: 13
- Non-Reduction Strategy: 14
- Pronoun-Retention Strategy: 20
- Gap Strategy: 55
- Not possible: 10
- Total: 112

(Comrie and Kuteva 2011)
Definitions cont.

- For Indo-European languages the most common strategy is the use of relative pronouns (and relative particles – e.g. dialectal German *wo*) (but generally speaking it is not the most common one)

- **English** uses relative pronouns such as: *who, which, whom, whose*, in relative clauses as the man who sold the book. *When* and *where* are sometimes called "relative adverbs", if they linking a relative clause to a main clause, e.g. *I went there the day (when) the accident happened,*

- *I remember the town where they filmed this movie.*
Detailed classification

free or headless, which have no head: What interests me is his motive

Paratactic relative clauses (relative looks like a declarative sentence with the full-fledged head):
That woman just passed by us, she helped me activate the card

Sentential: It’s said she baked this cake – which I just don’t believe;

and zero or contact relatives There’s the girl I saw earlier¹

non-restrictive (or non-defining) vs. restrictive (or defining)
The house which is empty is available
The house, which is empty, is available

¹Crystal (2008, pg 437)
Restrictive vs. non-restrictive

- We will deal with restrictive relative clauses using relative pronouns! Relative clauses are not *relative* in the sense that they are not absolute, as the pronoun used to construct them stands where the subject would in a sentence. **Restrictive clauses are a kind of intersective modifiers.** The difference between restrictive and non-restrictive is that **non-restrictive modifiers are assumed to be separate sentences.**

Thus:

- The house, which is empty, is available
- refers to – presupposes – one single house. The extension of "the" does not include the constituent between commas.
Restrictive relative clause - structure

Here, either the complementizer or the relative pronoun has to be deleted on the surface.
How do we manage to get the same derivation
〚 which is empty 〛 = 〚 empty 〛?

We know that we want the whole CP to get a value in $D_{<e,t>}$ and we also know that the VP inside denotes the same function as the CP. In (1) it will denote the characteristic function of the set of empty elements.

We could ignore “which”, “that” and the trace and treat them as semantically vacuous and assume that the semantic value is just passed from VP to S, to C’ and then to CP.

We could do that here but not in general.
Object traces

- What about object traces?
- (2) should denote the function $\lambda x \in D_e$. John abandoned $x$
- But this is NOT a value of any of its subtrees (Heim and Kratzer 98, p. 90)
- Should we really ignore the trace? If not what is its semantic value?
If in (2) the trace was treated as a semantically vacuous element, the entire relative clause would denote the characteristic function of the set of all individuals who abandoned John -- with John being the one abandoned. So we can’t treat the trace as semantically vacuous.

For instance, in “the job (which) John abandoned t ”, the trace is said to get its denotation from "which", which in turn gets it from the head (“job”). The whole DP then denotes an element.
“But if we said that the trace inherits its denotation from the head (through the CP), we would get into a vicious circle, since the denotation of the whole is supposed to be built up from the denotations of its parts.” (Heim and Kratzer 98, p. 91)

How are we then supposed to work with this restrictive construction?

Should we reconsider our structure?
Traces cont.

No, because what would the CP denote? In this structure, it couldn’t be a function of type \(<e,t>\) because then the whole DP would denote a truth value and not an individual. (Heim and Kratzer 98, p. 91)

**We need the trace to denote an individual.**
Solution

- Solution: working with variables!
- Variable:

“A variable denotes an individual, but only relative to a choice of an assignment of a value.” (Heim and Kratzer 98, p. 92)

- What is the "value assignment for a variable"?
"An assignment is an individual (element of D (=D_e))
The denotation of ‘t’ under the assignment ‘Texas’ is Texas
\[
\left[ t \right]^{\text{Texas}} = \text{Texas}
\]“ (Heim and Kratzer 98, p. 92)
We must allow the denotations of larger phrases that contain traces to be assignment-relative as well.

So what happens when we have sentences that already have truth conditions per se, like “John abandoned Fred.”?
Nothing relative about relative clauses

- We want our method to be as effective as possible, covering expressions such as "John abandoned Mary" as well as "John abandoned t".

- That’s why assignment-independent denotations are introduced through the following definition:

  **Assignment-Independent Denotation (AID):** For any tree α, α is in the domain of $〚 $ ] iff for all assignments a and b, $[α]^a = [α]^b$. If α is in the domain of $〚 $ ], then for all assignments a, $[α] = [α]^a$

- This changes nothing for lexical entries, for example: For any assignment a, $〚 \text{smokes} ⟩^a = 〚 \text{smokes} ⟩ = \lambda x ∈ D_e. x \text{smokes}$
Updated principles

- **Lexical Terminals**
  - If is a terminal node occupied by a lexical item, then $[\alpha]$ is specified in the lexicon.
  - Differentiating these Terminal Nodes from the ones with traces...
  - If $\alpha$ is a trace, then, for any assignment $a$, $[\alpha]^a = a$.

- **Non-Branching Nodes (NN)**
  - If $\alpha$ is a non-branching node and $\beta$ its daughter, then, for any assignment $a$, $[\alpha]^a = [\beta]^a$.

- **Functional Application (FA)**
  - If $\alpha$ is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment $a$, if $[\beta]^a$ is a function whose domain contains $[\gamma]^a$, then $[\alpha]^a = [\beta]^a([\gamma]^a)$.

- **Predicate Modification (PM)**
  - If $\alpha$ is a branching node and $\{\beta, \gamma\}$ the set of its daughters, then, for any assignment $a$, if $[\beta]^a$ and $[\gamma]^a$ are both functions of type $<e,t>$, then $[\alpha]^a = \lambda x \in D . [\beta]^a(x) = [\gamma]^a(x) = 1$. 
Denotation of a relative clause

How do we then get the denotation of the whole relative clause from “which” and “John abandoned t”?
Predicate abstraction! \textsuperscript{NEW}

- **Predicate Abstraction (PA)**
  
  If \( \alpha \) is a branching node whose daughters are a relative pronoun and \( \beta \), then \( \llbracket \alpha \rrbracket = \lambda x \in \mathcal{D}. \llbracket \beta \rrbracket x \)

- We use \( \lambda \)-notation to define the semantic value of \( \alpha \) as a function, this enables us to continue "using" the moved relative pronoun in the computation, without giving it a specific semantic value.
Do not mix up denotations applied to assignments with those under assignments:

\[
\text{〚\text{smokes}〛}^{\text{Ann}} \neq \text{〚\text{smokes}〛}(\text{Ann})
\]

- left side is a type \(<e,t>\) function; right side is the application of such function to an individual: a truth value (equals 0 or 1)

\[
\text{〚\text{Ann kisses t〛} \times \text{makes sense}} \text{ and stands for a truth value, depending on who is the individual } x
\]

\[
\text{〚\text{Ann kisses t〛}(x) \text{ makes absolutely no sense}, \text{ because:}
\]

\[
\text{〚\text{Ann kisses t〛} \text{ is not per sé in the domain of } \text{〚〛}, \text{ as it does not have any assignment-independent semantic value.}
\]

- Even if it did, its denotation would be a truth value (0/1), not a function. We would need to write something like \(1(x)\) or \(0(x)\).
The house which is empty is available.
The children (who) Pedro adopted.
References


