Local contexts and anaphora.

Keny Chatain (MIT)

September 26, 2017

1 Introduction

1.1 What's the problem?

- Co-variation without binding/c-command?[1]

(1) Frollo$_{\mathbf{j}}$ owned a donkey$_i$. He cherished it$_i$.

(2) Every farmer who owns a donkey$_i$ cherishes it$_i$.

Dynamic Approach:

1. words, constituents have *context-change potentials* ;
2. this potential is defined in their lexical entry ;
3. truth is defined in terms of context-change potential (failure as falsity)

(3) a. Frollo owned a donkey.
CCP: update the current context so that it contains Frollo as **j** and a donkey that he owns as **i**

b. He cherished it.
CCP: fail if the the referent for **i** does not cherish the referent for **j**.

- Quantifiers lexically specify how their arguments should update the context.

(4) a. Every **A B**.

b. **CCP:** for each x such that $\mathbf{A}(x)$ can update the context, update the current context with $\mathbf{A}(x)$ then update the context with $\mathbf{B}(x)$.
Fail if one of these updates fails.

Theoretical problem: monstruous items and universality

- One can easily construct monstruous lexical items.

[1] The main contribution of this talk is to "*take the violence out of donkey sentences*", in line with the research program initiated by Frank Staniszewski.

(5) **Quantifiers disallowing donkey pronouns**

a. Monster-Every **A B**

b. **CCP:** for each x such that $\mathbf{A}(x)$ can update the context, update the context with $\mathbf{B}(x)$. Fail if one of these updates fail.

- DS needs to posit lexical contraints so that such meanings are excluded cross-linguistically.

Empirical problem: uncomplying propositional logics.

- Negation operator is a "*test*" it can never update context with new referents.

(6) Not **A**
CCP: Fail if **A** updates the context. Update trivially if **A**'s update fails.

(7) Not Not **A**
CCP: Update trivially if **A** updates the context. Fail if **A** fails to update the context.

- But double negations can introduce referents:

(8) Alex: I was told that you didn't send Grandma a card for her birthday.
Lucas: It's not true that I didn't send Grandma a card for her birthday. It just arrived late.

- Similarly, disjunction in DS is not "*internally dynamic*": it does not pass referent from one disjunct to the next.
- But disjunction can pass referents from one disjunct to the next.

(9) Either it's false that there are bathrooms or they are not where she said they were.

1.2 Schlenker's witness-based proposal

- After processing a sentence or a portion of sentence, the assignment function maps i to the strongest restriction that one can add to a variable in the sentence without changing the truth conditions.

(10) Find the strongest R such that:
a donkey λx. Frollo owns $x \Leftrightarrow$ a donkey λx. Frollo owns $x \cap R$

(11) $R =$ donkeys owned by Frollo
so the new assignement function g' is such that $g'(i) = R$

- An *existential closure* principle is assumed to ensure that the indefinite really extends its scope.

(12) $\exists x \in g'(i)$, Frollo cherishes it$_{\mathrm{x}}$.

- The full analysis accounts for a variety of phenomena: donkey anaphora, paycheck sentences, functional dependencies of all sort, non-dynamic behaviours of negation and disjunction ...

Blindness of the semantics

- If indefinites are treated on a par with all other GQs, trouble ensues:

(13) Frollo owns every donkey.
R = all donkeys

(14) # Frollo owns every donkey. It brayed loudly.
= Frollo owns every donkey. Some donkey brayed loudly.

- One needs to distinguish variables introduced by indefinites from variables intoduced by universal quantifiers.
- An *ad hoc* move is good enough but can one derive this from an independent principle ?

$\exists/\forall$ reading

(15) Every farmer who owns a donkey spoils it.

a. $\forall$: every donkey-owning farmer spoils **all** of his donkeys

b. $\exists$: every donkey-owning farmer spoils **some** of his donkeys

- It is easy to construct dynamic entries to get the two readings.

(16) Every **A B**

a. for each x such that $\mathbf{A}(x)$ can update the context, update the context with $\mathbf{A}$(x) then $\mathbf{B}$(x). Fail if one update fails.

b. for each x such that $\mathbf{A}(x)$ can update the context, fail if update with $\mathbf{A}(x)$ then with (not $\mathbf{B}(x)$) succeeds.

- Within Schlenker's system, the *existential closure* principle only derives $\exists$ reading

Desiderata

➢ Independence: lexical items do not encode their effect on context.
$\rightarrow$ presented today

➢ Left-to-right bias
$\rightarrow$ presented today

➢ Predicts non-dynamic behaviours of *negation* and *disjunction*.
$\rightarrow$ presented today

➢ Differentiates between *every* and *a*.
$\rightarrow$ presented today

➢ Predicts $\exists/\forall$ distinction
$\rightarrow$ sketch of an idea in workshop

2 Alternative presentation

2.1 Basic assumptions

- Each node has a meaning and updates the context. This is reflected in the notation.

(17) a. **H&K notation:** $\llbracket \mathbf{T} \rrbracket^g = m$

b. **New notation:** $\begin{pmatrix} \mathbf{T} & \xmapsto{\llbracket\cdot\rrbracket} & m \\ g & \longrightarrow & g' \end{pmatrix}$

- The **left-to-right bias** is implemented in functional application ; this is the only principle that regulates left-to-right bias.

Functional Application

If

$$\begin{pmatrix} \mathbf{T} & \xmapsto{\llbracket\cdot\rrbracket} & f \\ g & \longrightarrow & g' \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \mathbf{T}' & \xmapsto{\llbracket\cdot\rrbracket} & x \\ g' & \longrightarrow & g'' \end{pmatrix}$$

Then:

$$\begin{pmatrix} [\,\mathbf{T}\ \ \mathbf{T}'\,] & \xmapsto{\llbracket\cdot\rrbracket} & f(x) \\ g & \to & g'' \end{pmatrix}$$

(18) a. Ada$_i$ walked in and she$_i$ ordered a beer.

b. She$_i$ ordered a beer and Ada$_i$ walked in.

(19) a. $\begin{pmatrix} \text{Ada walked in} & \xmapsto{\llbracket\cdot\rrbracket} & \texttt{walked-in}'(\texttt{ada}') \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{ada}'] \end{pmatrix}$

b. $\begin{pmatrix} \text{She}_i \text{ ordered a beer} & \xmapsto{\llbracket\cdot\rrbracket} & \texttt{ordered-a-beer}'(\underbrace{g[\boldsymbol{i} \to \texttt{ada}'](i)}_{\texttt{ada}'}) \\ g[\boldsymbol{i} \to \texttt{ada}'] & \longrightarrow & g[\boldsymbol{i} \to \texttt{ada}'] \end{pmatrix}$

c. $\begin{pmatrix} \text{(18b)} & \xmapsto{\llbracket\cdot\rrbracket} & (\exists! x, \texttt{person}'(x) \wedge \texttt{w-in}'(x)) \wedge \texttt{ordered-a-beer}'(\texttt{ada}') \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{ada}'] \end{pmatrix}$

d. FA does not apply the other way arounf for (18b)

- Lexical items do not encode effect on context.

(20) $\begin{pmatrix} \text{word} & \overset{[\![\cdot]\!]}{\longmapsto} & \texttt{word}' \\ g & \longrightarrow & g \end{pmatrix}$

Simple updates

- Any meaning of type e can update the assignment function.

Referent Introduction

If

$$\begin{pmatrix} \mathbf{T} & \overset{[\![\cdot]\!]}{\longmapsto} & x \\ g & \longrightarrow & g' \end{pmatrix} \quad \text{and} \quad x \text{ is type } e$$

Then:

$$\begin{pmatrix} \mathbf{T} & \overset{[\![\cdot]\!]}{\longmapsto} & x \\ g & \longrightarrow & g'[\boldsymbol{i} \to x] \end{pmatrix}$$

- This is the only principle that regulates referent introduction.

Summary

- Functional application requires the first node to update the context first (**left-to-right bias**)
- Lexical items do not encode any effect on context (**independence**).
- Any meaning of type e may be added to the context.

2.2 Indefinites and non-determinism

Non-determinism

(21) A woman_{i} walked in.

- If several women walked in (f. ex. *Mary* and *Ada*), the assignment function cannot be updated with a single individual.

(22) Do we update g to:

a. $g[i \to \texttt{mary}']$?

b. $g[i \to \texttt{ada}']$?

- Standard DS solution: consider updates to sets of assignments.
- Here, for simplicity, we'll assume that indices can map to sets of individuals[2].

[2]Using assignment functions that map to sets will yield problematic results when considering sentences with more than one indefinite. The strictly more powerful DS notion of context can be implemented within my system to account for those cases.

(23) Update g to $g[\boldsymbol{i} \to \texttt{mary}' \vee \texttt{ada}']$

- $p_1 \vee \ldots \vee p_n \overset{\text{def}}{=} \{p_1, \ldots, p_n\}$
- **Intuition:** the hearer does not know which particular referent is intended.
- Since $g(i)$ is a set of individuals, the interpretation of pronouns has to be revised. This revision leads to an account of the $\exists/\forall$ distinction.
  $\to$ not discussed here

Indefinites

- Here, I am going to adopt an alternative semantics for indefinites.
- Charlow (2014, 2017) have argued that exceptional scope and its restrictions are a natural consequence of the alternative semantics of indefinites.
- Here, I argue that non-deterministic referent introduction is a natural consequence of alternative semantics.
- **Implementation:** one and the same node may have several meanings.[3]

(24) a. $\begin{pmatrix} \text{a woman} & \xmapsto{[\![\cdot]\!]} & \texttt{ada}' \\ g & \longrightarrow & g \end{pmatrix}$

b. $\begin{pmatrix} \text{a woman} & \xmapsto{[\![\cdot]\!]} & \texttt{mary}' \\ g & \longrightarrow & g \end{pmatrix}$

c. ...

- Using the referent introduction rule, we can add the meaning of the indefinites to the assignment function.

(24') a. $\begin{pmatrix} \text{a woman} & \xmapsto{[\![\cdot]\!]} & \texttt{ada}' \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{ada}'] \end{pmatrix}$

b. $\begin{pmatrix} \text{a woman} & \xmapsto{[\![\cdot]\!]} & \texttt{mary}' \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{mary}'] \end{pmatrix}$

c. ...

Meaning postulate for the indefinite
For all choice functions f:

$$\begin{pmatrix} \text{a/some} & \xmapsto{[\![\cdot]\!]} & f \\ g & \longrightarrow & g \end{pmatrix}$$

[3]This is the simplest way of proceeding given what I said so far, but it will ultimately be insufficient since it forces us to stipulate the effect of alternative-taking operators as syncategorematic rules. Ultimately, it would be good to have a uniform principle to derive the effect of those operators on context without stipulation.

- As of now, alternatives percolate up the tree, without obstacles.
  → *we need to be able to limit the scope of the alternatives.*

(25) $\left(\begin{array}{lcl} \text{It's not the case that a woman walked in} & \xmapsto{[\![\cdot]\!]} & \neg(\texttt{walked-in}'(\texttt{ada}')) \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{ada}'] \end{array}\right)$

- We define the operator $\downarrow$ to existentially close t-type alternatives.
- This operator is the one that creates assignments to multiple individuals.
- (If g and g' are assignment functions defined on non-overlapping sets of indices, their union is denoted $g \cdot g'$)

Existential closure
For $\mathbf{T}$ of type t and g an assignment function, let $S =$

$$\left\{ \left(\begin{array}{ccc} \mathbf{T} & \xmapsto{[\![\cdot]\!]} & m_n \\ g & \longrightarrow & g \cdot g'_n \end{array} \right) \middle| \, n \in N \right\}$$

be the sets of meanings associated with node $\mathbf{T}$ under assignment function g. Then:

$$\left(\begin{array}{lcl} [\downarrow \ \mathbf{T}] & \xmapsto{[\![\cdot]\!]} & \exists n,\ m_n \\ g & \longrightarrow & g \cdot \left(i \mapsto \bigvee_{\substack{n \in N \\ m_n \text{ is } \mathbf{true}}} g'_n(i) \right) \end{array} \right)$$

- This operator only collects referents that were introduced by **true alternatives**. As a result, false existential statements have no effect on the context at all[4]!

Uncomplying behaviours in propositional logic

(26) It's not true that I didn't send Grandma a card for her birthday. It just arrived too late.
   a. not (not **A**) and **B**

- If **A** is true, then a card is introduced in the context (27).
  If **A** is false, then the assignment function is left as is (27b).

(27) a. $\left(\begin{array}{ccc} \text{I send Grandma a card} & \xmapsto{[\![\cdot]\!]} & \exists x,\ \texttt{card}'(x) \wedge \texttt{send}'(\texttt{gm}')(x)(\texttt{I}') = \mathbf{true} \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{card-I-sent}'] \end{array}\right)$

   b. $\left(\begin{array}{ccc} \text{I send Grandma a card} & \xmapsto{[\![\cdot]\!]} & \exists x,\ \texttt{card}'(x) \wedge \texttt{send}'(\texttt{gm}')(x)(\texttt{I}') = \mathbf{false} \\ g & \longrightarrow & g \end{array}\right)$

[4]Of course, we should ultimately revise this extensional fragment to world variables. After we discuss functional dependencies, it should be clear that in an intensional fragment, indefinites introduce partial individual concepts.

- The negation of a statement has the same context-change potential as its assertion.

(28) a. $\left(\begin{array}{ccc} \text{I didn't send Grandma a card} & \xmapsto{[\![\cdot]\!]} & \neg\exists x,\ \texttt{card}'(x) \wedge \texttt{send}'(\texttt{gm}')(x)(\texttt{I}') = \textbf{false} \\ g & \longrightarrow & g[\boldsymbol{i} \to \texttt{card-I-sent}'] \end{array}\right)$

b. $\left(\begin{array}{ccc} \text{I didn't send Grandma a card} & \xmapsto{[\![\cdot]\!]} & \neg\exists x,\ \texttt{card}'(x) \wedge \texttt{send}'(\texttt{gm}')(x)(\texttt{I}') = \textbf{true} \\ g & \longrightarrow & g \end{array}\right)$

- If **A** is true, then not (not **A**) introduces a referent and the pronoun receives a correct interpretation **B**.
- If **A** is false, then not (not **A**) does not introduce a referent and the pronoun triggers presupposition failure in **B**.
  → by standard rules of presupposition projection, the conjunction is **false** (since the first conjunct is **false**).
- Similarly for "(not **A**) or B" cases

Summary

- ➢ The system achieves **independence**, **left-to-right bias** and **non-determinism**
- ➢ **Non-determinism** is created when existential closure is applied on a set of alternatives.
- ➢ The standard rule for pronoun interpretation won't do but I'll keep it for the rest of the talk.
- ➢ In the next section, we show how to derive cases of **easy donkey sentences**, where the indefinite only ever has one witness ; pronouns can then be interpreted as in H&K.

2.3 Easy donkey sentences

Functional Dependencies

- A context should deal with *functional dependencies*.

(29) Every boy saw a marshmallow$_{\text{i}}$ on his plate ; no boy left **it**$_{\text{i}}$ on the plate.

- The meaning of *it* covaries with the boy.
- Following Schlenker, I assume that assignment functions can provide *unsaturated individuals* (e-ending types)[5].

(30) $g(i) = \lambda x.$ the marshmallow that x saw (type ee)

- Pronouns have complex subscripts.

(31) Every boy λ_j t_j saw exactly one marshmallow$_{\text{i}}$ on his plate ;
no boy λ_j t_j left $\textbf{it}_{\mathbf{i}(t_j)}$ on the plate.

[5]An alternative representation of these dependencies is plural assignments as in Brasoveanu. This representation is better suited to account for plural dependencies. Schlenker's presentation is used here because it is more intuitive.

- Textbook λ-abstraction written in new notation:

λ-Abstraction rule
If for all x, there is a meaning m_x such that:

$$\left(\begin{array}{ccc} \mathbf{T} & \overset{[\![\cdot]\!]}{\longmapsto} & m_x \\ & g[i \to x] & \end{array}\right)$$

Then:

$$\left(\begin{array}{ccc} [\,\lambda_i \;\; \mathbf{T}\,] & \overset{[\![\cdot]\!]}{\longmapsto} & \lambda x.\ m_x \\ & g & \end{array}\right)$$

- The new *dynamic* rule should reflect discourse referents that could have been introduced in each of the sub-contexts.
- Updates introduced in the scope of a λ-abstractor are preserved in the form of *unsaturated individuals.*
- **New λ-abstraction:**

λ-Abstraction rule
If for all x, there is a meaning m_x and an assignment g'_x such that:

$$\left(\begin{array}{lcl} \mathbf{T} & \overset{[\![\cdot]\!]}{\longmapsto} & m_x \\ g[i \to x] & \to & g[i \to x] \cdot g'_x \end{array}\right)$$

Then:

$$\left(\begin{array}{lcl} [\,\lambda_i \;\; \mathbf{T}\,] & \overset{[\![\cdot]\!]}{\longmapsto} & \lambda x.\ m_x \\ g[i \to x] & \to & g \cdot (i \mapsto \lambda x :\ i \in g'_x.\ g'_x(i)) \end{array}\right)$$

- To picture it:

$$\begin{array}{rcl} g[i \to \texttt{marc}'] & \xrightarrow{\quad m_{\texttt{john}'} \quad} & g[i \to \texttt{marc}', j \to d_1] \\ & \vdots & \\ g[i \to \texttt{amy}'] & \xrightarrow[\quad m_{\texttt{amy}'} \quad]{} & g[i \to \texttt{amy}'] \\ g[i \to \texttt{polly}'] & \xrightarrow[\quad m_{\texttt{polly}'} \quad]{} & g[i \to \texttt{polly}', j \to d_2] \end{array}$$

Updates for subcontexts

$$g \xrightarrow{\quad \lambda x.\ m_x \quad} g\left[j \to \left\{\begin{array}{ll} \texttt{marc}' & \mapsto \;\; d_1 \\ \texttt{polly}' & \mapsto \;\; d_2 \end{array}\right.\right]$$

Update after λ-abstraction

Application to easy donkey anaphora

(32) a. Every farmer who owns only one donkey cherishes it.

b. Every farmer [who λ_j t_j owns only one donkey$_{\mathtt{i}}$] [λ_j t_j cherishes it$_{\mathtt{i}(t_j)}$]

1. The relative clause is an existential statement ; it introduces referents iff it is true.

$$\begin{pmatrix} t_j \text{ owns only one donkey} & \xmapsto{[\![\cdot]\!]} & \exists! y,\ \mathtt{donkey}'(y) \wedge \mathtt{owns}'(y)(x) \\ g[j \to x] & \to & \begin{cases} g[j \to x] \text{ if } x \text{ doesn't own exactly one donkey} \\ g[j \to x, i \to \text{the donkey that } x \text{ owns}] \text{ else} \end{cases} \end{pmatrix}$$

2. The λ-abstraction rule preserves the donkeys that were introduced, in the form of a partial function.

$$\begin{pmatrix} \lambda_j.\ t_j \text{ owns only one donkey} & \xmapsto{[\![\cdot]\!]} & \lambda x.\ \exists! y,\ \mathtt{donkey}'(y) \wedge \mathtt{owns}'(y)(x) \\ g & \to & g[i \to \lambda x : x \text{ owns only one donkey. the donkey that } x \text{ owns}] \end{pmatrix}$$

3. Functional modification (and predicate modification) applies ; none of these have any effect on the assignment function.

$$\begin{pmatrix} \text{every farmer} \ldots \text{ donkey} & \xmapsto{[\![\cdot]\!]} & \mathtt{every}'(\mathtt{donkey\text{-}owning\text{-}farmer}') \\ g & \to & g[i \to \lambda x : (\ldots) \,.\, \text{the donkey that } x \text{ owns}] \end{pmatrix}$$

4. The part of the nuclear scope that is below the abstractor has to be evaluated wrt an assignment function that contains the *unsaturated* donkey.

$$\begin{pmatrix} t_j \text{ cherishes it}_{\mathtt{i}(t_j)} & \xmapsto{[\![\cdot]\!]} & \begin{array}{l} \mathtt{cherishes}'\, [g'(j)]\, [g'(i)(g'(j))] \\ = x \text{ cherishes the donkey that } x \text{ owns} \end{array} \\ g' = g \begin{bmatrix} j \to x, \\ i \to \lambda x : (\ldots) \,.\, \text{the donkey } x \text{ owns} \end{bmatrix} & \to & g \begin{bmatrix} j \to x, \\ i \to \lambda x : (\ldots) \,.\, \text{the donkey } x \text{ owns} \end{bmatrix} \end{pmatrix}$$

5. Nuclear scope does not introduce referents ; the λ-abstraction functions just like the textbook version.

$$\begin{pmatrix} \lambda_j.\ t_j \text{ cherishes it}_{\mathtt{i}(t_j)} & \xmapsto{[\![\cdot]\!]} & \lambda x.\ x \text{ cherishes the donkey that } x \text{ owns} \\ g[i \to \lambda x : (\ldots) \,.\, \text{the donkey } x \text{ owns}] & \to & g[i \to \lambda x : (\ldots) \,.\, \text{the donkey } x \text{ owns}] \end{pmatrix}$$

6. FA yields the right reading for *easy donkey sentences.*

$$\begin{pmatrix} (32) & \xmapsto{[\![\cdot]\!]} & \forall x,\ (\mathtt{donkey\text{-}owning\text{-}farmer}'(x)) \to (\mathtt{cherishes}'(\mathtt{the\text{-}d\text{-}x\text{-}owns}')(x)) \\ g & \to & g[i \to \lambda x : (\ldots) \,.\, \text{the donkey } x \text{ owns}] \end{pmatrix}$$

3 Conclusion and open questions

- In this talk, we dealt with an **easy problem**: obtaining context-change potentials from general universally available principles, to achieve only minimal empirical coverage.
- The goal is to extend this to more intriguing and problematic cases of anaphora: $\exists/\forall$ readings, plurals, etc.
- The former problem can be restated as the question how to define the correct interpretation rule for pronouns in an enriched context (ideas presented in workshop).