

Ellipsis: introduction

(87) John slept. Mary did, too.

- ellipsis analysis 1: **reconstructing** a semantically and pragmatically relevant constituent (mostly, a VP) that has only been deleted phonologically
 - this is the characteristic approach in Generative Grammar
 - e.g., in Sag (1976) or Fiengo and May (1994)
 - Sag's (1976) analysis of ACD cases still is one of the cornerstones for the assumption of LF
- ellipsis analysis 2: determining the meaning of a highly underspaced sentence with the help of a **structural similarity** to another sentence (parallelism)
 - this approach characterises computational approaches
- most difficult ellipsis cases were introduced in Gawron and Peters (1990)



Higher-order unification (HOU) and ellipsis: introduction 1

- basic intuition: elliptical sentences (TS) are **parallel** (structurally identical) to a source sentence (SS)
in (87) is the SS *John slept*, the TS, *Mary did, too*
- SS and TS differ w.r.t. **parallel elements**; in (87), *John* and *Mary*
- the SS meaning is the same as the TS meaning, if the semantic contribution of the SS parallel element is replaced by the one of its respective TS argument
- for (87) – simplified:

$[[\textit{John}]] = \mathbf{j}$ $[[\textit{Mary}]] = \mathbf{m}$

$[[\textit{John slept}]] = \mathbf{sleep}'(\mathbf{j})$

$[[\textit{Mary did, too}]] = \mathbf{sleep}'(\mathbf{m})$



HOU and ellipsis: introduction 2

- problem: how to ‘reconstruct’ the semantic contribution of a parallel element?
- during semantic construction, it has been **integrated** with the semantics of the rest of the sentence
- **higher-order unification** (HOU) can reformulate the fact that SS and TS mean the same modulo the parallel elements
 - there is a relation P which, when applied to the semantics of the parallel elements in the SS, is equal to the meaning of the SS
 - but the SS meaning is known
 - the meaning of the TS is then P applied to the semantics of the parallel elements of the TS



HOU and ellipsis: introduction 3

- for (87), we look for a P such that $\mathbf{sleep}'(\mathbf{j}) = P(\mathbf{j})$
- i.e., $P = \mathbf{sleep}'$
- the TS meaning is then $P(\mathbf{m}) = \mathbf{sleep}'(\mathbf{m})$
- such equations can be solved with specific algorithms, e.g., the one of Huet (1975)
- such algorithms can also cope with parallel elements that contribute higher-order semantic contributions, which were subsequently β -reduced
- this is necessary for cases like *Every man slept. Mary did too.*



HOU and ellipsis: introduction 4

- the proper treatment of (87) is thus:

$$\llbracket \text{John} \rrbracket = \lambda P.P(\mathbf{j})$$

$$\llbracket \text{Mary} \rrbracket = \lambda P.P(\mathbf{m})$$

$$\llbracket \text{John slept} \rrbracket = [\lambda P.P(\mathbf{j})](\mathbf{sleep}') = \mathbf{sleep}'(\mathbf{j})$$

$$\llbracket \text{Mary did, too} \rrbracket = [\lambda P.P(\mathbf{m})](Q) = Q(\mathbf{m})$$

- the equation to be solved is the following:

$$[\lambda P.P(\mathbf{j})](Q) = \mathbf{sleep}'(\mathbf{j})$$

- the derivation of the solution: $Q = \mathbf{sleep}'$ is much more difficult this time
- $P(\mathbf{j}) = \mathbf{sleep}'(\mathbf{j})$ has another, unwanted solution, viz., $\lambda x.\mathbf{sleep}'(\mathbf{j})$
- this would yield a false interpretation of the TS:

$\mathbf{sleep}'(\mathbf{j})$



HOU and ellipsis: anaphora 1

- this approach allows the derivation of the ‘strict’ and the ‘sloppy’ interpretation of ellipses that comprise anaphors
- the anaphor must have the SS parallel element as its antecedent

(88) Max loves his mother and Bill does too

- there are several solutions for the equations:

$$P(\mathbf{m}) = \mathbf{love}'(\mathbf{m}, \mathbf{mother-of}'(\mathbf{m}))$$

$$P = \lambda x. \mathbf{love}'(x, \mathbf{mother-of}'(\mathbf{m})) \text{ oder}$$

$$P = \lambda x. \mathbf{love}'(x, \mathbf{mother-of}'(x))$$

- this yields the following readings of the TS:

$$\mathbf{love}'(\mathbf{b}, \mathbf{mother-of}'(\mathbf{m})) \text{ oder}$$

$$\mathbf{love}'(\mathbf{b}, \mathbf{mother-of}'(\mathbf{b}))$$



HOU and ellipsis: anaphora 2

- but there are **unwanted solutions**, too

$$P = \lambda x. \text{love}'(\mathbf{m}, \text{mother-of}'(\mathbf{m}))$$

$$P = \lambda x. \text{love}'(\mathbf{m}, \text{mother-of}'(x))$$

- Dalrymple, Shieber, and Pereira (1991) distinguish ‘**primary**’ and ‘**secondary**’ occurrences of the semantic contribution of the parallel element of the SS
- every solution of the equation must **abstract over primary occurrences** of the SS parallel element
- they express this by underlining primary occurrences:

$$P(\mathbf{j}) = \text{sleep}'(\underline{\mathbf{j}})$$



HOU and ellipsis: anaphora 3

- the difference between primary and secondary occurrences can be formalised by **coloured unification** (Gardent and Kohlhase 1996)
 - the semantic contribution of parallel elements comprises a constant with the colour p_0 (primary occurrence)
 - anaphors that have this occurrence of the variable as their antecedent are modelled with the same constant, but this constant has the colour $\neg p_0$
 - the variable P in the TS (semantics of the pro-form) is of the colour $\neg p_0$
 - instantiations of variables with a colour f must be **f -monochrome**

$$P_{\neg p_0}(\mathbf{m}) = \mathbf{love}'(\mathbf{m}_{p_0}, \mathbf{mother-of}'(\mathbf{m}_{\neg p_0}))$$

i.e., $P_{\neg p_0} = \lambda x. \mathbf{love}'(x, \mathbf{mother-of}'(\mathbf{m}_{\neg p_0}))$ is possible

$P_{\neg p_0} = \lambda x. \mathbf{love}'(\mathbf{m}_{p_0}, \mathbf{mother-of}'(x))$ is ruled out



HOU and ellipsis: cascaded ellipsis 1

- an instance of the interaction of ellipsis and anaphora: **cascaded ellipsis**

(89) John revised his paper before the teacher did and Bill did, too

- five readings (whose papers are revised)

JJJJ/JJBJ/JJBB/JTJT/JTBT; but not JJJB

- DSP predict all six readings

- meaning of the sentence, according to Dalrymple et al. (1991)

before'[**revise'**(**j**, **paper-of'**(**j**)), $P(\mathbf{t})$] $\wedge Q(\mathbf{b})$

- equations:

$P(\mathbf{j}) = \mathbf{revise}'(\underline{\mathbf{j}}, \mathbf{paper-of}'(\mathbf{j}))$

$Q(\mathbf{j}) = \mathbf{before}'[\mathbf{revise}'(\underline{\mathbf{j}}, \mathbf{paper-of}'(\mathbf{j})), P(\mathbf{t})]$



HOU and ellipsis: cascaded ellipsis 2

- case 1: $P = \lambda x.\text{revise}'(x, \text{paper-of}'(\mathbf{j}))$
→ all readings derived by this solution are 'JJ**'
- $Q(\mathbf{j})$ is then $\text{before}'[\text{revise}'(\underline{\mathbf{j}}, \text{paper-of}'(\mathbf{j})), \text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{j}))]$
- three occurrences of \mathbf{j} , the first must always be abstracted: four subcases
- subcase 1.1: $Q = \lambda x.\text{before}'[\text{revise}'(x, \text{paper-of}'(\mathbf{j})), \text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{j}))]$
yields reading JJJJ:

$\text{before}'[\text{revise}'(\mathbf{j}, \text{paper-of}'(\mathbf{j})),$
 $\quad \text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{j}))] \wedge$
 $\text{before}'[\text{revise}'(\mathbf{b}, \text{paper-of}'(\mathbf{j})),$
 $\quad \text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{j}))]$



HOU and ellipsis: cascaded ellipsis 3

- subcase 1.2: $Q = \lambda x.\mathbf{before}'[\mathbf{revise}'(x, \mathbf{paper-of}'(x)), \mathbf{revise}'(t, \mathbf{paper-of}'(j))]$

yields reading JJBJ

- subcase 1c: $Q = \lambda x.\mathbf{before}'[\mathbf{revise}'(x, \mathbf{paper-of}'(j)), \mathbf{revise}'(t, \mathbf{paper-of}'(x))]$

yields reading JJJB

- subcase 1d: $Q = \lambda x.\mathbf{before}'[\mathbf{revise}'(x, \mathbf{paper-of}'(x)), \mathbf{revise}'(t, \mathbf{paper-of}'(x))]$

yields reading JJBB



HOU and ellipsis: cascaded ellipsis 4

- case 2: $P = \lambda x.\text{revise}'(x, \text{paper-of}'(x))$
→ all readings derived from that solution are 'JT**'
- $Q(\mathbf{j})$ is then
 $\text{before}'[\text{revise}'(\underline{\mathbf{j}}, \text{paper-of}'(\mathbf{j})),$
 $\text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{t}))]$
- subcase 2a: $Q = \lambda x.\text{before}'[\text{revise}'(x, \text{paper-of}'(\mathbf{j})),$
 $\text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{t}))]$

yields reading JTJT

- subcase 2b: $Q = \lambda x.\text{before}'[\text{revise}'(x, \text{paper-of}'(x)),$
 $\text{revise}'(\mathbf{t}, \text{paper-of}'(\mathbf{t}))]$

yields reading JJBT



HOU and ellipsis: quantifiers 1

- reconsider Hirschbühler sentences:

(90) John gave every student a test and Bill did, too.

- SS and TS have two readings each, (90) as a whole, only three
- problem: the scope in SS and TS is the same but not yet specified

- DSP combine HOU with a treatment of scope in the style of Hobbs and Shieber (1986):

(91) John greeted every person

- the underlying representation:

$\langle \forall x. \mathbf{person}'(x) \rangle \vdash \mathbf{greet}'(\mathbf{j}, x)$

- discharging the quantifier returns:

$\vdash \forall x. \mathbf{person}'(x) \rightarrow \mathbf{greet}'(\mathbf{j}, x)$



HOU and ellipsis: quantifiers 2

- the **order of discharging quantifiers and resolving ellipses** determines different readings

- underlying representation of (90):

$$\langle \exists x.\mathbf{test}'(x) \rangle, \langle \forall y.\mathbf{student}'(y) \rangle \vdash \mathbf{give}'(\mathbf{j}, x, y) \wedge P(\mathbf{b})$$

- if discharging comes first, quantifier scope is the same in SS and TS

– case 1:

discharging quantifiers (existential first)

$$\forall y.\mathbf{student}'(y) \rightarrow \exists x.\mathbf{test}'(x) \wedge \mathbf{give}'(\mathbf{j}, x, y) \wedge P(\mathbf{b})$$

solving the equation

$$P = \lambda z.\forall y.\mathbf{student}'(y) \rightarrow \exists x.\mathbf{test}'(x) \wedge \mathbf{give}'(z, x, y)$$

resulting reading

$$\forall y.\mathbf{student}'(y) \rightarrow \exists x.\mathbf{test}'(x) \wedge \mathbf{give}'(\mathbf{j}, x, y) \wedge$$

$$\forall y.\mathbf{student}'(y) \rightarrow \exists x.\mathbf{test}'(x) \wedge \mathbf{give}'(\mathbf{b}, x, y)$$



HOU and ellipsis: quantifiers 3

- if discharging comes first, quantifier scope is the same in SS and TS

– case 1 (ctd):

discharging quantifiers (universal first)

$$\exists x.\mathbf{test}'(x) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{j}, x, y) \wedge P(\mathbf{b})$$

solving the equation

$$P = \lambda z.\exists x.\mathbf{test}'(x) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(z, x, y)$$

resulting reading

$$\exists x.\mathbf{test}'(x) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{j}, x, y) \wedge$$

$$\exists x.\mathbf{test}'(x) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{b}, x, y)$$

- case 2: resolving the equation first (2 unattested readings)

solving the equation: $P = \lambda z.\mathbf{give}'(z, x, y)$

resulting reading, if universal quantifier is discharged first

$$\exists x.\mathbf{test}'(x) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{j}, x, y) \wedge \mathbf{give}'(\mathbf{b}, x, y)$$



HOU and ellipsis: quantifiers 4

- case 3: resolution after discharging of one quantifier (one attested reading [universal quantifier first], one reading unattested [existential first])

discharging the universal quantifier

$$\langle \exists x.\mathbf{test}'(x) \rangle \vdash \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{j}, x, y) \wedge P(\mathbf{b})$$

solving the equation

$$P = \lambda z.\forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(z, x, y)$$

discharging the existential quantifier

$$\exists x.\mathbf{test}'(x) \wedge (\forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{j}, x, y) \wedge \forall y.\mathbf{student}'(y) \rightarrow \mathbf{give}'(\mathbf{b}, x, y) \wedge)$$



HOU and ellipsis: quantifiers 5

- this also covers ACD cases

(92) John greeted everyone that Bill did

$\langle \forall x.\mathbf{person}'(x) \wedge P(\mathbf{b}) \rangle \vdash \mathbf{greet}'(\mathbf{j}, x)$

- resolve first

$$P = \lambda z.\mathbf{greet}'(z, x)$$

- then discharge:

$$\forall x.\mathbf{person}'(x) \wedge \mathbf{greet}'(\mathbf{b}, x) \rightarrow \mathbf{greet}'(\mathbf{j}, x)$$

- the other order would yield a not well-formed expression, as the variable P occurs on either side:

$$P(\mathbf{j}) = \forall x.\mathbf{person}'(x) \wedge P(\mathbf{b}) \rightarrow \mathbf{greet}'(\mathbf{j}, x)$$

- the ban against variables on both sides of the equation is the same strategy as the ban in CLLS against infinite trees

