Semantics 1

May 24, 2012

Gerhard Jäger

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

The copula verb *be*

Different uses of *be*

(1) Tully is Cicero. $\rightsquigarrow$ predicative is proper noun

(2) Cicero is a politician. $\rightsquigarrow$ predicative is indefinite NP

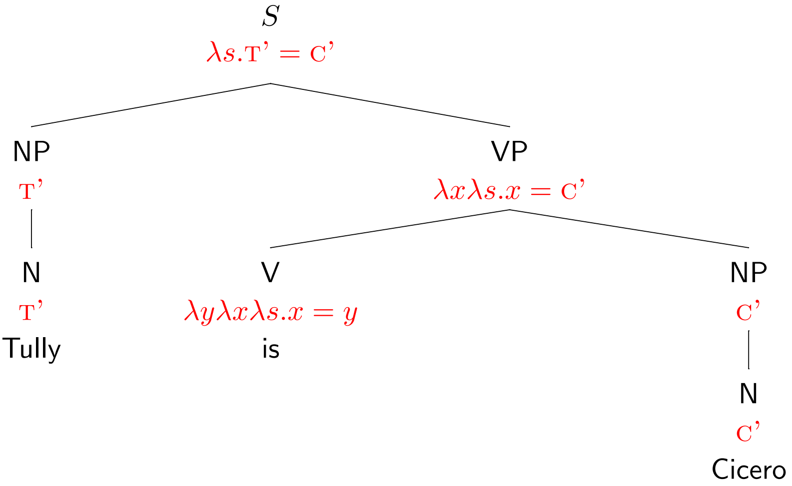
(3) Cicero is in Rome. $\rightsquigarrow$ predicative is PP

(4) Cicero is old. $\rightsquigarrow$ predicative is AP

Equative *be*

(1) Tully is Cicero.

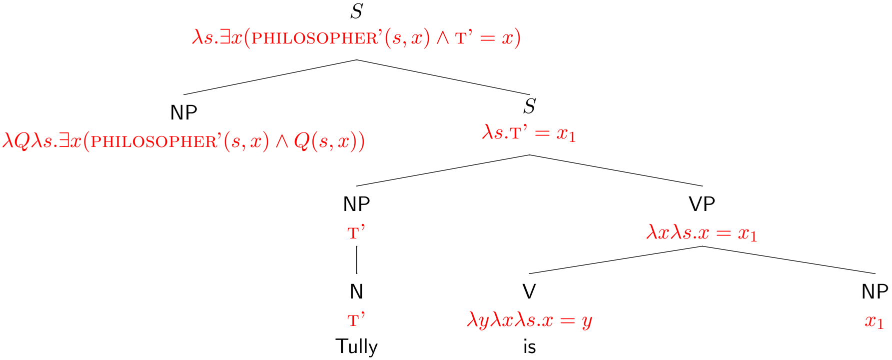
$$\text{is} \rightsquigarrow \lambda y \lambda x \lambda s.x = y$$



Equative *be*

Equative be also accounts for quantifiers in predicative position.

(1) Tully is a philosopher.

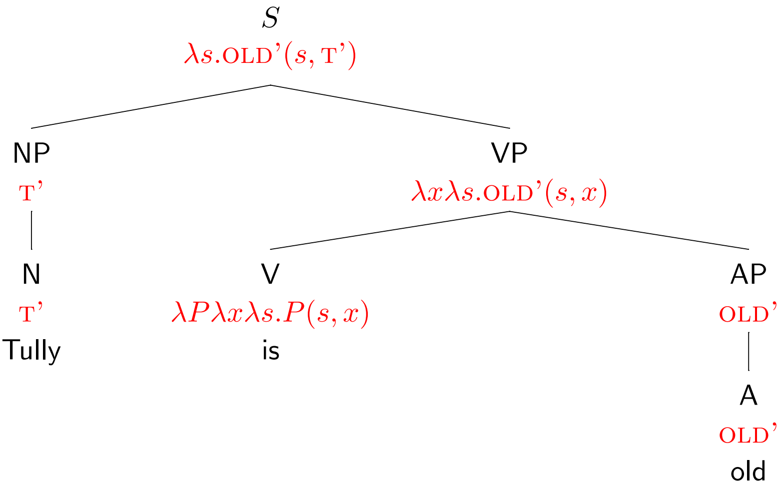


$$\lambda s.\exists x(\text{PHILOSOPHER'}(s,x) \wedge \text{T'} = x) \equiv \lambda s.\text{PHILOSOPHER'}(s,\text{T'})$$

Predicative *be*

(1) Tully is old.

$$\text{is} \rightsquigarrow \lambda P \lambda x \lambda s. P(s, x)$$



Predicative and attributive use of adjectives

- predicative use:

  (1) Tully is old. $\rightsquigarrow \lambda s.\text{OLD'}(s, \text{T'})$
- attributive use:

  (2) old man $\rightsquigarrow \lambda x \lambda s.\text{MAN'}(s, x) \wedge \text{OLD'}(s, x)$
- attributive use involves logical conjunction $\wedge$ that is missing in predicative use
- Where does this semantic content come from?

The syntactic solution

- Syntax: $\mathrm{NP}_1 \to \mathrm{AP}, \mathrm{NP}_2$
- Semantics: $\|\mathrm{NP}_1\| = \lambda x \lambda s.\|\mathrm{NP}_2\|(s, x) \wedge \|\mathrm{AP}\|(s, x)$
- Disadvantage:
  - does not work for all attributive adjectives:

(1) fake doctor
(2) alleged winner
(3) imaginary singers

The lexical solution

Lexical rule

If the lexicon contains an adjective A with the meaning

$$\lambda\vec{y}\lambda x\lambda s.\alpha(s,x)$$

for some predicate α, then the lexicon also contains an adjective A with the meaning

$$\lambda\vec{y}\lambda P\lambda x\lambda s.P(s,x)\wedge\alpha(s,x,\vec{y})$$

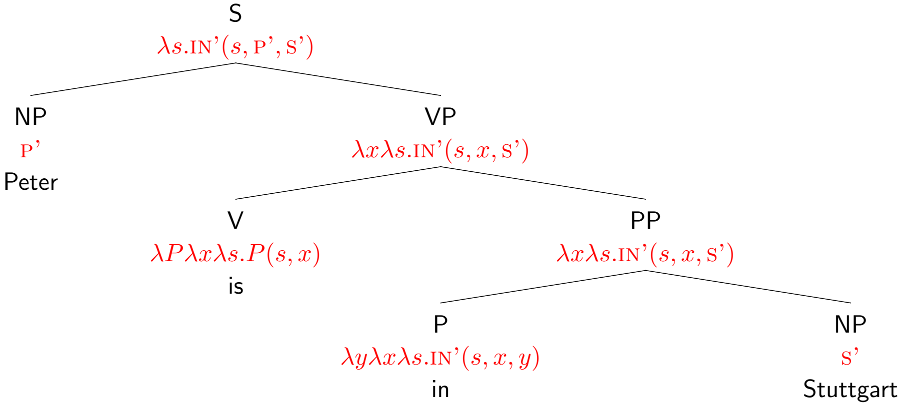
There is no consensus which solution is correct. In this course we will work with the lexical solution.

[0](The notation $\vec{y}$ represents a (possibly empty) sequence of additional arguments.)

Prepositions

- Just like APs, PPs have a predicative and a attributive use (plus an adverbial use, that will not be covered here)
- same systematic relationship between predicative and attributive use as above:
  - $\text{in}_{\text{pred}} \rightsquigarrow \lambda y \lambda x \lambda s.\text{IN'}(s, x, y)$
  - $\text{in}_{\text{attr}} \rightsquigarrow \lambda y \lambda P \lambda x \lambda s.P(s, x) \wedge \text{IN'}(s, x, y)$

Predicative use

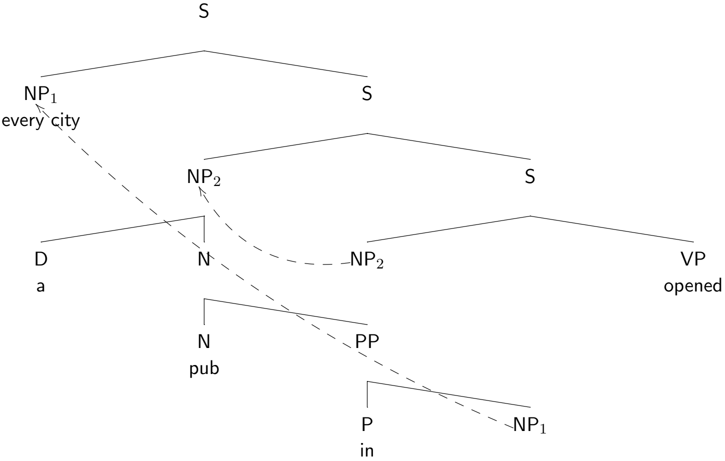


Attributive use

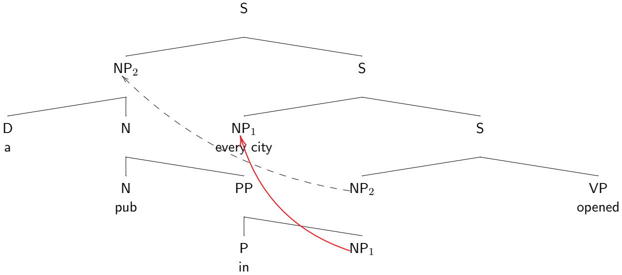
- N — $\lambda x \lambda s.\text{PUB'}(s,x) \wedge \text{IN'}(s,x,\text{S'})$
  - N — $\lambda x \lambda s \text{PUB'}(s,x)$
    - pub
  - PP — $\lambda P \lambda x \lambda s.P(s,x) \wedge \text{IN'}(s,x,\text{S'})$
    - P — $\lambda y \lambda P \lambda x \lambda s.P(s,x) \wedge \text{IN'}(s,x,y)$
      - in
    - NP — S'
      - Stuttgart

Inverse linking

(1) A pub in every city opened. $\rightsquigarrow$
$\lambda s.\forall y(\text{CITY'}(s,y) \rightarrow \exists x(\text{PUB'}(s,x) \wedge \text{IN'}(s,x,y) \wedge \text{OPEN'}(s,x)))$



- if we do QR in the reverse order...

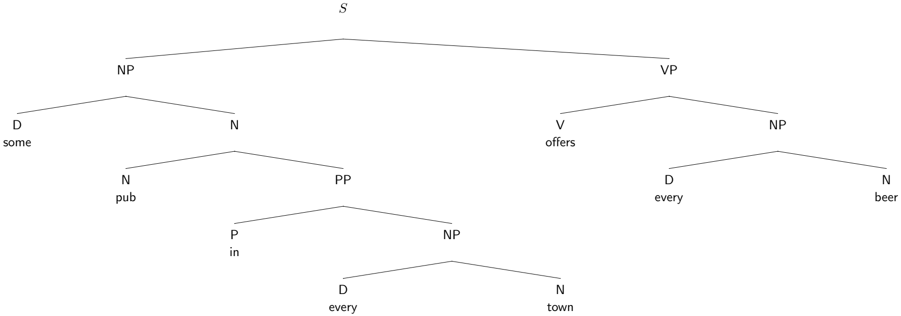


- NP$_1$ (every city) ends up not c-commanding its trace $\Rightarrow$ illicit movement!
- semantics would come out as
  $\lambda s.\exists x(\text{PUB'}(s,x) \wedge \text{IN'}(s,x,x_1) \wedge \forall y(\text{CITY'}(s,y) \rightarrow \text{OPEN'}(s,x)))$
- unbound variable (corresponds to non-c-commanded trace)

Inverse Linking

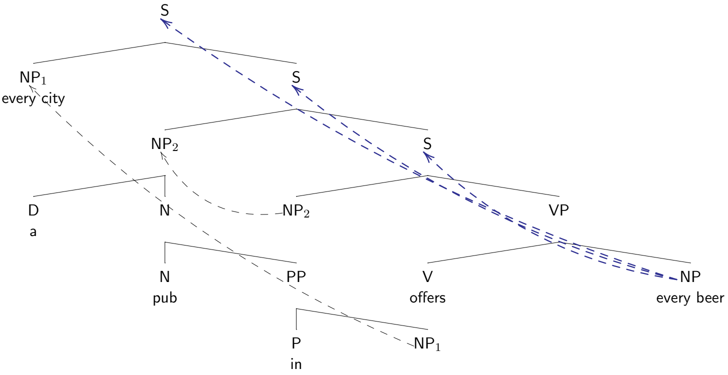
(1) Some pub in every town offers every beer.

S-Structure



Inverse linking

LF 1/2/3



Inverse Linking

(1) Some pub in every town offers every beer.

- with our current tools, we can derive three readings:
  - $\lambda s.\forall z(\textsc{beer'}(s,z) \to \forall y(\textsc{town'}(s,y) \to \exists x(\textsc{pub'}(s,x) \wedge \textsc{in'}(s,x,y) \wedge \textsc{offer'}(s,x,z))))$
  - $\lambda s.\forall y(\textsc{town'}(s,y) \to \forall z(\textsc{beer'}(s,z) \to \exists x(\textsc{pub'}(s,x) \wedge \textsc{in'}(s,x,y) \wedge \textsc{offer'}(s,x,z))))$
  - $\lambda s.\forall y(\textsc{town'}(s,y) \to \exists x(\textsc{pub'}(s,x) \wedge \textsc{in'}(s,x,y) \wedge \forall z(\textsc{beer'}(s,z) \to \textsc{offer'}(s,x,z))))$
- two more readings are possible but cannot be derived so far:
  - $\lambda s.\forall z(\textsc{beer'}(s,z) \to \exists x(\textsc{pub'}(s,x) \wedge \forall y(\textsc{town'}(s,y) \to \textsc{in'}(s,x,y)) \wedge \textsc{offer'}(s,x,z)))$
  - $\lambda s.\exists x(\textsc{pub'}(s,x) \wedge \forall y(\textsc{town'}(s,y) \to \textsc{in'}(s,x,y)) \wedge \forall z(\textsc{beer'}(s,z) \to \textsc{offer'}(s,x,z)))$