Topic 5

Quantification
Quantificational expressions

- *Everybody* praised Mary.
- *Everybody but Tina* praised Mary.
- One can find it *everywhere*.
- John *rarely* wears a cap.
- John *most often* wears a cap.
- We are *far from* Beijing.
- There is *a lot of* work to do today.
- *Everybody* needs *some* help *sometimes*.
- *Some* representatives of *every* department in *most* companies saw *a few* samples of *every* product
Taking stock

- utterance
- syntactic structure
- logical formula
- model
Tina is tall and thin
Taking stock

```
S
   /\  
  NP  VP
     /   /
    |   |   
   Tina V AP
      |   /   |
      is   A   C   A
           tall  and  thin
```
Taking stock

IS (AND TALL THIN) TINA = ($\lambda px. p x$) ($\lambda px. p x$) (AND TALL THIN) TINA

$\Rightarrow_\beta$ ($\lambda x. AND TALL THIN x$) TINA

$\Rightarrow_\beta$ AND TALL THIN TINA

= ($\lambda pqx. (p x) \land (q x)$) TALL THIN TINA

$\Rightarrow_\beta$ ($\lambda qx. (TALL x) \land (q x)$) THIN TINA

= ($\lambda qx. ((\lambda x. tall x) x) \land (q x))$ THIN TINA

$\Rightarrow_\beta$ ($\lambda qx. (tall x) \land (q x)$) THIN TINA

$\Rightarrow_\beta$ ($\lambda x. (tall x) \land (THIN x))$ TINA

= ($\lambda x. (tall x) \land ((\lambda x. thin x) x)$) TINA

$\Rightarrow_\beta$ ($\lambda x. (tall x) \land (thin x))$ TINA

$\Rightarrow_\beta$ (tALL TINA) \land (tINtina)

= (tall tina) \land (thin tina)
Taking stock

utterance → syntactic structure → logical formula → model

tall

E

tina

thin
Syntax-semantics interface

utterance → syntactic structure → logical formula → model
Syntax-semantics interface

- syntactic structure
  - type-theoretic abstract syntax
    - logical formula
Syntax-semantics interface

Syntactic structure

Type-theoretic abstract syntax

Logical formula

Logical constants

S is (and tall thin) tina

NP tina

Tina

VP is (and tall thin)

V is

AP and tall thin

A tall

C and

A thin
Syntax-semantics interface

syntactic structure → type-theoretic abstract syntax → logical formula

S IS (AND TALL THIN) TINA

NP TINA

Tina

VP IS (AND TALL THIN)

V IS

is

AP AND TALL THIN

A TALL

tall

A THIN

thin

A AND

and
Syntax-semantics interface

Syntactic structure -> type-theoretic abstract syntax -> logical formula

**Syntax**

**Semantics**

**Interface**

**Syntactic structure**

- **S**: is (and tall thin) Tina
- **NP**: Tina
  - **V**: is
    - **is**: is
      - **A**: tall
        - **tall**: tall
      - **C**: and
        - **and**: and
      - **A**: thin
        - **thin**: thin

**Type-theoretic abstract syntax**

- **is**: AP (NP S)
- **and**: AP (AP AP)
- **tall**: AP
- **thin**: AP
- **TINA**: NP

**Logical formula**
Syntax-semantics interface

```
IS : AP (NP S)
AND : AP (AP AP)
TALL : AP
THIN : AP
TINA : NP
```
Syntax-semantics interface

\[
\text{IS} : \text{AP (NP S)} \\
\text{AND} : \text{AP (AP AP)} \\
\text{TALL} : \text{AP} \\
\text{THIN} : \text{AP} \\
\text{TINA} : \text{NP}
\]

\[
\begin{align*}
S &:= t \\
\text{NP} &:= e \\
\text{AP} &:= e \, t
\end{align*}
\]
Syntax-semantics interface

\[
\begin{align*}
\text{TINA} & := \text{tina} \\
\text{TALL} & := \lambda x. \text{tall} \; x \\
\text{THIN} & := \lambda x. \text{thin} \; x \\
\text{AND} & := \lambda p q x. (p \; x) \land (q \; x) \\
\text{IS} & := \lambda p x. p \; x \\
\text{S} & := t \\
\text{NP} & := e \\
\text{AP} & := e \; t
\end{align*}
\]
Noun phrases (naive interpretation)
Noun phrases (naive interpretation)

Abstract syntax:

TINA : NP
MARY : NP
PRAISED : NP NP S
Noun phrases (naive interpretation)

Abstract syntax:

TINA : NP
MARY : NP
PRAISED : NP NP S

Semantic interpretation:
Noun phrases (naive interpretation)

Abstract syntax:

\[
\begin{align*}
\text{TINA} & : \text{NP} \\
\text{MARY} & : \text{NP} \\
\text{PRAISED} & : \text{NP NP S}
\end{align*}
\]

Semantic interpretation:

\[
\begin{align*}
\text{NP} & := e \\
\text{S} & := t
\end{align*}
\]
Noun phrases (naive interpretation)

Abstract syntax:

\[
\begin{align*}
Tina & : \text{NP} \\
Mary & : \text{NP} \\
\text{Praised} & : \text{NP NP S}
\end{align*}
\]

Semantic interpretation:

\[
\begin{align*}
\text{NP} & := e \\
\text{S} & := t
\end{align*}
\]

\[
\begin{align*}
Tina & := tina \\
Mary & := mary \\
\text{Praised} & := \lambda xy. \text{praised} y x
\end{align*}
\]

where:

\[
\begin{align*}
tina, mary & : e \\
\text{praised} & : e e t
\end{align*}
\]
Quantified noun phrases

► Tina praised Mary.
► Everybody praised Mary.
► Nobody praised Mary.
► Tina praised somebody.
► Everybody praised somebody.
► Everybody ran.
Generalized quantifiers & type raising
Generalized quantifiers & type raising

- Remember that $(\text{e t}) t$ is the type of the logical constants \textit{all} ($\forall$) and \textit{exists} ($\exists$).
Generalized quantifiers & type raising

- Remember that \((e \; t) \; t\) is the type of the logical constants \(\text{all } (\forall)\) and \(\text{exists } (\exists)\).

- Every term of type \((e \; t) \; t\) is called a generalized quantifier.
Generalized quantifiers & type raising

- Remember that \((e \ t) \ t\) is the type of the logical constants \(\text{all } (\forall)\) and \(\text{exists } (\exists)\).
- Every term of type \((e \ t) \ t\) is called a generalized quantifier.
- Semantically, a generalized quantifier corresponds to a set of sets of entities.
Generalized quantifiers & type raising

► Remember that \((e \text{ t}) \text{ t}\) is the type of the logical constants \textit{all} \((\forall)\) and \textit{exists} \((\exists)\).

► Every term of type \((e \text{ t}) \text{ t}\) is called a generalized quantifier.

► Semantically, a generalized quantifier corresponds to a set of sets of entities.

► The expected meaning of "everybody ran" might be captured by the following formula:

\[
\forall x. (\text{human } x) \rightarrow (\text{ran } x)
\]
Generalized quantifiers & type raising

- Remember that \((e \; t) \; t\) is the type of the logical constants \(\text{all} \; (\forall)\) and \(\text{exists} \; (\exists)\).

- Every term of type \((e \; t) \; t\) is called a generalized quantifier.

- Semantically, a generalized quantifier corresponds to a set of sets of entities.

- The expected meaning of “everybody ran” might be captured by the following formula:

\[
\forall x. \; (\text{human } x) \rightarrow (\text{ran } x)
\]

- Accordingly, the following \(\lambda\)-term, which is of type \((e \; t) \; t\), is a good candidate for the interpretation of “everybody”:

\[
\lambda p. \; \forall x. \; (\text{human } x) \rightarrow (p \; x)
\]
Generalized quantifiers & type raising

Syntax/semantics interface:

TINA : NP
MARY : NP
EVERYBODY : NP
SOMEBODY : NP
RAN : NP S
PRAISED : NP NP S

Semantic interpretation:

NP := (e t) t
S := t

TINA := ?
MARY := ?
EVERYBODY := λk. ∀x. (human x) → (k x)
SOMEBODY := λk. ∃x. (human x) ∧ (k x)
RAN := ?
PRAISED := ?
Proper names as generalized quantifiers
The interpretations of *TINA* and *MARY* must be of type \((e \; t) \; t\).
Proper names as generalized quantifiers

- The interpretations of **TINA** and **MARY** must be of type \((e \cdot t) \cdot t\).

- Semantically, it means that we must characterize an entity using a set of sets of entities.
Proper names as generalized quantifiers

★ The interpretations of TINA and MARY must be of type \((e \cdot t) \cdot t\).

★ Semantically, it means that we must characterize an entity using a set of sets of entities.

★ \(\{ S \in \mathcal{P}(E) : \text{tina} \in S \}\)
The interpretations of TINA and MARY must be of type \((e \ t) \ t\).

Semantically, it means that we must characterize an entity using a set of sets of entities.

\[
\{ \ S \in \mathcal{P}(E) : \text{tina} \in S \}
\]

\[
\lambda S_{(e\ t)}. \ S \ \text{tina}
\]
Applying type-raising to verb arguments
Applying type-raising to verb arguments

The syntactic type of RAN is (NP S), the semantic interpretation of NP is (e t) t, and the one of S is t. Accordingly the type of the interpretation of RAN must be (((e t) t) t) t
Applying type-raising to verb arguments

- The syntactic type of RAN is \( \text{(NP S)} \), the semantic interpretation of \( \text{NP} \) is \( \text{(e t) t} \), and the one of \( \text{S} \) is \( \text{t} \). Accordingly the type of the interpretation of RAN must be \( \text{((e t) t) t} \)

- \( \text{RAN} := \lambda s. s (\lambda x. \text{ran } x) \)
Applying type-raising to verb arguments

- The syntactic type of RAN is \((NP \ S)\), the semantic interpretation of \(NP\) is \((e \ t) \ t\), and the one of \(S\) is \(t\). Accordingly the type of the interpretation of RAN must be \(((e \ t) \ t) \ t\).

- \(RAN := \lambda s. \ s (\lambda x. \ ran \ x)\)

- Similarly, the type of the interpretation of PRAISED must be \(((e \ t) \ t) (((e \ t) \ t) \ t)\).
Applying type-raising to verb arguments

- The syntactic type of RAN is (NP S), the semantic interpretation of NP is (e t) t, and the one of S is t. Accordingly the type of the interpretation of RAN must be ((e t) t) t

- RAN := λs. s (λx. ran x)

- Similarly, the type of the interpretation of PRAISED must be ((e t) t) ((e t) t) t.

- PRAISED := λos. s (λx. o (λy. praised x y))
Generalized quantifiers & type raising

Syntax/semantics interface:

TINA : NP
MARY : NP
EVERYBODY : NP
SOMEBODY : NP
RAN : NP S
PRAISED : NP NP S

Semantic interpretation:

NP := (e t) t
S := t

TINA := λk. k tina
MARY := λk. k mary
EVERYBODY := λk. ∀x. (human x) → (k x)
SOMEBODY := λk. ∃x. (human x) ∧ (k x)
RAN := λs. s (λx. ran x)
PRAISED := λo. λs. s (λx. o (λy. praised x y))
Generalized quantifiers & type raising

Tina praised somebody.
Generalized quantifiers & type raising

Tina praised somebody.

PRAISED SOMEBODY TINA
Generalized quantifiers & type raising

Tina praised somebody.

\[
\text{PRAISED SOMEBODY TINA}
= (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x \ y))) \text{ SOMEBODY TINA}
\]
Generalized quantifiers & type raising

Tina praised somebody.

\[ \text{PRAISED SOMEBODY TINA} \]
\[ = \left( \lambda o. \lambda s. s \left( \lambda x. o \left( \lambda y. \text{praised } x \ y \right) \right) \right) \text{SOMEBODY TINA} \]
\[ \rightarrow_{\beta} \left( \lambda s. s \left( \lambda x. \text{SOMEBODY} \left( \lambda y. \text{praised } x \ y \right) \right) \right) \text{ TINA} \]
Generalized quantifiers & type raising

Tina praised somebody.

\[
\text{PRAISED SOMEBODY TINA} = (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x y))) \text{ SOMEBODY TINA}
\]

\[
\rightarrow_\beta (\lambda s. s (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x y))) \text{ TINA}
\]

\[
\rightarrow_\beta \text{ TINA} (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x y))
\]
Generalized quantifiers & type raising

Tina praised somebody.

\[
\text{PRAISED SOMEBODY TINA} \\
= (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised} \ x \ y))) \text{ SOMEBODY TINA} \\
\rightarrow _\beta (\lambda s. s (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y))) \text{ TINA} \\
\rightarrow _\beta \text{ TINA} (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y)) \\
= (\lambda k. k \text{ tina}) (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y))
\]
Generalized quantifiers & type raising

Tina praised somebody.

\[
PRAISED\ \SOMEBODY\ \TINA \\
= \ (\lambda o. \ \lambda s. \ s \ (\lambda x. \ o \ (\lambda y. \ \text{praised} \ x \ y))) \ \SOMEBODY\ \TINA \\
\rightarrow_\beta \ (\lambda s. \ s \ (\lambda x. \ \SOMEBODY \ (\lambda y. \ \text{praised} \ x \ y))) \ \TINA \\
\rightarrow_\beta \ \TINA \ (\lambda x. \ \SOMEBODY \ (\lambda y. \ \text{praised} \ x \ y)) \\
= \ (\lambda k. \ k \ \text{tina}) \ (\lambda x. \ \SOMEBODY \ (\lambda y. \ \text{praised} \ x \ y)) \\
\rightarrow_\beta \ (\lambda x. \ \SOMEBODY \ (\lambda y. \ \text{praised} \ x \ y)) \ \text{tina}
\]
Generalized quantifiers & type raising

Tina praised somebody.

PRAISED SOMEBODY TINA

\[
\begin{align*}
\text{PRAISED SOMEBODY TINA} & \quad = \quad (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised} \ x \ y))) \ \text{SOMEBODY TINA} \\
\rightarrow_{\beta} & \quad (\lambda s. s (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y))) \ \text{TINA} \\
\rightarrow_{\beta} & \quad \text{TINA} (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y)) \\
\quad = \quad (\lambda k. k \ \text{tina}) (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y)) \\
\rightarrow_{\beta} & \quad (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised} \ x \ y)) \ \text{tina} \\
\rightarrow_{\beta} & \quad \text{SOMEBODY} (\lambda y. \text{praised tina} \ y)
\end{align*}
\]
Tina praised somebody.

**PRAISED SOMEBODY TINA**

\[
= (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x \ y))) \ \text{SOMEBODY TINA}
\]

\[
\rightarrow_\beta (\lambda s. s (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x \ y))) \ \text{TINA}
\]

\[
\rightarrow_\beta \ \text{TINA} (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x \ y))
\]

\[
= (\lambda k. k \ \text{tina}) (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x \ y))
\]

\[
\rightarrow_\beta (\lambda x. \text{SOMEBODY} (\lambda y. \text{praised } x \ y)) \ \text{tina}
\]

\[
\rightarrow_\beta \ \text{SOMEBODY} (\lambda y. \text{praised } \text{tina} \ y)
\]

\[
= (\lambda k. \exists x. (\text{human } x) \land (k \ x)) (\lambda y. \text{praised } \text{tina} \ y)
\]
Generalized quantifiers & type raising

Tina praised somebody.

PRAISED SOMEBODY TINA

= \((\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x y)))\) SOMEBODY TINA

\(\rightarrow_\beta (\lambda s. s (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y))) \) TINA

\(\rightarrow_\beta \text{TINA } (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y)))\)

= \((\lambda k. k \text{tina}) (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y)))\)

\(\rightarrow_\beta (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y))) \text{ tina}\)

\(\rightarrow_\beta \text{SOMEBODY } (\lambda y. \text{praised tina } y)\)

= \((\lambda k. \exists x. (\text{human } x) \land (k x)) (\lambda y. \text{praised tina } y)\)

\(\rightarrow_\beta \exists x. (\text{human } x) \land ((\lambda y. \text{praised tina } y) x)\)
Generalized quantifiers & type raising

*Tina praised somebody.*

\[
\text{PRAISED SOMEBODY TINA} = (\lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x y))) \text{ SOMEBODY TINA}
\]

\[
\rightarrow_\beta (\lambda s. s (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y))) \text{ TINA}
\]

\[
\rightarrow_\beta \text{ TINA } (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y))
\]

\[
= (\lambda k. k \text{ tina}) (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y))
\]

\[
\rightarrow_\beta (\lambda x. \text{SOMEBODY } (\lambda y. \text{praised } x y)) \text{ tina}
\]

\[
\rightarrow_\beta \text{ SOMEBODY } (\lambda y. \text{praised tina } y)
\]

\[
= (\lambda k. \exists x. (\text{human } x) \land (k x)) (\lambda y. \text{praised tina } y)
\]

\[
\rightarrow_\beta \exists x. (\text{human } x) \land ((\lambda y. \text{praised tina } y) x)
\]

\[
\rightarrow_\beta \exists x. (\text{human } x) \land (\text{praised tina } x)
\]
Nouns & Determiners

Syntax/semantics interface:

TINA : NP
MARY : NP
EVERYBODY : NP
SOMEBODY : NP
MAN : N
WOMAN : N
EVERY : N NP
SOME : N NP
RAN : NP S
PRAISED : NP NP S

Semantic interpretation:

N := e t
NP := (e t) t
S := t
Nouns & Determiners

Semantic interpretation:

\[
\begin{align*}
\text{TINA} & := \lambda k. k \text{ tina} \\
\text{MARY} & := \lambda k. k \text{ mary} \\
\text{EVERYBODY} & := \lambda k. \forall x. (\text{human } x) \rightarrow (k x) \\
\text{SOMEBODY} & := \lambda k. \exists x. (\text{human } x) \wedge (k x) \\
\text{MAN} & := \lambda x. \text{ man } x \\
\text{WOMAN} & := \lambda x. \text{ woman } x \\
\text{EVERY} & := \lambda n. \lambda m. \forall x. n x \rightarrow m x \\
\text{SOME} & := \lambda n. \lambda m. \exists x. n x \wedge m x \\
\text{RAN} & := \lambda s. s (\lambda x. \text{ ran } x) \\
\text{PRAISED} & := \lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{ praised } x y)) \\
\end{align*}
\]

where:

woman, man : e t
Determiners as binary generalized quantifiers
Determiners as binary generalized quantifiers

- The type of the interpretations of EVERY and SOME is \((e \, t) \, (e \, t) \, t\).
Determiners as binary generalized quantifiers

- The type of the interpretations of EVERY and SOME is \((e\ t)\ (e\ t)\ t\).
- Every term of type \((e\ t)\ (e\ t)\ t\) is called a binary generalized quantifier.
Determiners as binary generalized quantifiers

- The type of the interpretations of EVERY and SOME is \((e \, t) \, (e \, t) \, t\).
- Every term of type \((e \, t) \, (e \, t) \, t\) is called a binary generalized quantifier.
- Semantically, a binary generalized quantifier corresponds to a relation between two sets of entities.
Determiners as binary generalized quantifiers

<table>
<thead>
<tr>
<th>Determiner</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOME $A \cap B$</td>
<td>$A \cap B \neq \emptyset$</td>
</tr>
<tr>
<td>EVERY $A \subset B$</td>
<td></td>
</tr>
<tr>
<td>NO $A \cap B$</td>
<td>$A \cap B = \emptyset$</td>
</tr>
<tr>
<td>(AT-LEAST $n$) $A \cap B$</td>
<td>$</td>
</tr>
<tr>
<td>(AT-MOST $n$) $A \cap B$</td>
<td>$</td>
</tr>
<tr>
<td>(EXACTLY $n$) $A \cap B$</td>
<td>$</td>
</tr>
<tr>
<td>MOST $A \cap B$</td>
<td>$</td>
</tr>
</tbody>
</table>
Scope ambiguities
Scope ambiguities

Every man praised a woman
Scope ambiguities

*Every man praised a woman*

\[
\forall x. \text{man } x \rightarrow (\exists y. \text{woman } y \land \text{praised } x y) \\
\exists y. \text{woman } y \land (\forall x. \text{man } x \land \text{praised } x y)
\]
Every man praised a woman

\[ \forall x. \text{man } x \rightarrow (\exists y. \text{woman } y \land \text{praised } x y) \]
\[ \exists y. \text{woman } y \land (\forall x. \text{man } x \land \text{praised } x y) \]

Subject wide scope:

\[ \text{PRAISED} = \lambda o. \lambda s. s (\lambda x. o (\lambda y. \text{praised } x y)) \]

Object wide scope:

\[ \text{PRAISED}_{\text{ows}} = \lambda o. \lambda s. o (\lambda y. s (\lambda x. \text{praised } x y)) \]