Determiners and generalized quantifiers

1. The

- A *the*-phrase denotes an referential individual of type $e$. The definite determiner *the* is of type $(e, t)$: it combines with a CN/NP of type $(e, t)$ to return an referential individual of type $e$.

(1) a. \[
\text{[the]}^w = \lambda P_{(e, t)} \lambda x_e[P(x)]
\]
    (The function that applies to a predicate $P$ and returns the unique entity $x$ s.t. $P(x)$ holds)

b. \[
\text{[the cat]}^w = \lambda x_e[\text{cat}_w(x)]
\]
    (The unique entity $x$ such that $x$ is a cat)

c. The cat snores.

\[
\text{DP} \\ e
\]
\[
\text{VP} \\ ⟨e, t⟩
\]
\[
\lambda x_e[\text{cat}_w(x)]
\]
\[
\text{snore}_w(\lambda x_e[\text{cat}_w(x)])
\]

- The definite determiner *the* presupposes uniqueness.

(2) a. [Pointing at one cat], the cat snores.

b. [Pointing at two cats], # the cat snores.

This presupposition is modeled as introduced by the presupposition of the $\iota$-operator:

(3) $\lambda x_e[P(x)]$ is defined iff there exists exactly one $x$ such that $P(x) = 1$. Formally:

a. \[
\text{[the]} = \lambda P_{(e, t)} : \exists! x[P(x)].\lambda x_e[P(x)]
\]

b. \[
\text{[the]} = \lambda P_{(e, t)} : \exists x[P(x) \land \forall y[P(y) \rightarrow y \leq x]].\lambda x_e[P(x) \land \forall y[P(y) \rightarrow y \leq x]]
\]

NB: Definition (b) is more preferable since it also extends to plural definite descriptions like *the cats*.  


• Adding relative clauses (REL)

Discussion: Which of the following (simplified) trees correctly describes the structure of the definite description “the girl who invited Andy”? [In other words, does the relative clause “who invited Andy” modify “girl” or “the girl”?] Why?

(4) the girl who invited Andy

a.

```
DP
  DP
    D
    the
    NP
    CN
    girl
  REL
    who invited Andy
```

b.

```
DP
  D
  the
  NP
  CN
  girl
  REL
    who invited Andy
```

Exercise: Compose the following sentence:

(5) The girl who invited Andy left.

```
S
  t
  left_w(λxₑ[girl_w(x) ∧ invite_w(x, a)])
    DP
      e
        λxₑ[girl_w(x) ∧ invite_w(x, a)]
          D
            ⟨et, e⟩
              λPₑ(x, e)λx[P(x)]
                the
          NP
            ⟨e, t⟩
              λxₑ[girl_w(x) ∧ invite_w(x, a)]
                λxₑ.girl_w(x)
                  CN
                    ⟨e, t⟩
                      λxₑ.girl_w(x)
                        girl
          REL
            ⟨e, t⟩
              λxₑ.invite_w(x, a)
                λxₑ.invite_w(x, a)
                  who invited Andy
    VP
      ⟨e, t⟩
        λxₑ.left_w(x)
          Vitr
            ⟨e, t⟩
              λxₑ.left_w(x)
                left
```
2. Generalized quantifiers and quantificational determiners

• Recall: We define quantificational determiners like some and every as relations between two sets of entities.

   (6)  a. Some cat meows.
   b. Every cat meows.
   c. No cat meows.

2.1. Generalized quantifiers

• Quantificational DPs (e.g. everything, something, nothing, every cat, some cat, no cat) are not individuals (of type e): (cf. proper names like John, definite DPs like the cat), nor individual sets (of type \( \langle e, t \rangle \)) (cf. common nouns like cat).

   – They are not individuals. Compare with e-type NPs:

   (7)  Law of Contradiction
      a. Mary is coming and Mary is not coming. (Contradiction)
      b. Someone is coming and someone is not coming. (Not contradiction)

   (8)  Law of Excluded middle
      a. Mary is coming or Mary is not coming. (Tautology)
      b. Every is coming or everyone is not coming. (Not tautology)

   (9)  Only an e-type NP can normally license a singular discourse pronoun.
      a. John /the man/ a man walked in. He looked tired.
      b. Every man /no man/ more than one man walked in. *He looked tired.

   – They are also not sets/predicates.

   * N-words: It is hard to think of nobody as a set. The best thing we can do is to treat it as an empty set. But then nobody and no linguist would be semantically equivalent, contra fact.

   * Numeral modified quantifiers: It’s unclear what set at least one question and at most three questions refer to.

Discussion: Can you think of more differences between generalized quantifiers and e-type NPs or common nouns?
• We treat quantificational DPs as second-order functions of type \( \langle et, t \rangle \), called **generalized quantifiers**. In (10), *meows* is an argument of *every cat*.

\[
(10) \quad \text{Every cat meows.}
\]

\[
\begin{aligned}
&\text{S}_t \\
&\text{DP}_{(et, t)} \quad \text{VP}_{(et, t)} \\
&\text{D} \quad \text{NP} \\
&\text{every} \quad \text{CN} \quad \text{meows}_{(et, t)} \\
&\text{cat}_{(et, t)}
\end{aligned}
\]

\[
\begin{aligned}
(11) \quad &\text{a. } \lambda P_{(et, t)} \cdot \forall x [\text{cat}_{et}(x) \rightarrow P(x)] \\
&\text{b. } \lambda P_{(et, t)} \cdot \forall x [\text{cat}_{et}(x) \rightarrow \text{meows}_{et}(x)]
\end{aligned}
\]

\[
\begin{aligned}
(12) \quad &\text{a. } \lambda P_{(et, t)} \cdot \exists x [\text{cat}_{et}(x) \land P(x)] \\
&\text{b. } \lambda P_{(et, t)} \cdot \exists x [\text{cat}_{et}(x) \land \text{meows}_{et}(x)]
\end{aligned}
\]

\[
\begin{aligned}
(13) \quad &\text{a. } \lambda P_{(et, t)} \cdot \exists x [\text{cat}_{et}(x) \land P(x)] \\
&\text{b. } \lambda P_{(et, t)} \cdot \exists x [\text{cat}_{et}(x) \land \text{meows}_{et}(x)]
\end{aligned}
\]

2.2. **Type-shifters**

• Individuals (of type \( e \)) can also be shifted into generalized quantifiers via **type-lifting**.

\[
\begin{aligned}
(14) \quad &\text{LIFT} = \lambda a_{et} \lambda P_{(et, t)} \cdot P(a) \\
(15) \quad &\text{a. } [\text{Kitty}]^w = k \\
&\text{b. } \text{LIFT}([\text{Kitty}]^w) = \lambda P_{(et, t)} \cdot P(k) \\
&\text{c. } \text{LIFT}([\text{Kitty}]^w)([\text{meows}]^w) = (\lambda P_{(et, t)} \cdot P(k))(\lambda x_{et} \cdot \text{meows}_{et}(x)) \\
&\quad = (\lambda x_{et} \cdot \text{meows}_{et}(x))(k) \\
&\quad = \text{meows}_{et}(k)
\end{aligned}
\]

• We can extract the quantification domain of an *∃*-quantifier via the BE-shifter (Partee 1986):

\[
\begin{aligned}
(16) \quad &\text{BE} = \lambda P \lambda z [P(\lambda y. y = z)] \\
(17) \quad &\text{BE}([\text{some cat}]^w) = (\forall f_{et, t} \cdot \exists x [\text{cat}_{et}(x) \land f(x)])(\lambda y. y = z) \\
&\quad = \lambda z \cdot \exists x [\text{cat}_{et}(z) \land x = z] \\
&\quad = \{ z \mid \text{cat}_{et}(z) \}
\end{aligned}
\]

**Discussion:** (i) What do we get by applying BE to \([\text{every cat}]^w\) and LIFT([\text{John}]^w)? (ii) What do we get by applying LIFT to \([\text{every cat}]^w\) and LIFT([\text{John}]^w)?
2.3. Quantificational determiners

• The determiner every combines with a common noun of type \((e, t)\) to return a generalized quantifier of type \((et, t)\). Therefore, its type is quite complex: \((et, (et, t))\).

\[(18)\]

\[\begin{align*}
\text{a. } & [\text{every}]^w = \lambda Q_{(e,t)} \lambda P_{(e,t)}. \forall x [Q(x) \to P(x)] \\
\text{b. } & [\text{some}]^w = \lambda Q_{(e,t)} \lambda P_{(e,t)}. \exists x [Q(x) \land P(x)] \\
\text{c. } & [\text{no}]^w = \lambda Q_{(e,t)} \lambda P_{(e,t)}. \neg \exists x [Q(x) \land P(x)]
\end{align*}\]

• A quantificational determiner takes two arguments (both of which are of type \((e, t)\)). The first argument is its restrictor, and the second argument is its scope.

\[(19)\]

Discussion: Identify the restrictor and scope of every in the following sentences.

\[(20)\]

\[\begin{align*}
\text{a. } & \text{Every student who read chapter 5 passed the exam.} \\
\text{b. } & \text{Everyone passed the exam.} \\
\text{c. } & \text{John read every chapter.}
\end{align*}\]
3. Properties of determiners and generalized quantifiers

- **Quantificational determiners as binary relations between sets**

Most generalized quantifiers can be decomposed into a quantificational determiner and a set-denoting term. In English, for example, quantificational determiners non-exclusively include those in (21) as well as their Boolean combinations.

(21) a. Aristotelian: *all, every, no, some*
   b. Proportional: *most, at least half, 10 percent of the, less than two-thirds of the*
   c. Numerical: *at least two, less than ten, between six and nine, finitely many, an odd number of*
   d. Exceptive: *no ... but John, every ... except Mary*

All of these quantificational determiners can be interpreted extensionally as relations between two sets of individuals in the discourse domain and are often called “type ⟨1, 1⟩ quantifiers” (‘1’ stands for 1-ary relation).

(22) Basic quantificational determiners
   a. \([\text{every}](A, B) =_{df} A \subseteq B\)
   b. \([\text{most}](A, B) =_{df} |A \cap B| > |A - B|\)
   c. \([\text{at least two}](A, B) =_{df} |A \cap B| \geq 2\)

(23) Exceptives (Gajewski 2008)
   a. \([\text{no ... but John}](A, B) =_{df} (A - \{j\}) \cap B = \emptyset\)
   b. \([\text{every ... except Mary}](A, B) =_{df} (A - \{m\}) \subseteq B\)

Comparative determiners express relations between three sets and thus are called “type ⟨1, 1, 1⟩ quantifiers”.

(24) a. \([\text{more ... than ...}](A, B, C) =_{df} |A \cap C| > |B \cap C|\)
   b. \([\text{less ... than ...}](A, B, C) =_{df} |A \cap C| < |B \cap C|\)

- **Conservativity of determiners**

  – Not every binary relation between sets of individuals can be lexicalized into determiners. All of the type ⟨1, 1⟩ quantifiers in natural languages are conservative — the part of the scope B that’s not in the restriction A does not matter for whether \(Q(A, B)\) holds.

(25) A type ⟨1, 1⟩ quantifier \(Q\) is **conservative** iff for every \(A\) and \(B\): \(Q(A, B) \iff Q(A, A \cap B)\).

   (Barwise and Cooper 1981; Higginbotham and May 1981; Keenan and Stavi 1986)

Example: *not every* versus *every not*. They both can be defined as a binary relation between two sets. However, *not every* can function as a determiner, while *every not* cannot. Conservativity captures this contrast.

(26) a. \([\text{not every}](A, B) =_{df} A \not\subseteq B\)
   b. \([\text{every not}](A, B) =_{df} \overline{A} \subseteq B\)

   \(\overline{A}\): the complement of \(A\’\) relative to the discourse universe.)
(27)  
   a. Not every student arrived.
       ⇔ Not every student is a student who arrived.
   b. Everyone who is not a student arrived.
       ⇔ Everyone who is not a student is a student who arrived.

**Exercise:** Show that *the* is conservative.

\*Live-on and range over\*

(28)  
   a. A generalized quantifier $\pi$ **lives on** a set $A$ iff for every $B$: $\pi(B) \iff \pi(B \cap A)$. (Barwise and Cooper 1981)
   b. A generalized quantifier is said to **range over** a set $A$ if and only if $A$ is the smallest live-on set (SMLO) of this GQ. (Szabolcsi 1997)

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>SMLO($\pi$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \upharpoonright$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$a \upharpoonright \sqcup b \upharpoonright$, $a \upharpoonright \sqcap b \upharpoonright$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>some/every/no student</td>
<td>student</td>
</tr>
<tr>
<td>some/two/most students</td>
<td>*student</td>
</tr>
<tr>
<td>every student but John</td>
<td>student – ${j}$</td>
</tr>
<tr>
<td>no student except John</td>
<td>student – ${j}$</td>
</tr>
</tbody>
</table>

Table 1: GQs and their smallest live-on sets (SMLO)

**Discussion:** For exceptives, the smallest live-on set is student – $\{j\}$, rather than student, can you tell why? Illustrate it using concrete examples.

**Discussion:** Try to decompose Montagovian individuals into a quantificational determiner and a set.