Plurality: Back to Generalized Quantifiers and Boolean Semantics

Yoad Winter

30 March 2012

1 Aim

Given: A sentence S = D-N'-VP.

D – a singular/plural determiner: *every, all, no, exactly one, exactly five*.

N' – a singular/plural atom/set nominal: *student(s), friend(s), students who met each other at school*.

VP – a singular/plural atom/set verb phrase: *is/are tall/similar, drink a whole glass of beer together*.

Question: What is the formal semantics of S?

Van Benthem's warning: Suppose we want to analyze *less than five students met* as entailing the existential sentence *there was a meeting set of students of less than five members*. We have to be careful that the sentence *less than five students sang* would not be predicted to entail *there was a singing set of students of less than five members*.
Conclusion: Determiners count; if they make an existence claim (e.g. by modification), this process must be restricted.

2 Principles

1. Barwise and Cooper: All determiners are of type $(et)((et)t)$.
2. Bennett: Plural predicates (N's and VPs) can be of type $(et)t$.
3. Partee and Rooth: Type shifting applies only in cases of type mismatch.
4. Only one type shifting operator per semantic category (predicate, quantifier, determiner).
5. $\Rightarrow$ No Van Benthem problem: determiners both count and modify at the same time.

3 Predicates

A. Lexical Predicates:

1. Atom predicates: *student, sing, tall*.
2. Set predicates: *friend, meet, similar*.

Atom/Set distinction: Atom predicates are lexically of type et. Set predicates are lexically of type ett.

B. Complex (number inflected) Predicates:

1. Singular predicates: *student, sings, is tall, friend, meets, is similar*.
2. Plural predicates: *students, (we/they) sing, are tall, friends, (we/they) meet, are similar*.

Singular/Plural distinction: Singular predicates are obligatorily of type et. Plural predicates are (optionally) of type ett.

C. Deriving these results:

1. Denotations of number features:
$$[\![+\text{SG}]\!] = sg_{(ett)(et)} \stackrel{def}{=} \lambda \mathcal{P}_{ett}.\lambda x_e.\mathcal{P}(\{x\})$$
$$[\![+\text{PL}]\!] = id_{(ett)(ett)} \stackrel{def}{=} \lambda \mathcal{P}_{ett}.\mathcal{P}$$
2. Consequently: atom predicates do not match the type of the number features.
3. Type mismatch resolution: using a *type fitting* operator for predicates (=Link's distributivity operator):
$$pfit_{(et)(ett)} \stackrel{def}{=} \lambda P_{et}.\lambda A_{et}.\emptyset \neq A \subseteq P$$
4. Fact: $sg \circ pfit = id_{(et)(et)}$.

D. Examples:

(1) a. $[\![\text{student}]\!] = [\![\text{student+SG}]\!] = \mathbf{student}'_{et}\ sg_{(ett)(et)}$ (mismatch)
$sg(pfit(\mathbf{student}')) = \mathbf{student}'$ (resolution)

b. $[\![\text{students}]\!] = [\![\text{student+PL}]\!] = \mathbf{student}'_{et}\ id_{(ett)(ett)}$ (mismatch)
$id(pfit(\mathbf{student}')) = \lambda A_{et}.\emptyset \neq A \subseteq \mathbf{student}'$ (resolution)

(2) a. $[\![\text{meets}]\!] = [\![\text{meet+SG}]\!] = sg_{(ett)(et)}(\mathbf{meet}'_{ett}) = \lambda x_e.\mathbf{meet}'(\{x\})$

b. $[\![\text{meet}]\!] = [\![\text{meet+PL}]\!] = id_{(ett)(ett)}(\mathbf{meet}'_{ett}) = \mathbf{meet}'$

4 Determiners

Singular determiners: a standard story.

(3) No student slept.

$\mathbf{no}'_{(et)(ett)}(\mathbf{student}'_{et})(\mathbf{sleep}'_{et})$
$\Leftrightarrow\ \mathbf{student}' \cap \mathbf{sleep}' = \emptyset$

(4) No committee met.

$\mathbf{no}'_{(et)(ett)}(\mathbf{committee}'_{et})(sg(\mathbf{meet}'_{ett}))$
$\Leftrightarrow \mathbf{committee}'_{et} \cap \{x_e : \mathbf{meet}'_{ett}(\{x\})\} = \emptyset.$

Plural determiners: Type mismatch between determiner denotation $D_{(et)(ett)}$ and noun denotation $\mathcal{A}_{ett}$. For instance:

(5) Exactly five students met.

Resolution of the mismatch using two independent processes:

1. *Counting*: $\mathbf{count}(D)(\mathcal{A})(\mathcal{B})$ iff $D(\cup\mathcal{A})(\cup(\mathcal{A}\cap\mathcal{B}))$.
(generates Scha's "neutral" reading)
For instance:

(6) $\mathbf{count}(\mathbf{exactly\_5}')(\mathbf{students}'_{ett})(\mathbf{meet}'_{ett})$
$\Leftrightarrow \mathbf{count}(\mathbf{exactly\_5}')(id(pfit(\mathbf{student}'_{et})))(\mathbf{meet}'_{ett})$
$\Leftrightarrow \mathbf{exactly\_5}'(\cup pfit(\mathbf{student}'))(\cup(\mathbf{meet}' \cap pfit(\mathbf{student}')))$
$\Leftrightarrow\ |\{x \in \mathbf{student}' : \exists A \subseteq \mathbf{student}'[x \in A \wedge \mathbf{meet}'(A)]\}| = 5$

In words: the total number of students who participated in student meetings is exactly five.

2. *Witnessing*: Given a standard $(et)((et)t)$ determiner, we say that $W \subseteq E$ is a *witness* of D and a set $A \subseteq E$ iff $W \subseteq A$ and $D(A)(W)$.
Examples: For the set $S \subseteq E$ of students, a set $A \subseteq E$ is a witness of the determiner SOME and the set S iff $A \subseteq S$ and $A \neq emptyset$. A set $A \subseteq E$ is a witness of the determiner EVERY and the set S iff $A = S$.
Question: what are the witnesses of NO, LESS_THAN_5, MOST and the set S?

The witness condition:
$\mathbf{wit}(D)(\mathcal{A})(\mathcal{B})$ iff either $\mathcal{A} \cap \mathcal{B} = \emptyset$ or $\exists W \in \mathcal{A} \cap \mathcal{B}$ such that $D(\cup\mathcal{A})(W)$.
In other words: D witnesses $\mathcal{A}_{ett}$ and $\mathcal{B}_{ett}$ iff whenever $\mathcal{A} \cap \mathcal{B}$ is not empty it includes a witness of D and $\cup\mathcal{A}$.
(generates Scha's "existential" reading)
For instance:

(7) $\mathbf{wit}(\mathbf{exactly\_5'})(\mathbf{students}'_{ett})(\mathbf{meet}'_{ett})$
$\Leftrightarrow \mathbf{wit}(\mathbf{exactly\_5'})(id(\mathit{pfit}(\mathbf{student}'_{et})))(\mathbf{meet}'_{ett})$
$\Leftrightarrow \mathbf{wit}(\mathbf{exactly\_5}'_{(et)(ett)})(\mathit{pfit}(\mathbf{student}'_{et}))(\mathbf{meet}'_{ett})$
$\Leftrightarrow [\exists A \subseteq \mathbf{student}'[A \neq \emptyset \wedge \mathbf{meet}'(A)]] \rightarrow \exists A \subseteq \mathbf{student}'[|A| = 5 \wedge \mathbf{meet}'(A)]$

In words: if any student(s) met then there was a meeting of exactly five students.

The resulting determiner fitting operator:

$$\mathit{dfit}_{((et)(ett))((ett)(ettt))} \stackrel{def}{=} \lambda D.\lambda\mathcal{A}.\lambda\mathcal{B}.\mathbf{count}(D)(\mathcal{A})(\mathcal{B}) \wedge \mathbf{wit}(D)(\mathcal{A})(\mathcal{B})$$

Consequences:

1. No essential witnessing, hence no Van Benthem problem, with atom predicates:
For all conservative determiners $D_{(et)(ett)} andsets A_{et}, B_{et}$:
$\mathit{dfit}(D)(\mathit{pfit}(A))(\mathit{pfit}(B)) \Leftrightarrow D(A)(A \cap B)$.

2. No essential witnessing, hence no Van Benthem problem, with downward monotone determiners: For all $D_{(et)(ett)} \in$ MON↓, $\mathcal{A}_{ett}, \mathcal{B}_{ett}$: $\mathit{dfit}(D)(\mathcal{A})(\mathcal{B}) \Leftrightarrow \mathbf{count}(\mathcal{A})(\mathcal{B})$.

3. *Conservativity* is preserved with collective quantification: *D Ns V* is equivalent to *D Ns are Ns that V* also when N is a noun like *friend* or V is a verb like *meet*. For instance:

(8) All the/exactly five/no friends met each other
$\Leftrightarrow$ All the/exactly five/no friends are friends who met each other

(9) All the/exactly five/no students who met in the bar shook hands with each other
$\Leftrightarrow$ All the/exactly five/no students who met in the bar are students who met in the bar and shook hands with each other

This is captured by **count**.

4. Lexical *monotonicity* of determiners is not always preserved when they combine with ett predicates. For instance:

(10) All the students drank together a whole glass of beer
$\not\Rightarrow$ All the rich students drank together a whole glass of beer

This is captured by **wit**.

5 Quantifiers

(11) All the students and every teacher smiled.

Type mismatch between an ett quantifier and an $ettt$ quantifier. Resolution by:

$qfit_{(ett)(ettt)} \stackrel{def}{=} \lambda Q_{ett}.\lambda \mathcal{A}_{ett}.Q(sg(\mathcal{A}))$

In words: a quantifier $qfit(Q)$ holds of the sets of sets whose singleton members' union is in Q.
Using the *qfit* operator, sentence (11) is analyzed as follows.

(12) $((dfit(\mathbf{all}'_{(et)(ett)})(pfit(\mathbf{student}'_{et}))) \cap (qfit(\mathbf{every}'_{(et)(ett)}(\mathbf{teacher}'_{et}))))(\mathbf{smile}'_{et})$
$\Leftrightarrow\ \mathbf{student}' \subseteq \mathbf{smile}' \ \wedge\ \mathbf{teacher}' \subseteq \mathbf{smile}'$

Note that *every* and *all* are treated as synonyms, both of which denoting the subset relation between et predicates.

Fact: The *qfit* operator preserves distributivity: For all $Q_{ett}, \mathcal{B}_{ett}$: $qfit(Q)(\mathcal{B}) \Leftrightarrow Q(sg(\mathcal{B}))$.
In particular: For all Q_{ett}, B_{et}: $qfit(Q)(pfit(B)) \Leftrightarrow Q(B)$.

6 How about Boolean *and*?

Again, having covert type flexibility operators allow us to preserve the boolean structures of our semantics.

(13) Mary and John met.

(14) $\mathbf{meet}'_{(et)t}(\{\mathbf{m}', \mathbf{j}'\})$

(15) $\{A : \mathbf{m}' \in A\} \cap \{A : \mathbf{j}' \in A\} \ = \ \{A : \{\mathbf{m}'\} \cup \{\mathbf{j}'\} \subseteq A\}$

Definition 1 (filter) *Let A be a boolean algebra and let F be a non-empty subset of A. F is called a* filter *of A iff the following hold:*

1. *For all $x, y \in F$: $x \wedge y \in F$.*
2. *For all $x \in F$, $y \in A$: if $x \leq y$ then $y \in F$.*

Definition 2 (principal filter) *Let A be a boolean algebra and $x \in A$. The* principal filter generated by *x is the set $\{y \in A : x \leq y\}$, which is denoted by F_x.*

Proposition 1 (The Principal Filter Property, PFP) *Let A be a boolean algebra and $x, y \in A$. Then $F_x \cap F_y = F_{x \vee y}$.*

Lemma 1 *$x \leq z$ and $y \leq z$ iff $x \vee y \leq z$.*

And even:

(16) Mary and [Sue or John] met.

a. $F_{\{\mathbf{m}'\}} \cap [F_{\{\mathbf{s}'\}} \cup F_{\{\mathbf{j}'\}}]$
b. $= \ [F_{\{\mathbf{m}'\}} \cap F_{\{\mathbf{s}'\}}] \cup [F_{\{\mathbf{m}'\}} \cap F_{\{\mathbf{j}'\}}]$
c. $\stackrel{PFP}{=} \ F_{\{\mathbf{m}',\mathbf{s}'\}} \cup F_{\{\mathbf{m}',\mathbf{j}'\}}$

Hypothesis 1 (the PFP hypothesis) *"Union" behaviour of* and *in NP coordination is a result of the principal filter property of its standard* intersective *denotation in the boolean domain of generalized quantifiers.*

(17) Mary and John met.

a. $[\mathrm{M} \sqcap \mathrm{J}]_{(et)t}$ $\mathbf{meet'}_{(et)t}$ (type mismatch)

b. $[\mathbf{C}(\mathrm{M} \sqcap \mathrm{J})]_{((et)t)t}(\mathbf{meet'})$

(18) **Minimum Sort**:

$min_{(\tau t)(\tau t)} \stackrel{def}{=} \lambda Q_{\tau t}.\lambda A_{\tau}.Q(A) \wedge \forall B \in Q[B \sqsubseteq A \rightarrow B = A]$

(19) **Existential Raising**: $\mathbf{E}_{(\tau t)((\tau t)t)} \stackrel{def}{=} \lambda A_{\tau t}.\lambda P_{\tau t}.\exists X_{\tau}[A(X) \wedge P(X)]$

(20) **Collectivity Raising**: $\mathbf{C}_{((et)t)(((et)t)t)} \stackrel{def}{=} \lambda Q_{(et)t}.\mathbf{E}(min(Q))$

(21) $min(\mathrm{M} \sqcap \mathrm{J}) = \{\{\mathbf{m'}, \mathbf{j'}\}\}$
$\mathbf{E}(min(\mathrm{M} \sqcap \mathrm{J})) = \mathbf{E}(\{\{\mathbf{m'}, \mathbf{j'}\}\})$
$= \lambda \mathcal{P}_{(et)t}.\exists B \in \{\{\mathbf{m'}, \mathbf{j'}\}\} [\mathcal{P}(B)]$
$= \lambda \mathcal{P}.\mathcal{P}(\{\mathbf{m'}, \mathbf{j'}\})$
Therefore, $[\mathbf{C}(\mathrm{M} \sqcap \mathrm{J})](\mathbf{meet'})$
$\Leftrightarrow [\lambda \mathcal{P}.\mathcal{P}(\{\mathbf{m'}, \mathbf{j'}\})](\mathbf{meet'})$
$\Leftrightarrow \mathbf{meet'}(\{\mathbf{m'}, \mathbf{j'}\})$