Class 2

Basic notions and tools

- NOTIONS
- TOOLS

Lambda Notation

IS as Identity Function

[[Tina is tall]] = 1 -- ***tina*** denotes an entity in the set for ***tall***

With types:

$$(\mathrm{IS}_{(et)(et)}(\mathbf{tall}_{et}))(\mathbf{tina}_{e})$$

Intuitively: **IS** maps any set to itself.

Formally:

$\mathrm{IS}_{(et)(et)}$ **=**

The function sending every element g of the domain D_{et} to g.

IS in lambda notation

IS$_{(et)(et)}$

The function sending every element g of the domain D_{et} to g.

Instead of writing "the function sending every element g of D_{et}" as in (55), we write "λg_{et}".

Instead of "to g" as in (55), we write ".g".

Thus: $\lambda g_{et} . g$ Summing up: IS $= \lambda g_{et} . g$

The letter 'λ' tells us that it is a function.

The notation 'g_{et}' before the dot introduces 'g' as an *ad hoc* name for the argument of this function. The type et in the subscript of g tells us that this argument can be any object in the domain D_{et}.

The re-occurrence of 'g' after the dot tells us that the function we define in (58) returns the value of its argument.

Lambda Notation

Lambda notation: *When writing "$\lambda x_\tau.\varphi$", where τ is a type, we mean: "the function sending every element x of the domain D_τ to φ".*

Function application with Lambda's

$(\lambda g_{et}.g)(\mathbf{tall}_{et})$
$= \mathbf{tall}$

Another example:

$\textsc{succ}(x) = x + 1$

$\textsc{succ}(22) = 22 + 1$

$(\lambda x_n.x + 1)(22)$ = $22 + 1$

Function application with lambda terms: *The result* $(\lambda x_\tau.\varphi)(a_\tau)$ *of applying a function described by a lambda term* $\lambda x_\tau.\varphi$ *to an argument* a_τ, *is equal to the value of the expression* φ, *with all occurrences of* x *replaced by* a.

Reflexives in object position

Tina praised herself

$$\mathbf{praise}_{e(et)}(\mathbf{tina}_e)(\mathbf{tina})$$

[[herself]] = **tina** ???

$$(\text{HERSELF}_{(e(et))(et)}(\mathbf{praise}_{e(et)}))(\mathbf{tina}_e)$$

$$= \mathbf{praise}(\mathbf{tina})(\mathbf{tina})$$

Generalizing:

For all functions R of the domain $D_{e(et)}$, for all entities x of the domain D_e:

$$(\text{HERSELF}_{(e(et))(et)}(R))(x) = R(x)(x)$$

Reflexives in object position (cont.)

For all functions R of the domain $D_{e(et)}$, for all entities x of the domain D_e:

$(\text{HERSELF}_{(e(et))(et)}(R))(x) = R(x)(x)$

$\text{HERSELF}_{(e(et))(et)}$ is
the function sending every element R of the domain $D_{e(et)}$ to the function sending every element x of the domain D_e to $R(x)(x)$.

$\text{HERSELF}_{(e(et))(et)}$
$= \lambda R_{e(et)}$.the function sending every element x of the domain D_e to $R(x)(x)$

$\text{HERSELF}_{(e(et))(et)}$
$= \lambda R_{e(et)}.(\lambda x_e.R(x)(x))$ = $\lambda R_{e(et)}.\lambda x_e.R(x)(x)$

Verifying the derivation

$(\text{HERSELF}_{(e(et))(et)}(\mathbf{praise}_{e(et)}))(\mathbf{tina}_e)$	▷ compositional analysis of structure (63)
$= ((\lambda R_{e(et)}.\lambda x_e.R(x)(x))(\mathbf{praise}))(\mathbf{tina})$	▷ definition (70) of HERSELF
$= (\lambda x_e.\mathbf{praise}(x)(x))(\mathbf{tina})$	▷ applying HERSELF to the argument **praise**
$= \mathbf{praise}(\mathbf{tina})(\mathbf{tina})$	▷ applying (HERSELF(**praise**)) to the argument **tina**

What have we learnt here?

- A useful notation for functions
- A useful rule for simplifying notation under function application

Exercise – Types for Conditionals

[If [you smile]] [you win]

you smile – of type *t*

you win – of type *t*

1. Write type equations.
2. What are the types of the following expressions?

   If you smile – If

3. Find denotation of ***if*** that explains:

   [[If [you smile]] [you win]] [and [you smile]]

   ➔ You win

Exercise – Lambdas

describe $\lambda f_{et}.\lambda u_e.f(\mathbf{john}_e)$ in words

describe $\lambda f_{(ee)t}.f(\lambda u_e.\mathbf{john}_e)$ in words

simplify $(\lambda f_{ee}.\lambda x_e.x = f(x))(\lambda u_e.\mathbf{john}_e)$

Restricting Denotations

Expressing NOT in lambda's

Tina [is [not tall]]

$\textsc{not}$ is the $(et)(et)$ function sending every et function g to the et function $\textsc{not}(g)$ that satisfies for every entity x:

$$(\textsc{not}(g))(x) = \begin{cases} 1 & \text{if } g(x) = 0 \\ 0 & \text{if } g(x) = 1 \end{cases}$$

$$\sim \;=\; \lambda x_t.1 - x$$

$$\text{NOT} \;=\; \lambda g_{et}.\lambda x_e.{\sim}(g(x))$$

$(\text{IS}_{(et)(et)}(\text{NOT}_{(et)(et)}(\mathbf{tall}_{et})))(\mathbf{tina}_e)$	▹ compositional analysis of structure (37)
$= ((\lambda g_{et}.g)(\textsc{not}(\mathbf{tall})))(\mathbf{tina})$	▹ definition of $\textsc{is}$ as identity function
$= (\textsc{not}(\mathbf{tall}))(\mathbf{tina})$	▹ applying identity function to $\textsc{not}(\mathbf{tall})$
$= ((\lambda g_{et}.\lambda x_e.{\sim}(g(x)))(\mathbf{tall}))(\mathbf{tina})$	▹ definition (55) of $\textsc{not}$
$= ((\lambda x.{\sim}(\mathbf{tall}(x))))(\mathbf{tina})$	▹ applying definition of $\textsc{not}$ to $\mathbf{tall}$
$= {\sim}(\mathbf{tall}(\mathbf{tina}))$	▹ application to $\mathbf{tina}$

Expressing ANDs in lambda's

[Tina [is tall]] [and [Tina [is thin]]]

For any two truth-values x and y: the truth-value $x \wedge y$ is $x \cdot y$, the multiplication of x by y.

$$\text{AND}^t = \lambda x_t.\lambda y_t.y \wedge x$$

Tina [is [tall [and thin]]]

For every two functions f_A and f_B in D_{et}, characterizing the subsets A and B of D_e: $(\text{AND}(f_A))(f_B)$ is defined as the function $f_{A \cap B}$, characterizing the intersection of A and B.

$$\text{AND}^{et} = \lambda f_{et}.\lambda g_{et}.\lambda x_e.g(x) \wedge f(x)$$

Attributive adjectives (1) - Intersective

Tina is a *tall* woman; the *tall* engineer visited us; I met five *tall* astronomers.

Tina is a Chinese pianist ⇔ Tina is Chinese and Tina is a pianist.

My doctor has a white Volkswagen ⇔ My doctor's Volkswagen is white.

Mary saw three carnivorous animals ⇔ Three animals that Mary saw are carnivorous.

Tina [is [a pianist]]

$\textsc{a}_{(et)(et)} = \textsc{is} = \lambda g_{et}.g$

$(\textsc{is}(\textsc{a}(\mathbf{pianist})))(\mathbf{tina})$

$= \mathbf{pianist}(\mathbf{tina})$

Attributive adjectives (2) - Intersective

Tina [is [a [Chinese pianist]]]

$$\textbf{chinese}^{\mathsf{mod}}_{(et)(et)} = \lambda f_{et}.\lambda x_e.\textbf{chinese}(x) \wedge f(x)$$

For any two truth-values x and y: the truth-value $x \wedge y$ is $x \cdot y$, the multiplication of x by y.

$(\text{IS}(\text{A}(\textbf{chinese}^{\mathsf{mod}}(\textbf{pianist}))))(\textbf{tina})$	▷ compositional analysis of (73)
$= (\textbf{chinese}^{\mathsf{mod}}(\textbf{pianist}))(\textbf{tina})$	▷ applying IS and A (identity functions)
$= ((\lambda f_{et}.\lambda x_e.\textbf{chinese}(x) \wedge f(x))(\textbf{pianist}))(\textbf{tina})$	▷ definition (74) of $\textbf{chinese}^{\mathsf{mod}}$
$= (\lambda x_e.\textbf{chinese}(x) \wedge \textbf{pianist}(x))(\textbf{tina})$	▷ applying modificational denotation to **pianist**
$= \textbf{chinese}(\textbf{tina}) \wedge \textbf{pianist}(\textbf{tina})$	▷ applying result to **tina**

Conclusion: with adjectives like *Chinese* the attributive (*et*)(*et*) denotation can be systematically derived from the predicative *et* denotation.

Note: this is not the case with all adjectives (cf. *skillful*).

Attributive adjectives (3) - Subsective

Jan is a <u>Chinese</u> surgeon & Jan is a violinist

➔ *Jan is a <u>Chinese</u> violinist*

Jan is a <u>skillful</u> surgeon & Jan is a violinist

➔ *Jan is a <u>skillful</u> violinist*

Conclusion 1: *skillful* is not intersective.

However, skillful has a weaker property, which we call <u>restrictivity</u>.

Jan is a <u>skillful</u> surgeon

➔ *Jan is a <u>surgeon</u>*

Attributive adjectives (4) - Subsective

Formally: ***M is subsective (or "restrictive") if for every set of entities A,*** **M(A) $\subseteq$ A.**

Conclusion 2: *skillful* is subsective.

In Lambdas:

$\mathbf{skillful}_{(et)(et)} = \lambda A.\ \lambda y.\ (\mathbf{skillful1}_{(et)(et)}(\mathbf{A}))(\mathbf{y}) \wedge \mathbf{A}(\mathbf{y})$

Summary – restrictions on denotations

Constant, Combinatorial:

IS, A, HERSELF

Constant, Logical:

ANDs, NOTs

Arbitrary:

tina, smile, praise, pianist, chinese (predicative use)

Logical operator on arbitrary:

chinese*mod*, skillful*mod* (attributive use)

Further: bachelor ➔ unmarried …

More Exercises

Exercise – part 1

Split the words in each sentence into 2 sets:
(i) words whose denotations are arbitrary across models;
(ii) words whose denotations are constant across models.

John is a man
Tina is a dancer and an artist
It is not the case that Trump respects Obama

Note: assume that it is not necessarily the case that is a word = an atomic constituent.

Exercise – part 2

For each sentence:

- Give lambda terms for the words from set (i)
- Define a model in which the sentence denotes 1 and a model in which the sentence denotes 0. This requires: a definition of the domain of entities, denotations of the words from set (i).

John is a man
Tina is a dancer and an artist
It is not the case that Obama respected Bush

Simple quantifiers

The problem

(i) Mary slept

slept(m)

m $\in$ sleep'

(ii) Every girl slept

slept(?)

girl' $\subseteq$ sleep'

Quantifiers – main claims

In order to describe the meaning of NPs with *determiners* (*every, some, most* etc.), we should let such NPs denote sets of subsets of E – type $(et)t$.

The same type is needed for describing *NP coordination* in a general way.

Montague's hypothesis about the matching between syntactic categories and semantic types leads us to adopt a uniform type for all NPs.

Some hard syntactic questions can then be given interesting semantic answers.

Keenan's typology of determiners (1)

Lexical Dets
every, each, all, some, a, no, several, neither, most, the, both, this, my, these, John's, ten, a few, a dozen, many, few

Cardinal Dets
exactly/approximately/more than/fewer than/at most/only ten, infinitely many, two dozen, between five and ten, just finitely many, an even/odd number of, a large number of

Approximative Dets
approximately/about/nearly/around fifty, almost all/no, hardly any, practically no

Definite Dets
the, that, this, these, my, his, John's, the ten, these ten, John's ten

Exception Dets
all but ten, all but at most ten, every...but John, no...but Mary,

Bounding Dets
exactly ten, between five and ten, most but not all, exactly half the, (just) one...in ten, only SOME (= some but not all; upper case = contrastive stress), just the LIBERAL, only JOHN's

Possessive Dets
my, John's, no student's, either John's or Mary's, neither John's nor Mary's

Value Judgment Dets
too many, a few too many, (not) enough, surprisingly few, ?many, ?few

Proportionality Dets
exactly half the/John's, two out of three, (not) one...in ten, less than half the/John's, a third of the/John's,

Keenan's typology (2)

Partitive Dets
most/two/none/only some of the/John's, more of John's than of Mary's, not more than two of the ten

Negated Dets
not every, not all, not a (single), not more than ten, not more than half, not very many, not quite enough, not over a hundred, not one of John's

Conjoined Dets
at least two but not more than ten, most but not all, either fewer than ten or else more than a hundred, both John's and Mary's, at least a third and at most two thirds of the, neither fewer than ten nor more than a hundred

Adjectively Restricted Dets
John's biggest, more male than female, most male and all female, the last...John visited, the first ...to set foot on the Moon, the easiest...to clean, whatever...are in the cupboard

Function-argument flip-flop

Flip: **NP:*e* + VP:*et* = S:*t***

(subject as argument)

Flop: *But can all those NPs denote entities?*

NP:(*et*)*t* + VP:*et* = S:*t*

(subject as function)

Flip: NP denotes (*et*)*t* function??? ☹

Flop: yes! let’s do some work on it! ☺

Generalized quantifiers - example

(1) Every man ran.

Let *every man* denote a *set of sets*: the set of subsets of E that include the set of men:

(2) $\{B \subseteq E : \mathbf{man}' \subseteq B\}$

In type-theoretical terms: *every man* denotes an $(et)t$ function.
Application of this function to the VP denotation:

(3) $\mathbf{run}' \in \{B \subseteq E : \mathbf{man}' \subseteq B\}$
$\Leftrightarrow \mathbf{man}' \subseteq \mathbf{run}'$
"every member of the set $\mathbf{man}'$ is a member of the set $\mathbf{run}'$"

For instance, if $E = \{a, b, c, d\}$, $\mathbf{man}' = \{a, b\}$ and $\mathbf{run}' = \{a, b, c\}$, then:
$[\![\text{every man}]\!] = \{B \subseteq E : \mathbf{man}' \subseteq B\}$
$= \{B \subseteq \{a, b, c, d\} : \{a, b\} \subseteq B\}$
$= \{\{a, b\}, \{a, b, c\}, \{a, b, d\}, \{a, b, c, d\}\}$,
and thus $\mathbf{run}' = \{a, b, c\} \in [\![\text{every man}]\!]$.

Universal GQ with Lambda's

set theory	typed lambda's
$A \subseteq E = D_e$	$\chi_A \in D_{et}$ is the char. func. of A
P characterizes $P^* \subseteq E = D_e$	$P \in D_{et}$
every man:	
$\{A \subseteq E \mid M \subseteq A\}$	$\lambda P_{et}.\forall x_e.\chi_M(x) \rightarrow P(x)$
every man ran:	
$R \in \{A \subseteq E \mid M \subseteq A\}$	$(\lambda P_{et}.\forall x_e.\chi_M(x) \rightarrow P(x))(\chi_R)$
$\Leftrightarrow \; M \subseteq R$	$\Leftrightarrow \; \forall x_e.\chi_M(x) \rightarrow \chi_R(x)$
or, equivalently:	
$\mathbf{run}^* \in \{A \subseteq E \mid \mathbf{man}^* \subseteq A\}$	$(\lambda P_{et}.\forall x_e.\mathbf{man} \rightarrow P(x))(\mathbf{run})$
$\Leftrightarrow \; \mathbf{man}^* \subseteq \mathbf{run}^*$	$\Leftrightarrow \; \forall x_e.\mathbf{man}(x) \rightarrow \mathbf{run}(x)$

GQs - more examples

(4) Some man ran.

(5) $\mathbf{run}' \in \{B \subseteq E : \mathbf{man}' \cap B \neq \emptyset\}$
$\Leftrightarrow \mathbf{man}' \cap \mathbf{run}' \neq \emptyset$
"there is an entity that is a member of both $\mathbf{man}'$ and $\mathbf{run}'$"

(6) No man ran.

(7) $\mathbf{run}' \in \{B \subseteq E : \mathbf{man}' \cap B = \emptyset\}$
$\Leftrightarrow \mathbf{man}' \cap \mathbf{run}' = \emptyset$

(8) *exactly five men*: $\{B \subseteq E : |\mathbf{man}' \cap B| = 5\}$

(9) *most men*: $\{B \subseteq E : |\mathbf{man}' \cap B| > |\mathbf{man}' \setminus B|\}$

GQ - definition

NP:(et)t + VP:et = S:t

(et)t functions ~= sets of sets of entities

Terminology: Any set $Q \subseteq \wp(E)$ (a set of subsets of E) is called a *generalized quantifier* (GQ) over E.

Other GQs with Lambda's

set theory	typed lambda's
every man:	
$\{A \subseteq E \mid M \subseteq A\}$	$\lambda P_{et}.\forall x_e.\mathbf{man}(x) \rightarrow P(x)$
some man:	
$\{A \subseteq E \mid M \cap A \neq \emptyset\}$	$\lambda P_{et}.\exists x_e.\mathbf{man}(x) \wedge P(x)$
no man:	
$\{A \subseteq E \mid M \cap A = \emptyset\}$	$\lambda P_{et}.\neg\exists x_e.\mathbf{man}(x) \wedge P(x)$
exactly five men:	
$\{A \subseteq E \mid \lvert M \cap A \rvert = 5\}$	$\lambda P_{et}.\lvert \mathbf{man}^* \cap P^* \rvert = 5$