Введение в Теоретическую Лингвистику

Филипп Дудчук
Саша Подобряев
E-mail: seminars@linguistics.msx.ru
http://seminars.narod.ru/spring2005

Лекция 4
11.03.2005

Ламбда-абстракция (λ-абстракция)

1 Композициональность значения предложения

Человеческий язык бесконечен. Несмотря на то, что его алфавит (лексикон) конечен, слова в этом алфавите могут быть бесконечной длины. Ср. (1).

(1) *Петя купил книгу, о которой ему говорил сосед его брата, который учится в университете, который находится в столице нашей страны, которая терпит острый экономический кризис, который был вызван ...*

➲ Проблема в том, что человеческий мозг конечен. Поэтому в нем знание языка должно хранится в каком-то конечном виде. Для этого мы ввели понятие порождающей грамматики, которая при помощи конечного количества правил порождает бесконечное количество синтаксических структур. Семантика предложения (условия истинности), так же как и синтаксическая структура должна «получаться» из конечного количества базовых элементов. Проще всего в качестве базовых элементов взять элементы словаря, т.к. можно спокойно предположить, что он конечен: психолингвистические исследования показывают, что весь словарь целиком хранится в голове человека в виде «начальных форм» лексем (исследования речи афатиков). Тогда значение предложение получается комбинацией значений частей этого предложения.

Каких частей?

(2) Вася дал Коле книгу.
(3) Коля дал Васе книгу.

(2) и (3) являются неэквивалентными: из (2) не следует (3) и из (3) не следует (2). При этом в (2) и (3) «используются» одни и те же слова: *Вася*, *дать*, *Коля*, *книга*. Отличия между (2) и (3) можно сформулировать только в синтаксических терминах. Это означает, что при «получении» (вычислении) семантики предложения необходимо учитывать его синтаксическую структуру.

> ПРИНЦИП КОМПОЗИЦИОНАЛЬНОСТИ
>
> Значение любого выражения есть функция от значений его частей, которая применяется к ним в том порядке, в котором они следуют друг за другом в синтаксической структуре.

2 Модельно-теоретическая семантика

Семантика должна сообщать для каждого грамматичного предложения его условия истинности. Можно написать формулу на каком-нибудь формальном языке и сказать, что условия истинности исходного предложения совпадают с условиями истинности этой формулы — ср. (4) и (4').

(4) Петя — студент.
(4') **студент** (p)

Предложение (4) может быть истинно в одно время (сейчас) и ложно в другое (через 5 лет). Кроме того, истинность (4) зависит от конкретной ситуации (кто говорит кому (4), что за мир вокруг них и проч.; можно представить себе, что через 1000 лет люди не будут знать смысла предиката **студент** — истинна или ложна тогда будет формула (4')?). Поэтому, кроме понятия условия истинности, вводится понятие **модели**, внутри которой будет «работать» наша семантика. Семантика, основанная на понятии модели, называется **модельно-теоретической**.

Модель нужна для **интерпретации** выражения. Исходно модель **M** устроена так:

(5) $\mathbf{M} = \langle D, I\rangle$

D — непустое множество индивидных констант в нашей модели, I — функция из множества E^D выражений, обозначающих индивидов из D, во множество D.

(6) $I: E^D \rightarrow D$

Как интерпретировать выражения, которые обозначают не индивидов, а более сложные вещи — например, отношения между индивидами? В **M** одноместные отношения (свойства) можно интерпретировать как подмножества D, двухместные отношения — как множества двоек индивидов, трехместные — как множество троек. Какая функция будет ставить в соответствие этим множествам соответствующие выражения естественного языка? Функция $[\![\ ]\!]^{\mathbf{M}}$. Она действует рекурсивно, последовательно разбивая на синтаксические части высказывания и ставя им в соответствие что-нибудь в модели **M**. В последний момент эта функция применяется к индивидам, тогда она эквивалентна функции I, которая уже в свою очередь каждому имени индивида ставит в соответствие элемент из D.

В качестве примера модели рассмотрим модель $M = \langle\{m, p, v\}, I\rangle$. Известно, что I(Маша) = m, I(Петя) = p, I(Вася) = v. Пусть в M будут истинны высказывания (7–9), а любые другие высказывания ложны.

(7) Маша любит Петю.
(8) Петя и Вася спят.
(9) Все счастливы.

Каким объектам нашей модели интуитивно «соответствуют» предложения (7–9)?

(7') $[\![\text{Маша любит Петю}]\!]^{\mathbf{M}} \rightarrow \{\langle I(\text{Маша}), I(\text{Петя})\rangle\} = \{\langle m, p\rangle\}$
множество, состоящее ровно из одной двойки, составленной из индивидов m и p.

(8') $[\![\text{Петя и Вася спят}]\!]^{\mathbf{M}} \rightarrow \{I(\text{Петя}), I(\text{Вася})\} = \{p, v\}$
множество, состоящее двух индивидов p и v.

(9') $[\![\text{Все счастливы}]\!]^{\mathbf{M}} \rightarrow D = \{m, p, v\}$
множество, состоящее всех индивидов множества D (домена модели M).

Оказывается, что (7) «соответствует» множеству множеств индивидов, а (8) и (9) соответствуют множествам индивидов. Отчего это зависит? По-видимому, от вершинного предиката. В (7) предикат двухместный, а в (8–9) — одноместный. Это означает, что существуют разные типы предикатов, или более широко, разные семантические т и п ы выражений.

3 Экстенсиональные типы

3.1 Основные понятия

Для того чтобы построить композициональную теорию семантики предложения, нам необходимо понимать, что значат синтаксические части предложений. Как стало понятно, эти части могут отличаться друг от друга **(экстенсиональным) типом**.

Интуитивно ясно, что все индивиды соответствуют какому-то одному типу. Обозначим этот тип так: e (англ. *entity* — сущность). Также очевидно, что все высказывания (предложения) языка соответствуют одному типу: все они могут быть либо ложными, либо истинными. Их тип обозначим так: t (англ. *truth value* — истинностное значение). Тогда можно построить типы для (почти) всех синтаксических частей (составляющих) предложения. Чтобы это понять, введем строгое определение типа.

Определение 1 (экстенсиональные типы). Пусть e — тип индивидов, t — тип истинностных значений. Тогда множество типов — такое минимальное множество TYPE, для которого верно:

1. $e \in TYPE$, $t \in TYPE$.
2. Если $\tau \in TYPE$ и $\sigma \in TYPE$, то функция $(\tau \rightarrow \sigma) \in TYPE$.

NB. Запись ‘функция $(\tau \rightarrow \sigma)$’ читается так: «(характеристическая) функция, которая ставит в соответствие элементу τ элемент σ».

Пример 1 (одноместные предикаты). Вообще о предикатах можно думать как о функциях, возвращающих истинностные значения. Если в некоторой формуле все валентности предиката заполнены константами или связанными переменными, эта формула является высказыванием. Для простоты проиллюстрируем это на примере одноместных предикатов.

(10) Коля спит.
(10') **спать** (k)

В (10') k — индивидная константа, которая заполняет единственную валентность предиката **спать**. Ср. (11), не являющийся высказыванием из-за того, что заполнены не все валентности предиката.

(11) *Коля дружит.
(11') **дружить** (k, ?)

Тогда одноместные предикаты — это функции, которые возвращают истинностные значения (предикат либо истинен, либо ложен). Что эти функции берут в качестве своего единственного аргумента? Выражение типа e (индивид). Тогда тип одноместного предиката (e → t). В дальнейшем мы будем обозначать это так: <e, t>.

Единственная польза от типов в том, что они могут композиционально комбинироваться друг с другом. Соединяясь друг с другом в дереве с самого низа, они должны на самом верху давать тип t. Как комбинируются типы между собой? По правилу аппликации.

3.2 Правило аппликации

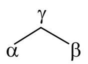
Не будем забывать, что типы — это функции. Они могут «брать» выражение одного типа и возвращать выражение другого (одноместные предикаты берут индивид и возвращают истинностное значение). Тогда они могут соединяться друг с другом следующим образом.

Правило аппликации. Комбинация типов $<\tau, \sigma>$ и τ дает выражение типа σ.

Пример 2.

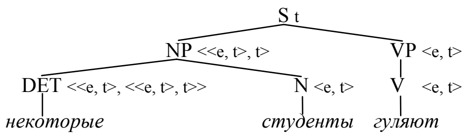
<e, t> и e дает t (второй тип является аргументом первого типа)
<<e, <e, t>>, <e, t>> и <e, <e, t>> дает <e ,t> (второй тип является аргументом первого типа)
<<e, <e, t>>, <e, t>> и <e, t> не сочетаются друг с другом (к ним неприменимо правило аппликации)

Поскольку композициональность требует учитывать синтаксическую структуру предложения, наложим синтаксическое ограничение на действие правила аппликации. Правило аппликации применяется только внутри составляющей $[_\gamma \alpha\ \beta]$ (см. (12)), причем один из узлов α и β является функцией, другой — ее аргументом. Результатом работы функции является тип составляющей γ

(12) 

Пример из жизни.

(13) Некоторые студенты гуляют.

(13') 

Почему у существительного *студенты* тип <e, t>, а не e?

Это видно из предложения *Вася — студент*. Вася имеет тип e (индивидная константа), чтобы у предложения получить тип t, составляющей [*студент*] необходимо придать тип <e, t>, как одноместному предикату. Все имена нарицательные N имеют тип <e, t> (в отличие от имен собственных, которые имеют тип e); о них можно думать как об одноместных предикатах.

3 Ламбда-абстракция

3.1 Общие сведения

Если у нас есть формула, написанная на языке исчисления предикатов первого порядка, соответствующая некоторому высказыванию русского языка, как получить формулы для частей этого высказывания? Ответ — примени λ-абстракцию.

Вообще все λ-выражения обозначают функции. $\lambda x[\varphi(x)]$ обозначает функцию, аргумент которой — переменная x и значение которой вычисляется подстановкой конкретного значения x в формулу $\varphi(x)$.

Пример 3 (λ-выражения). $\lambda x[x^2 - 5]$ обозначает функцию $x \to x^2 - 5$. По правилу аппликации, можем получить значение этой функции при x = 5: $\lambda x[x^2 - 5]\ (5) = 5^2 - 5 = 20$.

λ-исчисление было изобретено логиком Алонзо Чёрчем для того, чтобы связать функцию с ее именем. Это оказалось очень полезно в композициональной семантике.

3.2 Связь с экстенсиональными типами и λ-конверсия

Факт 1 (тип λ-выражений). Если φ — выражение некоторого типа a, а переменная x — типа b, то выражение $\lambda x[\varphi]$ — функция типа (b → a).

Поскольку все λ-выражения — функции, для них должно существовать правило аппликации (как сочетается аргумент с функцией). Это правило λ-конверсии.

Правило λ-конверсии. $\lambda x[\varphi]$ (k) есть выражение, эквивалентное формуле φ, в которой всюду заменены все свободные вхождения переменной x на значение k.

Пример 4 (λ-конверсия).

$\lambda x[x^2 - 5]\ (5) = 5^2 - 5 = 20$.
λx[**танцевать** (x)] (Вася) = **танцевать** (Вася) ⇜ Вася танцует.
λx[**любить** $(x$, Маша)] (Вася) = **любить** (Вася, Маша) ⇜ Вася любит Машу.
λx[**петь** $(x) \wedge$ **танцевать** (x)] (Вася) = **петь** (Вася) $\wedge$ **танцевать** (Вася) ⇜ Вася поет и танцует.

Вычислим тип последнего λ-выражения из Примера 4:

$$\underbrace{\lambda \underbrace{x}_{e}[\underbrace{\textbf{петь}\ (x) \wedge \textbf{танцевать}\ (x)}_{t}]}_{<e,\ t>}$$

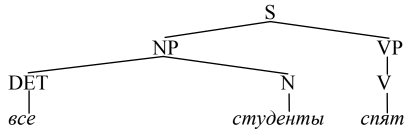
Получился тип <e, t> — тип одноместного предиката. Так оно и есть.

3.3 Конкретные примеры

Пусть у нас есть предложение русского языка (14) и его перевод на язык исчисления предикатов первого порядка (14').

(14) Все студенты спят.
(14') $\forall x$ [**студент** $(x) \to$ **спать** (x)]

Пусть мы хотим получить семантику слова *все*.

(14'') 

Для начала получим семантику именной группы [$_{NP}$ *все студенты*] для этого при помощи λ-абстракции а б с т р а г и р у е м с я от глагольной группы [$_{VP}$ *спать*]. Для этого возьмем переменную P того же типа, что глагольная группа [$_{VP}$ *спать*] (т.е. типа <e, t>, т.к. **спать** — одноместный предикат) и λ-абстрагируемся по этой переменной.

(15) ⟦все студенты⟧= $\lambda P[\forall x$ [**студент** $(x) \to P\ (x)$]

Из лекции прошлого семестра известно, что такая ИГ называется **обобщенным квантором**. Если посчитать тип выражения в (15) (так, как это делалось в Примере 4), получим тип <<e, t>, t>. Это и есть тип обобщенного квантора. Теперь при помощи другой переменной Q типа <e, t> абстрагируемся от вершины именной группы [$_N$ *студенты*]. Получим семантику кванторного слова *все*.

(16) ⟦все⟧= $\lambda Q \lambda P\ [\forall x\ [Q\ (x) \to P\ (x)]$

Аналогично можно получить λ-выражения для других обобщенных кванторов. У всех них будет тип <<e, t>, t>.

Маша:	$\lambda P\ [P(\mathsf{m})]$
любой студент:	$\lambda P[\forall x$ [**студент** $(x) \to P\ (x)$]
a student:	$\lambda P[\exists x$ [**student** $(x) \wedge P\ (x)$]
the king:	$\lambda P[\exists x$ [**king** $(x) \wedge \forall y$ [**king** $(y) \to x = y] \wedge P\ (x)$]