

Продолжения (continuations)

Г. Бронников

31 мая 2005

1 Цели семантической теории

Хотелось бы иметь теорию, в возможно бóльше степени отвечающую следующим требованиям:

1. **Композициональность.** В частности, хотелось бы, чтобы теория работала на уровне поверхностного синтаксиса, не постулируя, по возможности, нулей и невидимых передвижений.
2. **Простота.** Среди прочего, хотелось бы, чтобы единицы языка имели “естественный” тип. Так, кажется, что NP обозначают в большинстве случаев объекты типа e , непереходные глаголы имеют тип $e \rightarrow t$, переходные — тип $e \rightarrow (e \rightarrow t)$. Желательно также, чтобы каждому типу составляющей соответствовал один тип.

Таким образом, теория может содержать следующий фрагмент [Barker 2002]:

(1)	Синтаксис	Семантика
	$S \rightarrow NP VP$	$VP(NP)$
	$VP \rightarrow Vt NP$	$Vt(NP)$
	$NP \rightarrow John$	j
	$NP \rightarrow Mary$	m
	$VP \rightarrow left$	$\lambda x. left(x)$
	$Vt \rightarrow saw$	$\lambda x. \lambda y. saw(y, x)$

Сложности начинаются, когда требуется расширить теорию для работы с кванторными NP. Известны два пути решения: подъем кванторов и сдвиг типов.

Подъем кванторов позволяет добиться адекватного анализа, однако ценой уступок в композициональности. Сдвиг типов (по-видимому, наиболее полное описание такого подхода – [Hendricks 1993]) сохраняет синтаксис, однако возникают объекты сложных типов. Кроме того, анализ оказывается несимметричным относительно позиции именной группы в предложении.

2 Продолжения в информатике

Продолжения используются в Computer Science с шестидесятих годов. Наиболее известны работы [Plotkin 1975, Steele 1976].

Продолжение для данного выражения в программе – это то, что программа будет делать с его значением. Пусть, например, у нас есть выражение

$$a(2) + b(4)$$

и мы его вычисляем, вызывая операнды сложения слева направо. Тогда продолжением $a(2)$ будет

$$\lambda x.x + b(4)$$

Пусть вычисление $a(2)$ дало результат 3. Тогда продолжением выражения $b(4)$ будет

$$\lambda x.3 + x$$

Существует преобразование, которое по тексту исходной программы (написанной, как говорят, в *прямом стиле* (direct style)) получает программу, где каждая процедура в качестве дополнительного аргумента принимает продолжение – получается программа в *стиле с передачей продолжений* (continuation-passing style, CPS). Процедура эта называется *CPS-преобразованием* (CPS-transform). Так, будучи примененной к выражению

$$\lambda x.a(x) + b(x)$$

она дает

$$\lambda x.\lambda k.a'(x, \lambda y.b'(x, \lambda z.k(y + z)))$$

где a', b' – CPS-версии процедур a и b . CPS-процедура, если в качестве дополнительного аргумента ей дать “нулевое” продолжение $\lambda x.x$, эквивалентна исходной.

CPS-преобразование широко применяется в компиляторах, поскольку в CPS-версии программы разрешаются многие неоднозначности, которые могут присутствовать в исходном представлении (порядок вычисления аргументов, вызов по значению или по имени).

В выражении-результате CPS-преобразования вызов аргумента-продолжения — всегда самое последнее действие при выполнении процедуры. Однако если разрешить в CPS-программах использовать продолжения более свободно, можно смоделировать множество интересных структур управления. Самый простой и самый мощный вариант — предоставить исходной программе (написанной в “прямом стиле” доступ к объекту-продолжению. Именно это делает встроенная в язык Scheme процедура `call-with-current-continuation` (известная также как `call/cc`). Однако и помимо этого, с помощью продолжений можно описать семантику таких конструкций, как переходы (включая нелокальные — исключения), перебор с возвратами, кооперативная многозадачность.

Можно также рассматривать продолжения не относительно всей оставшейся программы, а относительно некоторого ее куска — ограниченные продолжения (delimited continuations). В языке, использующем такие продолжения, должно быть две конструкции: одна для того, чтобы отметить границу, относительно которой продолжения будут рассмариваться, и вторая, которая собственно вводит их в рассмотрение. В частности, [Danvy, Filinski 1989] предлагают *reset* и *shift* (“сброс” и “сдвиг”). Сдвиг (обозначается ξ) вводит в рассмотрение переменную, которая обозначает продолжение до ближайшего сброса (обозначается квадратными скобками). Кроме того, сдвиг имеет тело (где может использоваться введенная переменная), и это тело заменяет схваченное продолжение, то есть результат его вычисления возвращается как результат всего оператора “сброс”. Например,

$$\begin{aligned}
(2) \quad & [3 \times [10 \times (\xi f.1 + f(2))]] \\
& \triangleright [3 \times [1 + (\lambda v.[10 \times v])(2)]] \\
& \triangleright [3 \times [1 + [10 \times 2]]] \triangleright \dots \triangleright
\end{aligned}$$

3 Продолжения в формальной семантике

[Barker 2002] применяет CPS-преобразование к грамматике (1):

$$\begin{aligned}
(3) \quad S &\rightarrow \text{NP VP} && \lambda c_S.\underline{\text{NP}}(\lambda x.\underline{\text{VP}}(\lambda P.c_S(P(x)))) \\
&&& \lambda c_S.\underline{\text{VP}}(\lambda P.\underline{\text{NP}}(\lambda x.c_S(P(x)))) \\
\text{VP} &\rightarrow \text{Vt NP} && \lambda c_{\text{VP}}.\underline{\text{Vt}}(\lambda R.\underline{\text{NP}}(\lambda x.c_{\text{VP}}(R(x)))) \\
&&& \lambda c_{\text{VP}}.\underline{\text{NP}}(\lambda x.\underline{\text{Vt}}(\lambda R.c_{\text{VP}}(R(x)))) \\
\text{NP} &\rightarrow \text{John} && \lambda c_{\text{NP}}.c_{\text{NP}}(\mathbf{j}) \\
\text{NP} &\rightarrow \text{Mary} && \lambda c_{\text{NP}}.c_{\text{NP}}(\mathbf{m}) \\
\text{VP} &\rightarrow \text{left} && \lambda c_{\text{Vi}}.c_{\text{Vi}}(\lambda x.\mathbf{left}(x)) \\
\text{Vt} &\rightarrow \text{saw} && \lambda c_{\text{Vt}}.c_{\text{Vt}}(\lambda x.\lambda y.\mathbf{saw}(y, x))
\end{aligned}$$

Каждой составляющей, которая в исходной грамматике имеет тип α , CPS-грамматика присваивает тип $(\alpha \rightarrow t) \rightarrow t$ (в частности, именные группы получают знакомый тип обобщенного квантора: $(e \rightarrow t) \rightarrow t$). Значение любого предложения, полученное при помощи CPS-версии грамматики, будучи примененным к нулевому продолжению $\lambda x.x$, всегда дает тот же результат, что и исходная грамматика. Для ветвящихся вершин предлагается более одного значения (различающихся порядком вычисления аргументов), но пока что на результат они не влияют.

Теперь, однако, мы можем добавить к грамматике кванторные слова:

$$\begin{aligned}
(4) \quad \text{NP} &\rightarrow \text{everyone} && \lambda c_{\text{NP}}.\forall x : c_{\text{NP}}(x) \\
\text{NP} &\rightarrow \text{someone} && \lambda c_{\text{NP}}.\exists x : c_{\text{NP}}(x)
\end{aligned}$$

Получаем ожидаемые условия истинности:

$$\begin{aligned}
(5) \quad & \text{John saw everyone.} \\
& [\mathbf{s}[\underline{\text{NP}}_{\text{SU}}\text{John}][\underline{\text{VP}}[\underline{\text{Vt}}\text{saw}][\underline{\text{NP}}_{\text{DO}}\text{everyone}]]] \\
& \lambda c_S.\underline{\text{NP}}_{\text{SU}}(\lambda x.\underline{\text{VP}}(\lambda P.c_S(P(x)))) \\
& \lambda c_S.((\lambda c_{\text{NP}}.c_{\text{NP}}(\mathbf{j}))(\lambda x.\underline{\text{VP}}(\lambda P.c_S(P(x)))))) \\
& \lambda c_S.((\lambda x.\underline{\text{VP}}(\lambda P.c_S(P(x))))(\mathbf{j})) \\
& \lambda c_S.\underline{\text{VP}}(\lambda P.c_S(P(\mathbf{j}))) \\
& \lambda c_S.((\lambda c_{\text{VP}}.\underline{\text{Vt}}(\lambda R.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_{\text{VP}}(R(y)))))(\lambda P.c_S(P(\mathbf{j})))) \\
& \lambda c_S.\underline{\text{Vt}}(\lambda R.\underline{\text{NP}}_{\text{DO}}(\lambda y.(\lambda P.c_S(P(\mathbf{j}))))(R(y)))) \\
& \lambda c_S.\underline{\text{Vt}}(\lambda R.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_S(R(y)(\mathbf{j})))) \\
& \lambda c_S.((\lambda c_{\text{Vt}}.c_{\text{Vt}}(\lambda y'.\lambda x'.\mathbf{saw}(x', y')))(\lambda R.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_S(R(y)(\mathbf{j})))))) \\
& \lambda c_S.((\lambda R.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_S(R(y)(\mathbf{j}))))(\lambda y'.\lambda x'.\mathbf{saw}(x', y'))) \\
& \lambda c_S.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_S((\lambda y'.\lambda x'.\mathbf{saw}(x', y'))(y)(\mathbf{j}))) \\
& \lambda c_S.\underline{\text{NP}}_{\text{DO}}(\lambda y.c_S(\mathbf{saw}(\mathbf{j}, y))) \\
& \lambda c_S.((\lambda c_{\text{NP}}.\forall y' : c_{\text{NP}}(y'))(\lambda y.c_S(\mathbf{saw}(\mathbf{j}, y)))) \\
& \lambda c_S.(\forall y' : (\lambda y.c_S(\mathbf{saw}(\mathbf{j}, y)))(y')) \\
& \lambda c_S.\forall y' : c_S(\mathbf{saw}(\mathbf{j}, y'))
\end{aligned}$$

Остается применить полученное значение к нулевому продолжению $\lambda x.x$ и получить $\forall y'.\text{saw}(j, y')$.

Когда в предложении более одного квантора, начинают играть роль варианты для ветвящихся узлов дерева составляющих: в зависимости от того, какой порядок вычисления мы выберем, кванторы получают различные сферы действия.

Тут возникает интересное предсказание: поскольку выбор относительных сфер действия происходит только между ветвями одной составляющей, вариантов чтения должно получаться меньше, чем при подъеме кванторов или сдвиге типов (при трех кванторах — 4 вместо 6), а именно соблюдается следующее ограничение:

- (6) **Целостность.** Если имеется составляющая, содержащая B и C, но не содержащая A, то A должна либо включать в свою сферу действия одновременно B и C, либо ни одну из них.

примера предлагается

- (7) Most subjects put an object in every box.

Предсказываются такие результаты:

- (8) (a) most subjects > an object > every box
 (b) most subjects > every box > an object
 (c) *an object > most subjects > every box
 (d) an object > every box > most subjects
 (e) *every box > most subjects > an object
 (f) every box > an object > most subjects

Правила для вершины S приходится модифицировать: она не передает свое продолжение вовнутрь, и, таким образом, служит островом для сферы действия кванторов:

- (9) Старый вариант: $\lambda c_S.\underline{\text{NP}}(\lambda x.\underline{\text{VP}}(\lambda P.c_S(P(x))))$
 Новый вариант: $\lambda c_S.c_S(\underline{\text{NP}}(\lambda x.\underline{\text{VP}}(\lambda P.(P(x))))$

[Shan 2004] предлагает, помимо CPS-версии семантики, рассматривать семантику, записанную в “прямом стиле”, с использованием ограниченных продолжений и сдвигов. При этом синтаксические острова содержат сброс:

- (10) $S \rightarrow \text{NP VP} \quad [\text{VP}(\text{NP})]$

а семантика кванторов включает в себя сдвиг:

- (11) $\text{NP} \rightarrow \text{everyone} \quad \xi f.\forall x f(x)$
 $\text{NP} \rightarrow \text{someone} \quad \xi f.\exists x f(x)$

Этим единицам приписывается тип e_t^t , что следует читать так: данная единица ведет себя в контексте как имеющая тип e, при этом контекст должен иметь тип t, и единица преобразует его тоже в контекст типа t (в общем случае типу α_γ^δ в прямом стиле соответствует тип $(\alpha \rightarrow \gamma) \rightarrow \delta$ в CPS). Можно построить следующие варианты анализа для предложения *Someone loves everyone*:

- (12) (a) $[\mathbf{love}(\xi f_1.\exists x f_1(x), \xi f_2.\forall y f_2(y))]$
 $\triangleright[\exists x[\mathbf{love}(x, \xi f_2.\forall y f_2(y))]]$
 $\triangleright[\exists x[\forall y[\mathbf{love}(x, y)]]]$
- (b) $[\mathbf{love}(\xi f_1.\exists x f_1(x), \xi f_2.\forall y f_2(y))]$
 $\triangleright[\forall y[\mathbf{love}(\xi f_1.\exists x f_1(x), y)]]$
 $\triangleright[\forall y[\exists x[\mathbf{love}(x, y)]]]$

Шан, впрочем, предлагает другой метод управления сферами действия кванторов: всегда вычислять значение справа налево, но приписывать каждому сбросу и каждому сдвигу натуральное число, так что сброс уровня n оказывается прозрачен для сдвигов уровня $m < n$. Таким образом, предложение

- (13) *Some^m student likes everyⁿ course*

получает прямую сферу действия в случае, когда $m \leq n$, и обращенную, когда $m > n$.

Список литературы

- [Barker 2002] Barker, Chris. Continuations and the Nature of Quantification. *Natural Language Semantics* 10(3):211?242. <http://www.semanticsarchive.net/Archive/902ad5f7/barker.continuations.pdf>.
- [Danvy, Filinski 1989] Danvy, Olivier, and Andrzej Filinski. A functional abstraction of typed contexts. Tech. Rep. 89/12, DIKU, University of Copenhagen, Denmark. <http://www.daimi.au.dk/~danvy/Papers/fatc.ps.gz>.
- [Hendricks 1993] Hendricks, Herman. Studied flexibility: Categories and types in syntax and semantics. Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam.
- [Plotkin 1975] Plotkin, G.D. Call-by-name, Call-by-value and the λ -calculus. *Theoretical Computer Science* 1: 125-159. http://homepages.inf.ed.ac.uk/gdp/publications/cbn_cbv_lambda.pdf.
- [Shan 2004] Shan, Chung-chieh. Delimited Continuations in Natural Language. Quantification and Polarity Sensitivity. In *CW'04: Proceedings of the 4th ACM SIGPLAN workshop on continuations*, ed. Hayo Thielecke, 55-64. Technical report CSR-04-1, School of Computer Science, University of Birmingham. <http://arxiv.org/abs/cs.CL/0404006>.
- [Steele 1976] Steele, Guy Lewis Jr. *Lambda: The Ultimate Declarative*. MIT AI Lab. AI Lab Memo AIM-353. <http://repository.readscheme.org/ftp/papers/ai-lab-pubs/AIM-379.pdf>.