Uma linguagem para simplificação de demonstrações: "downcasing types"

Eduardo Nahum Ochs
Projeto de Pesquisa
Departamento de Matemática
Pólo Universitário de Rio das Ostras - UFF
Concurso para Professor Adjunto
Rio de Janeiro / Rio das Ostras,
9 de abril de 2008

1 Introdução

Tome uma prova de um enunciado geral; agora especialize-a para um caso particular. Essa "especialização" funciona mais ou menos como uma projeção: algumas distinções colapsam, alguns detalhes se perdem...

A técnica de simplificação de demonstrações na qual estou interessado é uma espécie de processo inverso desta "especialização". Começamos com uma "prova arquetipal" — uma prova de um certo caso particular, feita numa certa linguagem — e aí *mudamos o dicionário*; a interpretação de cada termo da prova muda, e a *mesma* prova se torna uma prova do caso geral. Chamamos isto de "levantamento"; provas arquetipais podem ser "levantadas" para o caso geral, e quando isto é feito obtemos uma prova do caso geral que usa uma linguagem herdada do caso particular arquetipal.

Estou interessado em entender melhor quatro tipos de levantamentos de provas; a linguagem de "downcased types" ("DNC", daqui por diante) é útil para todos eles.

(1) "Mundo funcional" $\to$ "Mundo lógico" (Curry-Howard)
(2) "Mundo real" $\to$ "Mundo sintático"
(3) Caso geral $\to$ caso arquetípico
(4) $\mathbf{Set}^{\mathbb{I}}/\mathcal{F} \to$ Análise não-standard.

1.1 "Mundo funcional" → "Mundo lógico" (Curry-Howard)

Compare a prova abaixo à esquerda, em Dedução Natural, de que $Q \supset R$ implica $P \land Q \supset P \land R$, com a construção do termo $\lambda d{:}A{\times}B.\langle \pi d, f(\pi' d)\rangle$: $(A{\times}B \to A{\times}C)$ em λ-cálculo simplesmente tipado:

$$\dfrac{\dfrac{\dfrac{[P\land Q]^1}{P} \qquad \dfrac{\dfrac{[P\land Q]^1}{Q} \qquad Q\supset R}{R}}{P\land R}}{(P\land Q\supset P\land R)}\,1 \qquad\qquad \dfrac{\dfrac{\dfrac{[d{:}A{\times}B]^1}{\pi d{:}A} \qquad \dfrac{\dfrac{[d{:}A{\times}B]^1}{\pi' d{:}B} \qquad f{:}B\to C}{f(\pi' d){:}C}}{\langle \pi d, f(\pi' d)\rangle{:}A{\times}C}}{\lambda d{:}A{\times}B.\langle \pi d, f(\pi' d)\rangle : A{\times}B\to A{\times}C}\,1$$

As duas têm exatamente a mesma estrutura. Isto é um exemplo do Isomorfismo de Curry-Howard em funcionamento; ele diz que há uma bijeção natural entre derivações em Dedução Natural e termos de λ-cálculo simplesmente tipado. Repare que na árvore um λ-cálculo os termos sempre crescem à medida que descemos; se usamos uma nova notação — “downcased types” — podemos não só manter os termos pequenos, como suprimir os tipos — os tipos podem ser reconstruídos “convertendo para maiúsculas” os termos. Note que os “conectivos” também têm que ser convertidos: ‘,’ convertido para maiúscula vira ‘$\times$’, e ‘$\mapsto$’ convertido para maiúscula vira ‘$\to$’.

$$\dfrac{\dfrac{\dfrac{[a,b]^1}{a} \qquad \dfrac{\dfrac{[a,b]^1}{b} \quad b{\mapsto}c}{c}}{a,c}}{a,b{\mapsto}a,c} \qquad\qquad \begin{array}{rcl} b & := & \pi'(a,b) \\ c & := & (b{\mapsto}c)(b) \\ a,c & := & \langle a,c\rangle \\ a,b{\mapsto}a,c & := & \lambda(a,b).(a,c) \end{array}$$

Agrora cada barra da árvore define um novo termo a partir de termos anteriores; isto gera o dicionário à direita... e a semântica de cada barra passar a ser: “se eu sei o significado dos termos acima da barra, eu sei o significado do termo abaixo da barra”, ou: “se eu sei ‘a’ e sei ‘c’ eu sei ‘a,c’ ”, “se eu sei ‘b’ e ‘$b{\mapsto}c$’ eu sei ‘c’ ”, etc.

Os “termos” em DNC funcionam de um modo bem diferente dos termos de λ-cálculo. Em DNC nós permitimos nomes longos para variáveis (por exemplo, ‘a,b’), a distinção sintática entre variáveis e termos não-primitivos não existe, e, aliás, sem o dicionário não é nem possível determinar só pelos nomes de dois termos qual é “mais primitivo” que o outro: por exemplo, $b{\mapsto}c$ é mais primitivo que c mas $a,b{\mapsto}a,c$ é menos primitivo que a,c.

O significado de cada termo em DNC é implícito, derivado a partir do contexto, e não explícito como em λ-cálculo... Isto faz com que no modo natural de se “pronunciar” termos e árvores em DNC os artigos sejam por default *indefinidos*: “um $b{\mapsto}c$ é uma função que leva cada b num c”, “um a,b é um elemento de $A \times B$, e é formado por um a e um b”... E, repare, daí faz sentido falar de “valor (ou significado) natural para um termo com um certo nome”. Qual é o significado natural para $a,b{\mapsto}b,a$, por exemplo? Usando o Isomorfismo de Curry-Howard podemos converter esta pergunta em: “$P \wedge Q \supset Q \wedge P$ é demonstrável em Dedução Natural? Qual é a prova?” — e uma vez encontrada a prova podemos convertê-la para um termo de λ-cálculo...

Perguntar se “$P \wedge Q \supset Q \wedge P$ é demonstrável intuicionisticamente” é o mesmo que perguntar se existe um morfimso de $P \wedge Q$ para $Q \wedge P$ numa Álgebra de Heyting livre com geradores P e Q; perguntar se “existe um termo cujo tipo é $A \times B \to B \times A$” é perguntar se existe um morfismo de $A \times B$ para $B \times A$ na categoria cartesiana fechada livre com geradores A e B. A diferença entre Álgebras de Heyting (“HAs”) e Categorias Cartesianas Fechadas (“CCCs”) é que numa HA temos no máximo um morfismo indo de um objeto dado para outro, enquanto numa CCC podemos ter vários; para converter uma CCC numa HA precisamos colapsar todos os morfismos com o mesmo domínio e mesmo codomínio num só... CCC $\to$ HA é uma projeção, e os levantamentos não são necessariamente únicos. Por exemplo, $P \land P \supset P \land P$ é demonstrável intuicionisticamente, mas existem quatro termos naturais indo de $A \times A$ em $A \times A$, não um só... uma notação em DNC não-ambígua para estes termos seria: $(a, a' \mapsto a, a)$, $(a, a' \mapsto a', a)$, etc. — ou seja, neste caso não basta converter $A \times A \to A \times A$ para minúsculas, é preciso distinguir as variáveis...

Daí, isto:

$$\begin{tikzcd} \mathrm{CCC} \arrow[d] & \text{(Mundo funcional)} \\ \mathrm{HA} & \text{(Mundo lógico)} \end{tikzcd}$$

é uma projeção; o levantamento, mesmo não funcionando sempre de modo não ambíguo, nos permite usar certos termos de DNC — por exemplo, a função "flip" $a, b \mapsto b, a$ — sem precisar definí-los, da mesma forma que usamos lemas simples, como $P \land Q \supset Q \land P$, sem demonstrá-los.

Curiosamente, várias outras operações têm bons "downcasings". Por exemplo, fixe um conjunto A; o funtor $(\times B)$, que leva cada conjunto A em $A \times B$, pode ser escrito em DNC como $a \Rightarrow a, b$ (pronúncia: "o funtor que leva cada espaço de 'a's no espaço dos 'a, b's), e o funtor $(B \to)$ pode ser escrito em DNC como $c \Rightarrow b \mapsto c$; a adjunção entre eles pode ser representada pelo diagrama à direita abaixo, que é o downcasing do diagrama à esquerda:

$$\begin{tikzcd} A \times B \arrow[d] \arrow[r, phantom, "\leftrightarrow"] & A \arrow[l, mapsto] \arrow[d] \\ C \arrow[r, mapsto] & B \to C \end{tikzcd} \qquad \begin{tikzcd} a, b \arrow[d, mapsto] \arrow[r, phantom, "\leftrightarrow"] & a \arrow[l, Leftarrow] \arrow[d, mapsto] \\ c \arrow[r, Rightarrow] & b \mapsto c \end{tikzcd}$$

Podemos escrever a versão lógica disto assim:

$$\begin{tikzcd} P \land Q \arrow[d, mapsto] \arrow[r, phantom, "\leftrightarrow"] & P \arrow[l, Leftarrow] \arrow[d, mapsto] \\ Q \arrow[r, Rightarrow] & Q \supset R \end{tikzcd}$$

No "mundo lógico" para checar que $P \Rightarrow P \land Q$ é um funtor basta checar que se um morfismo $P \mapsto P'$ existe então o morfismo $P \land Q \mapsto P' \land Q$ — sua imagem pelo funtor — também existe; não precisamos checar nem que morfismos identidade vão em identidades, nem que os funtores respeitam composição... uma prova de que algo é um funtor é menor no "mundo lógico", e para levantá-la para o "mundo funcional" precisamos provar coisas extras.

1.2 Mundo real $\to$ Mundo sintático

Os "proof assistants", como Coq ou Isabelle, são baseados em sistemas de tipos nos quais a idéia de "prova" e a de "ponto de um conjunto" são identificadas: para cada proposição P temos o conjunto das "testemunhas de que P é verdade", que ou é vazio ou é um singleton; escrevendo $W[P]$

para o conjunto das testemunhas de P, temos $W[P \land Q] = W[P] \times W[Q]$ e $W[P \supset Q] = W[P] \to W[Q]$.

Se tentamos formalizar "funtor" num proof assistant vemos que um funtor $F : \mathbf{A} \to \mathbf{B}$ tem quatro componentes:

* a sua ação nos objetos;
* a sua ação nos morfismos;
* uma prova de que ele leva identidades em identidades;
* uma prova de que ele respeita composição.

As duas últimas componentes são a "parte P" do funtor (de "provas"/"proposições"); as duas primeiras são a "parte T" (de "termos", e "tipos").

Podemos *definir* um "mundo sintático" na qual a definição de funtor só tem a parte T; idem para a definição de categoria — jogamos fora as equações $f \circ \mathrm{id} = f$, $\mathrm{id} \circ f = f$, $(f \circ g) \circ h = f \circ (g \circ h)$ —, idem para transformações naturais, etc.

Durante o meu doutorado eu imaginava que eu poderia provar algo sobre casos em que demonstrações no "mundo sintático" sobre fatos básicos de categorias se levantariam automaticamente para o "mundo real"; isto é, as "condições de funtorialidade" de um funtor se provariam automaticamente. Hoje em dia eu vejo que esse "mundo sintático" é interessante mesmo sem que o levantamento seja automático: provas em categorias se fatoram atráves de uma projeção no mundo sintático, de um modo que ainda não consegui formalizar totalmente.

1.3 Caso geral → caso arquetípico

Da mesma forma que um anel é um conjunto no qual podemos interpretar $(0, 1, +, \cdot, -)$, e em que estas operações se comportam "como deveriam" — isto é, certas equações são obedecidas; vamos imaginar que uma operação só pode receber o nome '+' se ela "merece este nome", i.e., se este '+' obedece certas equações pré-definidas — uma HA é uma estrutura (uma categoria) na qual podemos interpretar $(\top, \land, \lor, \supset, \bot)$, uma hiperdoutrina é uma estrutura na qual podemos interpretar lógica de 1^{a} ordem com igualdade, e um topos é uma estrutura na qual podemos interpretar "Local Set Theory" ([Bell]), que é um pouco mais que lógica de 1^{a} ordem com igualdade...

O diagrama abaixo mostra alguns tipos de estruturas, à esquerda, e à direita as operações que podem ser interpretadas nelas (suas "linguagens internas"). As setas verticais são "especializações"; a bijeção $\mathsf{Topos}_{-} \leftrightarrow \mathsf{Topos}_{+}$ diz que as duas definições de topos — uma com só quatro axiomas, a outra com uma operação para cada conectivo, quantificador, etc — são equivalentes; as diagonais pontilhadas são as projeções do isomorfismo de Curry-Howard, que mudam os conectivos.

$$\begin{tikzcd}
 & \mathsf{CCC_L} \arrow[d] & \mathsf{CCC_F} \arrow[l, dotted] \arrow[d] \\
 & \mathsf{BiCCC_L} \arrow[d] & \mathsf{BiCCC_F} \arrow[l, dotted] \arrow[dl] \\
 & \mathsf{Hyp} \arrow[d] & \\
\mathsf{Topos_-} \arrow[r, leftrightarrow] \arrow[dd] & \mathsf{Topos_+} \arrow[d] & \\
 & \mathsf{Set_{DTT}} \arrow[dl] & \\
\mathbf{Set} & &
\end{tikzcd}$$

$$\begin{tikzcd}
 & (\top,\wedge,\supset) \arrow[d] & (*,(,),\mapsto) \arrow[l, dotted] \arrow[d] \\
 & (\top,\wedge,\vee,\supset,\bot) \arrow[d] & (*,(,),\sqcup,\mapsto,\bot) \arrow[l, dotted] \arrow[dl] \\
 & \begin{array}{c}(*,(,),\sqcup,\mapsto,\bot)\\(\top,\wedge,\vee,\supset,\bot)\\(\exists,\forall,=,\mathrm{subst})\end{array} \arrow[d] & \\
(*,\mathrm{pb},\mapsto,\{|\}) \arrow[r, leftrightarrow] \arrow[dd] & \mathsf{LST} \arrow[d] & \\
 & \mathsf{Set_{DTT}} \arrow[dl] & \\
\mathsf{ZFC} & &
\end{tikzcd}$$

A prova de qualquer uma destas equivalências entre “categorias com certas estruturas extras” e “modelos para uma certa linguagem” é técnica, longa, e difícil... mas usando DNC é possível simplificar estas provas bastante. O topos arquetípico é **Set**; idem para a hiperdoutrina arquetípica, a CCC arquetípica, etc. As provas em DNC podem ser feitas usando a notação de **Set** e daí levantadas para o caso geral; os diagramas em DNC permanecem os mesmos, mas se usarmos o dicionário para expandir cada construção o resultado são expressões que só usam as operações categóricas.

Toposes e hiperdoutrinas também são usados para construir modelos para linguagens que não podem ser interpretadas consistentemente em **Set** — por exemplo, análise não-standard (infinitesimais numa ultrapotência de **Set**), geometria diferencial sintética (infinitesimais nilpotentes), e polimorfismo (). Em todos estes casos, curiosamente, a notação em DNC continua funcionando — a linguagem passa a falar de uma categoria de conjuntos com algumas operações a mais.

1.4 Análise Não-Standard

Um modo de provar que $\lim_{n\to+\infty}(1+\frac{a}{n})^n = e^a$, em análise não-standard, é provar que para qualque número natural infinitamente grande ω temos $(1+\frac{a}{\omega})^\omega \sim e^a$; os passos da prova são os abaixo:

$$
\begin{aligned}
\log{(1+\tfrac{a}{\omega})^\omega} &= \omega\ \log{(1+\tfrac{a}{\omega})} \\
&= \omega\ (\log 1 + ((\log' 1) + \mathbf{o})\ \tfrac{a}{\omega}) \\
&= \omega\ (0 + (1+\mathbf{o})\ \tfrac{a}{\omega}) \\
&= \omega\ (1+\mathbf{o})\ \tfrac{a}{\omega} \\
&= (1+\mathbf{o})\ a
\end{aligned}
$$

$$
\begin{aligned}
(1+\tfrac{a}{\omega})^\omega &= e^{((1+\mathbf{o})\ a)} \\
&= e^{(a+\mathbf{o}a)} \\
&= e^{(a+\mathbf{o}')} \\
&= e^a + \mathbf{o}''.
\end{aligned}
$$

Em alguns destes passos novos símbolos — $\mathbf{o}, \mathbf{o}', \mathbf{o}''$ — são introduzidos; seus nomes ('o'-zinhos) indicam que eles são infinitesimais, e há quantificadores implícitos: "existe um único valor para $\mathbf{o}$ (ou $\mathbf{o}'$, ou $\mathbf{o}''$) que faz a igualdade valer".

2 O projeto

Referências

[Bell]: Bell, J.L. Toposes and Local Set Theory, Oxford, 1988.

[Jacobs]: Jacobs, B. Categorical Logic and Type Theory, Studies in Logic and the Foundations of Mathematics 141, North Holland, Elsevier, 1999.

[Pitts]: Pitts, A.M. Polymorphism is Set Theoretic, Constructively. Springer Lecture Notes In Computer Science, vol. 283, pp. 12–39, 1987.

[Reynolds]: J. Reynolds. Polymorphism is not settheoretic. In Kahn, McQueen and Plotkin (editors), Symposium on semantics of data types, Volume 173 of Lecture Notes in Computer Science. Springer Verlag, 1984.

[Robinson]: Robinson, A. Non-standard analysis, Revised edition, Princeton University Press, 1976.

[SeelyHyp]: Seely, R.A.G. Hyperdoctrines, natural deduction and the Beck condition. Z. Math. Logik Grundlag. Math. 29 (1983), no. 6, 505–542.

[SeelyPLC]: Seely, R.A.G. Categorical semantics for higher order polymorphic lambda calculus. J. Symbolic Logic 52 (1987), no. 4, 969–989.

[S/L]: Stroyan, K., e Luxembourg, W.A.J. Introduction to the Theory of Infinitesimals. Academic Press, 1976.

[Wadler]: Wadler, P. Theorems for free! 4th International Conference on Functional Programming and Computer Architecture, London, September 1989.