UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ALEXANDRE COSTARD SOARES

Uma construção categórica de um modelo do cálculo lambda

RIO DE JANEIRO
2021

ALEXANDRE COSTARD SOARES

Uma construção categórica de um modelo do cálculo lambda

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Hugo de Holanda Cunha Nobrega, D.Sc.

RIO DE JANEIRO

2021

CIP - Catalogação na Publicação

S676c

Soares, Alexandre Costard
Uma construção categórica de um modelo do cálculo lambda / Alexandre Costard Soares. -- Rio de Janeiro, 2021.
28 f.

Orientador: Hugo de Holanda Cunha Nobrega.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2021.

1. Cálculo lambda. 2. Teoria das categorias. 3. Modelos. I. Nobrega, Hugo de Holanda Cunha, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

ALEXANDRE COSTARD SOARES

Uma construção categórica de um modelo do cálculo lambda

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 15 de JUNHO de 2021

BANCA EXAMINADORA:

Prof. Hugo de Holanda Cunha Nobrega, D.Sc.

Prof. João Antônio Recio da Paixão, D.Sc.

Prof. Severino Collier Coutinho, D.Sc.

AGRADECIMENTOS

Agradeço à Natália pela parceria nessa aventura. Seu apoio foi fundamental para eu chegar até aqui.

Ao professor Hugo, pelas aulas de categorias, pela orientação, pela paciência e pelas conversas quase semanais em tempos de isolamento social.

Aos professores João Paixão e Severino Collier, por aceitarem participar da banca examinadora e pelas críticas e sugestões.

Aos colegas do grupo de estudos de categorias, pela troca e companhia no início da pandemia.

À Larissa, Giselle, Fidel e Khadar, pelo apoio e carinho ao longo desses anos.

RESUMO

Desenvolvido na década de 1930, o cálculo lambda captura a noção de computabilidade de maneira distinta, porém equivalente, às máquinas de Turing, o que permitiu que ele encontrasse aplicações importantes na matemática e na computação, em particular na teoria das linguagens de programação. Seu primeiro modelo não trivial foi construído usando topologia e publicado em 1970. Este trabalho detalha uma derivação desse modelo usando teoria de domínios e teoria das categorias.

Palavras-chave: cálculo lambda. modelos. teoria de categorias. teoria de domínios.

ABSTRACT

Developed in the 1930's, the lambda calculus captures the notion of computability in a different but equivalent way to the Turing machines. This allowed important applications in mathematics and computer science, particularly in programming language theory. Its first non trivial model was constructed using topology and published in 1970. This text shows in details a derivation of such model using domain theory and category theory.

Keywords: lambda calculus. models. category theory. domain theory.

SUMÁRIO

1 INTRODUÇÃO

Cálculo λ é o nome usado atualmente para se referir a uma coleção de sistemas lógicos baseados em sistemas homônimos criados por Church ao longo da década de 1930 (CHURCH, 1932; CHURCH, 1933) para descrever propriedades gerais de funções. Tais sistemas acabaram posteriormente tendo aplicações em outras áreas da matemática, linguística e ciência da computação, em especial na teoria de linguagens de programação. A influência do cálculo λ é especialmente vista em linguagens ditas *funcionais*, a ponto de ser mencionado ou introduzido de maneira informal em livros de programação.

As primeiras versões do cálculo λ foram desenvolvidas em um período no qual se buscava um formalismo para estabelecer os fundamentos da matemática. Um dos grandes problemas dessa época era o problema da decisão (*Entscheidungsproblem*) proposto por Hilbert em 1928, no qual ele indaga sobre a existência de um algoritmo, no sentido de um método sistemático ou uma sequência de passos, que conseguiria decidir se uma fórmula da lógica de primeira ordem é consequência de outras fórmulas.

No final da década de 1920, Church começou a formular um sistema formal que ele julgava mais natural do que a teoria de tipos de Russell ou a teoria de conjuntos de Zermelo. Ele baseou seu sistema no conceito de funções. Contudo, diferente da concepção de mapeamento entre conjuntos atualmente difundida, Church considerava funções como regras que transformam um argumento em um valor, o que enfatiza seu aspecto computacional. Ele esperava conseguir enriquecer tal sistema a ponto de conseguir tomá-lo como fundação para a lógica e para a matemática. Contudo, os sistemas desenvolvidos inicialmente se mostraram inconsistentes e os sistemas consistentes posteriores eram demasiado fracos para tal objetivo.

O primeiro sistema desenvolvido foi publicado em 1932, uma lógica formal que tinha o cálculo λ como substrato. Rapidamente foi verificado que essa lógica possuía uma inconsistência que permitia provar fórmulas contraditórias, o que o levou a publicar uma revisão no ano seguinte. Church, com dois estudantes, Kleene e Rosser, fizeram várias descobertas sobre a lógica e o cálculo λ. Infelizmente, uma dessas descobertas foi que a lógica publicada em 1933 também era inconsistente. Mas o cálculo λ usado se mostrou bastante rico.

Um dos resultados obtidos foi a primeira resposta ao problema da decisão, publicado em 1936. Para isso foi necessário formalizar a noção de algoritmo. O cálculo λ, com sua propriedade de tratar funções como regras, se mostrou bastante adequado para isso. Church propôs formalizar a noção de algoritmo através da ideia de que uma função entre naturais é efetivamente computável caso possa ser representada por um λ-termo. Posteriormente, no mesmo ano, Turing publicou um resultado similar usando para isso o que chamamos atualmente de máquinas de Turing. Uma função é efetivamente computável

caso possa ser computada por uma máquina de Turing. Nos anos que seguiram foi provado que essas duas noções de computabilidade são equivalentes.

No final da década de 30, após uma nova tentativa fracassada de criar um sistema formal para fundamentar a matemática, Church decide mudar de foco e cria uma teoria de tipos para o cálculo λ que décadas à frente viria a influenciar as teorias de tipos usadas em linguagens de programação. Ao longo das décadas de 40 e 50 o interesse no cálculo λ diminuiu. As máquinas de Turing passaram a ser o modelo preferido para estudos em computabilidade, influenciando a criação dos primeiros computadores.

No fim da década de 50 McCarthy cria o LISP tendo o cálculo λ como uma de suas principais influências, e no início da década de 60 Landin propõe o uso de λ-termos no Algol 60. Essa influência sobre as linguagens de programação renova o interesse no cálculo λ e traz alguns descontentamentos sobre a fundamentação matemática dos sistemas desenvolvidos. Um desses críticos é Scott. Em 1969, um mês após advogar por uma alternativa tipada às linguagens não tipadas existentes, Scott consegue construir um modelo para o cálculo λ, conhecido como D_∞. Os métodos usados inspiraram diversos modelos posteriores. Na década de 70, o estudo das propriedades computacionais dos elementos de D_∞ teve grande influência no método de formalização de linguagens de programação chamado *semântica denotacional*.

Um aspecto curioso na história do cálculo λ é que o primeiro modelo tenha surgido mais de três décadas após a publicação das primeiras teorias. A construção de modelos é interessante tanto como prova de consistência quanto para a investigação das propriedades das estruturas que são modelos. Há alguns motivos para esse hiato. Um deles é que o interesse principal no cálculo λ nas suas primeiras décadas envolvia a criação de um formalismo independente da teoria de conjuntos. Outro é que, como veremos, a construção de modelos é relativamente sofisticada, já que o cálculo λ permite, por exemplo, a autoaplicação de funções.

No próximo capítulo vamos introduzir os conceitos básicos do cálculo λ, seus termos, uma de suas teorias mais usadas e descrever como é, de maneira abstrata, um modelo dessa teoria. Além disso, vamos ilustrar como o cálculo λ pode ser usado para codificar números naturais e funções computáveis.

No capítulo 3 iremos realizar, passo a passo, a construção do modelo D_∞, seguindo uma linha de raciocínio similar à originalmente apresentada por Scott (SCOTT, 1970) mas usando estruturas de teoria das categorias no lugar de conceitos de topologia originalmente usados.

Finalmente, no capítulo 4 indicaremos algumas possibilidades de expansão em torno do que desenvolvemos.

Nossa contribuição principal é o detalhamento da construção de D_∞, que não conseguimos encontrar na literatura de forma completa. Em (BARENDREGT, 1984) é feita declaração de que D_∞ é o ponto fixo de um funtor, com a demonstração deixada como

exercício para o leitor. Em (LAMBEK; SCOTT, 1988) é apresentada uma linha de raciocínio bastante similar a que apresentaremos na seção 3.3, contudo em outra categoria. Nosso objetivo, portanto, é mostrar como esse modelo pode ser construído de maneira que um leitor com domínio básico dos conceitos de teoria das categorias e de ordens parciais consiga acompanhar. Há alguns resultados que usaremos sem demonstração. Essa omissão foi feita para demonstrações diretas porém longas. Nesses casos entendemos que o benefício obtido por apresentá-las seria superado pela perda de concisão e foco do texto.

Ao leitor interessado na história do cálculo λ, recomendamos (CARDONE; HINDLEY, 2006), referência principal que usamos para as informações históricas apresentadas neste capítulo.

2 O CÁLCULO λ

Veremos agora o cálculo λ em sua versão mais simples, no sentido de ter menos símbolos e regras. Essa versão também é chamada de cálculo λK quando se deseja diferenciá-la de outras. Apesar de sua simplicidade, as regras estabelecidas são suficientes para capturar a noção de computabilidade.

2.1 TERMOS

Definição 2.1.1. As primitivas do cálculo λ são os símbolos λ, (,) e uma sequência infinita de símbolos $v_0, v_1, v_2, \ldots$ chamados *variáveis*. Um λ-*termo* é definido através das regras

1. Uma variável é um termo;
2. Se M for um termo, $\lambda x(M)$, em que x é uma variável, também é um termo. Termos com a forma $\lambda x(M)$ são comumente chamados de *abstrações*;
3. Se M e N forem termos, MN é um termo. Temos com a forma MN são comumente chamados de *aplicações*.

Ao longo do texto usaremos as seguintes convenções, relativamente comuns na literatura:

- letras minúsculas denotam variáveis;
- letras maiúsculas denotam termos arbitrários;
- parênteses externos são omitidos;
- $M \equiv N$ denota igualdade sintática, ou seja, que M é o mesmo termo que N;
- aplicações associam à esquerda, e.g. $MNP \equiv (MN)P$;
- abstrações associam à direita, e.g. $\lambda x \lambda y(M) \equiv \lambda x(\lambda y(M))$;
- a variável da abstração é separada do termo por um ponto, e.g. $\lambda x \,.\, M \equiv \lambda x(M)$;
- abstrações sucessivas são denotadas usando apenas um λ, e.g. $\lambda xy \,.\, M \equiv \lambda x \,.\, \lambda y \,.\, M$.

Temos, portanto, termos como xyz, $\lambda x \,.\, xy$, $\lambda xyz \,.\, xz(xz)$. Para entender como eles podem ser usados para formalizar computações, vamos apresentar de maneira informal algumas regras para operar com termos e interpretá-los. Posteriormente vamos formalizar essas regras.

Um termo com forma de abstração pode ser interpretado como uma função que mapeia a variável que segue o λ ao que está à direita do ponto. Assim, o termo $\lambda x\,.\,x$ representa a função identidade, que retorna seu argumento sem modificações. O termo $\lambda x\,.\,xx$ mapeia um termo M nele aplicado a si mesmo, MM. Interpretando abstrações dessa forma, o símbolo usado para a variável entre o λ e o ponto não possui significado em si. Os termos $\lambda x\,.\,xx$ e $\lambda y\,.\,yy$ são equivalentes.

Um termo com forma de aplicação MN é interpretado como a avaliação da função M com argumento N. Assim, se $M \equiv \lambda x\,.\,xyx$, teríamos $MN = NyN$. A aplicação exige alguns cuidados. Um exemplo ilustrativo disso é quando temos $M \equiv \lambda x\,.\,(\lambda y\,.\,x)$. M representaria uma função que, dado termo, retorna uma função constante. Por exemplo, $Mz = \lambda y\,.\,z$. Contudo, $My = \lambda y\,.\,y$, não é uma função constante e sim a identidade. Uma solução para essas situações de captura de variáveis é renomear a variável da abstração, o que, como discutimos há pouco, não altera o significado do termo M.

Finalmente, para um exemplo de como podemos realizar algumas operações simples com as regras apresentadas, sejam $S \equiv \lambda ufx\,.\,f(ufx)$ e $Z_0 \equiv \lambda fx\,.\,x$. Temos que

$$\begin{aligned} SZ_0 &\equiv (\lambda ufx\,.\,f(ufx))(\lambda fx\,.\,x) \\ &= (\lambda ufx\,.\,f(ufx))(\lambda gy\,.\,y) \\ &= \lambda fx\,.\,f((\lambda gy\,.\,y)fx) \\ &= \lambda fx\,.\,f((\lambda y\,.\,y)x) \\ &= \lambda fx\,.\,fx \end{aligned} \tag{2.1}$$

É fácil verificar que $S(SZ_0) = \lambda fx\,.\,f(fx)$ e assim sucessivamente, ou seja, começando com Z_0, cada aplicação de S introduz uma nova aplicação de f no corpo da abstração. Definindo $Z_{n+1} \equiv SZ_n$, para $n \geq 0$, temos uma codificação em λ-termos para os números naturais. Outras operações podem ser construídas, por exemplo, podemos definir $\texttt{soma} \equiv \lambda mnfx\,.\,mf(nfx)$, de modo que

$$\begin{aligned} \texttt{soma}\ Z_i\ Z_j &= (\lambda mnfx\,.\,mf(nfx))\ Z_i\ Z_j \\ &= \lambda fx\,.\,Z_i f(Z_j fx) \\ &= \lambda fx\,.\,(\lambda gy\,.\,\underbrace{g(g(\ldots(g}_{i\text{ vezes}}y)))f(Z_j fx) \\ &= \lambda fx\,.\,(\lambda y\,.\,\underbrace{f(f(\ldots(f}_{i\text{ vezes}}y)))(Z_j fx) \\ &= \lambda fx\,.\,\underbrace{f(f(\ldots(f}_{i\text{ vezes}}(Z_j fx)))) \\ &= \lambda fx\,.\,\underbrace{f(f(\ldots(f}_{i\text{ vezes}}(\underbrace{f(f(\ldots(f}_{j\text{ vezes}}(x)))))))) \\ &= Z_{i+j} \end{aligned}$$

Essa codificação dos naturais é conhecida como *numerais de Church* e foi a partir dela que Church chegou em seus resultados sobre a computabilidade de funções entre naturais. Veremos mais alguns detalhes sobre estes resultados mais adiante.

2.2 TEORIA FORMAL

Vamos agora apresentar uma maneira de formalizar a operação informal apresentada. Em uma abstração $\lambda x \,.\, M$, dizemos que M é o *escopo* de λx. Uma ocorrência de uma variável x é dita *ligada* caso esteja no escopo de algum λx ou seja a própria ocorrência de x no λx. Caso contrário, essa ocorrência é dita *livre*. Denotamos por $FV(M)$ o conjunto de variáveis que ocorrem livremente em M. Por exemplo, em $M \equiv \lambda z \,.\, xzxy$, as ocorrências de z são ligadas e as de x e y são livres. Temos nesse caso $FV(M) = \{x, y\}$.

Denotamos por $[N/x]M$ a substituição das ocorrências livres de x em M por N, renomeando as variáveis ligadas para evitar a captura de variáveis. Essa mudança de variáveis ligadas em geral é permitida pelas teorias e, como vimos, faz sentido quando interpretamos os termos como funções e aplicações.

Definição 2.2.1. A *teoria formal de β-equivalência*, ou simplesmente *teoria $\lambda\beta$*, possui como fórmulas equações entre termos do cálculo λ. Ela possui os esquemas de axiomas

$$M = M$$

$$\lambda x \,.\, M = \lambda y \,.\, [y/x]M \qquad \text{se } y \notin FV(M) \qquad (\alpha)$$

$$(\lambda x \,.\, M)N = [N/x]M \qquad (\beta)$$

E como regras de inferência[1]

$$\frac{M = N}{N = M} \qquad \frac{M = N \quad N = P}{M = P}$$

$$\frac{M = N}{PM = PN} \qquad \frac{M = N}{MP = NP}$$

$$\frac{M = N}{\lambda x \,.\, M = \lambda x \,.\, N}$$

O axioma (α), usualmente chamado de α-equivalência, é o que nos permite renomear variáveis ligadas. O axioma (β), chamado de β-redução ou β-conversão, permite reescrever uma aplicação através de substituição. A reescrita feita pela β-redução captura tanto a ideia de aplicação de função quanto a de um passo de computação. O raciocínio desenvolvido nas equações 2.1 consistiu do uso de α uma vez seguido de diversas β reduções.

Em (BARENDREGT, 1984) há um exemplo de um resultado relativamente simples de ser mostrado usando $\lambda\beta$ porém de grande importância.

[1] Essa representação das regras tem a semântica de que para se obter o que está abaixo da linha basta que se obtenha o que está acima.

Teorema 2.2.1. Todo termo possui um ponto fixo, ou seja, para todo termo M existe um termo X tal que $MX = X$.

Demonstração. Seja $W \equiv \lambda x \,.\, M(xx)$ e $X \equiv WW$. Então

$$X \equiv WW \equiv (\lambda x \,.\, M(xx))W = M(WW) = MX \qquad \square$$

Além disso podemos definir $Y \equiv \lambda f \,.\, (\lambda x \,.\, f(xx))(\lambda x \,.\, f(xx))$ tal que $YM = M(YM)$, ou seja, Y aplicado a um termo M é o ponto fixo de M.

Esse resultado aparentemente simples é de grande relevância pois o uso de pontos fixos é fundamental na codificação de funções recursivas, e portanto para formalizar a noção de função efetivamente computável, como visto em (HINDLEY; SELDIN, 2008, nota 4.15). Um exemplo simples de recursão é a função fatorial. Para não nos alongarmos nas diversas codificações, vamos supor que temos um termo `pred` tal que `pred` $Z_{n+1} = Z_n$, um termo `mul` tal que `mul` $Z_n\ Z_m = Z_{n\times m}$ e um termo `zero?` tal que `zero?` $N\ A\ B$ é A se $N \equiv Z_0$ e B caso contrário. Com isso poderíamos tentar definir fatorial como

$$\texttt{fac} \equiv \lambda n \,.\, (\texttt{zero?}\ n\ Z_1\ (\texttt{mul}\ n\ (\texttt{fac}\ (\texttt{pred}\ n))))$$

Contudo, `fac` está sendo definida em função de si, e a teoria desenvolvida não possui mecanismos para fazer isso de maneira direta. Contudo, se definirmos

$$F \equiv \lambda f n \,.\, (\texttt{zero?}\ n\ Z_1\ (\texttt{mul}\ n\ (f\ (\texttt{pred}\ n))))$$

Temos que

$$\begin{aligned} YF &= F(YF) \\ &= \lambda n \,.\, (\texttt{zero?}\ n\ Z_1\ (\texttt{mul}\ n\ (YF\ (\texttt{pred}\ n)))) \end{aligned}$$

de modo que podemos tomar `fac` como YF. A representação de funções recursivas no formalismo é fundamental para justificar a proposta de que o cálculo λ captura a noção de algoritmo e para mostrar sua equivalência com as máquinas de Turing.

2.3 MODELOS

Vamos agora apresentar uma noção de modelo para o cálculo λ relativamente familiar, dada sua semelhança com os modelos da lógica de primeira ordem. Ela é comumente usada quando se deseja verificar que uma determinada estrutura é de fato um modelo. É isso que faremos no próximo capítulo.

Definição 2.3.1. Uma *estrutura aplicativa* $\langle D, \cdot \rangle$ consiste de um conjunto D com pelo menos dois elementos e uma operação binária $\cdot$, potencialmente parcial, que mapeia D^2 em D. O conjunto D é chamado de domínio da estrutura.

Definição 2.3.2. Uma *valoração* é um mapeamento de variáveis em um domínio D. Dada uma valoração ρ, a notação $[d/x]\rho$ é usada para uma valoração ρ' tal que $\rho'(x) = d$ e $\rho'(v) = \rho(v)$ para $v \not\equiv x$.

Definição 2.3.3. Um *modelo* para a teoria $\lambda\beta$ é uma tripla $\langle D, \cdot, [\![\,]\!]\rangle$, com $\langle D, \cdot\rangle$ sendo uma estrutura aplicativa e $[\![\,]\!]$ tal que, dado um termo M e uma valoração ρ, $[\![M]\!]_\rho$ é mapeado em D da seguinte maneira:

1. $[\![x]\!]_\rho = \rho(x)$
2. $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$
3. $\forall d \in D \,.\, [\![\lambda x \,.\, M]\!]_\rho \cdot d = [\![M]\!]_{[d/x]\rho}$
4. $(\forall x \in FV(M) \,.\, \rho(x) = \sigma(x)) \Longrightarrow [\![M]\!]_\rho = [\![M]\!]_\sigma$
5. $y \notin FV(M) \Longrightarrow [\![\lambda x \,.\, M]\!]_\rho = [\![\lambda y \,.\, [y/x]M]\!]_\rho$
6. $(\forall d \in D \,.\, [\![\lambda x \,.\, M]\!]_\rho \cdot d = [\![\lambda x \,.\, N]\!]_\rho \cdot d) \Longrightarrow [\![\lambda x \,.\, M]\!]_\rho = [\![\lambda x \,.\, N]\!]_\rho$

1 e 2 são a definição de interpretação para variáveis e aplicações. 3 interpreta a regra β e 5 a regra α. 4 é uma propriedade comum de se exigir de modelos. 6 diz que se o resultado de aplicar as interpretações de duas abstrações em todo elemento de D é o mesmo, então essas interpretações são iguais. Isso é bastante similar a noção de que se duas funções f e g satisfazem $f(x) = g(x)$ em todo o domínio, então $f = g$, o que reforça a ideia de que abstrações podem ser interpretadas como funções.

Esse tipo de modelo por vezes é referenciado na literatura como um *modelo de ambiente* ou um *modelo sintático*.

Dados um modelo $\mathbb{D} = \langle D, \cdot, [\![\,]\!]\rangle$, uma valoração ρ, e dois termos M e N, dizemos que ρ e $\mathbb{D}$ satisfazem uma equação $M = N$ se, e somente se, $[\![M]\!]_\rho = [\![N]\!]_\rho$. Além disso, dizemos que $\mathbb{D}$ satisfaz $M = N$ se, e somente se, $M = N$ é satisfeita com qualquer valoração.

É possível mostrar (HINDLEY; SELDIN, 2008, teorema 15.12) que um modelo com as propriedades enumeradas satisfaz todas as equações demonstráveis na teoria $\lambda\beta$. Essa prova é feita por indução nos axiomas e regras da teoria.

3 O MODELO D_∞

Buscando uma maneira de formular uma teoria que permitisse "dar uma semântica matemática para linguagens de programação"(SCOTT, 1970) Scott formulou o primeiro modelo não trivial do cálculo λ não tipado, conhecido como D_∞. Os conceitos e técnicas usados na construção desse modelo influenciaram modelos mais simples posteriores.

A construção de Scott foi feita operando topologicamente sobre reticulados completos, um tipo de ordem parcial. Nossa abordagem será através de estruturas abstratas da teoria de categorias usando conceitos da teoria de domínios. Nossa linha de raciocínio será semelhante a apresentada por Scott em seu trabalho original (SCOTT, 1970) e por Lambek e Scott (LAMBEK; SCOTT, 1988).

Ao longo deste capítulo vamos usar uma notação chamada λ *informal* para representar funções usando o símbolo $\lambda\!\!\lambda$. Essa notação, embora obviamente inspirada no cálculo λ, é apenas uma abreviação conveniente. Por exemplo, usaremos $\lambda\!\!\lambda\, x\, .\, k$ para denotar uma função constante que mapeia qualquer x em seu domínio em k.

3.1 CONCEITOS BÁSICOS DE TEORIA DE DOMÍNIOS

Nesta seção vamos introduzir os conceitos de teoria dos domínios que usaremos. Para um tratamento mais completo o leitor pode consultar o (ABRAMSKY; JUNG, 1994, capítulo 2) ou (HINDLEY; SELDIN, 2008, capítulo 16).

Definição 3.1.1. Uma *ordem parcial* $\langle D, \sqsubseteq \rangle$ consiste de um conjunto D não vazio e uma relação $\sqsubseteq$ reflexiva, antissimétrica e transitiva.

Definição 3.1.2. Dada uma ordem parcial $\langle D, \sqsubseteq \rangle$ um subconjunto não vazio X de D é dito *direcionado*, caso para quaisquer dois elementos de X exista algum elemento de X maior ou igual a ambos, ou seja,

$$\forall x, y \in X\, .\, \exists z \in X\, .\, x \sqsubseteq z \wedge y \sqsubseteq z.$$

Definição 3.1.3. Dado um $X \subseteq D$, um elemento $d \in D$ é dito um *limite superior* de X se $\forall x \in X\, .\, x \sqsubseteq d$. Caso exista, o *supremo* de X, denotado por $\bigsqcup X$, é o menor limite superior de X, ou seja,

$$\forall d \in D\, .\, ((\forall x \in X\, .\, x \sqsubseteq d) \Longleftrightarrow \bigsqcup X \sqsubseteq d)$$

Definição 3.1.4. Uma *ordem parcial completa para direcionados* (usualmente abreviada por DCPO[1]) é uma ordem parcial $\langle D, \sqsubseteq \rangle$ que satisfaz as seguintes propriedades:

[1] Do inglês, *directed complete partial order*. Como é a forma mais usual na literatura, nesse texto vamos usar a abreviação em inglês.

- D possui um elemento *mínimo*, denotado por $\bot$, que satisfaz $\forall d \in D \,.\, \bot \sqsubseteq d$.
- Todo $X \subseteq D$ direcionado tem um supremo, denotado por $\bigsqcup X$.

Definição 3.1.5. Uma função $f : D \to D'$ entre as DCPOs $\langle D, \sqsubseteq \rangle$ e $\langle D', \sqsubseteq' \rangle$ é dita *monótona* se, para quaisquer $x, y \in D$ tais que $x \sqsubseteq y$ temos $f(x) \sqsubseteq' f(y)$.

Definição 3.1.6. Uma função monótona $f : D \to D'$ entre as DCPOs $\langle D, \sqsubseteq \rangle$ e $\langle D', \sqsubseteq' \rangle$ é dita *contínua* se, para qualquer subconjunto direcionado X de D

$$f(\bigsqcup X) = \bigsqcup \{f(x) \,|\, x \in X\}$$

Ou seja, a função mapeia o supremo de um subconjunto no supremo de sua imagem. Além disso, dadas $f \in [D \to D']$ e $g \in [D' \to D'']$, a composição $f \,;\, g \in [D \to D'']$[2].

Denotamos por $[D \to D']$ o conjunto de funções contínuas de D em D'. A ordem parcial em D' nos permite definir uma ordem parcial em $[D \to D']$ como[3].

$$f \sqsubseteq g \iff \forall d \in D \,.\, f(d) \sqsubseteq' g(d).$$

É direto mostrar que $[D \to D']$ possui um mínimo, dado pela função $\bot(d) = \bot'$. Também é possível mostrar que todo subconjunto direcionado de $[D \to D']$ possui um supremo. Logo, o conjunto $[D \to D']$ com a ordem parcial induzida por D' é uma DCPO.

Lema 3.1.7. Se $A = \bigcup_{i \in I} A_i$ então $\bigsqcup A = \bigsqcup_{i \in I} (\bigsqcup A_i)$ sempre que os supremos indicados existirem.

Demonstração. Ver (ABRAMSKY; JUNG, 1994, proposição 2.1.4) □

O lema 3.1.7, junto com a monotonicidade, nos permite chegar a outro resultado útil.

Lema 3.1.8. Seja $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ uma cadeia crescente de funções monótonas com supremo $\bigsqcup_{i \in \mathbb{N}} f_i$ e g uma função monótona arbitrária. Temos

$$\bigsqcup_{i,j \in \mathbb{N}} (f_i \,;\, g \,;\, f_j) = \bigsqcup_{i \in \mathbb{N}} \left(\bigsqcup_{j \in \mathbb{N}} (f_i \,;\, g \,;\, f_j) \right) = \bigsqcup_{j \in \mathbb{N}} \left(\bigsqcup_{i \in \mathbb{N}} (f_i \,;\, g \,;\, f_j) \right) = \bigsqcup_{i \in \mathbb{N}} (f_i \,;\, g \,;\, f_i)$$

[2] Neste texto usamos $f \,;\, g$ como sinônimo de $g \circ f$. Essa notação de composição na mesma ordem da leitura é conveniente quando estamos lidando com diagramas.

[3] A rigor, as duas relações de ordenamento que aparecem nessa equação são distintas. Seria mais correto escrever algo como

$$f \sqsubseteq^{\to} g \iff \forall d \in D \,.\, f(d) \sqsubseteq' g(d).$$

Em geral vamos usar a notação simplificada, especificando apenas quando necessário

Demonstração. Vamos inicialmente definir os conjuntos

$$\begin{aligned} A &= \{f_i \,;\, g \,;\, f_j \,|\, i \in \mathbb{N}, j \in \mathbb{N}\} \\ V_k &= \{f_k \,;\, g \,;\, f_j \,|\, j \in \mathbb{N}\} \\ H_k &= \{f_i \,;\, g \,;\, f_k \,|\, i \in \mathbb{N}\} \\ D_k &= \{f_i \,;\, g \,;\, f_j \,|\, i \leq k, j \leq k\} \end{aligned}$$

Portanto $A = \bigcup_{k\in\mathbb{N}} V_k = \bigcup_{k\in\mathbb{N}} H_k = \bigcup_{k\in\mathbb{N}} D_k$. Pelo lema 3.1.7,

$$\bigsqcup A = \bigsqcup_{k\in\mathbb{N}} \left(\bigsqcup V_k\right) = \bigsqcup_{k\in\mathbb{N}} \left(\bigsqcup_{j\in\mathbb{N}} (f_k \,;\, g \,;\, f_j)\right)$$

O resultado para H_k segue de forma análoga. Finalmente, para D_k, como os f_i estão em uma cadeia crescente e g é monótona, temos que $f_i \,;\, g \,;\, f_j \sqsubseteq f_k \,;\, g \,;\, f_k$ para $i, j \leq k$, de modo que

$$\bigsqcup A = \bigsqcup_{k\in\mathbb{N}} \left(\bigsqcup D_k\right) = \bigsqcup_{k\in\mathbb{N}} (f_k \,;\, g \,;\, f_k) \qquad \square$$

Definição 3.1.9. Uma *projeção* de D' em D é um par $\langle f, f^R\rangle$ de funções $f \in [D \to D']$ e $f^R \in [D' \to D]$ que satisfazem as seguintes propriedades

$$f \,;\, f^R = I_D$$

$$f^R \,;\, f \sqsubseteq I_{D'}$$

em que $I_X \in [X \to X]$ é a função identidade. Para que não haja confusão quando temos projeções mais complexas, vamos denotá-las por $\left[f \mid f^R\right]$. A composição de projeções é também é uma projeção e é dada por

$$\left[f \mid f^R\right] \,;\, \left[g \mid g^R\right] = \left[f \,;\, g \mid g^R \,;\, f^R\right]$$

É possível mostrar que, dada uma função f que é o lado esquerdo de uma projeção, seu par f^R é único e vice-versa. Assim, quando não houver ambiguidade, nos referiremos à projeção $\left[f \mid f^R\right]$ como f.

3.2 CONCEITOS BÁSICOS DE TEORIA DAS CATEGORIAS

Nesta seção iremos introduzir os conceitos básicos de teoria das categorias que usaremos no texto. Para um tratamento mais completo o leitor pode consultar (AWODEY, 2010).

Definição 3.2.1. Uma *categoria* consiste de

- uma coleção de *objetos*;

- uma coleção de *setas* (também chamadas de *morfismos*);
- uma função *domínio dom*, que mapeia cada seta em um objeto;
- uma função *codomínio cod*, que mapeia cada seta em um objeto;
- dadas duas setas f e g tais que $cod(f) = dom(g)$, existe uma seta $f\,;g$, chamada *composta* de f com g, com $dom(f\,;g) = dom(f)$ e $cod(f\,;g) = cod(g)$; e
- para cada objeto A, uma seta id_A tal que $dom(id_A) = cod(id_A) = A$

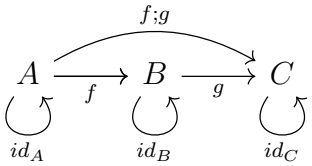
Por vezes será mais conveniente denotar uma seta f cujo domínio é A e codomínio B por $f : A \to B$. Vale ressaltar que setas não precisam ser funções, como ilustraremos mais adiante.

A composição de setas deve ser associativa, ou seja, $(f\,;g)\,;h = f\,;(g\,;h)$. Além disso, para qualquer seta $f : A \to B$, devemos ter que $id_A\,;f = f$ e $f\,;id_B = f$.

Um exemplo simples de categoria é uma ordem parcial $\langle D, \sqsubseteq \rangle$, cujos objetos são os elementos de D e as setas $f : A \to B$ indicam $A \sqsubseteq B$. As setas compostas vêm da transitividade de $\sqsubseteq$ e as identidades da reflexividade.

Outro exemplo é a categoria **Set**, que possui como objetos conjuntos e como setas funções. A composição de setas é a composição de funções usual e as identidades são as próprias funções identidade de cada conjunto.

Categorias podem ser convenientemente representadas usando diagramas. Por exemplo, no diagrama 3.1 abaixo temos os objetos A, B, C e as setas $f : A \to B$, $g : B \to C$ e a composta $f\,;g : A \to C$, além das identidades de cada objeto.



(3.1)

É comum a omissão das identidades e compostas quando não houver ambiguidade. O diagrama 3.1 ficaria

$$A \xrightarrow[f]{} B \xrightarrow[g]{} C$$

Definição 3.2.2. Uma seta $f : A \to B$ é dita um *isomorfismo* caso exista uma seta $g : B \to A$ tal que $f\,;g = id_A$ e $g\,;f = id_B$. Dizemos um objeto A é *isomorfo* a B, ou $A \cong B$ caso exista um isomorfismo entre eles.

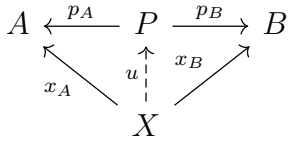
Como veremos, a noção de isomorfismo tem um papel fundamental na construção de estruturas abstratas. Essas estruturas em geral são definidas a partir de uma *propriedade de mapeamento universal*, ou UMP[4], em uma dada configuração de objetos e setas. A seguir definimos algumas estruturas abstratas.

[4] Do inglês, *universal mapping property*.

Definição 3.2.3. O *produto* de dois objetos A e B é um objeto P, domínio de duas setas $p_A : P \to A$ e $p_B : P \to B$, tal que, para qualquer outro objeto X e par de setas $x_A : X \to A$ e $x_B : X \to B$ existe uma única seta $u : X \to P$ que satisfaz

$$x_A = u \,;\, p_A$$
$$x_B = u \,;\, p_B$$

Usando diagramas, temos



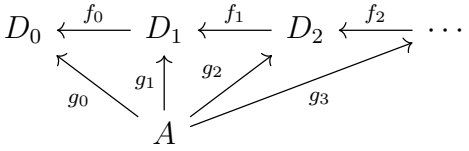
A propriedade de existência e unicidade da seta u é o que chamamos de UMP do produto.

Na categoria formada por uma ordem parcial, o produto entre dois objetos é o ínfimo entre os dois, ou seja, o maior elemento menor que ambos, caso exista. Em **Set**, o produto é o produto cartesiano usual.

Definição 3.2.4. Denominamos um *ω-diagrama*[5] em uma categoria a seguinte configuração de objetos e setas

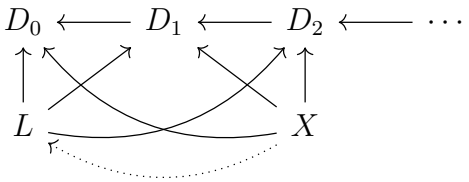
$$D_0 \xleftarrow{f_0} D_1 \xleftarrow{f_1} D_2 \xleftarrow{f_2} \cdots \tag{3.2}$$

Definição 3.2.5. Um *cone* para um ω-diagrama consiste de um objeto A, chamado *ápice*, e setas $g_i : A \to D_i$, para cada $i \in \mathbb{N}$, tais que $\forall i \in \mathbb{N} \,.\, g_i = g_{i+1} \,;\, f_i$. A configuração abaixo ilustra tal cone.



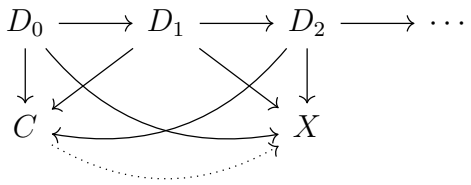
(3.3)

Definição 3.2.6. Um *limite* de um ω-diagrama, caso exista, é um cone com ápice L e setas $g_i : L \to D_i$ com $i \in \mathbb{N}$ tal que, para qualquer outro cone de ápice X com setas $h_i : X \to D_i$, existe uma única seta $u : X \to L$ tal que $\forall i \in \mathbb{N} \,.\, h_i = u \,;\, g_i$.



[5] Essa definição e as que seguem são casos particulares, suficientes para nosso interesse, dos conceitos mais gerais de cones e limites para diagramas arbitrários. Por exemplo, produtos também podem ser vistos como limites de certos diagramas. As definições gerais podem ser vistas em (AWODEY, 2010, capítulo 5)

Definição 3.2.7. Dado um ω-diagrama com as setas invertidas, um *cone dual* consiste de um ápice C e setas $g_i : D_i \to C$ satisfazendo $g_{i+1} = f \,;\, g_i$, com $i \in \mathbb{N}$. Um *colimite* de um ω-diagrama, caso exista, é um cone dual tal que, para qualquer outro cone dual com ápice X e setas $h_i : D_i \to X$, existe uma única seta $u : C \to X$ que satisfaz $\forall i \in \mathbb{N} \,.\, h_i = g_i \,;\, u$.



Definimos acima três estruturas abstratas: produtos, limites e colimites. Uma pergunta que podemos fazer é: o que acontece se mais de um objeto satisfizer a mesma UMP? Nesse caso, não é difícil mostrar que as setas únicas entre eles definem um isomorfismo. Assim, dizemos que os objetos que satisfazem as UMPs são únicos a menos de isomorfismo. Em geral não estamos interessados na igualdade literal entre objetos e sim em isomorfismos, o que justifica usarmos termos como *o* limite de um diagrama no lugar de *um* limite de um diagrama.

Esse não é o caso para as setas. Ao definir uma seta única, uma UMP acaba sendo uma ferramenta poderosa, permitindo concluir igualdade entre setas. Faremos uso dessa propriedade na seção 3.3.

Definição 3.2.8. Um *funtor* $F : \mathbf{A} \to \mathbf{B}$ é um mapeamento de objetos e setas da categoria $\mathbf{A}$ em, respectivamente, objetos e setas da categoria $\mathbf{B}$, que satisfaz as seguintes propriedades:

- $dom(F(f)) = F(dom(f))$, para qualquer seta f em $\mathbf{A}$
- $cod(F(f)) = F(cod(f))$, para qualquer seta f em $\mathbf{A}$
- $F(id_X) = id_{F(X)}$, para qualquer objeto X em $\mathbf{A}$
- $F(f \,;\, g) = F(f) \,;\, F(g)$, para quaisquer setas f e g em $\mathbf{A}$ com $cod(f) = dom(g)$.

Assim, um funtor mapeia uma categoria em outra preservando a estrutura categórica. Se pensarmos em diagramas, podemos, de maneira informal, pensar na ação de funtor como o mapeamento de um diagrama na categoria $\mathbf{A}$ em um diagrama semelhante na categoria $\mathbf{B}$, preservando as setas e, no máximo, "colapsando" vários objetos em um só.

Uma convenção relativamente comum que vamos adotar é a omissão dos parênteses na aplicação de funtores quando não houver ambiguidade. Assim, temos que $FX = F(X)$.

3.3 CONSTRUÇÃO DE D_∞

Vamos agora iniciar a construção do D_∞. Para isso, vamos trabalhar em duas categorias. A categoria **CPO**[6] tem como objetos DCPOs e como setas funções contínuas. A categoria **CPO*** é uma subcategoria de **CPO** com os mesmos objetos e cujas setas $f : D \to D'$ são projeções de D' em D, ou seja, dados dois objetos A e B e duas setas $f : A \to B$ e $f^R : B \to A$ em **CPO** tais que $\left[f \mid f^R\right]$ formam uma projeção, $\left[f \mid f^R\right]$ é uma seta entre A e B em **CPO***. Como mencionamos no fim da seção 3.1, quando não houver ambiguidade denotaremos a projeção $\left[f \mid f^R\right]$ apenas por f.

Como mencionamos anteriormente, buscamos uma estrutura aplicativa na qual podemos aplicar termos a si mesmos. Uma maneira de se ter essa propriedade é através de uma DCPO D que satisfaça

$$D \cong [D \to D]$$

com isomorfismo $\varphi : D \to [D \to D]$. Vamos primeiro verificar que, caso tenhamos tal objeto, podemos construir um modelo. Para isso vamos definir uma operação binária $\cdot$ e uma interpretação $[\![\,]\!]$.

Definição 3.3.1. Seja $\varphi : D \to [D \to D]$ um isomorfismo entre D e $[D \to D]$.

1. Para quaisquer x e $y \in D$, definimos a operação binária $\cdot$ como

$$x \cdot y = \varphi(x)(y)$$

2. Seja ρ uma valoração em D. Definimos $[\![\,]\!]$ por recursão:

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x) \\ [\![MN]\!]_\rho &= [\![M]\!]_\rho \cdot [\![N]\!]_\rho \\ [\![\lambda x \,.\, M]\!]_\rho &= \varphi^{-1}(\lambda\!\!\lambda\, d \,.\, [\![M]\!]_{[d/x]\rho}) \end{aligned}$$

Para mostrar que $\mathbb{D} = \langle D, \cdot, [\![\,]\!] \rangle$ é um modelo, temos que mostrar que $\mathbb{D}$ satisfaz as seis propriedades enumeradas em 2.3.3. 1 e 2 seguem direto da definição de $[\![\,]\!]$. Para mostrar 3, temos

$$\begin{aligned} [\![\lambda x \,.\, M]\!]_\rho \cdot d &= \varphi^{-1}(\lambda\!\!\lambda\, e \,.\, [\![M]\!]_{[e/x]}) \cdot d \\ &= \varphi(\varphi^{-1}(\lambda\!\!\lambda\, e \,.\, [\![M]\!]_{[e/x]}))(d) \\ &= (\lambda\!\!\lambda\, e \,.\, [\![M]\!]_{[e/x]})(d) \\ &= [\![M]\!]_{[d/x]} \end{aligned}$$

[6] A rigor, o nome **DCPO** seria mais apropriado, mas como o nome **CPO** é o que é usualmente encontrado na literatura, vamos usá-lo.

Para 6, temos que

$$\begin{aligned}\forall d \in D \,.\, (\llbracket M \rrbracket_{[d/x]\rho} = \llbracket M \rrbracket_{[d/x]\rho}) &\Longrightarrow \lambda\!\!\!\lambda\, d \,.\, \llbracket M \rrbracket_{[d/x]\rho} = \lambda\!\!\!\lambda\, d \,.\, \llbracket N \rrbracket_{[d/x]\rho} \\ &\Longrightarrow \varphi^{-1}(\lambda\!\!\!\lambda\, d \,.\, \llbracket M \rrbracket_{[d/x]\rho}) = \varphi^{-1}(\lambda\!\!\!\lambda\, d \,.\, \llbracket N \rrbracket_{[d/x]\rho}) \\ &\Longrightarrow \llbracket \lambda x \,.\, M \rrbracket_\rho = \llbracket \lambda x \,.\, N \rrbracket_\rho\end{aligned}$$

5 vem da substituição satisfazer $\llbracket [N/x]M \rrbracket_\rho = \llbracket M \rrbracket_{[\llbracket N \rrbracket_\rho / x]\rho}$ (BARENDREGT, 1984, lema 5.1.5). Finalmente, 4 pode ser demonstrada por indução em M.

Portanto, se construirmos um objeto D isomorfo a $[D \to D]$ podemos derivar daí um modelo. Como veremos, vamos construir esse objeto não em **CPO** mas em **CPO***. Definindo o funtor F que leva objetos D em $[D \to D]$, na mesma categoria, podemos reformular o problema como a busca de um objeto D que satisfaça

$$D \cong FD$$

logo, estamos buscando um ponto fixo do funtor F. Uma seta $\left[f \mid f^R\right] : D \to D'$ em **CPO*** é transformada pelo funtor F da seguinte maneira:

$$F\left[f \mid f^R\right] = \left[\lambda\!\!\!\lambda\, g \,.\, (f^R \,;\, g \,;\, f) \mid \lambda\!\!\!\lambda\, h \,.\, (f \,;\, h \,;\, f^R)\right]$$

em que $g : D \to D$ e $h : D' \to D'$. Os diagramas abaixo ilustram a ação de F sobre setas.

$$\begin{array}{ccc} D & \xrightarrow{\;g\;} & D \\ {\scriptstyle f^R}\uparrow & & \downarrow{\scriptstyle f} \\ D' & \dashrightarrow & D' \end{array} \qquad\qquad \begin{array}{ccc} D & \dashrightarrow & D \\ \downarrow{\scriptstyle f} & & \uparrow{\scriptstyle f^R} \\ D' & \xrightarrow[h]{} & D' \end{array}$$

Uma estratégia usada para encontrar o ponto fixos de funtores (MÉTAYER, 2003) é buscar o colimite do diagrama

$$D \xrightarrow{\;f\;} FD \xrightarrow{\;Ff\;} FFD \xrightarrow{\;FFf\;} \cdots$$

Para entender porque usar essa estratégia, vamos começar com um diagrama mais geral em **CPO***

$$D_0 \xrightarrow{\;f_0\;} D_1 \xrightarrow{\;f_1\;} D_2 \xrightarrow{\;f_2\;} \cdots \tag{3.4}$$

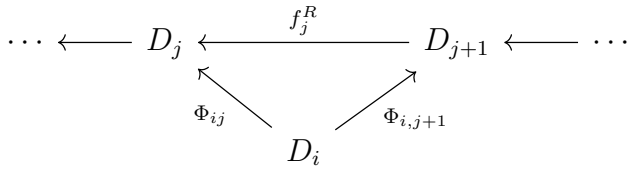
Uma primeira observação que podemos fazer sobre o diagrama 3.4 é que ele induz, em **CPO**, o diagrama

$$D_0 \xleftarrow{\;f_0^R\;} D_1 \xleftarrow{\;f_1^R\;} D_2 \xleftarrow{\;f_2^R\;} \cdots \tag{3.5}$$

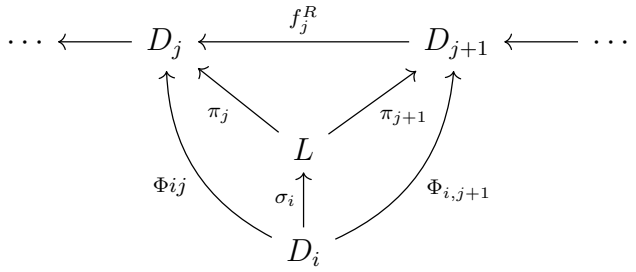
Usando as setas f_i e f_i^R, podemos construir setas $\Phi_{ij} : D_i \to D_j$ da seguinte maneira

$$\Phi_{ij} = \begin{cases} f_i \,;\, f_{i+1} \,;\, \ldots \,;\, f_{j-1} & \text{se } i < j, \\ I_{D_i} & \text{se } i = n, \\ f_{i-1}^R \,;\, f_{i-2}^R \,;\, \ldots \,;\, f_j^R & \text{se } i > j. \end{cases}$$

de modo que todos os D_i são ápices de cones



Como veremos posteriormente, o diagrama 3.5 possui limite, cujo ápice denotaremos por L e setas π_i, de maneira que, pela UMP do limite, temos a seta $\sigma_i : D_i \to L$ e a configuração representada pelo diagrama



ou seja, temos um par de setas $L \to D_i$ e $D_i \to L$. Da UMP do limite temos

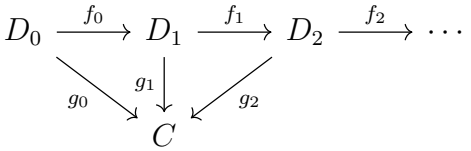
$$\begin{aligned}\Phi_{i,i+1} &= f_i \ = \sigma_i \,;\, \pi_{i+1} \\ \Phi_{i,i-1} &= f_i^R = \sigma_{i+1} \,;\, \pi_i\end{aligned}$$

e com isso podemos concluir que $\pi_i\sigma_i \sqsubseteq \pi_{i+1}\sigma_{i+1}$. Temos também que a cadeia crescente $\pi_i\sigma_i$ tem supremo id_L. Para mostrar isso, vamos analisar dois casos. Primeiro, se $i < j$ segue-se que $\pi_i \,;\, \Phi_{ij} = \pi_j \,;\, \Phi_{ji} \,;\, \Phi_{ij} \sqsubseteq \pi_j$. Segundo, se $i \geq j$ temos $\pi_i \,;\, \Phi_{ij} = \pi_j$. Portanto

$$\bigsqcup_{i\geq 0} (\pi_i \,;\, \Phi_{ij}) = \bigsqcup_{i\geq 0} (\pi_i \,;\, \sigma_i \,;\, \pi_j) = \pi_j.$$

Como π_j é contínua, $\bigsqcup_{i\in\mathbb{N}} (\pi_i \,;\, \sigma_i \,;\, \pi_j) = \bigsqcup_{i\in\mathbb{N}} (\pi_i \,;\, \sigma_i)\,;\pi_j$. Mas, se considerarmos o próprio L e as setas π_i como o "outro cone" na UMP do limite, vemos que a única seta s que satisfaz $\forall j \in \mathbb{N} \,.\, \pi_j = s \,;\, \pi_j$ é a identidade, e assim temos que $\bigsqcup_{i\in\mathbb{N}} (\pi_i \,;\, \sigma_i) = id_L$. Temos então que $[\sigma_i \mid \pi_i]$ é uma projeção de L em D_i. Finalmente, concluímos que L é o ápice de um cone dual do diagrama 3.4. O lema 3.3.2 abaixo nos diz que L é colimite do diagrama 3.4.

Lema 3.3.2. Seja C o ápice de um cone dual da forma



em **CPO***. Se $\bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, g_i\right) = id_C$, então C é colimite do ω-diagrama dual que forma sua base.

Demonstração. Para verificar que C satisfaz a UMP do colimite, temos que mostrar que, dado qualquer outro ápice A de cone dual com setas $h_i : D_i \to A$, existe uma única seta $a : C \to A$ que satisfaz

$$\forall i \in \mathbb{N} \,.\, h_i = g_i \,;\, a. \tag{3.6}$$

Para isso vamos primeiro supor que a satisfaz 3.6. Temos que

$$\begin{aligned} a &= id_C \,;\, a \\ &= \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, g_i\right) \,;\, a \\ &= \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, g_i \,;\, a\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, h_i\right) \end{aligned}$$

logo, apenas $a = \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, h_i\right)$ pode satisfazer 3.6. Para provar a existência, vamos verificar que a seta a encontrada de fato satisfaz a equação 3.6.

$$\begin{aligned} g_i \,;\, a &= g_i \,;\, \bigsqcup_{j\in\mathbb{N}} \left(g_j^R \,;\, h_j\right) \\ &= \bigsqcup_{j\in\mathbb{N}} \left(g_i \,;\, g_j^R \,;\, h_j\right) \\ &= \bigsqcup_{j\in\mathbb{N}} \left(\Phi_{ij} \,;\, h_j\right) \\ &= h_i \end{aligned}$$

Finalmente, vamos verificar que a seta $a^R = \bigsqcup_{i\in\mathbb{N}} \left(h_i^R \,;\, g_i\right)$ junto com a seta a formam uma projeção.

$$\begin{aligned} a \,;\, a^R &= \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, h_i\right) \,;\, \bigsqcup_{j\in\mathbb{N}} \left(h_j^R \,;\, g_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \bigsqcup_{j\in\mathbb{N}} \left(g_i^R \,;\, h_i \,;\, h_j^R \,;\, g_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \bigsqcup_{j\in\mathbb{N}} \left(g_i^R \,;\, \Phi_{ij} \,;\, g_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \left(g_i^R \,;\, g_i\right) \\ &= id_C \end{aligned}$$

$$\begin{aligned} a^R \,;\, a &= \bigsqcup_{i\in\mathbb{N}} \left(h_i^R \,;\, g_i\right) \,;\, \bigsqcup_{j\in\mathbb{N}} \left(g_j^R \,;\, h_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \bigsqcup_{j\in\mathbb{N}} \left(h_i^R \,;\, g_i \,;\, g_j^R \,;\, h_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \bigsqcup_{j\in\mathbb{N}} \left(h_i^R \,;\, \Phi_{ij} \,;\, h_j\right) \\ &= \bigsqcup_{i\in\mathbb{N}} \left(h_i^R \,;\, h_i\right) \\ &\sqsubseteq id_A \end{aligned}$$

Sabemos que os supremos usados nas definições de a e a^R existem pois são supremos de cadeias crescentes em uma DCPO. Portanto C é o colimite do diagrama 3.4. □

Uma consequência do lema 3.3.2 é que não importa o ponto inicial do diagrama tomado como base, já que, para uma cadeia crescente x_i, $\bigsqcup_{i\geq 0} x_i = \bigsqcup_{i\geq n} x_i$ para qualquer n finito.

Finalmente, estamos devendo mostrar que o diagrama 3.5 possui limite em **CPO**. Vamos fazer isso construindo esse limite concretamente. Para isso vamos definir um cone com ápice

$$L = \{g : \mathbb{N} \to \bigcup_{i\in\mathbb{N}} D_i \,|\, \forall i \in \mathbb{N} \,.\, g(i) \in D_i \text{ e } g(i) = f_i^R(g(i+1))\}$$

e setas $p_i : L \to D_i$, com $i \in \mathbb{N}$, tais que, dado um $g \in L$, $p_i(g) = g(i)$. Para mostrar que L é de fato o limite do diagrama 3.5, temos que mostrar que ele satisfaz a UMP do limite, ou seja, mostrar que existe uma única seta $u : X \to L$, em que X é o ápice de outro cone, com setas h_i tal que $\forall i \in \mathbb{N} \,.\, h_i = u \,;\, p_i$. Da UMP temos

$$\forall i \in \mathbb{N} \,.\, \forall x \in X \,.\, h_i(x) = p_i(u(x)) = u(x)(i)$$

Podemos, portanto, definir $u : X \to L$ como $\lambda\!\!\lambda\, x \,.\, \lambda\!\!\lambda\, i \,.\, h_i(x)$, ou seja, uma função que mapeia um $x \in X$ na função $\lambda\!\!\lambda\, i \,.\, h_i(x)$. Vamos verificar a continuidade de u. Seja Y um subconjunto direcionado de X. Então

$$\begin{aligned} u(\bigsqcup Y) &= \lambda\!\!\lambda\, i \,.\, h_i(\bigsqcup Y) \\ &= \lambda\!\!\lambda\, i \,.\, \bigsqcup_{y\in Y} h_i(y) \\ &= \bigsqcup_{y\in Y} (\lambda\!\!\lambda\, i \,.\, h_i(y)) \\ &= \bigsqcup_{y\in Y} u(y) \end{aligned}$$

Para verificar a unicidade de u, basta supor que exista um w que satisfaz $\forall i \in \mathbb{N} \,.\, (h_i = w \,;\, p_i)$ e verificar que $w = u$.

Com os resultados estabelecidos temos ferramentas o suficiente para voltar para o problema inicial. Desejamos encontrar o ponto fixo do funtor F que leva uma DCPO D

em $[D \to D]$. O passo inicial é construir uma seta $f : D \to FD$ em **CPO***. Para isso vamos definir as funções $f : D \to FD$ e $f^R : FD \to D$ abaixo:

$$f(d) = \lambda\!\!\lambda\, x \,.\, d \tag{3.7}$$

$$f^R(g) = g(\bot_{D_0}) \tag{3.8}$$

É fácil verificar que esse par de funções define uma projeção de FD em D. Podemos, portanto, construir o diagrama em **CPO***

$$D \xrightarrow{f} FD \xrightarrow{Ff} FFD \xrightarrow{FFf} \cdots$$

Esse diagrama é um caso particular de 3.4, possuindo, portanto, colimite C. Além disso, ele é mapeado pelo funtor F em

$$FD \xrightarrow{Ff} FFD \xrightarrow{FFf} FFFD \xrightarrow{FFFf} \cdots$$

que, como vimos, não só possui colimite como possui o mesmo colimite que o anterior. Assim, se FC for o colimite do segundo diagrama, temos que $C \cong FC$.

Temos o seguinte cone em **CPO***

$$\begin{array}{ccccc} \longrightarrow & D_i & \xrightarrow{F^i\left[f \mid f^R\right]} & D_{i+1} & \longrightarrow \\ & {\scriptstyle F[\sigma_i \mid \pi_i]}\downarrow & \swarrow_{F[\sigma_{i+1} \mid \pi_{i+1}]} & & \\ & FC & & & \end{array}$$

Caso $\bigsqcup_{i\in\mathbb{N}} \left((F\sigma_i)^R \,;\, (F\sigma_i)\right)$ seja $id_{[C\to C]}$ teremos que FC é o colimite do diagrama. Para verificar que essa igualdade vale, vamos relembrar a ação de F nas setas de **CPO***.

$$F\left[f \mid f^R\right] = \left[\lambda\!\!\lambda\, g \,.\, (f^R \,;\, g \,;\, f) \mid \lambda\!\!\lambda\, h \,.\, (f \,;\, h \,;\, f^R)\right].$$

Assim, temos que

$$(F\sigma_i)^R \,;\, (F\sigma_i) = \lambda\!\!\lambda\, k \,.\, (\pi_i \,;\, \sigma_i \,;\, k \,;\, \pi_i \,;\, \sigma_i).$$

Sabemos que $\pi_i \,;\, \sigma_i$ é uma cadeia crescente cujo supremo é id_C. Portanto, para um $k \in [C \to C]$ arbitrário, usando o lema 3.1.8, temos

$$\begin{aligned} \bigsqcup_{i\in\mathbb{N}} (\pi_i \,;\, \sigma_i \,;\, k \,;\, \pi_i \,;\, \sigma_i) &= \bigsqcup_{i\in\mathbb{N}} \bigsqcup_{j\in\mathbb{N}} (\pi_i \,;\, \sigma_i \,;\, k \,;\, \pi_j \,;\, \sigma_j) \\ &= \bigsqcup_{i\in\mathbb{N}} (\pi_i \,;\, \sigma_i) \,;\, k \,;\, \bigsqcup_{j\in\mathbb{N}} (\pi_j \,;\, \sigma_j) \\ &= id_C \,;\, k \,;\, id_C \\ &= k. \end{aligned}$$

Assim, temos que

$$(F\sigma_i)^R \,;\, (F\sigma_i) = \lambda\!\!\lambda\, k \,.\, k = id_{[C\to C]}$$

e podemos portanto concluir que FC é o colimite do diagrama 3.3. Essa DCPO que é o ponto fixo de F é o D_∞.

4 CONSIDERAÇÕES FINAIS E PERSPECTIVAS FUTURAS

Ao longo do texto apresentamos formalmente o cálculo λ, a teoria $\lambda\beta$ e uma definição de como são os modelos dessa teoria. A partir disso, formulamos o problema de se encontrar um modelo para o cálculo λ usando um ponto fixo de um funtor em **CPO***. Com essa formulação, detalhamos uma maneira de se encontrar tal ponto fixo. Podemos identificar alguns pontos de expansão a partir do detalhamento feito.

Primeiro, em nossa derivação de D_∞ usamos uma construção concreta do limite de um ω-diagrama em **CPO***. Uma expansão relativamente direta seria construir esse limite de maneira abstrata. Nessa mesma direção de tornar abstratas construções concretas, podemos investigar quais são as propriedades necessárias de um funtor para que ele mapeie o ω-diagrama dual 3.5 no colimite do ω-diagrama dual resultante. Essa investigação naturalmente leva ao conceito de funtores ω-contínuos (MÉTAYER, 2003).

Finalmente, podemos examinar modelos que generalizam a condição $D \cong [D \to D]$. Lembrando que introduzimos esse isomorfismo para dar conta do fato de os λ-termos poderem ser interpretados como valores e como funções. Contudo, a exigência de isomorfismo é muito forte. Na definição 3.3.1 poderíamos ter exigido apenas um par de setas $f : D \to [D \to D]$ e $g : [D \to D] \to D$ tais que $g \,;\, f = id_{[D \to D]}$. Quando exigimos o isomorfismo, estamos identificando elementos de D que são mapeados na mesma função. Essa exigência é conhecida como extensionalidade. Ao relaxar o isomorfismo, entramos no domínio dos modelos não extensionais.

Nosso objetivo com esse trabalho foi detalhar uma construção categórica de D_∞ de maneira que um leitor que domine os conceitos básicos de teoria das categorias conseguisse acompanhar. Acreditamos que essa construção seja um exercício interessante por intercalar a operação abstrata com os conceitos mais concretos em ordens parciais, o que pode facilitar o entendimento. Além disso, esperamos ter oferecido uma boa contextualização para o leitor que deseje se aprofundar no estudo do cálculo λ e áreas correlatas, como a lógica combinatória e a teoria de linguagens de programação.

REFERÊNCIAS

ABRAMSKY, S.; JUNG, A. **Domain Theory**. Oxford: Clarendon Press, 1994.

AWODEY, S. **Category theory**. Oxford: Oxford University Press, 2010.

BARENDREGT, H. P. **The lambda calculus**. Amsterdam: North-Holland, 1984.

CARDONE, F.; HINDLEY, J. R. History of lambda-calculus and combinatory logic. In: **Handbook of the History of Logic, vol. 5: Logic from Russell to Church**. Amsterdam: Elsevier, 2006. p. 723–817.

CHURCH, A. A set of postulates for the foundation of logic. **Annals of Mathematics**, Annals of Mathematics, v. 33, n. 2, p. 346–366, 1932.

CHURCH, A. A set of postulates for the foundation of logic. **Annals of Mathematics**, Annals of Mathematics, v. 34, n. 4, p. 839–864, 1933.

HINDLEY, J. R.; SELDIN, J. P. **Lambda-calculus and combinators, an introduction**. Cambridge: Cambridge University Press Cambridge, 2008.

LAMBEK, J.; SCOTT, P. J. **Introduction to higher-order categorical logic**. Cambridge: Cambridge University Press, 1988.

MÉTAYER, F. **Fixed points of functors**. 2003. Manuscrito não publicado, disponível em https://www.irif.fr/~metayer/PDF/fix.pdf. Acesso em: 28 de maio de 2021.

SCOTT, D. **Outline of a mathematical theory of computation**. Oxford, 1970. Relatório técnico n. PRG02, Oxford University Computing Laboratory, Programming Research Group.