FUNDAMENTOS DE PROGRAMAÇÃO FUNCIONAL

UMA INTRODUÇÃO AO CÁLCULO LAMBDA

Carlos Caleiro

Secção de Ciência da Computação
Departamento de Matemática
Instituto Superior Técnico

Dezembro de 2002

Estas notas destinam-se ao acompanhamento da disciplina de Programação Funcional correntemente leccionada ao 2o ano das Licenciaturas em Matemática Aplicada e Computação e em Ciências Informáticas do Instituto Superior Técnico, cujos objectivos se descrevem de seguida.

Objectivos

Gerais
Dominar a teoria matemática da programação funcional, nomeadamente os mecanismos básicos da definição recursiva de funções, da teoria da computabilidade e dos sistemas de tipos. Conhecer os princípios fundamentais da semântica denotacional. Desenvolver programas no paradigma funcional.
Operacionais
Aprender os fundamentos do cálculo-lambda, incluindo: estratégias de redução e principais teoremas de confluência e decidibilidade, definibilidade lambda das funções parciais recursivas, lógica combinatória, sistemas de tipos e modelos do cálculo-lambda. Entender a semântica rigorosa de uma linguagem funcional. Desenvolver a capacidade de programar no paradigma funcional.

Programa

Fundamentos
Perspectiva histórica. Cálculo lambda: sintaxe, substituição de variáveis, esquema de redução, conversão alfa, redução beta, redução eta, teorias lambda, coerência, divergência, formas normais, confluência, primeiro teorema de Church-Rosser, estratégias de redução (normal, aplicativa e preguiçosa), reduções normais, segundo teorema de Church-Rosser, combinadores $\boldsymbol{K}$ e $\boldsymbol{S}$. Lógica combinatória: completude combinatória, determinadores de pontos fixos, combinador $\boldsymbol{Y}$, equações combinatórias e emergência do paradigma da programação funcional, representação do cálculo proposicional e da aritmética básica, postulado de Church, definibilidade e computabilidade, teorema de Kleene, teorema de Gödel da lógica combinatória, indecidibilidade do problema da paragem. Sistemas de tipos: tipos ímplicitos, tipos funcionais, polimorfismo, inferência de tipos, resultados de redução do sujeito, normalização e decidibilidade. Semântica denotacional: perspectiva computacional de Scott-Strachey, álgebras combinatórias, teorema de Curry-Shönfinkel, álgebras lambda e modelos lambda, argumento de Cantor, ordens parciais completas e topologia de Scott, continuidade, domínios reflexivos, elementos finitos, domínios algébricos, teorema da correcção, extensionalidade, modelo concreto baseado na álgebra dos termos, modelos de Engeler, Plotkin e dos limites projectivos.
Programação
Panorama das linguagens de programação funcional. Programação na linguagem Scheme, com ênfase na definição recursiva de funções, técnicas de programação em grande escala, operações sobre listas, funções de ordem superior, avaliação preguiçosa e computação simbólica. Pequenos projectos (eg, jogo da vida, interpretador de cálculo lambda, demonstrador automático de teoremas proposicionais, calculadora simbólica).

O material contido nestas notas foi, na sua maioria, adaptado da grande referência moderna ao Cálculo Lambda, o livro *The Lambda Calculus - Its Syntax and Semantics* de H.P. Barendregt, volume 203 da série "Studies in Logic and the Foundations of Mathematics", publicado em 1984 pela North-Holland, e ainda do capítulo *Lambda Calculi with Types*, do mesmo autor, do volume II da colectânea "Handbook of Logic in Computer Science", editado por S. Abramsky, D. Gabbay e T. Maibaum e publicado pela Oxford University Press em 1992.

Índice

Introdução **1**

1 Sintaxe **3**
Exercícios 6

2 Redução **8**
Exercícios 13

3 Teorias lambda **16**
Exercícios 18

4 Lógica combinatória **20**
Exercícios 23

5 Definibilidade lambda **25**
Exercícios 28

6 Computabilidade **30**
Exercícios 34

7 Sistemas de tipos **36**
Exercícios 43

8 Semântica denotacional **45**
Exercícios 49

9 Teoria de domínios **50**
Exercícios 55

A Indução por Regras **57**
Exercícios 58

Introdução

Antes de mais, a programação funcional emerge essencialmente da adopção de uma forma diferente de observar os fenómenos computacionais, que se baseia na intuição de que um programa pode ser visto como uma função (no sentido matemático do termo), que recebe um argumento e calcula o correspondente resultado. Mas como surge este paradigma de programação?

A história remonta a meados do século XVIII quando Gottfried Leibniz, filósofo e matemático alemão que simultaneamente com Isaac Newton lançou também as bases do cálculo diferencial e integral, formulou o que viria a ficar conhecido por *Entscheidungsproblem* ou *Problema da decisão*. O problema consistia na possibilidade de conceber, primeiro, uma linguagem universal na qual se pudesse expressar qualquer problema e, por fim, um método de decisão para os problemas expressos nessa linguagem. Esta questão resistiu aos mais variados esforços durante o século XIX, até que já no século XX, mais precisamente em 1936, Alonzo Church e Alan Turing propuseram as primeiras definições rigorosas da noção de "método de decisão". As propostas de Church e Turing, juntamente com os trabalhos do matemático austríaco Kurt Gödel acabaram por demonstrar a impraticabilidade da ideia original de Leibniz e ditaram o fracasso do *Entscheidungsproblem*. No entanto, estes trabalhos viriam a revelar-se muitíssimo frutuosos. De facto, o que Church e Turing propuseram, cada um por si e de formas bem distintas mas equivalentes, foram modelos matemáticos do que deveria ser um computador, alguns anos antes de haver computadores. Outra proposta importante também equivalente, surgiu em 1954 por intermédio do matemático russo Aleksander Markov. Pelo facto de estas e várias outras propostas, de cariz substancialmente diferente, se terem revelado equivalentes e de nunca ter sido encontrado nenhum exemplo em contrário, é comummente aceite, ainda hoje, que encerram em si a "verdadeira" noção de computabilidade. Este princípio é usualmente conhecido como *Postulado de Church-Markov-Turing*.

A proposta de Turing, a *máquina de Turing*, baseava-se na ideia de uma memória dividida em células onde se podem guardar valores que vão sendo sucessivamente alterados por execução de acções. A proposta de Church era uma teoria simbólica, muito mais abstracta, que dá pelo nome de *cálculo lambda*. Por razões tecnológicas, foi a proposta de Turing que acabou por ser concretizada após a II Guerra Mundial por John Von Neumann, dando origem à arquitectura dos computadores que todos conhecemos. Por esse facto, também, uma parte substancial das linguagens de programação tiram partido das ideias de Turing: são as linguagens *imperativas* ou *procedimentais* como ALGOL, FORTRAN, PASCAL, BASIC ou C, em que um programa prescreve a sequência de passos segundo a qual os valores guardados nas células de memória vão sendo alterados. Apesar disso, também as ideias subjacentes a abordagens alternativas à de Turing acabaram por se reflectir em linguagens de programação: é o caso da linguagem SNOBOL, amplamente utilizada na ex-URSS e baseada na noção de *algoritmo de Markov*; ou das linguagens *funcionais* como LISP,

Scheme, ML, Miranda, Clean ou Haskell, baseadas no cálculo lambda de Church. As linguagens funcionais acabaram por ter um impacto fortíssimo devido ao seu modelo matemático rigoroso, aliado à intuição muito clara de que um programa corresponde a uma função. Pela sua clareza, em oposição às linguagens imperativas, as linguagens funcionais fazem parte da família de linguagens ditas *declarativas*, que inclui também linguagens de programação em lógica como PROLOG.

Para além disso, as linguagens funcionais são implementações quase imediatas do cálculo lambda, onde muitos problemas prementes relacionados com os fenómenos computacionais podem ser estudados com detalhe e a um nível de abstracção adequado[1]. É o caso de mecanismos de modularização como as funções de ordem superior (que facilitam a concepção, depuração, legibilidade e verificação de programas), das definições recursivas, dos sistemas de tipos, e mesmo dos fenómenos emergentes da computação concorrente e distribuída. Do ponto de vista da lógica, o cálculo lambda tem tido também um impacto decisivo no desenvolvimento da teoria da prova, o que se reflecte nos mais modernos sistemas computacionais de demonstração. Por estas razões, e ainda mais algumas, o cálculo lambda é muitas vezes utilizado como "linguagem mínima", onde outras linguagens de programação ganham significado usando técnicas da semântica denotacional.

[1]São bem conhecidos os problemas levantados no paradigma imperativo, por exemplo, pela intratabilidade dos efeitos colaterais ou da chamada "programação em esparguete".

Capítulo 1

Sintaxe

Os objectos de estudo do cálculo lambda (puro) são *termos* construídos sobre um alfabeto constituído por:

- $\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2, \ldots$ - símbolos de constante;
- $v_0, v_1, v_2, \ldots$ - símbolos de variável;
- λ - símbolo de abstracção lambda;
- $(,)$ - parênteses esquerdo e direito.

Usamos *Const* para denotar o conjunto $\{\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2, \ldots\}$ dos símbolos de constante e *Var* para denotar o conjunto $\{v_n \mid n \in \mathbb{N}\}$ dos símbolos de variável. Usualmente, no entanto, trabalhamos com alfabetos sem quaisquer símbolos de constante, ou seja, em que $Const = \emptyset$. De facto, como veremos mais adiante, as constantes de interesse (computáveis) poderão ser codificadas no cálculo sem símbolos de constante.

Definição 1.1 *Termos lambda.*
O conjunto Λ dos termos lambda é definido indutivamente[1] pelas seguintes regras:

$$\text{constantes: } \frac{}{\mathtt{c}_n} \qquad \text{variáveis: } \frac{}{v_n}$$

$$\text{abstracção: } \frac{M}{(\lambda v_n M)} \qquad \text{aplicação: } \frac{M \quad N}{(MN)}.$$

É usual adoptar as seguintes convenções de notação:

- $M, N, L, \ldots$ denotam termos lambda arbitrários;
- $\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots$ denotam símbolos de constante arbitrários;
- $x, y, z, \ldots$ denotam símbolos de variável arbitrários;
- o símbolo $\equiv$ denota a relação de igualdade sintáctica entre termos;
- $\lambda x_1 x_2 \ldots x_n.M \equiv (\lambda x_1 x_2 \ldots x_n.M) \equiv (\lambda x_1 (\lambda x_2 (\ldots (\lambda x_n M) \ldots)))$;
- $MN_1N_2 \ldots N_n \equiv (MN_1N_2 \ldots N_n) \equiv (\ldots ((MN_1)N_2) \ldots N_n)$.

[1] A definição indutiva faz-se sobre o conjunto U de todas as sequências de símbolos do alfabeto.

A intuição por detrás de um termo lambda é simples: uma variável x ou uma constante $\mathtt{a}$ representam um *valor*; uma abstracção $\lambda x.M$ representa uma "função" cujo parâmetro formal é x e cuja regra simbólica de cálculo é dada pelo termo M; uma aplicação MN representa a aplicação da "função" M ao argumento N.

No cálculo lambda não há qualquer distinção entre valores "passivos" (como uma variável) e valores "activos" (como uma "função"). Qualquer valor pode ser aplicado a outro. Outra consequência deste facto é não sermos capazes de distinguir (por definição) entre $\lambda xy.M$ (a "função" binária que recebe x e y e devolve M) e $\lambda x.(\lambda y.M)$ (a "função" unária que recebe x e devolve a "função" unária que devolve M após receber y). Esta promoção das "funções" unárias como primitivas e a correspondente representação de "funções" de aridade superior deve-se a uma ideia original de Curry.

Aprofundemos um pouco as propriedades sintácticas da abstracção, tendo em conta que o *âmbito* da abstracção λx em $\lambda x.M$ é precisamente o termo $\lambda x.M$.

Definição 1.2 *Ocorrências livres e mudas.*
Diz-se que x ocorre livre num termo M se e só se existe uma ocorrência de x em M fora do âmbito de qualquer abstracção λx. Qualquer ocorrência de x em M no âmbito de uma abstracção λx é dita uma ocorrência muda.

Claramente, nada impede que uma variável possa ocorrer simultaneamente livre e muda no mesmo termo.

Proposição 1.3 *Variáveis livres.*
O conjunto $vl(M) \subseteq Var$ das variáveis que ocorrem livres num termo M é definido indutivamente por:

- $vl(\mathtt{a}) = \emptyset$;
- $vl(x) = \{x\}$;
- $vl(\lambda x.M) = vl(M) \setminus \{x\}$;
- $vl(M_1 M_2) = vl(M_1) \cup vl(M_2)$.

Proposição 1.4 *Variáveis mudas.*
O conjunto $vm(M) \subseteq Var$ das variáveis que ocorrem mudas num termo M é definido indutivamente por:

- $vm(\mathtt{a}) = \emptyset$;
- $vm(x) = \emptyset$;
- $vm(\lambda x.M) = \{x\} \cup vm(M)$;
- $vm(M_1 M_2) = vm(M_1) \cup vm(M_2)$.

Os termos lambda sem variáveis livres são particularmente interessantes.

Definição 1.5 *Combinador.*
Um combinador é um termo $M \in \Lambda$ tal que $vl(M) = \emptyset$. Denota-se o conjunto de todos os combinadores por Λ^0.

Os combinadores dizem-se também termos *fechados*. Chamamos *fecho de* M a qualquer termo $\lambda x_1 \ldots x_n.M$ onde $\{x_1, \ldots, x_n\} = vl(M)$. Claramente, um fecho de M é sempre um termo fechado, podendo no entanto haver vários fechos de M correspondentes a diferentes permutações de $vl(M)$. Em particular, se M é já um termo fechado então o seu único fecho é o próprio M.

Há vários combinadores importantes cujas definições usamos frequentemente:

- $\boldsymbol{I} \equiv \lambda x.x$;
- $\boldsymbol{1} \equiv \lambda xy.xy$;
- $\boldsymbol{K} \equiv \lambda xy.x$;
- $\boldsymbol{S} \equiv \lambda xyz.xz(yz)$;
- $\boldsymbol{B} \equiv \lambda xyz.x(yz)$;
- $\boldsymbol{C} \equiv \lambda xyz.xzy$;
- $\boldsymbol{Y} \equiv \lambda f.((\lambda x.f(xx))(\lambda x.f(xx)))$;
- $\boldsymbol{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$;

Atente-se agora na operação sintáctica de *substituição de variáveis livres*.

Definição 1.6 *Substituição de variáveis livres.*
Denota-se por $M[x := N]$ o termo lambda que resulta de *substituir* em M todas as ocorrências livres de x por N.

De facto, o mecanismo fundamental do cálculo lambda, dito *redução-*β, estabelece o modo como deve ser avaliada a aplicação de um termo $\lambda x.M$ a um argumento N. Tendo em conta a intuição, o resultado de $(\lambda x.M)N$ deverá corresponder precisamente ao termo $M[x := N]$.

No entanto, há que tomar alguns cuidados. Tome-se o termo $\boldsymbol{K} \equiv \lambda xy.x$. Será de esperar que dados argumentos M e N, o resultado da aplicação $\boldsymbol{K}MN$ seja M. No entanto, seguindo estritamente a ideia acima, ter-se-á como resultado de $\boldsymbol{K}MN \equiv (\lambda xy.x)MN$ precisamente $x[x := M][y := N] \equiv M[y := N]$. Como é óbvio isto levanta um problema sempre que $y \in vl(M)$. Como resolvê-lo? O problema surge, antes de mais, porque $\boldsymbol{K} \equiv \lambda xy.x \equiv \lambda x.\lambda y.x$ e x ocorre livre em $\lambda y.x$ no âmbito da abstracção λy. Daí que, ao substituirmos x por M, qualquer possível ocorrência livre de y em M passa a ser muda, pois é capturada pela abstracção λy. Será possível assumir sempre que, numa situação como esta, $y \notin vl(M)$?

Suponha-se que ao avaliar $(\lambda xy.x)MN$ notamos que $y \in vl(M)$. Será razoável avaliar, em vez disso, o termo $(\lambda xz.x)MN$ onde escolhemos $z \notin vl(M)$? Se o fizermos o resultado corresponderá ao que pretendemos, já que $x[x := M][z := N] \equiv M[z := N] \equiv M$. É claro que os termos $\boldsymbol{K} \equiv \lambda xy.x$ e $\lambda xz.x$ são sintacticamente distintos. No entanto é perfeitamente claro que, enquanto regras de definição de uma "função", eles são equivalentes: ambos representam a "função" que retorna o primeiro de dois argumentos recebidos. A solução passa então por desenvolver o cálculo lambda considerando uma noção mais forte que a simples igualdade sintáctica e que seja capaz de identificar termos como $\lambda xy.x$ e $\lambda xz.x$. A essa noção damos o nome de *congruência-*α.

Definição 1.7 *Subtermos.*
O conjunto $sub(M)$ dos subtermos de um termo M é definido indutivamente por:

- $sub(\mathtt{a}) = \{\mathtt{a}\}$;
- $sub(x) = \{x\}$;
- $sub(\lambda x.N) = \{\lambda x.N\} \cup sub(N)$;
- $sub(M_1 M_2) = \{M_1 M_2\} \cup sub(M_1) \cup sub(M_2)$.

Definição 1.8 *Congruência-α.*
Uma mudança de variáveis mudas num termo M consiste na substituição em M de um subtermo da forma $\lambda x.N$ por $\lambda y.(N[x := y])$, para alguma variável y que não ocorra em N.

Dois termos M e N dizem-se congruentes-α, o que denotamos por $M \equiv_\alpha N$, se N resulta de M por uma sequência de mudanças de variáveis mudas.

Módulo congruência-α, cada termo lambda deve ser visto então como representante de toda uma classe de termos. Para resolver o problema definitivamente adoptamos daqui em diante a seguinte convenção, permitida pela congruência-α.

Convenção 1.9 *Utilização de variáveis.*
Em qualquer contexto que contenha os termos $M_1, \ldots, M_n$, assume-se que toda a variável que ocorra muda nalgum M_k não ocorre livre em nenhum dos termos $M_1, \ldots, M_n$.

A robustez desta convenção segue dos resultados seguintes.

Lema 1.10 *Troca de substituições.*
Se $x \notin vl(L)$ então:

$$M[x := N][y := L] \equiv_\alpha M[y := L][x := N[y := L]].$$

Proposição 1.11 *Congruência-α versus substituição.*
Se $M_1 \equiv_\alpha M_2$ e $N_1 \equiv_\alpha N_2$ então, para qualquer variável x, tem-se:

$$M_1[x := N_1] \equiv_\alpha M_2[x := N_2].$$

Exercícios

A. Mostre que todo o termo lambda tem exactamente o mesmo número de parênteses esquerdos e direitos.

B. Mostre que:

   (i) uma variável x ocorre em M se e só se $x \in vl(M) \cup vm(M)$;

   (ii) o conjunto de variáveis que ocorre num termo é necessariamente finito.

C. Demonstre que o termo $M[x := N]$ pode ser definido indutivamente por:

   - $\mathtt{a}[x := N] \equiv \mathtt{a}$;
   - $x[x := N] \equiv N$;
   - $y[x := N] \equiv y$;
   - $(\lambda x.M_1)[x := N] \equiv (\lambda x.M_1)$;

- $(\lambda y.M_1)[x := N] \equiv \lambda y.(M_1[x := N])$;
- $(M_1 M_2)[x := N] \equiv (M_1[x := N])(M_2[x := N])$.

D. Recorde a definição de congruência-α e mostre que $\equiv_\alpha$ é:

   (i) uma relação de equivalência no conjunto Λ;

   (ii) compatível com a abstracção e aplicação, i.e., se $M_1 \equiv_\alpha N_1$ e $M_2 \equiv_\alpha N_2$ então $\lambda x.M_1 \equiv_\alpha \lambda x.N_1$ e $M_1 M_2 \equiv_\alpha N_1 N_2$.

E. Um termo M diz-se *α-simples* se $vl(M) \cap vm(M) = \emptyset$.

   (i) Mostre que qualquer que seja o termo lambda M existe um termo α-simples N tal que $M \equiv_\alpha N$;

   (ii) Conclua que de facto a congruência-α permite cumprir a convenção de utilização de variáveis.

F. Mostre que o termo $M[x := N]$, construído de acordo com a definição de substituição de variáveis livres em termos usando a convenção de utilização de variáveis, é congruente-α com o termo $M[x/N]$ que resulta da substituição *sensível ao contexto* das ocorrências livres de x em M por N, e que se define indutivamente por:

- $\mathtt{a}[x/N] \equiv \mathtt{a}$;
- $x[x/N] \equiv N$;
- $y[x/N] \equiv y$;
- $(\lambda x.M_1)[x/N] \equiv \lambda x.M_1$;
- $(\lambda y.M_1)[x/N] \equiv \lambda y.(M_1[x/N])$ se $x \notin vl(M_1)$ ou $y \notin vl(N)$;
- $(\lambda y.M_1)[x/N] \equiv \lambda z.(M_1[y/z][x/N])$ se $x \in vl(M_1)$, $y \in vl(N)$ e z não ocorre em M_1 ou N;
- $(M_1 M_2)[x/N] \equiv (M_1[x/N])(M_2[x/N])$.

Capítulo 2

Redução

Tendo estabelecido a sintaxe do cálculo lambda, podemos agora estudar com mais detalhe os seus mecanismos. Como já foi referido, o mecanismo fundamental do cálculo lambda é a *redução-β*, que prescreve que o termo $(\lambda x.M)N$ se *reduz* ao termo $M[x := N]$ (não esquecer que adoptamos sempre a convenção de utilização de variáveis). De qualquer forma, este mecanismo é apenas um caso particular da seguinte noção geral.

Definição 2.1 *Esquema e relação de redução.*
Um esquema de redução é uma relação binária em Λ. Uma relação de redução é um esquema de redução fechado para as seguintes regras de *compatibilidade*:

$$\frac{\langle M, N\rangle}{\langle ML, NL\rangle} \qquad \frac{\langle M, N\rangle}{\langle LM, LN\rangle} \qquad \frac{\langle M, N\rangle}{\langle \lambda x.M, \lambda x.N\rangle}.$$

Definição 2.2 *Esquema de redução-β.*
O esquema de redução-β é definido por:

$$\beta = \{\langle (\lambda x.M)N, M[x := N]\rangle \mid M, N \in \Lambda, x \in Var\}.$$

Outro mecanismo interessante no contexto do cálculo lambda, nomeadamente em conjugação com a redução-β, é o esquema de redução-η. Esta noção está intimamente ligada à noção de *extensionalidade*, que aprofundaremos mais adiante, e identifica termos que tenham o mesmo resultado quando aplicados a qualquer possível argumento.

Definição 2.3 *Esquemas de redução-η e redução-$\beta\eta$.*
O esquema de redução-η é definido por:

$$\eta = \{\langle (\lambda x.Mx), M\rangle \mid M \in \Lambda, x \notin vl(M)\}.$$

O esquema de redução-$\beta\eta$ é definido por:

$$\beta\eta = \beta \cup \eta.$$

Facilmente, qualquer esquema de redução induz uma relação de redução.

Definição 2.4 *Redução num passo, redução e conversão.*
Seja R um esquema de redução. Então define-se:

- a relação $\rightarrow_R$ de *redução-R num passo* é definida indutivamente a partir de R pelas regras de compatibilidade;

- a relação $\twoheadrightarrow_R$ de *redução-R* é o fecho reflexivo e transitivo de $\rightarrow_R$;
- a relação $=_R$ de *conversão-R* é a relação de equivalência gerada por $\twoheadrightarrow_R$.

Claramente, as relações $\rightarrow_R$, $\twoheadrightarrow_R$ e $=_R$ são todas fechadas para as regras de compatibilidade.

Considere-se, como exemplo, o esquema de redução-β e recorde-se a definição dos combinadores $\boldsymbol{M} \equiv \lambda x.xx$ e $\boldsymbol{I} \equiv \lambda x.x$. Obviamente tem-se

$$\boldsymbol{M}\boldsymbol{I}y \rightarrow_\beta \boldsymbol{I}\boldsymbol{I}y \rightarrow_\beta \boldsymbol{I}y \rightarrow_\beta y$$

e portanto também

$$\boldsymbol{M}\boldsymbol{I}y \twoheadrightarrow_\beta y \ , \ \ \boldsymbol{M}\boldsymbol{I}y =_\beta y \ \text{ e } \ y =_\beta \boldsymbol{M}\boldsymbol{I}y.$$

Definição 2.5 *Redex e forma normal.*
Seja R um esquema de redução. Diz-se que:

- M é um R-redex se $\langle M, N\rangle \in R$ para algum termo N;
- N está na forma normal-R se nenhum dos seus subtermos é um R-redex;
- N é uma forma normal-R de M se $M =_R N$ e N está na forma normal-R.

Interpretando a redução de um termo como um processo computacional, podemos reconhecer cada passo de redução como um passo computacional e uma forma normal como a terminação do respectivo processo. No exemplo anterior, é claro que y está na forma normal-β e é portanto uma forma normal-β de $\boldsymbol{M}\boldsymbol{I}y$. No entanto, enquanto por exemplo $\lambda x.yx$ está também na forma normal-β o termo pode ainda ser reduzido-η,

$$(\lambda x.yx) \rightarrow_\eta y$$

para a sua forma normal-$\beta\eta$, que é y.

Proposição 2.6 *Redução versus forma normal.*
Seja R um esquema de redução e M um termo na forma normal-R. Então:

- *não existe N tal que $M \rightarrow_R N$;*
- *se $M \twoheadrightarrow_R N$ então $M \equiv N$.*

Note-se no entanto que, em geral, o recíproco do segundo ponto deste resultado não é válido. Nomeadamente, tomando o esquema de redução-β, é possível encontrar um termo M que verifica a condição do segundo ponto mas não está na forma normal-β. Basta considerar $\boldsymbol{\Omega} \equiv \boldsymbol{M}\boldsymbol{M}$, onde $\boldsymbol{M} \equiv \lambda x.xx$ e notar que o único caminho de redução-β possível a partir de $\boldsymbol{\Omega}$ é

$$(\lambda x.xx)(\lambda x.xx) \rightarrow_\beta (\lambda x.xx)(\lambda x.xx) \rightarrow_\beta (\lambda x.xx)(\lambda x.xx) \rightarrow_\beta \ldots$$

Isto significa, em particular, que o termo não possui nenhuma forma normal-β. Computacionalmente, podemos interpretar este facto como correspondendo à não terminação do seu processo de avaliação.

Considere-se um outro exemplo, ainda no contexto do esquema de redução-β. O termo $(\lambda y.\boldsymbol{M}y)z$ tem claramente uma forma normal zz de acordo com a redução

$$(\lambda y.\boldsymbol{M}y)z \rightarrow_\beta \boldsymbol{M}z \rightarrow_\beta zz.$$

Independentemente de já sabermos que existem termos sem qualquer forma normal, será que o facto de sabermos que zz é uma forma normal de $(\lambda y.\boldsymbol{M}y)z$ exclui a existência de outras formas normais? Será possível que uma redução do termo com outra escolha de redexes nos leve a uma forma normal distinta? Neste caso, por exemplo, podemos iniciar outra redução escolhendo primeiro o redex mais interior $\boldsymbol{M}y$. No entanto, tem-se

$$(\lambda y.\boldsymbol{M}y)z \to_\beta (\lambda y.yy)z \to_\beta zz$$

mostrando que neste caso é irrelevante a escolha. Mas será que isto se passa sempre?

Definição 2.7 *Confluência.*
Uma relação de redução $\twoheadrightarrow$ diz-se confluente quando, para quaisquer M, N_1, N_2, se $M \twoheadrightarrow N_1$ e $M \twoheadrightarrow N_2$ então existe L tal que $N_1 \twoheadrightarrow L$ e $N_2 \twoheadrightarrow L$.

Definição 2.8 *Propriedade de Church-Rosser.*
Diz-se que um esquema de redução R tem a propriedade de Church-Rosser se a relação de redução-R é confluente.

Proposição 2.9 *Propriedade de Church-Rosser e forma normal.*
Seja R um esquema de redução com a propriedade de Church-Rosser. Então:

- *se $M =_R N$ então existe L tal que $M \twoheadrightarrow_R L$ e $N \twoheadrightarrow_R L$;*
- *se N é uma formal normal-R de M então $M \twoheadrightarrow_R N$;*
- *todo o termo tem, no máximo, uma forma normal-R.*

De facto, a redução-β (e também η e $\beta\eta$) é insensível à escolha entre redexes e as formas normais, quando existem, são únicas.

Proposição 2.10 *Teorema de Church-Rosser.*
Os esquemas de redução β, η e $\beta\eta$ têm a propriedade de Church-Rosser.

Logo, qualquer escolha de redex numa redução-β (ou η ou $\beta\eta$) passo a passo do termo levará, se terminar, ao mesmo resultado. No entanto, põe-se a questão de poder haver determinadas escolhas de redex que possam levar a um caminho de redução infinito, mesmo quando o termo tem uma forma normal. Por exemplo, considerando de novo a redução-β, o termo $(\lambda xy.y)\boldsymbol{\Omega}z$ pode reduzir-se facilmente

$$(\lambda xy.y)\boldsymbol{\Omega}z \to_\beta (\lambda y.y)z \to_\beta z$$

tendo z como forma normal. No entanto, escolhendo sempre primeiro o redex correspondente ao problemático subtermo $\boldsymbol{\Omega}$, teremos um caminho de redução infinito

$$(\lambda xy.y)\boldsymbol{\Omega}z \to_\beta (\lambda xy.y)\boldsymbol{\Omega}z \to_\beta (\lambda xy.y)\boldsymbol{\Omega}z \to_\beta \ldots$$

Como escolher então, em cada caso, o redex adequado?

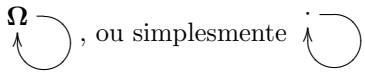
Esta questão pode ser analisada mais facilmente recorrendo ao grafo (dirigido) de redução de um termo. Recorde-se que um grafo dirigido é simplesmente um quádruplo $G = \langle \mathcal{N}, \mathcal{A}, o, d\rangle$ onde $\mathcal{N}$ é um conjunto (de *nós*), $\mathcal{A}$ é um conjunto (de *arcos*) e $o, s : \mathcal{A} \to \mathcal{N}$ são funções (que associam a cada arco o seu nó *origem* e o seu nó *destino*, respectivamente). Note-se que um grafo dirigido pode ser visualizado representando os nós como pontos no plano e cada arco como uma seta da sua origem para o seu destino.

Definição 2.11 *Grafo de redução.*
Seja R um esquema de redução. O grafo $G_R(M)$ de redução-R de um termo M tem como nós todos os termos N tais que $M \twoheadrightarrow_R N$, e um arco do nó N_1 para o nó N_2 por cada possível redução $N_1 \rightarrow_R N_2$.

Considere-se, por exemplo, o termo $\boldsymbol{I}(\boldsymbol{I}x)$, cujo grafo de redução-$\beta$ é

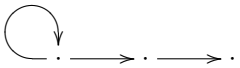
$$\boldsymbol{I}(\boldsymbol{I}x) \Longrightarrow \boldsymbol{I}x \longrightarrow x \text{ , ou simplesmente } \cdot \Longrightarrow \cdot \longrightarrow \cdot$$

Outro exemplo, considerando o termo $\boldsymbol{\Omega}$, é



Definição 2.12 *Normalização forte.*
Seja R um esquema de redução. Um termo M diz-se fortemente normalizável se não existe nenhum caminho infinito no grafo $G_R(M)$.

Já sabemos então que o resultado de uma computação a partir de um termo fortemente normalizável (como $\boldsymbol{I}(\boldsymbol{I}x)$) não é sensível à escolha de redexes, podendo no entanto variar o número de passos necessários. Para termos cujo grafo de redução tenha apenas caminhos infinitos (como $\boldsymbol{\Omega}$), obviamente, a questão nem se põe. Mas para termos cujo grafo de redução contém caminhos finitos e infinitos, como escolher os redexes em cada passo por forma a garantir a terminação da computação? É o caso do termo $(\lambda xy.y)\boldsymbol{\Omega}z$ cujo grafo de redução é



Definição 2.13 *Estratégia de redução.*
Seja R um esquema de redução. Uma estratégia de redução-R (num passo) é uma função $\iota : \Lambda \rightarrow \Lambda$ tal que:

- se M está na forma normal-R então $\iota(M) \equiv M$ senão $M \rightarrow_R \iota(M)$.

Assim, seguindo uma dada estratégia de redução-R, ter-se-á para cada termo M um caminho de redução determinado

$$M \rightarrow_R \iota(M) \rightarrow_R \iota^2(M) \rightarrow_R \iota^3(M) \rightarrow_R \ldots$$

Esta sequência de redução pode ser infinita (caso em que não é encontrada forma normal) ou terminar, obviamente, com uma forma normal. Claramente, é desejável que o caminho de redução seja finito para todo o termo com forma normal-R.

Para a redução-β, há três estratégias de redução particularmente interessantes.

Definição 2.14 *Estratégias de redução-β.*
No contexto da redução-β definem-se as seguintes estratégias de redução, para um termo arbitrário M que não esteja na forma normal-β:

- *normal*: $\iota(M)$ corresponde à redução em M do β-redex cuja abstracção surja mais à esquerda;
- *aplicativa*: $\iota(M)$ corresponde à redução em M do β-redex $(\lambda x.N)L$, em que L não contenha nenhum outro β-redex, cuja abstracção surja mais à esquerda;
- *preguiçosa*: $\iota(M)$ corresponde à redução em M do β-redex $(\lambda x.N)L$, em que L não contenha nenhum outro β-redex ou $x \notin vl(N)$, cuja abstracção surja mais à esquerda.

Atente-se no seguinte exemplo por forma a ilustrar as diferenças entre as três estratégias. Considere-se o termo $(\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z)$. Usando cada uma das três estratégias teríamos, respectivamente,

$$(\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z) \to_\beta (\lambda y.yy)(\boldsymbol{I}z) \to_\beta \boldsymbol{I}z(\boldsymbol{I}z) \to_\beta z(\boldsymbol{I}z) \to_\beta zz,$$

$$(\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z) \to_\beta (\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z) \to_\beta (\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z) \to_\beta \ldots$$

e

$$(\lambda xy.yy)\mathbf{\Omega}(\boldsymbol{I}z) \to_\beta (\lambda y.yy)(\boldsymbol{I}z) \to_\beta (\lambda y.yy)z \to_\beta zz.$$

No primeiro e terceiro casos obtemos a forma normal zz. No entanto, o número de passos de redução é maior na estratégia normal. Isto passa-se, em geral, pois a estratégia preguiçosa optimiza a estratégia normal com a ideia essencial da estratégia aplicativa: não replicar um termo antes de o avaliar. A questão é que a estratégia aplicativa avalia sempre o argumento, mesmo quando o seu valor não é necessário enquanto a estratégia preguiçosa o faz apenas se o valor é imprescindível. Como consequência, embora seja em geral mais eficiente que a estratégia normal, a estratégia aplicativa pode levar a caminhos divergentes que não permitem encontrar a forma normal do termo, caso exista.

Na terminologia das linguagens de programação estas três estratégias estão associadas, respectivamente, aos paradigmas da *chamada por nome*, *por valor* e *por necessidade*. No primeiro caso os argumentos só são avaliados quando necessários (até lá usa-se o seu *nome* e não o seu *resultado*, tal como na estratégia preguiçosa), enquanto no segundo caso os argumentos são sempre avaliados antes. No terceiro caso, mais eficiente mas mais difícil de implementar, os argumentos são avaliados antes apenas se necessários, para evitar a sua replicação. Embora a estratégia aplicativa (por vezes também denominada de estratégia "eager", por oposição a "lazy"[1]) seja a mais intuitiva (razão porque é usada muitas vezes) e mais eficiente que a normal, evitando a replicação de avaliações do mesmo termo, o exemplo anterior mostra que nem sempre é a mais desejável. De facto é possível demonstrar que a estratégia normal, embora eventualmente menos eficiente, conduz a uma forma normal sempre que ela existe. Este resultado é devido a Curry e Feys.

Proposição 2.15 *Normalização-β e chamada por nome.*
Se M tem forma normal-β então o caminho de redução normal-β a partir de M é finito.

Para o caso da redução-$\beta\eta$ será suficiente usar esta mesma estratégia enquanto possível, resolvendo os η-redexes no final.

Definição 2.16 *Estratégia de redução normal-$\beta\eta$.*
No contexto da redução-$\beta\eta$ define-se a seguinte estratégia de redução normal, para um termo arbitrário M que não esteja na forma normal-$\beta\eta$:

- $\iota(M)$ corresponde, sempre que M não está na forma normal-β, à redução em M do β-redex cuja abstracção surja mais à esquerda, e ao η-redex cuja abstracção surja mais à esquerda no caso contrário.

O seguinte resultado deve-se também a Curry e Feys.

[1] Em inglês, "lazy" significa preguiçoso, ocioso, enquanto "eager" significa ávido, sôfrego.

Proposição 2.17 *Normalização-$\beta\eta$ e chamada por nome.*
Se M tem forma normal-$\beta\eta$ então o caminho de redução normal-$\beta\eta$ a partir de M é finito.

No cálculo lambda com constantes, é usual haver esquemas de redução específicos associados a cada símbolo de constante, a que se chama *regras-δ*. Apesar de, como vamos ver, as constantes correspondentes a regras computáveis poderem ser directamente codificadas no cálculo sem símbolos de constante, há razões de eficiência que motivam muitas vezes a sua inclusão. No entanto, a introdução de regras-δ pode levantar problemas e deve ser extremamente cuidadosa. Atente-se no seguinte exemplo para um alfabeto com símbolos de constante $\mathtt{cons}, \mathtt{car}, \mathtt{cdr}$. O esquema de redução-$\delta$ associado, onde para quaisquer $M, N \in \Lambda$ se tem

$$\mathtt{cons}\,(\mathtt{car}\,M)\,(\mathtt{cdr}\,M) \to_\delta M$$
$$\mathtt{car}\,(\mathtt{cons}\,M\,N) \to_\delta M$$
$$\mathtt{cdr}\,(\mathtt{cons}\,M\,N) \to_\delta N$$

destina-se a lidar com a construção e destrução de pares ordenados. É relativamente simples verificar que o esquema de redução-$\beta\delta$ correspondente não tem a propriedade de Church-Rosser.

No entanto, tudo corre bem se tivermos alguns cuidados. Considere-se que são dados $X_1, \ldots, X_k \subseteq \Lambda^0$ (combinadores), $f : X_1 \times \cdots \times X_k \to \Lambda$ uma função e que o alfabeto contém uma constante $\mathtt{f}$.

Proposição 2.18 *Propriedade de Church-Rosser e regras-δ.*
Se todos os combinadores em $X_1 \cup \cdots \cup X_k$ estão na forma normal-$\beta\delta$ então o esquema de redução-$\beta\delta$ obtido a partir das regras-δ

$$\mathtt{f}\,M_1 \ldots M_k \to_\delta f(M_1, \ldots, M_k), \text{ com } M_1 \in X_1, \ldots, M_k \in X_k$$

tem a propriedade de Church-Rosser.

Um exemplo paradigmático é o da igualdade entre formas normais-β fechadas. Considerando símbolos de constante $\mathtt{ig}, \mathtt{verd}$ e $\mathtt{falso}$ o esquema de redução-$\beta\delta$ obtido a partir das regras-δ

$$\mathtt{ig}\,MN \to_\delta \mathtt{verd}, \text{ se } M \equiv N$$
$$\mathtt{ig}\,MN \to_\delta \mathtt{falso}, \text{ se } M \not\equiv N$$

onde M, N são combinadores em que as constantes $\mathtt{ig}, \mathtt{verd}$ e $\mathtt{falso}$ não ocorrem, tem a propriedade de Church-Rosser.

Exercícios

A. Seja R um esquema de redução. Mostre que:

   (i) $\twoheadrightarrow_R$ é a menor relação de redução reflexiva e transitiva que contém R;

   (ii) $=_R$ é uma relação de equivalência fechada para as regras de compatibilidade.

B. Seja R um esquema de redução. Mostre que $M \to_R N$ se e só se existem $M_1 \in \mathit{sub}(M)$ e $N_1 \in \mathit{sub}(N)$ tais que $\langle M_1, N_1 \rangle \in R$ e N se obtém de M substituindo precisamente uma ocorrência de M_1 por N_1.

C. Mostre que se R é um esquema de redução e $N_1 \twoheadrightarrow_R N_2$ então $M[x := N_1] \twoheadrightarrow_R M[x := N_2]$.

D. Mostre que se $M \twoheadrightarrow_{\beta\eta} N$ então $vl(N) \subseteq vl(M)$.

E. Um esquema de redução R diz-se *substitutivo* se é fechado para a regra:

$$\frac{\langle M, N\rangle}{\langle M[x := L], N[x := L]\rangle}.$$

Mostre que se R é substitutivo então também o são $\rightarrow_R$, $\twoheadrightarrow_R$ e $=_R$.

Verifique se os esquemas de redução β, η e $\beta\eta$ são ou não substitutivos.

F. Seja $\twoheadrightarrow$ uma relação de redução. Mostre que se $\twoheadrightarrow$ é confluente então o seu fecho reflexivo e transitivo $\twoheadrightarrow^*$ também é confluente (*Lema do diamante*).

G. Mostre que o fecho reflexivo da relação $\rightarrow_\beta$ não é confluente.

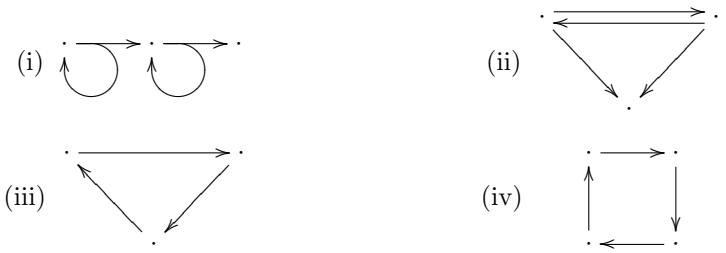
H. Diz-se que duas relações de redução $\twoheadrightarrow_1$ e $\twoheadrightarrow_2$ *comutam* quando, para quaisquer termos M, N_1, N_2, se $M \twoheadrightarrow_1 N_1$ e $M \twoheadrightarrow_2 N_2$ então existe L tal que $N_1 \twoheadrightarrow_2 L$ e $N_2 \twoheadrightarrow_1 L$. Mostre que se $\twoheadrightarrow_1$ e $\twoheadrightarrow_2$ comutam e são ambas confluentes então a relação $(\twoheadrightarrow_1 \cup \twoheadrightarrow_2)^*$ também é confluente (*Lema de Hindley-Rosen*).

I. Mostre que $\twoheadrightarrow_\beta$ e $\twoheadrightarrow_\eta$ comutam (no sentido da alínea anterior).

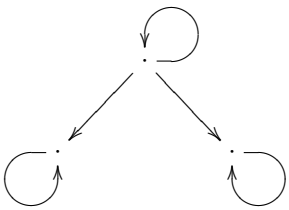
J. Defina os grafos de redução-β dos seguintes termos:

(i) $\omega_3\omega_3$, com $\omega_3 \equiv \lambda x.xxx$;
(ii) $(\lambda x.\boldsymbol{I})(\omega_3\omega_3)$;
(iii) $(\lambda x.\boldsymbol{I})\boldsymbol{\Omega}$;
(iv) TTT, com $T \equiv (\lambda xy.yxx)$.

K. Encontre termos cujos grafos de redução-β sejam:



L. Mostre que não existe nenhum termo cujo grafo de redução-β seja da forma:



M. Diz-se que um esquema de redução R tem a *propriedade fraca de Church-Rosser* quando, para quaisquer termos M, N_1, N_2, se $M \rightarrow_R N_1$ e $M \rightarrow_R N_2$ então existe L tal que $N_1 \twoheadrightarrow_R L$ e $N_2 \twoheadrightarrow_R L$. Diz-se ainda que R é *fortemente normalizante* se não existe nenhuma sequência infinita de reduções-R num passo $M_0 \rightarrow_R M_1 \rightarrow_R M_2 \rightarrow_R \ldots$. Mostre que:

(i) a propriedade de Church-Rosser implica a propriedade fraca de Church-Rosser;

(ii) a propriedade fraca de Church-Rosser não implica a propriedade de Church-Rosser;

(iii) se R é fortemente normalizável e tem a propriedade fraca de Church-Rosser então também tem a propriedade de Church-Rosser (*Lema de Newmann*).

N. Tomando $Const = \{\mathtt{d}, \mathtt{e}\}$ e o esquema de redução-δ definido por

$$\mathtt{d}\, MM \to_\delta \mathtt{e}$$

para qualquer $M \in \Lambda$, verifique que:

(i) o termo $C \equiv AA(\lambda yx.\, \mathtt{d}\, x(yx))$, onde $A \equiv \lambda xy.y(xxy)$, é tal que para qualquer termo M se tem $CM \twoheadrightarrow_\beta \mathtt{d}\, M(CM)$;

(ii) o termo $X \equiv AAC$ é tal que $X \to_\beta CX$;

(iii) $X \twoheadrightarrow_{\beta\delta} \mathtt{e}$ e $X \twoheadrightarrow_{\beta\delta} C\mathtt{e}$;

(iv) $\mathtt{e}$ está na forma normal-$\beta\delta$ e $C\mathtt{e} \not\twoheadrightarrow_{\beta\delta} \mathtt{e}$, concluindo-se que falha a propriedade de Church-Rosser.

O. Tomando $Const = \{\mathtt{cons}, \mathtt{car}, \mathtt{cdr}\}$ e o esquema de redução-δ anteriormente descrito, verifique que:

(i) o termo $S \equiv \lambda xy.\mathtt{cons}(\mathtt{car}(\lambda z.zx))(\mathtt{cdr}(\lambda z.zy))(\boldsymbol{K}\,\mathtt{cons})$ é tal que para qualquer termo M se tem $SMM \twoheadrightarrow_{\beta\delta} \mathtt{cons}$;

(ii) a técnica do exercício anterior pode ser usada para mostrar que também neste caso a propriedade de Church-Rosser falha.

P. Defina sistemas de regras-δ nas condições da Proposição 2.18 que caracterizem as seguintes operações:

(i) $\mathtt{e}, \mathtt{ou}, \mathtt{neg}$, sobre os Booleanos, com $Const = \{\mathtt{e}, \mathtt{ou}, \mathtt{neg}, \mathtt{verd}, \mathtt{falso}\}$;

(ii) $\mathtt{mais}, \mathtt{vezes}$, sobre os naturais, com $Const = \{\mathtt{mais}, \mathtt{vezes}, \mathtt{0}, \mathtt{1}, \mathtt{2}, \dots\}$.

Capítulo 3

Teorias lambda

Tendo estabelecido a forma como podemos operar termos lambda por redução, podemos agora estudar os princípios lógicos do cálculo lambda.

Definição 3.1 *Teoria-λ.*
Uma teoria-λ é um conjunto de equações entre termos lambda, da forma $M = N$[1], fechado para as seguintes regras:

$$\begin{array}{rl}
\text{conversão-}\beta\text{:} & \dfrac{}{(\lambda x.M)N = M[x:=N]} \\[2ex]
\text{reflexividade:} & \dfrac{}{M = M} \\[2ex]
\text{simetria:} & \dfrac{M = N}{N = M} \\[2ex]
\text{transitividade:} & \dfrac{M = N \quad N = L}{M = L} \\[2ex]
\text{aplicação-esq:} & \dfrac{M = N}{ML = NL} \\[2ex]
\text{aplicação-dir:} & \dfrac{M = N}{LM = LN} \\[2ex]
\text{regra-}\xi\text{:} & \dfrac{M = N}{\lambda x.M = \lambda x.N}\,.
\end{array}$$

Dado um conjunto de equações $\mathcal{T}$, denotamos por $\mathcal{T}^\lambda$ a menor teoria-λ que contém $\mathcal{T}$. Se $M = N \in \mathcal{T}^\lambda$ denotamos esse facto por $\mathcal{T} \vdash_\lambda M = N$. A menor teoria-$\lambda$ é, obviamente, $\emptyset^\lambda$. Se $M = N \in \emptyset^\lambda$ denotamos esse facto simplesmente por $\vdash_\lambda M = N$. Será que esta teoria é coerente, por forma a ter algum significado?

Sendo a lógica associada ao cálculo lambda relativamente pobre (não existe por exemplo negação) necessitamos de usar uma versão adequada da noção de *coerência*.

Definição 3.2 *Coerência.*
Uma teoria-λ $\mathcal{T}$ diz-se coerente se existem termos M, N tais que $M = N \notin \mathcal{T}$. Um conjunto $\mathcal{T}$ de equações é λ-coerente se $\mathcal{T}^\lambda$ é uma teoria coerente.

De facto, é muito fácil construir teorias-λ incoerentes. Tome-se por exemplo o conjunto formado apenas pela equação $\boldsymbol{S} = \boldsymbol{K}$ onde, recorde-se, $\boldsymbol{S} \equiv \lambda xyz.xz(yz)$ e $\boldsymbol{K} \equiv \lambda xy.x$. Recorde-se ainda que $\boldsymbol{I} \equiv \lambda x.x$. A teoria $\{\boldsymbol{S} = \boldsymbol{K}\}^\lambda$ é incoerente pois, qualquer que seja M, $\boldsymbol{S} = \boldsymbol{K} \vdash_\lambda M = \boldsymbol{I}$. A partir de $\boldsymbol{S} = \boldsymbol{K}$, por aplicação à esquerda, também se tem $\boldsymbol{S}\boldsymbol{I}(\boldsymbol{K}M)\boldsymbol{I} = \boldsymbol{K}\boldsymbol{I}(\boldsymbol{K}M)\boldsymbol{I}$. Agora, usando sucessivamente a redução-β de cada um dos lados da equação chegamos facilmente a $M = \boldsymbol{I}$.

[1] Obviamente, há que não confundir $=$ (a relação de *conversão*) com a igualdade sintáctica $\equiv$.

Ainda assim, felizmente, é possível construir teorias-λ coerentes.

Proposição 3.3 *Dedução-λ versus conversão-β.*
Para quaisquer termos M, N, tem-se $\vdash_\lambda M = N$ se e só se $M =_\beta N$.

A coerência da dedução-λ surge então como corolário da propriedade de Church-Rosser do esquema de redução-β.

Proposição 3.4 *Coerência da dedução-λ.*
A teoria $\emptyset^\lambda$ é coerente.

Por forma a incorporar também a noção de extensionalidade, é possível proceder à extensão da noção de teoria-λ de duas formas equivalentes.

Definição 3.5 *Extensionalidades.*
A regra *ext* corresponde a

$$\frac{Mx = Nx}{M = N},\ x \notin vl(MN).$$

A regra de conversão-η corresponde a

$$\frac{}{\lambda x.Mx = M},\ x \notin vl(M).$$

Definição 3.6 *Teoria-$(\lambda + ext)$ e teoria-$\lambda\eta$.*
Uma teoria-$(\lambda + ext)$ é uma teoria-λ fechada para a regra *ext*. Analogamente, uma teoria-λ fechada para a regra de conversão-η diz-se uma teoria-$\lambda\eta$.

Tal como antes, dado um conjunto de equações $\mathcal{T}$, denotamos por $\mathcal{T}^{(\lambda+ext)}$ e $\mathcal{T}^{\lambda\eta}$, respectivamente, a menor teoria-$(\lambda + ext)$ e a menor teoria-$\lambda\eta$ que contêm $\mathcal{T}$. Se $M = N \in \mathcal{T}^{(\lambda+ext)}$ denotamos esse facto por $\mathcal{T} \vdash_{(\lambda+ext)} M = N$. Se $M = N \in \mathcal{T}^{\lambda\eta}$ denotamos esse facto por $\mathcal{T} \vdash_{\lambda\eta} M = N$.

Proposição 3.7 *Teorias extensionais.*
Para qualquer conjunto $\mathcal{T}$ de equações, tem-se $\mathcal{T}^{(\lambda+ext)} = \mathcal{T}^{\lambda\eta}$.

A dedução-$\lambda\eta$ também é coerente, de novo como corolário da propriedade de Church-Rosser, mas agora do esquema de redução-$\beta\eta$.

Proposição 3.8 *Dedução-$\lambda\eta$ versus conversão-$\beta\eta$.*
Para quaisquer termos M, N, tem-se $\vdash_{\lambda\eta} M = N$ se e só se $M =_{\beta\eta} N$.

Proposição 3.9 *Coerência da dedução-$\lambda\eta$.*
A teoria $\emptyset^{\lambda\eta}$ é coerente.

Uma vez que os termos lambda representam "funções" e a sua redução o processo computacional associado à sua avaliação, é usual assumir que os termos sem qualquer forma normal são de certa forma desprovidos de significado. Nesse pressuposto, então, seria admissível identificá-los e trabalhar na teoria-λ que colapsa entre si todos os termos sem forma normal-β, ou na teoria-$\lambda\eta$ que colapsa entre si todos os termos sem forma normal-$\beta\eta$.

Proposição 3.10 *Formas normais versus coerência.*
O conjunto $\{M = N \mid M, N \in \Lambda$ não têm forma normal-$\beta\}$ é λ-incoerente.
O conjunto $\{M = N \mid M, N \in \Lambda$ não têm forma normal-$\beta\eta\}$ é $\lambda\eta$-incoerente.

De facto, este procedimento acaba por identificar demasiados termos. O que há a fazer é identificar todos os termos que possam de facto servir de suporte à noção de "função" indefinida.

Definição 3.11 *Solubilidade.*
Um combinador F diz-se λ-solúvel, respectivamente $\lambda\eta$-solúvel, se existem termos $N_1, \ldots, N_n$ tais que $\vdash_\lambda FN_1 \ldots N_n = \boldsymbol{I}$, respectivamente $\vdash_{\lambda\eta} FN_1 \ldots N_n = \boldsymbol{I}$.
Um termo M diz-se λ-solúvel ou $\lambda\eta$-solúvel se algum fecho $\lambda x_1 \ldots x_n.M$ de M o é.

Note-se que a noção de solubilidade está intimamente ligada à existência de solução para equações envolvendo termos lambda. Em particular, se F é solúvel com $N_1 \ldots N_n$, a equação $F\vec{X} = Y$ tem como solução $\vec{X} \equiv N_1 \ldots N_n Y$.

Obviamente, qualquer termo λ-solúvel é também $\lambda\eta$-solúvel. Facilmente, por exemplo, o combinador $\boldsymbol{S}$ é λ-solúvel pois

$$\boldsymbol{SIII} \to_\beta \boldsymbol{II}(\boldsymbol{II}) \to_\beta \boldsymbol{I}(\boldsymbol{II}) \to_\beta \boldsymbol{II} \to_\beta \boldsymbol{I}.$$

Também o termo $x\boldsymbol{I\Omega}$ é λ-solúvel pois

$$(\lambda x.x\boldsymbol{I\Omega})\boldsymbol{K} \to_\beta \boldsymbol{KI\Omega} \to_\beta (\lambda y.\boldsymbol{I})\boldsymbol{\Omega} \to_\beta \boldsymbol{I}.$$

No entanto, por exemplo, o combinador $\boldsymbol{\Omega}$ é claramente $\lambda\eta$-insolúvel. São os termos insolúveis que devem colapsar-se entre si, como representantes da classe das "funções" indefinidas. A relação entre a insolubilidade e a inexistência de formas normais, no entanto, não é directa.

Definição 3.12 *Forma normal prefixa.*
Um termo M diz-se na forma normal prefixa se $M \equiv \lambda x_1 \ldots x_n.xN$.
Diz-se que um termo N tem uma forma normal prefixa-λ, respectivamente forma normal prefixa-$\lambda\eta$, se existe M na forma normal prefixa tal que $\vdash_\lambda N = M$, respectivamente $\vdash_{\lambda\eta} N = M$.

Por exemplo, o termo $\lambda x.x\boldsymbol{I\Omega}$ não tem forma normal-β mas está na forma normal prefixa-λ. De facto, é fácil verificar que a noção de forma normal é estritamente mais forte que a noção de forma normal prefixa.

Proposição 3.13 *Solubilidade versus forma normal prefixa.*
Um termo M é λ-solúvel se e só se M tem uma forma normal prefixa-λ.
Um termo M é $\lambda\eta$-solúvel se e só se M tem uma forma normal prefixa-$\lambda\eta$.

Veriquemos por fim que as teorias correspondentes à identificação dos termos insolúveis são coerentes.

Proposição 3.14 *Insolubilidade versus coerência.*
O conjunto $\{M = N \mid M, N \in \Lambda$ λ-insolúveis$\}$ é λ-coerente.
O conjunto $\{M = N \mid M, N \in \Lambda$ $\lambda\eta$-insolúveis$\}$ é $\lambda\eta$-coerente.

Exercícios

A. Sejam $\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2$ conjuntos de equações. Mostre que:

   (i) $\mathcal{T} \subseteq \mathcal{T}^\lambda$;

   (ii) se $\mathcal{T}_1 \subseteq \mathcal{T}_2$ então $\mathcal{T}_1^\lambda \subseteq \mathcal{T}_2^\lambda$;

   (iii) $(\mathcal{T}^\lambda)^\lambda \subseteq \mathcal{T}^\lambda$.

B. Seja $\mathcal{T}$ um conjunto de equações e $M, N \in \Lambda^0$. Mostre que $\mathcal{T} \vdash_{\lambda\eta} M = N$ se e só se $\mathcal{T} \cup \{\boldsymbol{I} = \boldsymbol{1}\} \vdash_\lambda M = N$.

C. Mostre que para qualquer conjunto de equações $\mathcal{T}$ se tem $\mathcal{T}^{\lambda} \subseteq \mathcal{T}^{\lambda\eta}$.

D. Mostre que são λ-incoerentes os seguintes conjuntos:

(i) $\{\boldsymbol{I} = M\}$ onde M é tal que $\vdash_{\lambda} \boldsymbol{K}M = M$;

(ii) $\{\boldsymbol{I} = WW\}$ onde $W \equiv \lambda x.xxx$;

(iii) $\{xx = xy\}$;

(iv) $\{(xy)z = x(yz)\}$.

E. Mostre que:

(i) se M está na forma normal-β então M está na forma normal prefixa-λ;

(ii) se M está na forma normal-$\beta\eta$ então M está na forma normal prefixa-$\lambda\eta$.

F. Mostre que o combinador $\boldsymbol{Y}$ é λ-solúvel.

G. Mostre, para as teorias λ e $\lambda\eta$, que:

(i) M é solúvel se e só se $\lambda x.M$ é solúvel;

(ii) se M é insolúvel então (MN) e $M[x := N]$ também são insolúveis.

H. Mostre, para os esquemas de redução β e $\beta\eta$, que:

(i) M tem uma forma normal prefixa se e só se $\lambda x.M$ tem uma forma normal prefixa;

(ii) se M não tem uma forma normal prefixa então (MN) e $M[x := N]$ também não têm forma normal prefixa.

I. Considere os seguintes esquemas de redução:

$$\Omega_{\lambda} = \{\langle M, \boldsymbol{\Omega}\rangle \mid \boldsymbol{\Omega} \not\equiv M \in \Lambda,\ \text{M } \lambda\text{-insolúvel}\},$$

$$\Omega_{\lambda\eta} = \{\langle M, \boldsymbol{\Omega}\rangle \mid \boldsymbol{\Omega} \not\equiv M \in \Lambda,\ \text{M } \lambda\eta\text{-insolúvel}\},$$

$$\beta\Omega = \beta \cup \Omega_{\lambda}$$

e

$$\beta\eta\Omega = \beta\eta \cup \Omega_{\lambda\eta}.$$

Mostre que:

(i) todos são substitutivos;

(ii) todos têm a propriedade de Church-Rosser;

(iii) $\{M = N \mid M, N \in \Lambda\ \lambda\text{-insolúveis}\} \vdash_{\lambda} P = Q$ se e só se $P =_{\beta\Omega} Q$;

(iv) $\{M = N \mid M, N \in \Lambda\ \lambda\eta\text{-insolúveis}\} \vdash_{\lambda\eta} P = Q$ se e só se $P =_{\beta\eta\Omega} Q$.

Capítulo 4

Lógica combinatória

A lógica combinatória corresponde a um fragmento do cálculo lambda, mas que provaremos ser suficientemente poderoso para o representar. A ideia, desenvolvida por Schönfinkel e Curry, consiste em tomar como primitivos (leia-se constantes) os combinadores $\boldsymbol{K} \equiv \lambda xy.x$ e $\boldsymbol{S} \equiv \lambda xyz.xz(yz)$ e pura e simplesmente eliminar a abstracção-λ. Para evitar qualquer confusão sintáctica entre os combinadores acima e as constantes correspondentes, designamo-las por $\mathtt{k}$ e $\mathtt{s}$. Para simplificar, assumimos que o alfabeto subjacente não contém outros símbolos de constante, ou seja, $\mathit{Const} = \{\mathtt{k}, \mathtt{s}\}$.

Definição 4.1 *Termos combinatórios.*
O conjunto LC dos termos combinatórios é definido indutivamente pelas seguintes regras:

$$\text{constantes:} \quad \frac{}{\mathtt{k}} \qquad \frac{}{\mathtt{s}}$$

$$\text{variáveis:} \quad \frac{}{v_n}$$

$$\text{aplicação:} \quad \frac{P \quad Q}{(PQ)}.$$

É usual adoptar $P, Q, R, \ldots$ para denotar termos combinatórios arbitrários. Note-se que se $P \in LC$ e x ocorre em P então $x \in vl(P)$. Isto porque, não havendo abstracções, $vm(P) = \emptyset$. Em particular, sendo tomados como símbolos constantes primitivos, tem-se que $vm(\mathtt{k}) = vm(\mathtt{s}) = \emptyset$.

Definição 4.2 *Teoria-LC.*
Uma teoria-LC é um conjunto de equações entre termos combinatórios fechado para as seguintes regras:

$$\text{constantes:} \quad \frac{}{\mathtt{k}PQ=P} \qquad \frac{}{\mathtt{s}PQR=PR(QR)}$$

$$\text{reflexividade:} \quad \frac{}{P=P}$$

$$\text{simetria:} \quad \frac{P=Q}{Q=P}$$

$$\text{transitividade:} \quad \frac{P=Q \quad Q=R}{P=R}$$

$$\text{aplicação-esq:} \quad \frac{P=Q}{PR=QR}$$

$$\text{aplicação-dir:} \quad \frac{P=Q}{RP=RQ}.$$

Uma teoria-$LC\eta$ é uma teoria-LC fechada para a regra *ext*, onde $x \notin vl(PQ)$:

$$ext: \quad \frac{Px=Qx}{P=Q}.$$

Dado um conjunto de equações combinatórias $\mathcal{T}$, denotamos por $\mathcal{T}^{LC}$ a menor teoria-LC que contém $\mathcal{T}$ e por $\mathcal{T}^{LC\eta}$ a menor teoria-$LC\eta$ que contém $\mathcal{T}$. Tal como antes usamos $\vdash_{LC}$ e $\vdash_{LC\eta}$ para denotar as noções de dedução associadas.

De facto, é fácil de perceber a analogia com as teorias lambda, se pura e simplesmente substituirmos cada ocorrência das constantes $\mathtt{k}$ e $\mathtt{s}$ num termo combinatório pelo combinador respectivo.

Definição 4.3 *Representação de termos combinatórios.*
A função $\__\lambda : LC \to \Lambda$ é definida indutivamente por:

- $\mathtt{k}_\lambda \equiv \boldsymbol{K} \equiv \lambda xy.x$;
- $\mathtt{s}_\lambda \equiv \boldsymbol{S} \equiv \lambda xyz.xz(yz)$;
- $x_\lambda \equiv x$;
- $(MN)_\lambda \equiv M_\lambda N_\lambda$.

Dado um conjunto de equações combinatórias $\mathcal{T}$ denotamos o conjunto de equações entre termos lambda $\{P_\lambda = Q_\lambda \mid P = Q \in \mathcal{T}\}$ por $\mathcal{T}_\lambda$.

Proposição 4.4 *Correcção combinatória.*
Para qualquer conjunto $\mathcal{T}$ de equações combinatórias e quaisquer termos combinatórios P, Q tem-se:

- *se $\mathcal{T} \vdash_{LC} P = Q$ então $\mathcal{T}_\lambda \vdash_\lambda P_\lambda = Q_\lambda$.*
- *se $\mathcal{T} \vdash_{LC\eta} P = Q$ então $\mathcal{T}_\lambda \vdash_{\lambda\eta} P_\lambda = Q_\lambda$.*

No entanto, em geral, o recíproco não é verdadeiro. Note-se por exemplo que $(\mathtt{sk})_\lambda \equiv \boldsymbol{SK}$, $(\mathtt{k(skk)})_\lambda \equiv \boldsymbol{K}(\boldsymbol{SKK})$, como $\boldsymbol{SK} \twoheadrightarrow_\beta \lambda xy.y$ e $\boldsymbol{K}(\boldsymbol{SKK}) \twoheadrightarrow_\beta \lambda xy.y$ tem-se $\vdash_\lambda \boldsymbol{SK} = \boldsymbol{K}(\boldsymbol{SKK})$ e no entanto $\not\vdash_{LC} \mathtt{sk} = \mathtt{k(skk)}$, apesar de se ter de facto $\vdash_{LC\eta} \mathtt{sk} = \mathtt{k(skk)}$. Isto acontece porque as teorias-LC (ao contrário das teorias-$LC\eta$) não são fechadas para nenhum análogo da regra-ξ: $\frac{M=N}{\lambda x.M=\lambda x.N}$. No entanto, é possível definir uma noção correspondente à abstracção-λ mesmo no fragmento combinatório.

Definição 4.5 *Abstracção combinatória.*
Para qualquer termo combinatório P define-se o termo combinatório $\lambda^* x.P$ indutivamente por:

- $\lambda^* x.x \equiv \mathtt{skk}$;
- $\lambda^* x.Q \equiv \mathtt{k}Q$ se $x \notin vl(Q)$;
- $\lambda^* x.QR \equiv \mathtt{s}(\lambda^* x.Q)(\lambda^* x.R)$ se $x \in vl(QR)$.

Tem-se, por exemplo, que

$$\lambda^* x.\lambda^* y.yx \equiv \lambda^* x.\mathtt{s(skk)(k}x\mathtt{)} \equiv \mathtt{s(k(s(skk)))(s(kk)(skk))}.$$

Note-se que, para qualquer termo combinatório P, se tem $\vdash_{LC} \mathtt{skk}P = P$. Isto explica, à luz do usual combinador $\boldsymbol{I} \equiv \lambda x.x$, a razão porque se define $\lambda^* x.x \equiv \mathtt{skk}$. Por este facto tomamos como boa a representação de $\boldsymbol{I}$ por $\mathtt{skk}$.

Proposição 4.6 *Aplicação versus abstracção combinatória.*
Para quaisquer termos combinatórios P, Q tem-se $\vdash_{LC} (\lambda^* x.P)Q = P[x := Q]$.

Entretanto, note-se que a noção combinatória correspondente à regra-ξ pode agora ser enunciada:

$$\text{regra-}\xi\text{ combinatória:} \quad \frac{P=Q}{\lambda^* x.P=\lambda^* x.Q}.$$

No entanto, as teorias-LC não são de facto fechadas para esta regra. Por exemplo, tem-se que $\vdash_{LC} \mathtt{skk}x = x$, as correspondentes abstracções combinatórias são $\lambda^* x.\mathtt{skk}x \equiv \mathtt{s}(\mathtt{k}(\mathtt{skk}))(\mathtt{skk})$ e $\lambda^* x.x \equiv \mathtt{skk}$, mas $\nvdash_{LC} \mathtt{s}(\mathtt{k}(\mathtt{skk}))(\mathtt{skk}) = \mathtt{skk}$. No entanto, claramente, $\vdash_{LC\eta} \mathtt{s}(\mathtt{k}(\mathtt{skk}))(\mathtt{skk}) = \mathtt{skk}$.

Para ultrapassar este problema, então, há que considerar o cálculo combinatório extensional. A solução assenta na possibilidade, talvez surpreendente de, tal como $\boldsymbol{I}$ corresponde ao termo combinatório $\mathtt{skk}$, podermos associar um correspondente combinatório a todo o termo lambda.

Definição 4.7 *Representação combinatória de termos lambda.*
A função $\_{}_{LC} : \Lambda \to LC$ é definida indutivamente por:

- $x_{LC} \equiv x$;
- $(\lambda x.M)_{LC} \equiv \lambda^* x.M_{LC}$;
- $(MN)_{LC} \equiv M_{LC}N_{LC}$.

Como seria de esperar tem-se $\boldsymbol{I}_{LC} \equiv \mathtt{skk}$. Tem-se também, por exemplo, $(\lambda xy.yx)_{LC} = \mathtt{s}(\mathtt{k}(\mathtt{s}(\mathtt{skk})))(\mathtt{s}(\mathtt{kk})(\mathtt{skk}))$.

Proposição 4.8 *Lemas da completude combinatória.*
Para qualquer termo lambda M tem-se $\vdash_{\lambda} (M_{LC})_{\lambda} = M$.
Para qualquer termo combinatório P tem-se $\vdash_{LC\eta} (P_{\lambda})_{LC} = P$.

Pelas razões já apontadas, naturalmente, a segunda condição deste lema não se verifica para o cálculo combinatório não extensional. Em particular, tem-se que $\nvdash_{LC} (\mathtt{k}_{\lambda})_{LC} = \mathtt{k}$ e $\nvdash_{LC} (\mathtt{s}_{\lambda})_{LC} = \mathtt{s}$.

Dado um conjunto de equações entre termos lambda $\mathcal{T}$ denotamos o conjunto de equações combinatórias $\{M_{LC} = N_{LC} \mid M = N \in \mathcal{T}\}$ por $\mathcal{T}_{LC}$.

Proposição 4.9 *Completude combinatória.*
Para qualquer conjunto $\mathcal{T}$ de equações entre termos lambda e quaisquer termos lambda M, N tem-se:

- $\mathcal{T} \vdash_{\lambda\eta} M = N$ *se e só se* $\mathcal{T}_{LC} \vdash_{LC\eta} M_{LC} = N_{LC}$.

Analogamente, para qualquer conjunto $\mathcal{T}$ de equações combinatórias e quaisquer termos combinatórios P, Q tem-se:

- $\mathcal{T} \vdash_{LC\eta} P = Q$ *se e só se* $\mathcal{T}_{\lambda} \vdash_{\lambda\eta} P_{\lambda} = Q_{\lambda}$.

Este resultado mostra que o fragmento aplicativo do cálculo lambda correspondente aos combinadores $\boldsymbol{K}$ e $\boldsymbol{S}$ é suficientemente expressivo. Os combinadores $\boldsymbol{K}$ e $\boldsymbol{S}$ formam assim aquilo a que se chama uma *base combinatória*. É interessante verificar que o combinador $\boldsymbol{X} \equiv \lambda x.x\boldsymbol{KSK}$ constitui também, por si próprio, uma base combinatória.

Exercícios

A. Mostre que ficam bem definidas as funções:

   (i) $\_\_\lambda : LC \to \Lambda$ da Definição 4.3;

   (ii) $\lambda^* x.\_ : LC \to LC$ da Definição 4.5;

   (iii) $\_\_LC : \Lambda \to LC$ da Definição 4.7.

B. Mostre que as teorias-$LC\eta$ são fechadas para a regra-ξ combinatória.

C. Considere as abreviaturas $\mathtt{i} \equiv \mathtt{skk}$, $\mathtt{b} \equiv \mathtt{s(ks)k}$ e $\mathtt{c} \equiv \mathtt{s(bs(bks))(kk)}$. Verifique que se tem:

   (i) $\vdash_{LC\eta} \mathtt{k} = \mathtt{s(kk)i}$;

   (ii) $\vdash_{LC\eta} \mathtt{s} = \mathtt{c(cbs)i}$;

   (iii) $\vdash_{LC\eta} \boldsymbol{\Omega}_{LC} = \mathtt{sii(sii)}$;

   (iv) $\vdash_{LC\eta} \boldsymbol{B}_{LC} = \mathtt{b}$;

   (v) $\vdash_{LC\eta} \boldsymbol{C}_{LC} = \mathtt{c}$;

   (vi) $\vdash_{LC\eta} \boldsymbol{Y}_{LC} = \mathtt{s(cb(sii))(cb(sii))}$.

D. Considerando ainda as abreviaturas $\mathtt{i}, \mathtt{b}, \mathtt{c}$ da alínea anterior, mostre que as seguintes regras para termos combinatórios são correctas em $LC\eta$, para quaisquer termos combinatórios P, Q :

   - $\mathtt{s(k}P\mathtt{)i} = P$;
   - $\mathtt{s(k}P\mathtt{)(k}Q\mathtt{)} = \mathtt{k}(PQ)$;
   - $\mathtt{s(k}P\mathtt{)}Q = \mathtt{b}PQ$;
   - $\mathtt{s}P\mathtt{(k}Q\mathtt{)} = \mathtt{c}PQ$;

E. A tradução (via $\_\_LC$) de termos lambda para termos combinatórios sobre $\mathtt{k}, \mathtt{s}$ é, regra geral, pouco eficiente. Considere, para os efeitos deste exercício o conjunto LC dos termos combinatórios definidos tal como na Definição 4.1, mas directamente a partir das constantes combinatórias $\mathtt{k}, \mathtt{s}, \mathtt{i}, \mathtt{b}, \mathtt{c}$.

   (i) Defina $\_\_\lambda$ tal como na Definição 4.3, mas agora substituindo também cada uma das constantes $\mathtt{i}, \mathtt{b}, \mathtt{c}$ pelo respectivo combinador $\boldsymbol{I}, \boldsymbol{B}, \boldsymbol{C}$, e verifique que fica bem definida;

   (ii) Defina teoria-LC juntando à Definição 4.2 as três regras seguintes

   $$\overline{\mathtt{i}P = P} \qquad \overline{\mathtt{b}PQR = P(QR)} \qquad \overline{\mathtt{c}PQR = PRQ}$$

   e demonstre que ainda se verifica o resultado de correcção combinatória da Proposição 4.4.

   (iii) Redefina a construção de $\lambda^* x.P$ da seguinte forma:

   - $\lambda^* x.x \equiv \mathtt{i}$;
   - $\lambda^* x.Q \equiv \mathtt{k}Q$ se $x \notin vl(Q)$;
   - $\lambda^* x.QR \equiv \mathtt{b}Q(\lambda^* x.R)$ se $x \notin vl(Q)$ e $x \in vl(R)$.
   - $\lambda^* x.QR \equiv \mathtt{c}(\lambda^* x.Q)R$ se $x \in vl(Q)$ e $x \notin vl(R)$.
   - $\lambda^* x.QR \equiv \mathtt{s}(\lambda^* x.Q)(\lambda^* x.R)$ se $x \in vl(Q)$ e $x \in vl(R)$.

   Mostre que a construção fica bem definida e demonstre que ainda se verifica a Proposição 4.6.

(iv) De acordo com a nova definição calcule $\boldsymbol{Y}_{LC}$.

(v) Demonstre que ainda se verificam os lemas e o resultado de completude combinatória das Proposições 4.8 e 4.9.

F. Mostre que para quaisquer $P \in LC$ e $M \in \Lambda$ se tem:

(i) $vl(P_\lambda) = vl(P)$;

(ii) $vl(\lambda^* x.P) = vl(P) \setminus \{x\}$;

(iii) $vl(M_{LC}) = vl(M)$.

G. Verifique que $\vdash_{LC\eta} (\mathtt{s}_\lambda)_{LC} = \mathtt{s}$.

H. Mostre que, para $M, N \in \Lambda$, se tem $(M[x := N])_{LC} \equiv M_{LC}[x := N_{LC}]$.

I. Mostre que:

(i) qualquer que seja o conjunto $\mathcal{T}$ de equações entre termos lambda e $M, N \in \Lambda$ se tem $\mathcal{T} \vdash_\lambda M = N$ se e só se $(\mathcal{T}_{LC})_\lambda \vdash_\lambda M = N$, e também $\mathcal{T} \vdash_{\lambda\eta} M = N$ se e só se $(\mathcal{T}_{LC})_\lambda \vdash_{\lambda\eta} M = N$;

(ii) qualquer que seja o conjunto $\mathcal{T}$ de equações entre termos combinatórios e $P, Q \in LC$ se tem $\mathcal{T} \vdash_{LC\eta} P = Q$ se e só se $(\mathcal{T}_\lambda)_{LC} \vdash_{LC\eta} P = Q$.

J. Prove que de facto se verifica ainda o recíproco do resultado de correcção combinatória no caso extensional, nomeadamente mostrando que qualquer que seja o conjunto $\mathcal{T}$ de equações entre termos lambda e $M, N \in \Lambda$,

$$\text{se } \mathcal{T} \vdash_{\lambda\eta} M = N \text{ então } \mathcal{T}_{LC} \vdash_{LC\eta} M_{LC} = N_{LC}.$$

K. Mostre que de facto o combinador $\boldsymbol{X} \equiv \lambda x.x\boldsymbol{KSK}$ constitui por si só uma base combinatória, verificando que $\vdash_\lambda \boldsymbol{K} = \boldsymbol{XXX}$ e $\vdash_\lambda \boldsymbol{S} = \boldsymbol{X}(\boldsymbol{XX})$.

Capítulo 5

Definibilidade lambda

A noção de ponto fixo, como veremos, é a característica fundamental da programação recursiva e, por consequência, da noção de computabilidade. Ao longo deste capítulo adoptamos, por ser suficiente, a noção de teoria-λ.

Definição 5.1 *Ponto fixo.*
Um ponto fixo de um termo lambda F é um termo M tal que $\vdash_\lambda FM = M$.

Por exemplo, obviamente, qualquer termo é ponto fixo de $\boldsymbol{I} \equiv \lambda x.x$. Mas já não será nada óbvio encontrar um ponto fixo de $\boldsymbol{1} \equiv \lambda xy.xy$. No entanto, talvez algo surpreendentemente, todo o termo lambda tem pontos fixos.

Proposição 5.2 *Teorema do ponto fixo.*
Todo o termo lambda tem um ponto fixo.

Em particular, o termo NN com $N \equiv \lambda x.\boldsymbol{1}(xx)$ é ponto fixo de $\boldsymbol{1}$. De facto, tem-se $\vdash_\lambda \boldsymbol{1}(NN) = NN$ pois

$$NN \equiv (\lambda x.\boldsymbol{1}(xx))N \to_\beta \boldsymbol{1}(NN).$$

O modo uniforme como pode obter-se um ponto fixo de um termo dado sugere a seguinte definição.

Definição 5.3 *Determinador de pontos fixos.*
Um termo lambda M diz-se um determinador de pontos fixos se, para qualquer termo F, o termo MF é ponto fixo de F.

Proposição 5.4 *Combinador $\boldsymbol{Y}$.*
O combinador $\boldsymbol{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ é um determinador de pontos fixos.

A existência de determinadores de pontos fixos como $\boldsymbol{Y}$ (existem muitos outros) permite resolver facilmente certo tipo de equações combinatórias. Considere, por exemplo, a igualdade $F = \lambda xy.FyxF$. Será possível encontrar um termo F que a verifique? Facilmente, o que procuramos é um termo F que seja ponto fixo de $M \equiv \lambda fxy.fyxf$ (obviamente a equação reduz-se a $F = MF$). Defina-se então $F \equiv \boldsymbol{Y}M$. De imediato tem-se:

$$F \equiv \boldsymbol{Y}M = M(\boldsymbol{Y}M) \equiv MF \equiv (\lambda fxy.fyxf)F \to_\beta \lambda xy.FyxF.$$

Este exemplo é apenas uma instância do seguinte resultado geral.

Proposição 5.5 *Resolução de equações combinatórias.*
Seja C um termo com $vl(C) \subseteq \{f, x_1, \ldots, x_n\}$. Então, existe um termo F tal que, para quaisquer $M_1, \ldots, M_n$, se tem $\vdash_\lambda FM_1 \ldots M_n = (\lambda f x_1 \ldots x_n.C)FM_1 \ldots M_n$.

Este é o resultado essencial que permite a emergência do paradigma de programação funcional, nomedamente do tratamento das definições recursivas. É claro que, tendo em vista a construção de programas com base no cálculo lambda, é útil dispor de algumas construções básicas. Em vez de as introduzir directamente como constantes, o que já sabemos poder levantar diversos problemas, vamos ver como codificá-las no cálculo lambda sem constantes.

Definição 5.6 *Representação de Booleanos.*
Definem-se os combinadores correspondentes aos valores lógicos verdadeiro e falso, respectivamente, por $\boldsymbol{Verd} \equiv \lambda xy.x$ e $\boldsymbol{Falso} \equiv \lambda xy.y$.

Estas definições permitem facilmente representar também expressões condicionais do tipo

$$\boldsymbol{Se}\ \textit{condição}\ \boldsymbol{Então}\ \textit{consequente}\ \boldsymbol{Senão}\ \textit{subsequente}.$$

Definição 5.7 *Representação de expressões condicionais.*
Define-se o combinador correspondente à construção de expressões condicionais por $(\boldsymbol{Se\_Então\_Senão\_}) \equiv \lambda bxy.bxy$.

Para quaisquer termos B, M, N, usamos a abreviatura

$$(\boldsymbol{Se}\ B\ \boldsymbol{Então}\ M\ \boldsymbol{Senão}\ N) \equiv (\boldsymbol{Se\_Então\_Senão\_})BMN.$$

De facto, facilmente, tem-se

$$(\boldsymbol{Se}\ \boldsymbol{Verd}\ \boldsymbol{Então}\ M\ \boldsymbol{Senão}\ N) \equiv (\lambda bxy.bxy)\,\boldsymbol{Verd}MN \twoheadrightarrow_\beta M$$

e

$$(\boldsymbol{Se}\ \boldsymbol{Falso}\ \boldsymbol{Então}\ M\ \boldsymbol{Senão}\ N) \equiv (\lambda bxy.bxy)\boldsymbol{Falso}MN \twoheadrightarrow_\beta N.$$

Outra estrutura interessante são os pares ordenadados.

Definição 5.8 *Representação de pares ordenados e projecções.*
Definem-se os combinadores correspondentes à construção e destrução de pares ordenados, respectivamente, por $[\_,\_] \equiv \lambda xyz.zxy$, $(\_)_0 \equiv \lambda x.x\,\boldsymbol{Verd}$ e $(\_)_1 \equiv \lambda x.x\boldsymbol{Falso}$.

Para quaisquer termos M, N, usamos as abreviaturas

$$[M, N] \equiv [\_,\_]MN$$

$$(M)_0 \equiv (\_)_0 M$$

$$(M)_1 \equiv (\_)_1 M.$$

De facto, facilmente, tem-se

$$([M, N])_0 \equiv (\lambda x.x\,\boldsymbol{Verd})((\lambda xyz.zxy)MN) \twoheadrightarrow_\beta M$$

e

$$([M, N])_1 \equiv (\lambda x.x\boldsymbol{Falso})((\lambda xyz.zxy)MN) \twoheadrightarrow_\beta N.$$

À luz do que vimos no final do Capítulo 2, como seria de esperar, estas definições não satisfazem em geral $\vdash_\lambda M = [(M)_0, (M)_1]$.

Ainda mais interessante é a possibilidade de representar os números naturais.

Definição 5.9 *Representação de números naturais.*
Definem-se indutivamente os combinadores correspondentes aos números naturais por $\ulcorner 0 \urcorner \equiv \lambda x.x$ e $\ulcorner n+1 \urcorner \equiv [\textbf{\textit{Falso}}, \ulcorner n \urcorner]$.

Por exemplo, ter-se-á

$$\ulcorner 3 \urcorner \equiv [\textbf{\textit{Falso}}, [\textbf{\textit{Falso}}, [\textbf{\textit{Falso}}, \ulcorner 0 \urcorner]]].$$

Usando esta definição, é fácil construir combinadores correspondentes a algumas operações básicas sobre os naturais.

Definição 5.10 *Sucessor, predecessor e nulidade.*
Definem-se os combinadores correspondentes às operações de sucessor, predecessor e nulidade de um número natural, respectivamente, por $\textbf{\textit{Suc}} \equiv \lambda x.[\textbf{\textit{Falso}}, x]$, $\textbf{\textit{Pred}} \equiv \lambda x.(x)_1$ e $\textbf{\textit{Zero?}} \equiv \lambda x.(x)_0$.

Note que se tem $\vdash_\lambda \textbf{\textit{Suc}} \ulcorner n \urcorner = \ulcorner n+1 \urcorner$, $\vdash_\lambda \textbf{\textit{Pred}} \ulcorner n+1 \urcorner = \ulcorner n \urcorner$, $\vdash_\lambda \textbf{\textit{Zero?}} \ulcorner 0 \urcorner = \textbf{\textit{Verd}}$ e $\vdash_\lambda \textbf{\textit{Zero?}} \ulcorner n+1 \urcorner = \textbf{\textit{Falso}}$. Sendo o predecessor uma função parcial nos naturais tem-se $\vdash_\lambda \textbf{\textit{Pred}} \ulcorner 0 \urcorner = \textbf{\textit{Falso}}$.

Podemos definir, com generalidade, o que se entende por *definibilidade-λ* de funções numéricas do tipo $\varphi : \mathbb{N}^k \nrightarrow \mathbb{N}$ com $k \in \mathbb{N}$. A seta cortada $\nrightarrow$ indica que a função φ pode não ser total. Comecemos no entanto por considerar apenas as usuais funções totais.

Definição 5.11 *Definibilidade-λ de funções totais.*
Uma função total $\varphi : \mathbb{N}^k \to \mathbb{N}$, com $k \in \mathbb{N}$, diz-se definível-λ se existe um termo lambda F tal que, para quaisquer $n_1, \ldots, n_k \in \mathbb{N}$, se tem:

$$\vdash_\lambda F \ulcorner n_1 \urcorner \ldots \ulcorner n_k \urcorner = \ulcorner \varphi(n_1, \ldots, n_k) \urcorner.$$

Se notarmos que cada número natural n é representado por um termo na forma normal-β, o teorema de Church-Rosser garante que se tem, inclusive, que

$$F \ulcorner n_1 \urcorner \ldots \ulcorner n_k \urcorner \twoheadrightarrow_\beta \ulcorner \varphi(n_1, \ldots, n_k) \urcorner.$$

Então, por exemplo, a função $\varphi : \mathbb{N} \to \mathbb{N}$ tal que $\varphi(n) = n+1$ para qualquer $n \in \mathbb{N}$ é definível-λ, bastando para tal considerar o combinador $\textbf{\textit{Suc}}$ definido anteriormente.

A noção de definibilidade pode ser estendida a funções parciais. Recorde-se que uma função parcial $\varphi : \mathbb{N}^k \nrightarrow \mathbb{N}$ é essencialmente uma função total $\varphi : D \to \mathbb{N}$ com $D \subseteq \mathbb{N}^k$. Se $\vec{n} \in D$ então $\varphi(\vec{n})$ está definido, o que denotamos por $\varphi(\vec{n}) \downarrow$. Caso contrário, se $\vec{n} \notin D$, $\varphi(\vec{n})$ está indefinido, o que denotamos por $\varphi(\vec{n}) \uparrow$.

Definição 5.12 *Definibilidade-λ de funções parciais.*
Uma função parcial $\varphi : \mathbb{N}^k \nrightarrow \mathbb{N}$, com $k \in \mathbb{N}$, diz-se definível-λ se existe um termo lambda F tal que, para quaisquer $n_1, \ldots, n_k \in \mathbb{N}$, se tem:

$$\vdash_\lambda F \ulcorner n_1 \urcorner \ldots \ulcorner n_k \urcorner = \ulcorner \varphi(n_1, \ldots, n_k) \urcorner, \text{ se } \varphi(n_1, \ldots, n_k) \downarrow$$

e

$$F \ulcorner n_1 \urcorner \ldots \ulcorner n_k \urcorner \ \lambda\text{-insolúvel, se } \varphi(n_1, \ldots, n_k) \uparrow.$$

Recorde-se aqui a intuição por detrás da definição de termos insolúveis, como o combinador $\mathbf{\Omega}$, que vimos no final do Capítulo 3. Obviamente, esta noção coincide com a anterior noção de definibilidade sempre que a função em causa seja total.

Note-se no entanto que, de acordo com esta definição, o combinador ***Pred*** não demonstra que a função parcial $\varphi : \mathbb{N} \nrightarrow \mathbb{N}$ tal que $\varphi(0) \uparrow$ e $\varphi(n+1) = n$ para qualquer $n \in \mathbb{N}$ é definível-λ. Isto porque o termo ***Pred*** $\ulcorner 0 \urcorner$ não é insolúvel. Para tal, haveria que considerar por exemplo, o termo $\lambda x.(\textbf{\textit{Zero?}}\ x)\ \mathbf{\Omega}\ (\textbf{\textit{Pred}}\ x)$.

Exercícios

A. Verifique que não se tem $\boldsymbol{Y}M \twoheadrightarrow_\beta M(\boldsymbol{Y}M)$. Mostre que:

   (i) o combinador $\mathbf{\Theta} \equiv (\lambda xy.y(xxy))(\lambda xy.y(xxy))$ é um determinador de pontos fixos;

   (ii) para qualquer M, tem-se $\mathbf{\Theta}M \twoheadrightarrow_\beta M(\mathbf{\Theta}M)$.

B. Considere o combinador $\boldsymbol{G} \equiv \lambda xy.y(xy)$. Mostre que se M é ponto fixo de $\boldsymbol{G}$ então M é um determinador de pontos fixos.

C. Defina-se $\boldsymbol{Y}^0 \equiv \boldsymbol{Y}$ e $\boldsymbol{Y}^{n+1} \equiv \boldsymbol{Y}^n(\textbf{\textit{SI}})$ para cada $n \in \mathbb{N}$. Mostre que todos os termos $\boldsymbol{Y}^k$ para $k \in \mathbb{N}$ são determinadores de pontos fixos.

D. Dado um termo M defina-se $\boldsymbol{Y}_M \equiv \lambda y.WWM$, com $W \equiv \lambda xz.y(xxz)$. Mostre que $\boldsymbol{Y}_M$ é um determinador de pontos fixos.

E. Considere a seguinte definição proposta por Klop:

$$\pounds \equiv \lambda abcdefghijklmnopqstuvwxyzr.r(thisisafixedpointcombinator).$$

   Mostre que

$$\$ \equiv \pounds \equiv \pounds^{26}$$

   é um determinador de pontos fixos.

F. Verifique que $\vdash_\lambda \boldsymbol{YI} = \mathbf{\Omega}$. Interprete o facto à luz da insolubilidade de $\mathbf{\Omega}$.

G. Em cada um dos casos, encontre um termo M que satisfaça a igualdade:

   (i) $Mx = M$;

   (ii) $Mxy = xMy$;

   (iii) $Mx = xM$.

H. Defina combinadores ***Neg***, ***Conj***, ***Disj***, ***Impl*** e ***Equiv*** correspondentes aos habituais conectivos Booleanos.

I. Verifique que, de facto, se tem:

   (i) $\vdash_\lambda \textbf{\textit{Suc}}\ \ulcorner n \urcorner = \ulcorner n+1 \urcorner$;

   (ii) $\vdash_\lambda \textbf{\textit{Pred}}\ \ulcorner 0 \urcorner = \textbf{\textit{Falso}}$;

   (iii) $\vdash_\lambda \textbf{\textit{Pred}}\ \ulcorner n+1 \urcorner = \ulcorner n \urcorner$;

   (iv) $\vdash_\lambda \textbf{\textit{Zero?}}\ \ulcorner 0 \urcorner = \textbf{\textit{Verd}}$;

   (v) $\vdash_\lambda \textbf{\textit{Zero?}}\ \ulcorner n+1 \urcorner = \textbf{\textit{Falso}}$,

J. Considere o termo $\boldsymbol{Soma} \equiv \boldsymbol{Y}(\lambda fxy.(\boldsymbol{Zero?}\ x)\ y\ (\boldsymbol{Suc}\ (f\ (\boldsymbol{Pred}\ x)\ y)))$. Verifique que, para quaisquer $n, m \in \mathbb{N}$, se tem $\vdash_\lambda \boldsymbol{Soma}\ \ulcorner n \urcorner\ \ulcorner m \urcorner = \ulcorner n+m \urcorner$. Conclua que a função $(\_ + \_) : \mathbb{N}^2 \to \mathbb{N}$ é definível-λ.

K. Mostre que as seguintes funções totais são definíveis-λ:

(i) a função produto $(\_ \times \_) : \mathbb{N}^2 \to \mathbb{N}$;

(ii) a função factorial $(\_!) : \mathbb{N} \to \mathbb{N}$;

(iii) a função mínimo $\min(\_, \_) : \mathbb{N}^2 \to \mathbb{N}$.

L. Mostre que a função parcial $\varphi : \mathbb{N} \nrightarrow \mathbb{N}$ tal que $\varphi(n) = n$ se n é par e $\varphi(n) \uparrow$ se n é ímpar é definível-λ.

M. Considere a representação de Church $\widetilde{\ }$ dos números naturais definida por $\widetilde{n} \equiv \lambda fx.f^n(x)$ onde $f^0(x) \equiv x$ e $f^{m+1}(x) \equiv ff^m(x)$. Mostre que:

(i) existe G tal que $\vdash_\lambda G\ \ulcorner n \urcorner = \widetilde{n}$;

(ii) existe H tal que $\vdash_\lambda H\ \widetilde{n} = \ulcorner n \urcorner$;

(iii) existe P tal que $\vdash_\lambda P\ \widetilde{n+1} = \widetilde{n}$.

Capítulo 6

Computabilidade

Neste capítulo mostramos que as funções numéricas definíveis-λ concordam com a noção clássica de computabilidade efectiva, permitindo abordar o estudo da teoria da computabilidade no contexto do cálculo lambda.

Postulado 6.1 *Church-Markov-Turing.*
As funções numéricas efectivamente computáveis são precisamente as funções parciais recursivas.

Antes de recordarmos a definição da classe das funções parciais recursivas, comecemos por considerar a classe restrita das funções totais ditas recursivas.

Definição 6.2 *Funções básicas.*
As funções básicas são:

- $U_i^k : \mathbb{N}^k \to \mathbb{N}$, com $i \leq k$, tal que $U_i^k(\vec{n}) = n_i$ para qualquer $\vec{n} \in \mathbb{N}^k$;
- $Z : \mathbb{N} \to \mathbb{N}$ tal que $Z(n) = 0$ para qualquer $n \in \mathbb{N}$;
- $S : \mathbb{N} \to \mathbb{N}$ tal que $S(n) = n + 1$ para qualquer $n \in \mathbb{N}$.

As funções básicas U_i^k, Z e S são geralmente denominadas de "projecções", "zero" e "sucessor", respectivamente. Todas elas são obviamente definíveis-λ.

Lema 6.3 *Definibilidade-λ das funções básicas.*
As funções básicas são definíveis-λ por:

- $\lambda x_1 \ldots x_k.x_i$, *para cada* U_i^k;
- $\lambda x.\ulcorner 0 \urcorner$, *para* Z;
- $\boldsymbol{Suc}$, *para* S.

Definição 6.4 *Composição de funções totais.*
Sejam $\varphi : \mathbb{N}^j \to \mathbb{N}$ e $\psi_1, \ldots, \psi_j : \mathbb{N}^k \to \mathbb{N}$ funções totais. Diz-se que a função total $\gamma : \mathbb{N}^k \to \mathbb{N}$ se obtém por composição a partir de φ e $\psi_1, \ldots, \psi_j$, o que se denota por $\gamma = \varphi \circ \langle \psi_1, \ldots, \psi_j \rangle$, se para qualquer $\vec{n} \in \mathbb{N}^k$ se tem

$$\gamma(\vec{n}) = \varphi(\psi_1(\vec{n}), \ldots, \psi_j(\vec{n})).$$

Lema 6.5 *Definibilidade-λ da composição de funções totais.*
Sejam $\varphi : \mathbb{N}^j \to \mathbb{N}$ e $\psi_1, \ldots, \psi_j : \mathbb{N}^k \to \mathbb{N}$ funções totais definíveis-λ, respectivamente, por F e $G_1, \ldots, G_j$. Então, a função composta $\varphi \circ \langle \psi_1, \ldots, \psi_j \rangle$ é definível-λ por

$$\lambda x_1 \ldots x_k.F(G_1 x_1 \ldots x_k) \ldots (G_j x_1 \ldots x_k).$$

Definição 6.6 *Recursão.*
Sejam $\varphi : \mathbb{N}^k \to \mathbb{N}$ e $\psi : \mathbb{N}^{k+2} \to \mathbb{N}$ funções totais. Diz-se que $\gamma : \mathbb{N}^{k+1} \to \mathbb{N}$ se obtém por recursão a partir de φ e ψ, o que se denota por $\gamma = \mathrm{rec}(\varphi, \psi)$, se para quaisquer $n \in \mathbb{N}$ e $\vec{m} \in \mathbb{N}^k$ se tem

$$\gamma(0, \vec{m}) = \varphi(\vec{m})$$

e

$$\gamma(n+1, \vec{m}) = \psi(\gamma(n, \vec{m}), n, \vec{m}).$$

Lema 6.7 *Definibilidade-λ da recursão.*
Sejam $\varphi : \mathbb{N}^k \to \mathbb{N}$ e $\psi : \mathbb{N}^{k+2} \to \mathbb{N}$ funções totais definíveis-λ, respectivamente, por F e G. Então, a função $rec(\varphi, \psi)$ é definível-λ por

$$\boldsymbol{Y}(\lambda h y x_1 \dots x_k.(\boldsymbol{Zero?}\ y)(F x_1 \dots x_k)(G\ (h\ (\boldsymbol{Pred}\ y)\ x_1 \dots x_k)\ (\boldsymbol{Pred}\ y)\ x_1 \dots x_k)).$$

Definição 6.8 *Minimização garantida.*
Seja $\varphi : \mathbb{N}^{k+1} \to \mathbb{N}$ uma função total que satisfaz, para qualquer $\vec{n} \in \mathbb{N}^k$, a condição $\{m \in \mathbb{N} : \varphi(m, \vec{n}) = 0\} \neq \emptyset$. Diz-se que $\gamma : \mathbb{N}^k \to \mathbb{N}$ se obtém por minimização garantida a partir de φ, o que se denota por $\gamma = \mu\varphi$, se para qualquer $\vec{n} \in \mathbb{N}^k$ se tem

$$\gamma(\vec{n}) = \min\{m \in \mathbb{N} \mid \varphi(m, \vec{n}) = 0\}.$$

Uma função numérica φ nas condições da definição acima de minimização garantida diz-se *anulável*.

Lema 6.9 *Definibilidade-λ da minimização garantida.*
Seja $\varphi : \mathbb{N}^{k+1} \to \mathbb{N}$ uma função total anulável definível-λ por F. Então, a função $\mu\varphi$ é definível-λ por

$$\lambda x_1 \dots x_k.\, \boldsymbol{Y}(\lambda h y.(\boldsymbol{Zero?}\ (F y x_1 \dots x_k))\ y\ (h\ (\boldsymbol{Suc}\ y)))\ \ulcorner 0 \urcorner.$$

Definição 6.10 *Funções recursivas.*
A classe $\mathcal{R}$ das funções (totais) recursivas é definida indutivamente pelas seguintes regras:

básicas: $\dfrac{}{U_i^k} \quad \dfrac{}{Z} \quad \dfrac{}{S}$

composição total: $\dfrac{\varphi \quad \psi_1 \ \dots \ \psi_k}{\varphi \circ \langle \psi_1, \dots, \psi_k \rangle}$

recursão: $\dfrac{\varphi \quad \psi}{\mathrm{rec}(\varphi, \psi)}$

minimização garantida: $\dfrac{\varphi}{\mu\varphi}$, φ anulável.

Proposição 6.11 *Teorema de Kleene das funções recursivas.*
As funções numéricas totais definíveis-λ são precisamente as funções recursivas.

Este resultado pode agora ser generalizado também às funções parciais. Para tal necessitamos, no entanto, de ter algum cuidado.

Definição 6.12 *Composição de funções parciais.*
Sejam $\varphi : \mathbb{N}^j \nrightarrow \mathbb{N}$ e $\psi_1, \dots, \psi_j : \mathbb{N}^k \nrightarrow \mathbb{N}$ funções. Diz-se que $\gamma : \mathbb{N}^k \nrightarrow \mathbb{N}$ se obtém por composição a partir de φ e $\psi_1, \dots, \psi_j$, o que se denota por $\gamma = \varphi \circ \langle \psi_1, \dots, \psi_j \rangle$, se para qualquer $\vec{n} \in \mathbb{N}^k$ se tem

$$\gamma(\vec{n}) = \varphi(\psi_1(\vec{n}), \dots, \psi_j(\vec{n})), \text{ se } \psi_1(\vec{n}) \downarrow \text{ e } \dots \text{ e } \psi_j(\vec{n}) \downarrow \text{ e } \varphi(\psi_1(\vec{n}), \dots, \psi_j(\vec{n})) \downarrow$$

e

$$\gamma(\vec{n}) \uparrow \text{, caso contrário.}$$

A questão da definibilidade-λ da composição de funções parciais é relativamente mais complexa. Tomem-se, por exemplo, a função básica $Z : \mathbb{N} \to \mathbb{N}$ e a função $\varphi : \mathbb{N} \not\to \mathbb{N}$ tal que $\varphi(n) \uparrow$ para qualquer $n \in \mathbb{N}$. Obviamente, elas são definíveis-λ pelos termos $F \equiv \lambda x.\ulcorner 0 \urcorner$ e $G \equiv \lambda x.\mathbf{\Omega}$, respectivamente. No entanto, usando a construção utilizada para compôr funções totais obter-se-ia o termo $\lambda x.F(Gx)$ que, claramente, também define-λ a função Z e não a função $Z \circ \varphi = \varphi$. De facto, tem-se

$$(\lambda x.F(Gx))\ \ulcorner n \urcorner \to_\beta F(G\ \ulcorner n \urcorner) \equiv (\lambda x.\ulcorner 0 \urcorner)(G\ \ulcorner n \urcorner) \to_\beta \ulcorner 0 \urcorner.$$

A forma de ultrapassar este problema reside na própria noção de insolubilidade. De facto é fácil verificar que para qualquer numeral se tem $\vdash_\lambda \ulcorner n \urcorner \boldsymbol{KII} = \boldsymbol{I}$. Por outro lado, se M é um termo insolúvel então também o é o termo $M\boldsymbol{KII}$.

Lema 6.13 *Definibilidade-λ da composição de funções parciais.*
Sejam $\varphi : \mathbb{N}^j \not\to \mathbb{N}$ e $\psi_1, \ldots, \psi_j : \mathbb{N}^k \not\to \mathbb{N}$ funções definíveis-λ, respectivamente, por F e $G_1, \ldots, G_j$. Então, a função composta $\varphi \circ \langle \psi_1, \ldots, \psi_j \rangle$ é definível-λ por

$$\lambda x_1 \ldots x_k.(G_1 x_1 \ldots x_k \boldsymbol{KII}) \ldots (G_j x_1 \ldots x_k \boldsymbol{KII})(F(G_1 x_1 \ldots x_k) \ldots (G_j x_1 \ldots x_k)).$$

Para obtermos a classe das funções parciais recursivas há ainda que considerar uma outra construção que generaliza a minimização garantida.

Definição 6.14 *Minimização.*
Seja $\varphi : \mathbb{N}^{k+1} \not\to \mathbb{N}$ uma função. Diz-se que $\gamma : \mathbb{N}^k \not\to \mathbb{N}$ se obtém por minimização a partir de φ, o que se denota por $\gamma = \mu\varphi$, se para qualquer $\vec{n} \in \mathbb{N}^k$ se tem

$$\gamma(\vec{n}) = \min X, \text{ se } X = \{m \in \mathbb{N} \mid \varphi(m, \vec{n}) = 0\} \neq \emptyset \text{ e } \varphi(m, \vec{n}) \downarrow \text{ para } m < \min X$$

e

$$\gamma(\vec{n}) \uparrow, \text{ caso contrário.}$$

Lema 6.15 *Definibilidade-λ da minimização.*
Seja $\varphi : \mathbb{N}^{k+1} \not\to \mathbb{N}$ uma função definível-λ por F. Então, a função $\mu\varphi$ é definível-λ por

$$\lambda x_1 \ldots x_k.\, \boldsymbol{Y}(\lambda hy.(\boldsymbol{Zero?}\ (Fyx_1 \ldots x_k))\ y\ (h\ (\boldsymbol{Suc}\ y)))\ \ulcorner 0 \urcorner.$$

Definição 6.16 *Funções parciais recursivas.*
A classe $\mathcal{PR}$ das funções parciais recursivas é definida indutivamente a partir da classe $\mathcal{R}$ das funções recursivas pelas seguintes regras:

$$\text{composição parcial:} \quad \frac{\varphi \quad \psi_1 \ \ldots \ \psi_k}{\varphi \circ \langle \psi_1, \ldots, \psi_k \rangle}$$

$$\text{minimização:} \quad \frac{\varphi}{\mu\varphi}.$$

Proposição 6.17 *Teorema de Kleene das funções parciais recursivas.*
As funções numéricas definíveis-λ são precisamente as funções parciais recursivas.

Com este resultado, podemos agora desenvolver alguns fundamentos da teoria da computabilidade no contexto do cálculo lambda. O primeiro facto notável é a possibilidade de estabelecer uma bijecção efectiva entre os termos lambda e os números naturais. Embora consideremos os termos lambda definidos sobre um alfabeto sem símbolos de constante, a definição é facilmente transportável para o caso geral.

Definição 6.18 *Números de Gödel.*
A função $\sharp : \Lambda \to \mathbb{N}$ é definida indutivamente por:

- $\sharp v_n = 3n$;
- $\sharp(\lambda v_n M) = 3P(n, \sharp M) + 1$;
- $\sharp(MN) = 3P(\sharp M, \sharp N) + 2$,

onde $P : \mathbb{N}^2 \to \mathbb{N}$ é definida por $P(m, n) = 2^m(2n + 1) - 1$.

Proposição 6.19 *Bijecção da numeração de Gödel.*
A função $\sharp : \Lambda \to \mathbb{N}$ *é uma bijecção.*

Nomeadamente, a função $P : \mathbb{N}^2 \to \mathbb{N}$ definida acima também estabelece uma bijecção. Usando $\sharp$ é possível importar para o cálculo lambda algumas noções e resultados fundamentais da teoria da computabilidade.

Definição 6.20 *Conjunto recursivo e conjunto recursivamente enumerável.*
Um conjunto $A \subseteq \mathbb{N}$ diz-se:

- recursivo se é recursiva a função característica $\chi_A : \mathbb{N} \to \mathbb{N}$ definida por

$$\chi_A(n) = \begin{cases} 1 & \text{se } n \in A \\ 0 & \text{se } n \notin A \end{cases} ;$$

- recursivamente enumerável (r.e.) se é parcial recursiva a função $\varphi_A : \mathbb{N} \rightharpoonup \mathbb{N}$ definida por

$$\varphi_A(n) = \begin{cases} 1 & \text{se } n \in A \\ \text{indefinido} & \text{se } n \notin A \end{cases} .$$

É relativamente fácil verificar que A é recursivo se e só se A e $\mathbb{N} \setminus A$ são ambos recursivamente enumeráveis. Daí que qualquer conjunto recursivo seja também recursivamente enumerável. O recíproco, no entanto, não se passa. Veremos mais adiante como construir um conjunto recursivamente enumerável que não seja recursivo.

As noções de conjunto recursivo e recursivamente enumerável estão fortemente ligadas às questões da decidibilidade.

Definição 6.21 *Decidibilidade, semi-decidibilidade.*
Seja $\mathcal{P}$ uma propriedade dos números naturais. A propriedade $\mathcal{P}$ diz-se:

- decidível se $\{n \in \mathbb{N} \mid \mathcal{P}(n) = 1\}$ é recursivo;
- semi-decidível se $\{n \in \mathbb{N} \mid \mathcal{P}(n) = 1\}$ é recursivamente enumerável.

Como já foi afirmado antes, é possível que uma propriedade não decidível seja semi-decidível. (In)felizmente, no entanto, existem muitas propriedades interessantes que não são sequer semi-decidíveis.

Proposição 6.22 *Segundo teorema do ponto fixo.*
Para qualquer termo lambda F *existe um termo* M *tal que* $\vdash_\lambda F \ulcorner \sharp M \urcorner = M$.

Este resultado permite-nos demonstrar o seguinte facto, bem conhecido, que enuncia a impossibilidade de determinar se um "programa" dado vai ou não terminar.

Proposição 6.23 *Problema da paragem.*
O conjunto $\{\sharp M \mid M \text{tem forma normal-}\beta\}$ *é recursivamente enumerável mas não recursivo.*

Assim sendo, a propriedade $\mathcal{P}$ tal que $\mathcal{P}(\sharp M) = 1$ se e só se M tem uma forma normal-λ não é decidível, embora seja semi-decidível. Daí, imediatamente, resulta que a propriedade dual $\mathcal{Q}$ tal que $\mathcal{Q}(\sharp M) = 1$ se e só se M não tem uma forma normal-λ não é sequer semi-decidível.

Proposição 6.24 *Indecidibilidade do cálculo-*λ.
Se $\mathcal{T} \subseteq \Lambda$ *e o conjunto* $\{P(\sharp M, \sharp N) \mid \mathcal{T} \vdash_\lambda M = N\}$ *é recursivo então* $\mathcal{T}$ *é* λ-*incoerente.*

Em particular, a teoria coerente $\emptyset^\lambda$ é necessariamente indecidível.

Exercícios

A. Demonstre os Lemas 6.3, 6.5, 6.7 e 6.9.

B. Demonstre que:

   (i) $\vdash_\lambda \ulcorner n \urcorner \boldsymbol{KII} = \boldsymbol{I}$;

   (ii) se M é insolúvel então também o é o termo $M\boldsymbol{KII}$.

C. Demonstre os Lemas 6.13 e 6.15.

D. Use as construções subjacentes aos teoremas de Kleene para definir termos lambda que representem as seguintes funções:

   - $\varphi_1 : \mathbb{N}^2 \to \mathbb{N}$ tal que $\varphi_1(m,n) = n^2.(m+1)$;
   - $\varphi_2 : \mathbb{N}^2 \nrightarrow \mathbb{N}$ tal que $\varphi_2(m,n) = \begin{cases} m-n & \text{se } m \geq n \\ \text{indef} & \text{caso contrário} \end{cases}$;
   - $\varphi_3 : \mathbb{N}^2 \nrightarrow \mathbb{N}$ tal que $\varphi_3(m,n) = \begin{cases} m \mod n & \text{se } n \neq 0 \\ \text{indef} & \text{caso contrário} \end{cases}$;
   - $\varphi_4 : \mathbb{N} \to \mathbb{N}$ tal que $\varphi_4(n)$ é a raíz quadrada inteira de n;
   - $\varphi_5 : \mathbb{N} \to \mathbb{N}$ tal que $\varphi_5(n)$ é o dígito de valência 10^{-n} de $\sqrt{2}$.

E. Supondo um alfabeto com infinitas constantes $Const = \{\mathsf{c}_n \mid n \in \mathbb{N}\}$, mostre que a função $\flat : \Lambda \to \mathbb{N}$ definida indutivamente por

   - $\flat \mathsf{c}_n = 6n$;
   - $\flat v_n = 6n + 3$;
   - $\flat(\lambda v_n M) = 3P(n, \flat M) + 1$;
   - $\flat(MN) = 3P(\flat M, \flat N) + 2$,

   constitui uma bijecção efectiva.

F. Sabendo que $A, B \subseteq \mathbb{N}$ são conjuntos recursivos, mostre que são recursivos os conjuntos $\emptyset$, $\mathbb{N}$, $A \cup B$, $A \cap B$ e $A \setminus B$.

G. Sabendo que $A, B \subseteq \mathbb{N}$ são r.e, mostre que são r.e. os conjuntos $A \cup B$ e $A \cap B$.

H. Demonstre que $A \subseteq \mathbb{N}$ é recursivo se e só se A e $\mathbb{N} \setminus A$ são ambos r.e.

I. Mostre que existe um termo M tal que $\vdash_\lambda M = \ulcorner \sharp M \urcorner$.

J. Um conjunto $A \subseteq \mathbb{N}$ diz-se λ-fechado quando, para quaisquer termos lambda M, N, se $\vdash_\lambda M = N$ e $\sharp M \in A$ então $\sharp N \in A$. Demonstre que:

   (i) se $B, C \subseteq \mathbb{N}$ são conjuntos não vazios e λ-fechados então não existe nenhum conjunto recursivo $D \subseteq \mathbb{N}$ tal que $B \subseteq D$ e $C \cap D = \emptyset$ (versão-λ do *Lema de Rice*);

   (ii) se $B \subseteq \mathbb{N}$ e $\emptyset \neq B \neq \mathbb{N}$ é λ-fechado então B não é recursivo (versão-λ do *Teorema de Rice*).

K. Demonstre as duas alíneas do exercício anterior directamente, sem recorrer ao segundo teorema do ponto fixo.

L. Demonstre a indecidibilidade do problema da paragem directamente, sem recorrer ao Teorema de Rice.

M. Mostre que o conjunto $\{\sharp M \mid \vdash_\lambda M = \boldsymbol{I}\}$ não é recursivo.

Capítulo 7

Sistemas de tipos

Há duas escolas de sistemas de tipos para o cálculo lambda (e actuais linguagens de programação): os tipos implícitos e explícitos. A ideia central comum a ambas é que associar um tipo a um termo M é uma garantia (parcial) de que M tem significado computacional. Enquanto nas abordagens implícitas cada termo é descrito sem referência a tipos, que podem ser inferidos depois, nas abordagens explícitas os termos são definidos conjuntamente com o seu tipo. Embora as abordagens se relacionem, os tipos explícitos são menos flexíveis e exigem uma sintaxe mais elaborada, pelo que nos concentramos apenas no estudo de sistemas de tipos implícitos. O mais simples de todos é o sistema-$\lambda\!\rightarrow$ de Curry.

A linguagem dos *tipos* é definida sobre um conjunto $TVar = \{\gamma_n \mid n \in \mathbb{N}\}$ de variáveis de tipo[1].

Definição 7.1 *Tipos-$\lambda\!\rightarrow$.*
O conjunto $T_{\lambda\rightarrow}$ dos tipos do sistema $\lambda\!\rightarrow$ é definido indutivamente pelas regras:

$$\text{variáveis de tipo: } \frac{}{\gamma_n} \qquad\qquad \text{espaços de funções: } \frac{\sigma \quad \tau}{(\sigma\rightarrow\tau)}.$$

É usual adoptar as seguintes convenções de notação:

- $\alpha, \beta, \gamma, \ldots$ com ou sem índices ou plicas denotam variáveis de tipo arbitrárias;
- $\rho, \sigma, \tau, \ldots$ com ou sem índices ou plicas denotam tipos arbitrários;
- o símbolo $\equiv$ denota a relação de igualdade sintáctica entre tipos;
- $\sigma_1 \rightarrow \sigma_2 \rightarrow \cdots \rightarrow \sigma_{n-1} \rightarrow \sigma_n \equiv (\sigma_1 \rightarrow (\sigma_2 \rightarrow (\cdots \rightarrow (\sigma_{n-1} \rightarrow \sigma_n)\ldots)))$.

Cada tipo funcional $\sigma \rightarrow \tau$ representa, intuitivamente, os termos lambda que quando aplicados a um termo de tipo σ resultam num termo de tipo τ.

Definição 7.2 *Atribuições, declarações e bases.*
Uma atribuição de tipo é uma expressão $M : \sigma$ onde $M \in \Lambda$ e σ é um tipo. Uma declaração de tipo $x : \sigma$ é uma atribuição de tipo a uma variável. Uma base é um conjunto de declarações de tipo a variáveis distintas.

Em $M : \sigma$, M diz-se e o *sujeito* e σ o *predicado* da atribuição. Uma base Γ exprime um contexto em que certas variáveis têm os tipos indicados. Obviamente a definição exclui a hipótese de Γ conter declarações distintas para a mesma variável, mas admite que apenas algumas variáveis tenham tipo declarado. Escrevemos $x \in \Gamma$ se existe uma declaração para x em Γ e usamos $\Gamma(x)$ para denotar o tipo declarado.

[1] É usual haver também um conjunto $TConst$ de tipos básicos. Para simplificar, assumimos $TConst = \emptyset$.

A partir de uma base é possível *inferir* tipos para termos compostos.

Definição 7.3 *Inferência de tipos-$\lambda\!\rightarrow$.*
A relação de inferência de tipos $\vdash_{\lambda\rightarrow}$, entre bases e atribuições, é definida indutivamente pelas seguintes regras:

$$\text{axioma base:} \quad \frac{}{\Gamma \vdash_{\lambda\rightarrow} x:\sigma},\ \Gamma(x) \equiv \sigma$$

$$\text{eliminação de } \rightarrow\text{:} \quad \frac{\Gamma \vdash_{\lambda\rightarrow} M:(\sigma\rightarrow\tau) \qquad \Gamma \vdash_{\lambda\rightarrow} N:\sigma}{\Gamma \vdash_{\lambda\rightarrow} (MN):\tau}$$

$$\text{introdução de } \rightarrow\text{:} \quad \frac{\Gamma \cup \{x:\sigma\} \vdash_{\lambda\rightarrow} M:\tau}{\Gamma \vdash_{\lambda\rightarrow} (\lambda x.M):(\sigma\rightarrow\tau)},\ x \notin \Gamma.$$

Tal como antes, se $\emptyset \vdash_{\lambda\rightarrow} M : \sigma$ escrevemos simplesmente $\vdash_{\lambda\rightarrow} M : \sigma$.

Assim, por exemplo, é relativamente simples concluir que $\vdash_{\lambda\rightarrow} \boldsymbol{K} : \sigma \rightarrow \tau \rightarrow \sigma$:

$$\dfrac{\dfrac{\{x:\sigma, y:\tau\} \vdash_{\lambda\rightarrow} x:\sigma}{\{x:\sigma\} \vdash_{\lambda\rightarrow} (\lambda y.x) : (\tau \rightarrow \sigma)}}{\vdash_{\lambda\rightarrow} (\lambda xy.x) : (\sigma \rightarrow (\tau \rightarrow \sigma))},$$

ou que também $\vdash_{\lambda\rightarrow} \boldsymbol{K} : \sigma \rightarrow \sigma \rightarrow \sigma$:

$$\dfrac{\dfrac{\{x:\sigma, y:\sigma\} \vdash_{\lambda\rightarrow} x:\sigma}{\{x:\sigma\} \vdash_{\lambda\rightarrow} (\lambda y.x) : (\sigma \rightarrow \sigma)}}{\vdash_{\lambda\rightarrow} (\lambda xy.x) : (\sigma \rightarrow (\sigma \rightarrow \sigma))}.$$

Noutro exemplo, tem-se que $\{y:\sigma\} \vdash_{\lambda\rightarrow} \boldsymbol{I}y : \sigma$:

$$\dfrac{\dfrac{\{y:\sigma, x:\sigma\} \vdash_{\lambda\rightarrow} x:\sigma}{\{y:\sigma\} \vdash_{\lambda\rightarrow} (\lambda x.x) : \sigma \rightarrow \sigma} \qquad \{y:\sigma\} \vdash_{\lambda\rightarrow} y:\sigma}{\{y:\sigma\} \vdash_{\lambda\rightarrow} (\lambda x.x)y : \sigma}.$$

Em geral um termo pode ter vários tipos, mas também há termos não tipificáveis-$\lambda\!\rightarrow$. Um exemplo paradigmático é xx. Claramente, seria necessário um contexto em que x pudesse ter, simultaneamente, tipos $\sigma \rightarrow \tau$ e σ (como função e argumento), o que é manifestamente impossível. Em contraste, verifica-se que $\vdash_{\lambda\rightarrow} \boldsymbol{II} : \sigma \rightarrow \sigma$.

A tipificação em $\lambda\!\rightarrow$ tem várias propriedades importantes. Analisemos a relevância da informação declarada na base. Dada uma base Γ e um conjunto de variáveis $V \subseteq \mathit{Var}$, denotamos por $\Gamma \restriction V$ a sub base $\{x:\sigma \mid x \in V, \Gamma(x) \equiv \sigma\}$.

Proposição 7.4 *Lema da base em $\lambda\!\rightarrow$.*
Seja Γ uma base e suponha-se que $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$. Então:

- *se Γ' é outra base e $\Gamma \subseteq \Gamma'$ então $\Gamma' \vdash_{\lambda\rightarrow} M : \sigma$;*
- *$vl(M) \subseteq \Gamma$;*
- *$\Gamma \restriction vl(M) \vdash_{\lambda\rightarrow} M : \sigma$.*

Garante-se assim que as tipificações de M dependem de maneira essencial da declaração na base do tipo das suas variáveis livres, e apenas dessas.

Vejamos como a estrutura de um termo determina a estrutura dos seus tipos.

Proposição 7.5 *Lema da geração de tipos-$\lambda\!\rightarrow$.*
Seja Γ uma base. Então:

- *se $\Gamma \vdash_{\lambda\rightarrow} x : \sigma$ então $\Gamma(x) \equiv \sigma$;*
- *se $\Gamma \vdash_{\lambda\rightarrow} (MN) : \tau$ então existe σ tal que $\Gamma \vdash_{\lambda\rightarrow} M : \sigma \rightarrow \tau$ e $\Gamma \vdash_{\lambda\rightarrow} N : \sigma$;*
- *se $\Gamma \vdash_{\lambda\rightarrow} (\lambda x.M) : \rho$ então existem σ, τ tais que $\Gamma \cup \{x : \sigma\} \vdash_{\lambda\rightarrow} M : \tau$ e $\rho \equiv (\sigma \rightarrow \tau)$.*

Na verdade, é claro que o sistema de inferência apenas permite construir tipos para um termo a partir de tipos para todos os seus subtermos.

Proposição 7.6 *Lema da tipificação de subtermos em $\lambda\!\rightarrow$.*
Se $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$ e M' é um subtermo de M então existem uma base Γ' e um tipo σ' tais que $\Gamma' \vdash_{\lambda\rightarrow} M' : \sigma'$.

A inferência de tipos-$\lambda\!\rightarrow$ também é robusta face à substituição de variáveis.

Proposição 7.7 *Lema da substituição em $\lambda\!\rightarrow$.*
Se $\Gamma \cup \{x : \sigma\} \vdash_{\lambda\rightarrow} M : \tau$ e $\Gamma \vdash_{\lambda\rightarrow} N : \sigma$ então $\Gamma \vdash_{\lambda\rightarrow} M[x := N] : \tau$.

A propriedade seguinte garante que as tipificações-$\lambda\!\rightarrow$ resistem à redução-β.

Proposição 7.8 *Teorema da invariância do tipo-$\lambda\!\rightarrow$ face à redução-β.*
Se $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$ e $M \twoheadrightarrow_\beta M'$ então $\Gamma \vdash_{\lambda\rightarrow} M' : \sigma$.

Esta propriedade é também conhecida como teorema da redução do sujeito. Em geral a propriedade recíproca, a expansão do sujeito, não se verifica. Considerando $\boldsymbol{I}$ e $\boldsymbol{KI}(\lambda x.xx)$, obviamente $\boldsymbol{KI}(\lambda x.xx) \twoheadrightarrow_\beta \boldsymbol{I}$. No entanto, tem-se $\vdash_{\lambda\rightarrow} \boldsymbol{I} : \sigma \rightarrow \sigma$ mas $\not\vdash_{\lambda\rightarrow} \boldsymbol{KI}(\lambda x.xx) : \sigma \rightarrow \sigma$ pois já sabemos que o subtermo xx não é tipificável. Mas a expansão do sujeito pode falhar mesmo quando os termos são tipificáveis. Considerem-se agora os termos $\boldsymbol{Falso}$ e $\boldsymbol{SK}$. Novamente, é claro que $\boldsymbol{SK} \twoheadrightarrow_\beta \boldsymbol{Falso}$. Para além disso, tem-se $\vdash_{\lambda\rightarrow} \boldsymbol{Falso} : \alpha \rightarrow \beta \rightarrow \beta$ e $\vdash_{\lambda\rightarrow} \boldsymbol{SK} : (\beta \rightarrow \alpha) \rightarrow \beta \rightarrow \beta$ mas $\not\vdash_{\lambda\rightarrow} \boldsymbol{SK} : \alpha \rightarrow \beta \rightarrow \beta$.

O sistema-$\lambda\!\rightarrow$ é suficientemente forte para garantir que se um termo é tipificável então não existe nenhum caminho de redução infinito a partir dele.

Proposição 7.9 *Teorema da normalização forte em $\lambda\!\rightarrow$.*
Se $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$ então M é fortemente normalizável.

Em linguagens de programação e sistemas de demonstração automática é importante que vários problemas relacionados com o sistema de tipos sejam decidíveis. Fixada uma base Γ, os problemas mais usuais consistem em determinar se:

- *tipificação de termos*: dado $M \in \Lambda$ existe $\sigma \in T_{\lambda\rightarrow}$ tal que $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$;
- *verificação de tipos*: dados $M \in \Lambda$ e $\sigma \in T_{\lambda\rightarrow}$ se tem $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$;
- *habitação de tipos*: dado $\sigma \in T_{\lambda\rightarrow}$ existe $M \in \lambda$ tal que $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$.

Como facilmente se verifica que $\Gamma \vdash_{\lambda\rightarrow} M : \sigma$ se e só se $\vdash_{\lambda\rightarrow} (\lambda x_1 \ldots x_n.M) : \Gamma(x_1) \rightarrow \cdots \rightarrow \Gamma(x_n) \rightarrow \sigma$, onde $vl(M) \equiv \{x_1, \ldots, x_n\}$, é suficiente considerar os casos em que $\Gamma = \emptyset$ e M é um combinador.

Os dois primeiros problemas estão fortemente relacionados e a solução para ambos assenta na ideia de tipo principal.

Uma *substituição de tipos* é uma função $s : TVar \to T_{\lambda\to}$. Denotamos por $[\alpha_1 := \sigma_1, \ldots, \alpha_n := \sigma_n]$ a substituição s tal que $s(\alpha_i) \equiv \sigma_i$ para $1 \leq i \leq n$ e $s(\beta) \equiv \beta$ se $\beta \notin \{\alpha_1, \ldots, \alpha_n\}$, e usamos id para denotar a substituição trivial. Denotamos por σs o tipo que se obtém a partir de σ por substituição simultânea das ocorrência de cada $\alpha \in TVar$ por $s(\alpha)$. Dadas s_1 e s_2 denotamos por $s_2 \circ s_1$ a substituição definida por $(s_2 \circ s_1)(\alpha) \equiv s_1(\alpha)s_2$.

Definição 7.10 *Tipos variantes.*
Diz-se que σ e τ são tipos variantes se existem s_1 e s_2 tais que $\sigma \equiv \tau s_1$ e $\tau \equiv \sigma s_2$.

Tipos variantes são portanto iguais, a menos de uma troca de variáveis de tipo.

Definição 7.11 *Tipo principal.*
Diz-se que σ é um tipo principal de $M \in \Lambda^0$ se são satisfeitas as seguintes condições:

- $\vdash_{\lambda\to} M : \sigma$;
- se $\vdash_{\lambda\to} M : \sigma'$ então existe s tal que $\sigma' \equiv \sigma s$.

Note-se que o tipo principal de um combinador é único a menos de uma troca de variáveis de tipo. Obviamente, será útil garantir que todo o combinador tipificável-$\lambda\to$ tem um tipo principal. Para tal recorremos à noção de unificação.

Definição 7.12 *Tipos unificáveis.*
Um conjunto de pares $P \subseteq T_{\lambda\to} \times T_{\lambda\to}$ diz-se unificável se existe uma substituição de tipos s tal que $\sigma s \equiv \tau s$ para cada par $\langle \sigma, \tau \rangle \in P$.

Nas condições acima, s diz-se uma substituição *unificadora* de P. Por exemplo $[\alpha := \beta, \gamma := \beta]$ é uma unificadora de $\{\langle \alpha \to \beta \to \alpha, \beta \to \beta \to \gamma \rangle\}$. A substituição $[\alpha := \beta \to \theta, \beta := \beta \to \theta, \gamma := \beta \to \theta]$ é outra unificadora.

Definição 7.13 *Unificadora mais geral.*
Seja $P \subseteq T_{\lambda\to} \times T_{\lambda\to}$ um conjunto de pares. Uma substituição de tipos s diz-se uma unificadora mais geral (umg) de P se são satisfeitas as seguintes condições:

- s é unificadora de P;
- se s_1 é unificadora de P existe s_2 tal que $s_1 = s_2 \circ s$.

É fácil verificar que $s \equiv [\alpha := \beta, \gamma := \beta]$ é umg de $\{\langle \alpha \to \beta \to \alpha, \beta \to \beta \to \gamma \rangle\}$. Em particular, $[\alpha := \beta \to \theta, \beta := \beta \to \theta, \gamma := \beta \to \theta] \equiv [\beta := \beta \to \theta] \circ s$.

Note-se que a umg de um conjunto de pares é única a menos de uma troca de variáveis de tipo. Vejamos qual a utilidade da noção de unificação de tipos.

Proposição 7.14 *Teorema de Wand.*
Seja Γ uma base e $M \in \Lambda$ tal que $vl(M) \subseteq \Gamma$. Dado $\sigma \in T_{\lambda\to}$, define-se o conjunto finito de pares $P(\Gamma, M, \sigma) \subseteq T_{\lambda\to} \times T_{\lambda\to}$ indutivamente por:

- $P(\Gamma, x, \sigma) \equiv \{\langle \Gamma(x), \sigma \rangle\}$;
- $P(\Gamma, NL, \sigma) \equiv P(\Gamma, N, \alpha \to \sigma) \cup P(\Gamma, L, \alpha)$, *se α é fresca (ainda não usada);*
- $P(\Gamma, \lambda x.N, \sigma) \equiv \{\langle \sigma, \alpha \to \beta \rangle\} \cup P(\Gamma \cup \{x : \alpha\}, N, \beta)$, *se α, β são frescas.*

Então tem-se:

- *se s unifica $P(\Gamma, M, \sigma)$ então $\Gamma s \vdash_{\lambda\to} M : \sigma s$;*
- *se $\Gamma s \vdash_{\lambda\to} M : \sigma s$ então existe s_1 que coincide com s em todas as variáveis de tipo que ocorrem em Γ ou σ tal que s_1 unifica $P(\Gamma, M, \sigma)$.*

Tem-se por exemplo que $P(\emptyset, \lambda x.xx, \alpha) \equiv \{\langle \alpha, \beta \to \gamma \rangle\} \cup P(\{x : \beta\}, xx, \gamma) \equiv \{\langle \alpha, \beta \to \gamma \rangle\} \cup P(\{x : \beta\}, x, \delta \to \gamma) \cup P(\{x : \beta\}, x, \delta) \equiv \{\langle \alpha, \beta \to \gamma \rangle, \langle \beta, \delta \to \gamma \rangle, \langle \beta, \delta \rangle\}$, que se verifica facilmente não ser unificável.

Proposição 7.15 *Corolário do teorema de Wand.*
Seja M um combinador. Então:

- *se $\vdash_{\lambda\to} M : \sigma s$ então existe s_1 com $\sigma s_1 \equiv \sigma s$ que unifica $P(\emptyset, M, \sigma)$;*
- *se s unifica $P(\emptyset, M, \sigma)$ então $\vdash_{\lambda\to} M : \sigma s$;*
- *se s é umg de $P(\emptyset, M, \alpha)$ então $s(\alpha)$ é um tipo principal de M.*

À luz deste resultado, é imediato que os problemas da tipificação de termos e da verificação de tipos podem ser decididos tirando partido de um algoritmo que calcule, se existirem, umgs dos conjuntos de pares obtidos no teorema de Wand.

Proposição 7.16 *Algoritmo de unificação de Robinson.*
Dado $P \subseteq T_{\lambda\to} \times T_{\lambda\to}$ finito, o procedimento definido a seguir é tal que:

- *se P é unificável então termina com* `unif = sim` *e s umg de P;*
- *se P não é unificável então termina com* `unif = não`.

---

1. `INÍCIO`
2. `unif` $:=$ `sim`; $s := \mathrm{id}$; $Q := P$;
3. `ENQUANTO unif = sim &` $Q \neq \emptyset$ `FAZER`
4. `TOMAR` $\langle \sigma_1, \sigma_2 \rangle \in Q$;
5. `SE` $\sigma_1 \equiv \sigma_2$
6. `ENTÃO` $Q := Q \setminus \{\langle \sigma_1, \sigma_2 \rangle\}$
7. `SENÃO SE` $\sigma_i \equiv \alpha \in TVar$ `&` α `não ocorre em` σ_j
8. `ENTÃO` $s := [\alpha := \sigma_j] \circ s$; $Q := (Q \setminus \{\langle \sigma_1, \sigma_2 \rangle\})[\alpha := \sigma_j]$
9. `SENÂO SE` $\sigma_1 \equiv \rho_1 \to \tau_1$ `&` $\sigma_2 \equiv \rho_2 \to \tau_2$
10. `ENTÃO` $Q := (Q \setminus \{\langle \sigma_1, \sigma_2 \rangle\}) \cup \{\langle \rho_1, \rho_2 \rangle, \langle \tau_1, \tau_2 \rangle\}$
11. `SENÃO unif` $:=$ `não`
12. `FIM`

Por exemplo, confirmando que $P(\emptyset, \lambda x.xx, \alpha)$ não é unificável, tem-se:

Passo 0:
`unif = sim`, $s \equiv \mathrm{id}$, $Q \equiv \{\langle \alpha, \beta \to \gamma \rangle, \langle \beta, \delta \to \gamma \rangle, \langle \beta, \delta \rangle\}$;

Passo 1: tomando $\langle \alpha, \beta \to \gamma \rangle$,
`unif = sim`, $s \equiv [\alpha := \beta \to \gamma]$, $Q \equiv \{\langle \beta, \delta \to \gamma \rangle, \langle \beta, \delta \rangle\}$;

Passo 2: tomando $\langle \beta, \delta \to \gamma \rangle$,
`unif = sim`, $s \equiv [\alpha := (\delta \to \gamma) \to \gamma, \beta := \delta \to \gamma]$, $Q \equiv \{\langle \delta \to \gamma, \delta \rangle\}$;

Passo 3: tomando $\langle \delta \to \gamma, \delta \rangle$,
`unif = não`.

Fica claro que integrando os conceitos de tipo principal e unificação se obtém o resultado desejado. Um combinador M é tipificável-$\lambda\!\rightarrow$ se e só se $P(\emptyset, M, \alpha)$ é unificável, bastando aplicar-lhe o algoritmo de unificação. Para determinar se $\vdash_{\lambda\!\rightarrow} M : \sigma$ podemos, de forma semelhante, verificar se $P(\emptyset, M, \sigma)$ é unificável e obter, no caso afirmativo, uma umg s do conjunto. O problema reduz-se então a verificar se σ e σs são variantes. Para tanto, basta verificar se s é uma troca de variáveis de tipo para σ, ou seja, se s mapeia injectivamente as variáveis de tipo que ocorrem em σ em variáveis de tipo.

Proposição 7.17 *Teorema de Hindley-Milner.*
Os problemas da tipificação de termos e da verificação de tipos-$\lambda\!\rightarrow$ são decidíveis.

Suponha-se que pretendíamos determinar se $\boldsymbol{K}$ seria tipificável-$\lambda\!\rightarrow$. Facilmente se constrói $P(\emptyset, \boldsymbol{K}, \alpha) \equiv \{\langle \alpha, \beta \rightarrow \gamma\rangle, \langle \gamma, \delta \rightarrow \epsilon\rangle, \langle \beta, \epsilon\rangle$ e usando o algoritmo de unificação obtém-se a umg $[\alpha := \epsilon \rightarrow \delta \rightarrow \epsilon, \gamma := \delta \rightarrow \epsilon, \beta := \epsilon]$, pelo que podemos concluir que $\epsilon \rightarrow \delta \rightarrow \epsilon$ é um tipo principal de $\boldsymbol{K}$, que é portanto tipificável.

Suponhamos agora que queríamos verificar a tipificação $\vdash_{\lambda\!\rightarrow} \boldsymbol{K} : \alpha \rightarrow \alpha \rightarrow \alpha$. Aplicando a $P(\emptyset, \boldsymbol{K}, \alpha \rightarrow \alpha \rightarrow \alpha) \equiv \{\langle \alpha \rightarrow \alpha \rightarrow \alpha, \beta \rightarrow \gamma\rangle, \langle \gamma, \delta \rightarrow \epsilon\rangle, \langle \beta, \epsilon\rangle\}$ o algoritmo de unificação obtém-se a umg $s \equiv [\alpha := \epsilon, \beta := \epsilon, \gamma := \epsilon \rightarrow \epsilon, \delta := \epsilon]$. Como $s(\alpha) \equiv \epsilon$ os tipos $\alpha \rightarrow \alpha \rightarrow \alpha$ e $(\alpha \rightarrow \alpha \rightarrow \alpha)s \equiv \epsilon \rightarrow \epsilon \rightarrow \epsilon$ são variantes.

Da mesma forma, para verificar que $\not\vdash_{\lambda\!\rightarrow} \boldsymbol{K} : \alpha \rightarrow \alpha \rightarrow \beta$, vejamos o que acontece com $P(\emptyset, \boldsymbol{K}, \alpha \rightarrow \alpha \rightarrow \beta) \equiv \{\langle \alpha \rightarrow \alpha \rightarrow \beta, \gamma \rightarrow \delta\rangle, \langle \delta, \theta \rightarrow \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$:

Passo 0:
`unif = sim`, $s \equiv \mathrm{id}$, $Q \equiv \{\langle \alpha \rightarrow \alpha \rightarrow \beta, \gamma \rightarrow \delta\rangle, \langle \delta, \theta \rightarrow \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$;

Passo 1: tomando $\langle \alpha \rightarrow \alpha \rightarrow \beta, \gamma \rightarrow \delta\rangle$,
`unif = sim`, $s \equiv \mathrm{id}$, $Q \equiv \{\langle \alpha, \gamma\rangle, \langle \alpha \rightarrow \beta, \delta\rangle, \langle \delta, \theta \rightarrow \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$;

Passo 2: tomando $\langle \alpha, \gamma\rangle$,
`unif = sim`, $s \equiv [\alpha := \gamma]$, $Q \equiv \{\langle \gamma \rightarrow \beta, \delta\rangle, \langle \delta, \theta \rightarrow \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$;

Passo 3: tomando $\langle \gamma \rightarrow \beta, \delta\rangle$,
`unif = sim`, $s \equiv [\alpha := \gamma, \delta := \gamma \rightarrow \beta]$, $Q \equiv \{\langle \gamma \rightarrow \beta, \theta \rightarrow \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$;

Passo 4: tomando $\langle \gamma \rightarrow \beta, \theta \rightarrow \epsilon\rangle$,
`unif = sim`, $s \equiv [\alpha := \gamma, \delta := \gamma \rightarrow \beta]$, $Q \equiv \{\langle \gamma, \theta\rangle, \langle \beta, \epsilon\rangle, \langle \gamma, \epsilon\rangle\}$;

Passo 5: tomando $\langle \gamma, \theta\rangle$,
`unif = sim`, $s \equiv [\alpha := \theta, \delta := \theta \rightarrow \beta, \gamma := \theta]$, $Q \equiv \{\langle \beta, \epsilon\rangle, \langle \theta, \epsilon\rangle\}$;

Passo 6: tomando $\langle \beta, \epsilon\rangle$,
`unif = sim`, $s \equiv [\alpha := \theta, \delta := \theta \rightarrow \epsilon, \gamma := \theta, \beta := \epsilon]$, $Q \equiv \{\langle \theta, \epsilon\rangle\}$;

Passo 7: tomando $\langle \theta, \epsilon\rangle$,
`unif = sim`, $s \equiv [\alpha := \epsilon, \delta := \epsilon \rightarrow \epsilon, \gamma := \epsilon, \beta := \epsilon, \theta := \epsilon]$, $Q \equiv \emptyset$.

Apesar de o procedimento terminar com sucesso temos $s(\alpha) \equiv s(\beta) \equiv \epsilon$ pelo que o mapeamento não é injectivo. Logo $\alpha \rightarrow \alpha \rightarrow \beta$ e $(\alpha \rightarrow \alpha \rightarrow \beta)s \equiv \epsilon \rightarrow \epsilon \rightarrow \epsilon$ não são variantes e portanto $\alpha \rightarrow \alpha \rightarrow \beta$ não é tipo de $\boldsymbol{K}$.

O problema da habitação de tipos é de certa forma ortogonal aos dois primeiros. Dada uma base Γ e $\sigma \in T_{\lambda\!\rightarrow}$, não é difícil demonstrar que existe $M \in \Lambda$ tal que $\Gamma \vdash_{\lambda\!\rightarrow} M : \sigma$ se e só se σ, visto como fórmula, é dedutível na lógica proposicional positiva *PROP* a partir de $\{\rho \mid x : \rho \in \Gamma\}$. A decidibilidade das deduções em *PROP*, que está fora do âmbito destas notas, é bem conhecida.

Proposição 7.18 *Decidibilidade do problema da habitação de tipos-$\lambda\!\rightarrow$.*
O problema da habitação de tipos-$\lambda\!\rightarrow$ é decidível.

Atentemos agora no enriquecimento do sistema-$\lambda\!\rightarrow$ com tipos polimórficos. A noção de *polimorfismo*, muito importante nas linguagens de programação modernas, assenta no facto de, por exemplo, $\vdash_{\lambda\rightarrow} \boldsymbol{I} : \sigma \rightarrow \sigma$ para qualquer $\sigma \in T_{\lambda\rightarrow}$. A ideia original de Girard e Reynolds na construção do sistema $\lambda 2$ em 1972-74 (também conhecido por sistema de tipos polimórficos, ou sistema F) foi precisamente a de capturar situações como esta, permitindo atribuir a $\boldsymbol{I}$ o tipo polimórfico $\forall\alpha.\alpha \rightarrow \alpha$.

Definição 7.19 *Tipos-$\lambda 2$.*
O conjunto $T_{\lambda 2}$ dos tipos do sistema $\lambda 2$ é definido indutivamente pelas regras:

variáveis de tipo: $\dfrac{}{\gamma_n}$ espaços de funções: $\dfrac{\sigma \quad \tau}{(\sigma\rightarrow\tau)}$ polimorfismos: $\dfrac{\sigma}{(\forall\gamma_n.\sigma)}$.

É usual adoptar as seguintes convenções de notação:

- $\forall\alpha_1\alpha_2\ldots\alpha_n.\sigma \equiv (\forall\alpha_1.(\forall\alpha_2.\ldots.(\forall\alpha_n.\sigma)\ldots))$;
- $\forall\alpha.\sigma \rightarrow \tau \equiv \forall\alpha.(\sigma \rightarrow \tau) \not\equiv (\forall\alpha.\sigma) \rightarrow \tau$.

O quantificador universal, inexistente em tipos-$\lambda\!\rightarrow$, captura variáveis de tipo da mesma forma que a abstracção lambda captura variáveis. Uma ocorrência de α num tipo-$\lambda 2$ σ diz-se livre se está fora do âmbito de qualquer quantificação $\forall\alpha$. Qualquer ocorrência de α no âmbito de uma quantificação $\forall\alpha$ é diz-se muda. Denotamos por $vtl(\sigma)$ e $vtm(\sigma)$, respectivamente, os conjuntos de variáveis de tipo que ocorrem livres e mudas em σ. Quanto à substituição de variáveis em tipos-$\lambda 2$, $\sigma[\alpha := \tau]$, para evitar a captura indevida das variáveis livres de τ por quantificadores de σ após a substituição, deve exigir-se que τ seja livre para α em σ, ou seja, que nenhuma ocorrência livre de α em σ surja no âmbito de uma quantificação $\forall\beta$ para $\beta \in vtl(\tau)$. Por exemplo, o tipo $\beta \rightarrow \beta$ não é livre para α no tipo $\forall\beta.\beta \rightarrow \alpha$.

Definição 7.20 *Inferência de tipos-$\lambda 2$.*
A relação de inferência de tipos $\vdash_{\lambda 2}$, entre bases e atribuições, é definida indutivamente pelas seguintes regras:

axioma base: $\dfrac{}{\Gamma\vdash_{\lambda 2} x:\sigma}$, $\Gamma(x) \equiv \sigma$

eliminação de $\rightarrow$: $\dfrac{\Gamma\vdash_{\lambda 2} M:(\sigma\rightarrow\tau) \qquad \Gamma\vdash_{\lambda 2} N:\sigma}{\Gamma\vdash_{\lambda 2}(MN):\tau}$

introdução de $\rightarrow$: $\dfrac{\Gamma\cup\{x:\sigma\}\vdash_{\lambda 2} M:\tau}{\Gamma\vdash_{\lambda 2}(\lambda x.M):(\sigma\rightarrow\tau)}$, $x \notin \Gamma$

eliminação de $\forall$: $\dfrac{\Gamma\vdash_{\lambda 2} M:\forall\alpha.\sigma}{\Gamma\vdash_{\lambda 2} M:\sigma[\alpha:=\tau]}$, τ livre para α em σ

introdução de $\forall$: $\dfrac{\Gamma\vdash_{\lambda 2} M:\sigma}{\Gamma\vdash_{\lambda 2} M:\forall\alpha.\sigma}$, $\alpha \notin vtl(\Gamma)$.

Como sempre, se $\emptyset \vdash_{\lambda 2} M : \sigma$ escrevemos simplesmente $\vdash_{\lambda 2} M : \sigma$.

É relativamente simples concluir, por exemplo, que de facto $\vdash_{\lambda 2} \boldsymbol{I} : \forall\alpha.\alpha \rightarrow \alpha$:

$$\dfrac{\dfrac{\{x : \alpha\} \vdash_{\lambda 2} x : \alpha}{\vdash_{\lambda 2} (\lambda x.x) : \alpha \rightarrow \alpha}}{\vdash_{\lambda 2} (\lambda x.x) : \forall\alpha.\alpha \rightarrow \alpha.}$$

Mais interessante é o facto de $(\lambda x.xx)$ ser tipificável-$\lambda 2$, ao contrário do que acontecia em $\lambda\!\rightarrow$:

$$\dfrac{\dfrac{\dfrac{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} x:\forall\alpha.\alpha}{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} x:\beta\to\alpha} \qquad \dfrac{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} x:\forall\alpha.\alpha}{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} x:\beta}}{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} xx:\alpha}}{\{x:\forall\alpha.\alpha\}\vdash_{\lambda 2} xx:\forall\alpha.\alpha}}{\vdash_{\lambda 2} (\lambda x.xx):(\forall\alpha.\alpha)\to(\forall\alpha.\alpha).}$$

Muitas propriedades estruturais importantes de $\lambda\!\rightarrow$ verificam-se ainda em $\lambda 2$. É o caso dos lemas da base, dos subtermos e da substituição. O lema da geração de tipos e os teoremas de redução do sujeito e da normalização forte são mais complexos mas ainda são válidos. Os resultados de decidibilidade em $\lambda 2$ são, no entanto, bem diferentes. De facto, até hoje, os problemas da tipificação de termos e da verificação de tipos estão em aberto. Devido à forte relação entre ambos, sabe-se pelo menos que se o segundo for decidível então também será decidível o primeiro. Por outro lado, o problema da habitação de tipos-$\lambda 2$ está bem estudado e é ... indecidível! Na verdade, mostra-se que um tipo é habitado se e só se corresponde a um teorema da lógica proposicional de segunda-ordem, que se sabe não ser decidível.

O estudo de sistemas de tipos implícitos mais ricos, envolvendo tipos recursivos, intersecção de tipos ou aproximações finitárias, bem como de sistemas de tipos explícitos e do conhecido λ-cubo está para além dos objectivos destas notas.

Exercícios

A. Mostre que para quaisquer tipos σ,τ,ρ se tem:

   (i) $\vdash_{\lambda\rightarrow} \boldsymbol{S}:(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to\sigma\to\rho$;

   (ii) $\vdash_{\lambda\rightarrow} \boldsymbol{SK}:(\sigma\to\tau)\to\sigma\to\sigma$;

   (iii) $\vdash_{\lambda\rightarrow} \boldsymbol{KI}:\tau\to\sigma\to\sigma$.

B. Supondo que $\sigma,\tau\in T_{\lambda\rightarrow}$ e $\alpha\in TVar$, mostre que se $\Gamma\vdash_{\lambda\rightarrow} M:\sigma$ então $\Gamma[\alpha:=\tau]\vdash_{\lambda\rightarrow} M:\sigma[\alpha:=\tau]$.

C. Considere-se $FN\equiv\{M\in\Lambda\mid M \text{ fortemente normalizável}\}$, e para quaisquer $A,B\subseteq\Lambda$ defina-se $A\to B\equiv\{M\in\Lambda\mid \text{se } N\in A \text{ então } MN\in B\}$.

   Um conjunto $X\subseteq FN$ diz-se *saturado* se verifica as seguintes condições, para quaisquer $n\geq 0$, $M_1,\ldots,M_n,N\in FN$ e $x\in Var$:

   - $xM_1\ldots M_n\in X$;
   - se $P[x:=N]M_1\ldots M_n\in X$ então $(\lambda x.P)NM_1\ldots M_n\in X$.

   Seja $SAT\equiv\{X\subseteq FN\mid X \text{ saturado}\}$. Demonstre que:

   (i) $FN\in SAT$;

   (ii) se $A,B\in SAT$ então $A\to B\in SAT$.

D. Mostre que se $\{A_i\}_{i\in I}\subseteq SAT$ então $\bigcap_{i\in I}A_i\in SAT$ (vide exercício 4.1.1.C).

E. A denotação $[\![\sigma]\!]\subseteq\Lambda$ de um tipo $\sigma\in T_{\lambda\rightarrow}$ é definida indutivamente por:

- $[\![\alpha]\!] \equiv FN$;
- $[\![\rho \to \tau]\!] \equiv [\![\rho]\!] \to [\![\tau]\!]$.

Demonstre que $[\![\sigma]\!] \in SAT$ (vide exercício 4.1.1.C).

F. Uma *substituição* é uma função $s : Var \to \Lambda$. Dado $M \in \Lambda$ com $vl(M) \equiv \{x_1, \ldots, x_n\}$, denota-se por Ms o termo $M[x_1 := s(x_1), \ldots, x_n := s(x_n)]$ obtido a partir de M por substituição simultânea das ocorrências livres de cada variável x_i por $s(x_i)$. Usa-se $s \Vdash M : \sigma$ para indicar que $Ms \in [\![\sigma]\!]$, $s \Vdash \Gamma$ para indicar que $s \Vdash x : \Gamma(x)$ para cada $x \in \Gamma$, e $\Gamma \vDash M : \sigma$ para indicar que, qualquer que seja a substituição s, se $s \Vdash \Gamma$ então $s \Vdash M : \sigma$.

Demonstre que se $\Gamma \vdash_{\lambda\to} M : \sigma$ então $\Gamma \vDash M : \sigma$ (vide exercício 4.1.1.E).

G. Demonstre que $\Gamma \vdash_{\lambda\to} M : \sigma$ sse $\vdash_{\lambda\to} (\lambda x_1 \ldots x_n.M) : \sigma_1 \to \cdots \to \sigma_n \to \sigma$, onde $vl(M) \equiv \{x_1, \ldots, x_n\}$ e cada $\sigma_i \equiv \Gamma(x_i)$.

H. Demonstre que se σ, τ são tipos principais de um combinador M então σ, τ são variantes.

I. Demonstre que se s_1, s_2 são umgs de um conjunto de pares P então existem s_1', s_2' tais que $s_1 \equiv s_1' \circ s_2$ e $s_2 \equiv s_2' \circ s_1$.

J. Mostre que $\boldsymbol{KI}(\lambda x.xx)$ não é tipificável-$\lambda\!\to$.

K. Verifique que $\not\vdash_{\lambda\to} \boldsymbol{SK} : \alpha \to \beta \to \beta$.

L. Em $\lambda\!\to$, encontre tipos principais para os combinadores $\boldsymbol{I}$, $\boldsymbol{1}$, $\boldsymbol{B}$ e $\boldsymbol{C}$.

M. A lógica proposicional positiva *PROP* tem como fórmulas os tipos $T_{\lambda\to}$, onde as variáveis de tipo correspondem a símbolos proposicionais e $\to$ ao conectivo de implicação. O conjunto das fómulas dedutíveis em *PROP* a partir de um conjunto de fórmulas Φ é definido indutivamente pelas seguintes regras:

$$\text{hipótese:} \quad \frac{}{\varphi}, \text{ se } \varphi \in \Phi$$

$$\text{axioma 1:} \quad \frac{}{\sigma \to (\tau \to \sigma)}$$

$$\text{axioma 2:} \quad \frac{}{(\sigma \to (\tau \to \rho)) \to ((\sigma \to \tau) \to (\sigma \to \rho))}$$

$$\textit{modus ponens:} \quad \frac{\sigma \to \tau \quad \sigma}{\tau}.$$

Demonstre que existe $M \in \Lambda$ tal que $\Gamma \vdash_{\lambda\to} M : \sigma$ se e só se σ é dedutível em *PROP* a partir de $\Phi \equiv \{\rho \mid x : \rho \in \Gamma\}$.

N. Mostre que se $\vdash_{\lambda\to} M : \sigma$ então também $\vdash_{\lambda 2} M : \sigma$, e verifique que o recíproco não é verdadeiro em geral.

O. Mostre que se tem:

(i) $\vdash_{\lambda 2} \boldsymbol{Verd} : \forall\alpha\beta.\alpha \to \beta \to \alpha$;

(ii) $\vdash_{\lambda 2} \widetilde{n} : \forall\alpha.(\alpha \to \alpha) \to \alpha \to \alpha$, para $n \in \mathbb{N}$ (vide exercício 3.1.1.M);

(iii) $\vdash_{\lambda 2} (\lambda x.xx) : \forall\beta.((\forall\alpha.\alpha) \to \beta)$.

P. Demonstre os lemas da base, da tipificação de subtermos e da substituição para $\lambda 2$.

Q. Supondo que $\sigma, \tau \in T_{\lambda 2}$ e $\alpha \in TVar$, mostre que se $\Gamma \vdash_{\lambda 2} M : \sigma$ então $\Gamma[\alpha := \tau] \vdash_{\lambda 2} M : \sigma[\alpha := \tau]$.

Capítulo 8

Semântica denotacional

A necessidade de estudar não apenas as propriedades operacionais mas também as propriedades funcionais dos programas levou ao estudo detalhado da semântica denotacional do cálculo lambda, que veio a servir de base à semântica denotacional de muitas linguagens de programação. Por semântica denotacional entende-se um mapa $[\![\_]\!]$ que associa a cada programa P a sua denotação $[\![P]\!]$ nalgum domínio apropriado. A base da semântica denotacional das linguagem de programação (também aplicável ao caso do cálculo lambda), comummente adoptado desde a sua publicação por Scott e Strachey no início dos anos 1970, é o princípio da *transparência referencial*. Quer isto dizer que a cada construção θ da linguagem que possa seja usada para compor os programas $P_1, \ldots P_n$ no programa $\theta(P_1, \ldots, P_n)$ deve corresponder uma operação $\widehat{\theta}$ sobre o domínio que verifique a condição

$$[\![\theta(P_1, \ldots, P_n)]\!] \equiv \widehat{\theta}([\![P_1]\!], \ldots, [\![P_n]\!]).$$

Este princípio, por si só, implica imediatamente que o domínio denotacional dos programas deve ser uma *álgebra* para as construções da linguagem. No caso do cálculo lambda, como vimos no Capítulo 4, a única construção fundamental é a aplicação (a abstracção pode sempre ser eliminada).

Definição 8.1 *Estrutura aplicativa e extensionalidade.*
Uma estrutura aplicativa é um par $\mathcal{M} = \langle X, \bullet \rangle$ onde X é um conjunto não vazio e $\bullet$ é uma operação binária em X. Uma estrutura aplicativa diz-se extensional quando, para quaisquer $a, b \in X$, é satisfeita a condição:

$$\text{se } a \bullet c \equiv b \bullet c \text{ para todo o } c \in X \text{ então } a \equiv b.$$

As estruturas aplicativas cujo conjunto de suporte singular dizem-se *triviais*.

Por abuso de notação, passamos a escrever $a_1 \bullet a_2 \bullet a_3 \bullet \cdots \bullet a_n$ para representar $(\ldots((a_1 \bullet a_2) \bullet a_3) \bullet \ldots) \bullet a_n$.

Obviamente, a noção de estrutura aplicativa é muito vaga. Ainda mais se tivermos em conta tudo o que vimos anteriormente. Nomeadamente, o estudo da lógica combinatória sugere a seguinte extensão da noção de estrutura aplicativa.

Definição 8.2 *Álgebra combinatória.*
Uma álgebra combinatória é um quádruplo $\mathcal{M} = \langle X, \bullet, k, s \rangle$ onde $\langle X, \bullet \rangle$ é uma estrutura aplicativa, $k, s \in X$ são elementos distinguidos e, quaisquer que sejam $a, b, c \in X$, são satisfeitas as condições:

$$k \bullet a \bullet b \equiv a \quad \text{e} \quad s \bullet a \bullet b \bullet c \equiv a \bullet c \bullet (b \bullet c).$$

É fácil verificar que uma álgebra combinatória é trivial precisamente quando $k \equiv s$. Claramente, estamos particularmente interessados nas álgebras combinatórias não triviais. Note-se, no entanto, que as álgebras combinatórias não triviais são estruturas algébricas bastante estranhas. Em particular, são sempre não comutativas (em geral, $a \bullet b \not\equiv b \bullet a$), não associativas (em geral, $(a \bullet b) \bullet c \not\equiv a \bullet (b \bullet c)$) e infinitas. Apesar disso, vejamos como podemos interpretar os termos do cálculo lambda numa álgebra combinatória.

Sendo quase imediata a interpretação de termos combinatórios numa álgebra combinatória, basta definir a interpretação de cada termo lambda M como a interpretação do termo combinatório M_{LC} correspondente.

Definição 8.3 *Valoração, interpretação de termos.*
Uma valoração na álgebra combinatória $\mathcal{M} = \langle X, \bullet, k, s\rangle$ é uma função $\rho : Var \to X$.

Dada uma valoração ρ em $\mathcal{M}$, o mapa de interpretação de termos combinatórios $(\_)^{\mathcal{M}}_{\rho} : LC \to X$ é definido indutivamente por:

- $(\mathtt{k})^{\mathcal{M}}_{\rho} \equiv k$;
- $(\mathtt{s})^{\mathcal{M}}_{\rho} \equiv s$;
- $(v_n)^{\mathcal{M}}_{\rho} \equiv \rho(v_n)$;
- $(PQ)^{\mathcal{M}}_{\rho} \equiv (P)^{\mathcal{M}}_{\rho} \bullet (Q)^{\mathcal{M}}_{\rho}$.

Dada uma valoração ρ em $\mathcal{M}$, o correspondente mapa de interpretação de termos lambda $[\![\_]\!]^{\mathcal{M}}_{\rho} : \Lambda \to X$ é definido por:

- $[\![M]\!]^{\mathcal{M}}_{\rho} \equiv (M_{LC})^{\mathcal{M}}_{\rho}$, para cada $M \in \Lambda(\mathcal{M})$.

Podemos agora definir o que significa a satisfação de uma equação pela álgebra combinatória.

Definição 8.4 *Satisfação de equações.*
Diz-se que uma álgebra combinatória $\mathcal{M}$ satisfaz, com uma valoração ρ, uma equação $P = Q$ entre termos combinatórios, o que se denota por $\mathcal{M}, \rho \models P = Q$, se $(P)^{\mathcal{M}}_{\rho} \equiv (Q)^{\mathcal{M}}_{\rho}$. Diz-se que $\mathcal{M}$ satisfaz $P = Q$, o que se denota por $\mathcal{M} \models P = Q$, se $\mathcal{M}, \rho \models P = Q$ para toda a valoração ρ.

Analogamente, diz-se que $\mathcal{M}$ satisfaz com ρ uma equação $M = N$ entre termos lambda, o que se denota por $\mathcal{M}, \rho \models M = N$, se $[\![M]\!]^{\mathcal{M}}_{\rho} \equiv [\![N]\!]^{\mathcal{M}}_{\rho}$. Diz-se também que $\mathcal{M}$ satisfaz $M = N$, o que se denota por $\mathcal{M} \models M = N$, se $\mathcal{M}, \rho \models M = N$ para toda a valoração ρ.

Obviamente, $\mathcal{M}, \rho \models M = N$ se e só se $\mathcal{M}, \rho \models M_{LC} = N_{LC}$, e $\mathcal{M} \models M = N$ se e só se $\mathcal{M} \models M_{LC} = N_{LC}$.

Ainda assim, há muitas álgebras combinatórias desinteressantes, nomeadamente por não satisfazerem sequer as equações $M = N$ tais que $\vdash_{\lambda} M = N$. Eis um exemplo.

Definição 8.5 *Álgebra quociente dos termos combinatórios.*
A álgebra combinatória quociente dos termos combinatórios é

$$\mathcal{M}_{LC} = \langle LC/{=_{LC}}, \bullet, [\mathtt{k}]_{LC}, [\mathtt{s}]_{LC}\rangle,$$

onde:

- $=_{LC}$ é a relação de equivalência definida por $P =_{LC} Q$ se e só se $\vdash_{LC} P = Q$;

- a classe de equivalência de $P \in LC$ é $[P]_{LC} \equiv \{Q \in LC : P =_{LC} Q\}$;
- o conjunto quociente é $LC/=_{LC} \equiv \{[P]_{LC} : P \in LC\}$;
- a operação binária $\bullet$ é definida por $[P]_{LC} \bullet [Q]_{LC} \equiv [PQ]_{LC}$.

Facilmente, tem-se que $\vdash_\lambda \lambda z.(\lambda x.x)z = \lambda x.x$. No entanto, é também verdade que $\mathcal{M}_{LC} \not\models \lambda z.(\lambda x.x)z = \lambda x.x$, já que $\not\vdash_{LC} \mathtt{s(k(skk))(skk)} = \mathtt{skk}$.

De facto, se usarmos também a representação de termos combinatórios verificamos que numa álgebra combinatória, em geral, $(P)^{\mathcal{M}}_\rho \not\equiv [\![P_\lambda]\!]^{\mathcal{M}}_\rho$ para um termo combinatório P. Há portanto que enriquecer algo mais a noção de álgebra combinatória.

Definição 8.6 *Álgebra-λ.*
Uma álgebra combinatória $\mathcal{M}$ diz-se uma álgebra-λ se, para quaisquer $P, Q \in LC$, é satisfeita a condição:

$$\text{se } \vdash_\lambda P_\lambda = Q_\lambda \text{ então } \mathcal{M} \models P = Q.$$

As álgebras-λ podem ser caracterizadas de uma forma mais directa, sem recorrer ao mapa $\_\_{}_\lambda$.

Proposição 8.7 *Caracterização de álgebras-λ.*
Uma álgebra combinatória $\mathcal{M}$ é uma álgebra-λ se e só se são satisfeitas as seguintes três condições:

- *se $\vdash_\lambda M = N$ então $\mathcal{M} \models M = N$, para quaisquer $M, N \in \Lambda$;*
- *$\mathcal{M} \models \boldsymbol{K}_{LC} = \mathtt{k}$;*
- *$\mathcal{M} \models \boldsymbol{S}_{LC} = \mathtt{s}$.*

Pelas razões já apontadas, $\mathcal{M}_{LC}$ não é uma álgebra-λ.

De facto, facilmente se constata que $(P)^{\mathcal{M}}_\rho \equiv [\![P_\lambda]\!]^{\mathcal{M}}_\rho$ em qualquer álgebra-λ. As álgebras-λ formam portanto uma das classes desejáveis de domínios de interpretação para o cálculo lambda. É usual, no entanto, considerar uma classe ainda mais restrita constituída por álgebras mais facilmente manipuláveis. Recorde-se a definição do combinador $\boldsymbol{1} \equiv \lambda xy.xy$. Facilmente se verifica que, para qualquer termo lambda M, se tem $\vdash_\lambda \boldsymbol{1}M = \lambda x.Mx$ e $\vdash_\lambda \boldsymbol{1}(\lambda x.M) = (\lambda x.M)$. Em termos dos combinadores $\boldsymbol{K}$ e $\boldsymbol{S}$ tem-se $\vdash_\lambda \boldsymbol{1} = \boldsymbol{S}(\boldsymbol{K}(\boldsymbol{SKK}))$.

Definição 8.8 *Modelo-λ.*
Um modelo-λ é uma álgebra-λ $\mathcal{M} = \langle X, \bullet, k, s\rangle$ que satisfaz, para quaisquer $a, b \in X$, e tomando $1 \in X$ como abreviatura de $s \bullet (k \bullet (s \bullet k \bullet k))$, a condição:

$$\text{se } a \bullet c = b \bullet c \text{ para todo } c \in X \text{ então } 1 \bullet a = 1 \bullet b.$$

Note-se que em qualquer álgebra-λ $\mathcal{M}$, e para qualquer valoração ρ, $[\![\boldsymbol{1}]\!]^{\mathcal{M}}_\rho \equiv 1$.

É também possível caracterizar precisamente a classe dos modelos-λ, sem recorrer ao combinador $\boldsymbol{1}$. Dada uma valoração $\rho : \mathit{Var} \to X$ e $c \in X$ denota-se por $\rho(x := a)$ a valoração ρ' tal que $\rho'(x) \equiv c$ e $\rho'(y) \equiv \rho(y)$ para $y \not\equiv x$.

Definição 8.9 *Extensionalidade fraca.*
Uma álgebra combinatória $\mathcal{M} = \langle X, \bullet, k, s\rangle$ diz-se fracamente extensional se, para quaisquer $P, Q \in LC$ e qualquer valoração ρ, é satisfeita a condição:

$$\text{se } \mathcal{M}, \rho(x := c) \models P = Q \text{ para todo } c \in X \text{ então } \mathcal{M}, \rho \models \lambda x.P_\lambda = \lambda x.Q_\lambda.$$

De facto a extensionalidade fraca coincide precisamente com a propriedade exigida aos modelos-λ.

Proposição 8.10 *Caracterização de modelos-λ.*
Uma álgebra-λ é um modelo-λ se e só se é fracamente extensional.

Fazendo a ponte para a extensionalidade, recorde-se o combinador $\boldsymbol{I} \equiv \lambda x.x$. É sabido que, em termos dos combinadores $\boldsymbol{K}$ e $\boldsymbol{S}$, se tem $\vdash_\lambda \boldsymbol{I} = \boldsymbol{SKK}$. Na verdade também é imediato verificar que em qualquer álgebra-λ $\mathcal{M}$, e para qualquer valoração ρ, $[\![\boldsymbol{I}]\!]^{\mathcal{M}}_\rho \equiv s \bullet k \bullet k$.

Proposição 8.11 *Caracterização da extensionalidade.*
Uma álgebra-λ é extensional se e só se é um modelo-λ e satisfaz a equação $\boldsymbol{1} = \boldsymbol{I}$.

Será que refazendo, de alguma forma, a álgebra quociente construída em 8.5 é possível encontrar uma álgebra-λ e/ou um modelo-λ?

Definição 8.12 *Álgebra quociente dos termos lambda.*
Seja $\mathcal{T}$ uma teoria-λ. A álgebra combinatória quociente dos termos lambda induzida por $\mathcal{T}$ é

$$\mathcal{M}_\mathcal{T} = \langle \Lambda/{=_\mathcal{T}}, \bullet, [\boldsymbol{K}]_\mathcal{T}, [\boldsymbol{S}]_\mathcal{T} \rangle,$$

onde:

- $=_\mathcal{T}$ é a relação de equivalência definida por $M =_\mathcal{T} N$ se e só se $\mathcal{T} \vdash_\lambda M = N$;
- a classe de equivalência de $M \in \Lambda$ é $[M]_\mathcal{T} \equiv \{N \in \Lambda : M =_\mathcal{T} N\}$;
- o conjunto quociente é $\Lambda/{=_\mathcal{T}} \equiv \{[M]_\mathcal{T} : M \in \Lambda\}$;
- a operação binária $\bullet$ é definida por $[M]_\mathcal{T} \bullet [N]_\mathcal{T} \equiv [MN]_\mathcal{T}$.

Podemos fazer uma construção análoga considerando apenas os termos lambda fechados, ou seja, os combinadores.

Definição 8.13 *Álgebra quociente dos termos lambda fechados.*
Seja $\mathcal{T}$ uma teoria-λ. A álgebra combinatória quociente dos termos lambda fechados induzida por $\mathcal{T}$ é

$$\mathcal{M}^0_\mathcal{T} = \langle \Lambda^0/{=_\mathcal{T}}, \bullet, [\boldsymbol{K}]^0_\mathcal{T}, [\boldsymbol{S}]^0_\mathcal{T} \rangle,$$

onde:

- $[P]^0_\mathcal{T} \equiv [P]_\mathcal{T} \cap \Lambda^0$;
- $\Lambda^0/{=_\mathcal{T}} \equiv \{[P]^0_\mathcal{T} : P \in \Lambda^0\}$;
- a operação binária $\bullet$ é definida por $[P]^0_\mathcal{T} \bullet [Q]^0_\mathcal{T} \equiv [PQ]^0_\mathcal{T}$.

Na verdade, qualquer destas duas estruturas é bastante interessante.

Proposição 8.14 *Álgebras quociente.*
Dada uma teoria-λ $\mathcal{T}$, tem-se:

- *$\mathcal{M}_\mathcal{T}$ é um modelo-λ;*
- *$\mathcal{M}^0_\mathcal{T}$ é uma álgebra-λ;*
- *$\mathcal{M}_\mathcal{T}$ e $\mathcal{M}^0_\mathcal{T}$ são triviais se e só se $\mathcal{T}$ é λ-incoerente;*

- *se $\mathcal{T}$ é uma teoria extensional então $\mathcal{M}_{\mathcal{T}}$ é um modelo-λ extensional;*
- *se $\mathcal{T}$ é tal que $\mathcal{T} \vdash_\lambda M = N$ quando $\mathcal{T} \vdash_\lambda MZ = NZ$ para qualquer $Z \in \Lambda^0$, então $\mathcal{M}^0_{\mathcal{T}}$ é um modelo-λ extensional.*

Estes factos permitem-nos demonstrar o seguinte resultado de adequação das noções de álgebra-λ e modelo-λ.

Proposição 8.15 *Adequação.*
Para $M, N \in \Lambda$, são equivalentes entre si as seguintes condições:

- $\vdash_\lambda M = N$;
- *$M = N$ é satisfeita por todas as álgebras-λ;*
- *$M = N$ é satisfeita por todos os modelos-λ.*

Se $\mathcal{T}$ é um conjunto de equações entre termos lambda, ainda são equivalentes as condições:

- $\mathcal{T} \vdash_\lambda M = N$;
- *$M = N$ é satisfeita por todos os modelos-λ que satisfazem as equações de $\mathcal{T}$.*

Após a construção destas estruturas de carácter puramente sintáctico, põe-se agora a seguinte questão: existirão álgebras-λ e/ou modelos-λ com um conteúdo denotacional mais próximo da intuição? Note-se que há uma dificuldade básica a ser ultrapassada, e que só foi resolvida satisfatoriamente, por Scott, já no final dos anos 1960: no cálculo lambda confundem-se "valores" e "funções" pelo que faz sentido procurar um domínio D tal que $D \simeq D^D$. No entanto, Cantor mostrou que os únicos conjuntos D que verificam esta propriedade são os conjuntos singulares.

Exercícios

A. Mostre que uma álgebra combinatória é trivial se e só se $k \equiv s$.

B. Mostre que qualquer álgebra combinatória comutativa, associativa ou finita é necessariamente trivial.

C. Seja $\mathcal{M}$ uma álgebra combinatória, $M \in \Lambda$ e ρ, ρ' duas valorações tais que $\rho(x) \equiv \rho'(x)$ para qualquer $x \in vl(M)$. Verifique que então $[\![M]\!]^{\mathcal{M}}_\rho \equiv [\![M]\!]^{\mathcal{M}}_{\rho'}$.

D. Sejam $\mathcal{M} = \langle X, \bullet, k, s\rangle$ uma álgebra combinatória. Mostre que para quaisquer $M, N \in \Lambda$ e valoração ρ em $\mathcal{M}$ se tem $[\![M[x := N]]\!]^{\mathcal{M}}_\rho \equiv [\![M]\!]^{\mathcal{M}}_{\rho(x:=[\![N]\!]^{\mathcal{M}}_\rho)}$.

E. Mostre que uma álgebra-λ satisfaz qualquer equação do tipo $P = (P_\lambda)_{LC}$, onde P é um termo combinatório.

F. Verifique que, para qualquer álgebra-λ $\mathcal{M}$ e qualquer valoração ρ, $[\![\mathbf{1}]\!]^{\mathcal{M}}_\rho = 1$ e $[\![\boldsymbol{I}]\!]^{\mathcal{M}}_\rho = s \bullet k \bullet k$.

G. Mostre que $\mathcal{M}_{LC}$, tal como definida em 8.5, é de facto uma álgebra combinatória. Verifique ainda que, para quaisquer $M \in \Lambda^0$ e valoração ρ, se tem $[\![M]\!]^{\mathcal{M}_{LC}}_\rho \equiv [M_{LC}]_{LC}$.

H. Mostre que $\mathcal{M}_{\mathcal{T}}$ e $\mathcal{M}^0_{\mathcal{T}}$, tal como definidas como em 8.12 e 8.13 a partir de uma teoria-λ $\mathcal{T}$, são de facto álgebras combinatórias. Verifique ainda que, para quaisquer $M \in \Lambda^0$ e valoração ρ, se tem $[\![M]\!]^{\mathcal{M}_{\mathcal{T}}}_\rho \equiv [(M_{LC})_\lambda]_{\mathcal{T}}$ e $[\![M]\!]^{\mathcal{M}^0_{\mathcal{T}}}_\rho \equiv [(M_{LC})_\lambda]^0_{\mathcal{T}}$.

Capítulo 9

Teoria de domínios

A solução para a procura de domínios de interpretação para o cálculo lambda passa, de facto, por encontrar um domínio D que possa ser munido de uma estrutura topológica que o torne isomorfo ao conjunto $[D \to D]$ das funções contínuas de D para D, de acordo com essa topologia. Vejamos, em abstracto, o que é um espaço topológico.

Definição 9.1 *Espaço topológico.*
Um espaço topológico é um par $\langle X, \mathcal{O} \rangle$ onde X é um conjunto e $\mathcal{O} \subseteq \wp(X)$ tal que:

- $\emptyset, X \in \mathcal{O}$;
- se $\{O_i\}_{i \in I} \subseteq \mathcal{O}$ então $(\bigcup_{i \in I} O_i) \in \mathcal{O}$;
- se $O_1, O_2 \in \mathcal{O}$ então $O_1 \cap O_2 \in \mathcal{O}$.

Nestas condições é usual dizer-se que $\mathcal{O}$ é uma topologia sobre o conjunto X. Cada conjunto $O \in \mathcal{O}$ diz-se um *aberto* da topologia. As condições significam que $\emptyset$ e X são necessariamente abertos, que uma união arbitrária de conjuntos abertos é ainda um conjunto aberto, e que a intersecção de dois conjuntos abertos também é um conjunto aberto. É usual dizer-se que um conjunto $Y \subseteq X$ é *fechado* se o seu complementar $\overline{Y} = X \setminus Y$ é aberto.

Dado um espaço topológico, define-se a seguinte noção de função *contínua*.

Definição 9.2 *Função contínua entre espaços topológicos.*
Sejam $\langle X_1, \mathcal{O}_1 \rangle$ e $\langle X_2, \mathcal{O}_2 \rangle$ espaços topológicos. Uma função $h : X_1 \to X_2$ diz-se contínua se, para qualquer $O \in \mathcal{O}_2$, se tem $h^{-1}(O) \in \mathcal{O}_1$.

Como é usual, $h^{-1}(O) = \{a \in X_1 : h(a) \in O\}$ é a imagem inversa de O por h. As funções contínuas entre dois espaços topológicos são precisamente aquelas cuja imagem inversa de qualquer conjunto aberto é ainda um conjunto aberto.

Para os fins em vista, estamos particularmente interessados num certo tipo de espaço topológico que se pode definir sobre um conjunto parcialmente ordenado, ou seja, um conjunto munido de uma relação binária reflexiva, transitiva e anti-simétrica.

Definição 9.3 *Conjunto orientado.*
Seja $\langle D, \leq \rangle$ uma ordem parcial. Um conjunto $X \subseteq D$ diz-se orientado se $X \neq \emptyset$ e para quaisquer $a, b \in X$ existe $c \in X$ tal que $a \leq c$ e $b \leq c$.

Para simplificar, no seguimento, referir-nos-emos a uma ordem parcial $\langle D, \leq \rangle$ apenas por D sempre que seja clara qual é a relação de ordem. Recorde-se que, numa ordem parcial D, um elemento $d \in D$ se diz *majorante* de um conjunto $X \subseteq D$ se $a \leq d$ para qualquer $a \in X$. O supremo de X, quando existe, corresponde ao menor dos majorantes de X, ou seja, um elemento $\bigvee X \in D$ majorante de X tal que $\bigvee X \leq d$ para qualquer majorante d de X.

Definição 9.4 *Ordem parcial completa.*
Uma ordem parcial D diz-se completa se são satisfeitas as seguintes condições:

- existe $\bot \in D$ tal que $\bot \leq d$ para qualquer $d \in D$;
- qualquer conjunto orientado $X \subseteq D$ tem supremo $\bigvee X$.

Atente-se na definição da topologia de Scott sobre uma ordem parcial completa.

Definição 9.5 *Abertos de Scott.*
A colecção $\mathcal{O}$ dos abertos de Scott sobre uma ordem parcial completa D é constituída precisamente pelos conjuntos $O \subseteq D$ que satisfazem as seguintes condições:

- se $a \in O$ e $a \leq b$ então $b \in O$;
- se $X \subseteq D$ é um conjunto orientado e $\bigvee X \in O$ então $X \cap O \neq \emptyset$.

Proposição 9.6 *Topologia de Scott.*
Se $\mathcal{O}$ é a colecção de abertos de Scott sobre uma ordem parcial completa D então $\langle D, \mathcal{O} \rangle$ é um espaço topológico.

Daqui em diante, no contexto de uma ordem parcial completa, assumimos sempre a sua topologia de Scott.

A principal razão do interesse da topologia de Scott é a seguinte caracterização de continuidade.

Proposição 9.7 *Continuidade na topologia de Scott.*
Sejam D_1 e D_2 uma ordens parciais completas. Uma função $h : D_1 \to D_2$ é contínua se e só se

$$h(\bigvee_1 X) = \bigvee_2 h(X), \text{ para qualquer } X \subseteq D_1 \text{ orientado.}$$

Como é usual, $h(X) = \{h(a) : a \in X\}$.

Note-se que, como corolário, toda a função contínua é necessariamente monótona, ou seja, se $a \leq_1 b$ então $h(a) \leq_2 h(b)$.

Passemos agora a analisar algumas operações interessantes sobre ordens parciais completas.

Definição 9.8 *Produto de ordens parciais completas.*
Sejam D_1 e D_2 ordens parciais completas. Define-se sobre $D_1 \times D_2$ a relação de ordem $\leq$ tal que $\langle a_1, a_2 \rangle \leq \langle b_1, b_2 \rangle$ se e só se $a_1 \leq_1 b_1$ e $a_2 \leq_2 b_2$.

Proposição 9.9 *Ordem parcial completa produto.*
Se D_1 e D_2 são ordens parciais completas então o seu produto $D_1 \times D_2$ é também uma ordem parcial completa.

Atente-se na seguinte propriedade.

Proposição 9.10 *Continuidade nos argumentos.*
Sejam D_1, D_2 e D ordens parciais completas. Uma função $h : D_1 \times D_2 \to D$ é contínua se e só se, para quaisquer $a \in D_1$ e $b \in D_2$, são contínuas as funções $h(a, \_) : D_2 \to D$ e $h(\_, b) : D_1 \to D$.

Obviamente, $h(a, \_) = \lambda\!\!\lambda d.f(a, d)$ e $h(\_, b) = \lambda\!\!\lambda d.f(d, b)$.

Outra operação interessante consiste na construção do espaço das funções contínuas entre duas ordens parciais completas.

Definição 9.11 *Funções contínuas entre ordens parciais completas.*
Sejam D_1 e D_2 ordens parciais completas. Define-se sobre o conjunto $[D_1 \to D_2]$ das funções contínuas de D_1 para D_2 a relação de ordem $\leq$ tal que $h \leq h'$ se e só se $h(a) \leq_2 h'(a)$ para qualquer $a \in D_1$.

Proposição 9.12 *Ordem parcial completa das funções contínuas.*
Se D_1 e D_2 são ordens parciais completas então o espaço $[D_1 \to D_2]$ das funções contínuas entre elas é também uma ordem parcial completa.

Podemos agora estudar com detalhe a operação de aplicação.

Proposição 9.13 *Continuidade da aplicação.*
Se D_1 e D_2 são ordens parciais completas então é uma função contínua a operação de aplicação Ap: $[D_1 \to D_2] \times D_1 \to D_2$ definida por:

$$Ap(h, a) = h(a), \text{ para } h \in [D_1 \to D_2] \text{ e } a \in D_1.$$

Analisemos também a operação de abstracção.

Proposição 9.14 *Continuidade da abstracção.*
Se D_1, D_2 e D são ordens parciais completas então é uma função contínua a operação de abstracção Abs: $[D_1 \times D_2 \to D] \to [D_1 \to [D_2 \to D]]$ definida por:

$$Abs(h) = \lambda\!\!\lambda a.h(a, \_), \text{ para } h \in [D_1 \times D_2 \to D].$$

Como sabemos, um dos conceitos fundamentais da programação recursiva é a noção de ponto fixo. Como se comportarão as funções contínuas no que diz respeito a esta questão fundamental?

Proposição 9.15 *Teorema de Knaster-Tarski.*
Seja D uma ordem parcial completa. Então:

- *qualquer função $h \in [D \to D]$ tem um ponto fixo;*
- *existe uma função $Fix \in [[D \to D] \to D]$ tal que, para qualquer $h \in [D \to D]$, $Fix(h)$ é o menor ponto fixo de h.*

A capacidade de confundirmos "valores" e "funções" advém da seguinte definição.

Definição 9.16 *Retracção.*
Sejam D_1 e D_2 ordens parciais completas. Então, D_1 diz-se uma retracção de D_2, o que se denota por $D_1 \sqsubseteq D_2$, se existem $G \in [D_1 \to D_2]$ e $F \in [D_2 \to D_1]$ tais que $F \circ G$ é a função identidade em D_1.

Nestas circunstâncias diz-se que D_1 é uma retracção de D_2 com F e G. Verifica-se que, necessariamente, a função F é sobrejectiva e a função G injectiva. Mais ainda, tem-se sempre que $(G \circ F) \circ (G \circ F) = G \circ F$.

Definição 9.17 *Domínio reflexivo.*
Um domínio reflexivo é uma ordem parcial completa D tal que $[D \to D] \sqsubseteq D$.

Se a retracção de D em $[D \to D]$ é com F e G, diz-se também que D é um domínio reflexivo com F e G. Os domínios reflexivos constituem já uma boa base de partida para a construção de modelos-λ.

Definição 9.18 *Modelo-λ induzido por domínio reflexivo.*
O modelo-λ induzido por um domínio reflexivo D com F e G é

$$\mathcal{M}_D = \langle D, \bullet, k, s \rangle,$$

onde:

- $a \bullet b \equiv F(a)(b)$;
- $k \equiv G(\lambda\!\!\lambda a.G(\lambda\!\!\lambda b.a))$;
- $s \equiv G(\lambda\!\!\lambda a.G(\lambda\!\!\lambda b.G(\lambda\!\!\lambda c.(a \bullet c) \bullet (b \bullet c))))$.

Para simplificar a notação, escrevemos $\lambda\!\!\lambda^G d.e$ em vez de $G(\lambda\!\!\lambda d.e)$. Assim, em $\mathcal{M}_D$, tem-se $k \equiv \lambda\!\!\lambda^G a.\lambda\!\!\lambda^G b.a$ e $s \equiv \lambda\!\!\lambda^G a.\lambda\!\!\lambda^G b.\lambda\!\!\lambda^G c.(a \bullet c) \bullet (b \bullet c)$.

Proposição 9.19 *Domínios reflexivos.*
A estrutura $\mathcal{M}_D$ induzida por um domínio reflexivo D é de facto um modelo-λ. Para além disso, $\mathcal{M}_D$ é extensional se e só se $G \circ F$ é a função identidade em $[D \to D]$.

Note-se que se $\mathcal{M}_D$ é extensional isso significa precisamente que $G \equiv F^{-1}$ e portanto $D \simeq [D \to D]$.

É bastante útil considerar também ordens parciais completas com certas propriedades particulares.

Definição 9.20 *Elemento compacto.*
Seja D uma ordem parcial completa. Um elemento $e \in D$ diz-se compacto quando, para qualquer conjunto orientado $X \subseteq D$, é satisfeita a seguinte condição:

$$\text{se } e \leq \bigvee X \text{ então } e \leq a \text{ para algum } a \in X.$$

Definição 9.21 *Domínio algébrico.*
Uma ordem parcial completa D diz-se um domínio algébrico se, para qualquer $d \in D$, o conjunto $\{e \leq d : e \text{ é compacto}\}$ é orientado e $d = \bigvee\{e \leq d : e \text{ é compacto}\}$.

Num domínio algébrico é possível dar uma caracterização local da noção de continuidade.

Proposição 9.22 *Continuidade em domínio algébrico.*
Seja D um domínio algébrico e $h : D \to D$ uma função. Então, h é contínua se e só se

$$h(d) = \bigvee\{h(e) : e \leq d,\ e \text{ é compacto}\}, \text{ para qualquer } d \in D.$$

Põe-se agora a questão de encontrar domínios de interpretação plausíveis para o cálculo-λ. O primeiro modelo que vamos considerar foi proposto por Engeler em 1981, e embora seja interessante, serve apenas de aperitivo ao modelo seguinte, historicamente muito mais importante.

Dado um conjunto A, seja B_A o conjunto definido indutivamente a partir de A pela seguinte regra:

$$\frac{b_1 \ \ldots \ b_n \ \ b}{\langle \{b_1, \ldots, b_n\}, b \rangle}.$$

Definição 9.23 *Construção de Engeler.*
A ordem parcial completa de Engeler sobre o conjunto A é $D_A = \langle \wp(B_A), \subseteq \rangle$.

Proposição 9.24 *Domínio de Engeler.*
Qualquer que seja o conjunto A, a ordem parcial completa de Engeler D_A é um domínio algébrico reflexivo com F e G definidas por:

- $F(X) \equiv \lambda\!\!\lambda Y.\{z \in B_A : \langle Z, z \rangle \in X \ \textit{com} \ Z \subseteq Y\}$;
- $G(h) \equiv \{\langle X, x \rangle \in B_A : x \in h(X)\}$.

Proposição 9.25 *Modelo de Engeler.*
Qualquer que seja o conjunto A, o domínio de Engeler D_A induz um modelo-λ não extensional.

Não sendo muito concreto, o modelo de Engeler acima descrito, contém os ingredientes essenciais do modelo $P\omega$, proposto por Plotkin, que é baseado nos números naturais.

Definição 9.26 *Construção de Plotkin.*
A ordem parcial completa de Plotkin é $P\omega \equiv \langle \wp(\mathbb{N}), \subseteq \rangle$.

É óbvio que $P\omega$ é um domínio algébrico. De facto, toda a essência do modelo $P\omega$ reside no facto de qualquer função contínua sobre $P\omega$ poder ser completamente caracterizada pela sua acção nos conjuntos finitos.

Recorde-se que é possível construir uma bijecção[1] $(\_,\_) : \mathbb{N}^2 \to \mathbb{N}$ por:

- $(n, m) \equiv \frac{1}{2}(n + m)(n + m + 1) + m$.

Analogamente, é possível construir uma bijecção Fin: $\wp_{fin}(\mathbb{N}) \to \mathbb{N}$ por:

- $Fin(\{n_1, \ldots, n_k\}) \equiv \sum_{i=1}^{k} 2^{n_i}$.

No que se segue usamos E_n para referir o único subconjunto de $\mathbb{N}$ tal que $Fin(E_n) \equiv n$.

Proposição 9.27 *Domínio de Plotkin.*
$P\omega$ é um domínio algébrico reflexivo com Fun e Gr definidas por:

- $Fun(X) \equiv \lambda\!\!\lambda Y.\{m \in \mathbb{N} : (n, m) \in X \ \textit{com} \ E_n \subseteq Y\}$;
- $Gr(h) \equiv \{(n, m) : m \in h(E_n)\}$.

Proposição 9.28 *Modelo de Plotkin.*
O domínio de Plotkin $P\omega$ induz um modelo-λ não extensional.

Para obtermos finalmente um modelo extensional vamos recuperar a construção original de Scott, usando limites projectivos.

[1] Usamos esta bijecção, em vez de P tal como definida em 6.18, por ser a originalmente usada por Plotkin. No entanto, todos os resultados apresentados seriam ainda válidos se a bijecção usada fosse P.

Definição 9.29 *Limite projectivo.*
Seja $\{D_n\}_{n\in\mathbb{N}}$ uma sequência de ordens parciais completas e $\{g_n\}_{n\in\mathbb{N}}$ uma sequência de funções tal que $g_i \in [D_{i+1} \to D_i]$ para cada $i \in \mathbb{N}$. O limite projectivo $\lim_{n\to\infty}\langle D_n, g_n\rangle$ é a ordem parcial completa $\langle D_\infty, \leq\rangle$ definida por:

- $D_\infty \equiv \{\langle a_n\rangle_{n\in\mathbb{N}} : g_i(a_{i+1}) \equiv a_i \in D_i \text{ para cada } i \in \mathbb{N}\}$;
- $\langle a_n\rangle_{n\in\mathbb{N}} \leq \langle b_n\rangle_{n\in\mathbb{N}}$ se $a_i \leq_i b_i$ para cada $i \in \mathbb{N}$.

Sequências $\{D_n\}_{n\in\mathbb{N}}$ e $\{g_n\}_{n\in\mathbb{N}}$ tal como acima dizem-se um sistema projectivo. Felizmente, é possível construir um sistema projectivo a partir de qualquer ordem parcial completa D.

Proposição 9.30 *Sistema projectivo normal.*
Seja D uma ordem parcial completa. Então, as sequências $\{D_n\}_{n\in\mathbb{N}}$ e $\{g_n\}_{n\in\mathbb{N}}$ definidas por:

- $D_0 \equiv D$ *e* $D_{n+1} \equiv [D_n \to D_n]$*;*
- $g_0 \equiv \lambda\!\!\lambda h.h(\bot)$ *e* $g_{n+1} \equiv \lambda\!\!\lambda h.g_n \circ h \circ f_n$*,*

onde a sequência $\{f_n\}_{n\in\mathbb{N}}$, tal que $f_i \in [D_i \to D_{i+1}]$ para cada $i \in \mathbb{N}$, é definida por:

- $f_0(d) \equiv \lambda\!\!\lambda a.d$ *e* $f_{n+1} \equiv \lambda\!\!\lambda h.f_n \circ h \circ g_n$*,*

constituem um sistema projectivo, dito normal.

Definição 9.31 *Construção de Scott.*
A ordem parcial completa de Scott sobre a ordem parcial completa D é o limite D_∞ do sistema projectivo normal sobre D.

Proposição 9.32 *Domínio de Scott.*
Qualquer que seja a ordem parcial completa D, a ordem parcial completa de Scott D_∞ é um domínio algébrico reflexivo.

Proposição 9.33 *Modelo de Scott.*
Qualquer que seja a ordem parcial completa D, o domínio de Scott D_∞ induz um modelo-λ extensional.

O estudo dos modelos categoriais do cálculo lambda, baseados em categorias cartesianas fechadas, está já fora do âmbito deste texto.

Exercícios

A. Mostre que o conjunto $\mathbb{R}$ munido da usual definição de conjunto aberto constitui um espaço topológico.

B. Mostre que a noção usual de continuidade para funções reais de variável real coincide com a noção de continuidade baseada na topologia usual sobre $\mathbb{R}$.

C. Sejam $\langle X_1, \mathcal{O}_1\rangle$ e $\langle X_2, \mathcal{O}_2\rangle$ espaços topológicos e $h : X_1 \to X_2$ uma função. Demonstre que h é contínua se e só se $h^{-1}(Y) \subseteq X_1$ é fechado para qualquer $Y \subseteq X_2$ fechado.

D. Mostre que em qualquer ordem parcial $\langle D, \leq\rangle$ existe, no máximo:

- um elemento mínimo $\bot$;
- um elemento $\bigvee X$, para cada $X \subseteq D$.

E. Seja D uma ordem parcial e $X_i \subseteq D$ um conjunto com supremo, para qualquer $i \in I$. Mostre que:

- se para cada $a \in X_i$ existe $b \in X_j$ tal que $a \leq b$ então $\bigvee X_i \leq \bigvee X_j$;
- se $X_i \subseteq X_j$ então $\bigvee X_i \leq \bigvee X_j$;
- se $X \subseteq D$ e existe $\bot \in D$ então X tem supremo sse $X \cup \{\bot\}$ tem supremo, e nesse caso $\bigvee(X \cup \{\bot\}) \equiv \bigvee X$;
- $\bigcup_{i \in I} X_i$ tem supremo sse $\{\bigvee X_i : i \in I\}$ tem supremo, e nesse caso $\bigvee\{\bigvee X_i : i \in I\} \equiv \bigvee(\bigcup_{i \in I} X_i)$.

F. Uma ordem parcial D diz-se um reticulado completo se qualquer conjunto $X \subseteq D$ tem supremo $\bigvee X$. Verifique que:

- todo o reticulado completo é uma ordem parcial completa;
- dado um conjunto X, $\langle \wp(X), \subseteq \rangle$ é um reticulado completo.

G. Seja D uma ordem parcial completa. Então, para qualquer $d \in D$, o conjunto $U_d = \{a \in D : a \not\leq d\}$ é um aberto de Scott.

H. Um espaço topológico $\langle X, \mathcal{O} \rangle$ diz-se:

- T_0 se para quaisquer $a, b \in X$ com $a \not\equiv b$ existe $O \in \mathcal{O}$ tal que $\{a, b\} \cap O \neq \emptyset$ mas $\{a, b\} \not\subseteq O$;
- T_1 se para quaisquer $a, b \in X$ com $a \not\equiv b$ existem $O_a, O_b \in \mathcal{O}$ tais que $\{a, b\} \cap O_a = \{a\}$ e $\{a, b\} \cap O_b = \{b\}$.

Mostre que se D é uma ordem parcial completa então a sua topologia de Scott define um espaço T_0 que, em geral, não é T_1.

I. Mostre que de facto, numa ordem parcial completa, toda a função contínua é monótona. O recíproco será verdadeiro?

J. Mostre que se D é um domínio extensional com F e G então, para quaisquer $M \in \Lambda$ e valoração ρ, tem-se $[\![\lambda x.M]\!]_\rho^{\mathcal{M}_D} \equiv \lambda\!\!\lambda^G d.[\![M]\!]_{\rho(x:=d)}^{\mathcal{M}_D}$.

K. Mostre que, para qualquer conjunto X, $\langle \wp(X), \subseteq \rangle$ é um domínio algébrico onde os elementos compactos são os conjuntos finitos. Supondo que $X \equiv \mathbb{N}$, determine ainda o ponto fixo mínimo da função $\Phi : \wp(\mathbb{N}) \to \wp(\mathbb{N})$ tal que $\Phi(A) \equiv \{0\} \cup \{n + 2 : n \in A\}$.

L. Mostre que num domínio algébrico D os conjuntos abertos de Scott são precisamente os que resultam de uniões arbitrárias de conjuntos da forma $\{d \in D : e \leq d\}$ com $e \in D$ um elemento compacto.

M. Demonstre que o produto de domínios algébricos é ainda um domínio algébrico.

N. Verifique que as aplicações $(\_,\_) : \mathbb{N}^2 \to \mathbb{N}$ e Fin: $\wp_{fin}(\mathbb{N}) \to \mathbb{N}$ definidas acima são de facto bijecções.

O. Calcule em $P\omega$ e em D_A, para A arbitrário, as denotações dos combinadores $\boldsymbol{I}$, $\boldsymbol{K}$ e $\boldsymbol{S}$.

Apêndice A

Indução por Regras

Seja U um conjunto.

Definição A.1 *Regra.*
Uma regra é um par $\langle X, u\rangle$ onde $X \subseteq U$ e $u \in U$.

É usual representar uma regra $\langle X, u\rangle$ por

$$\frac{X}{u}.$$

Se $X = \{x_1, \ldots, x_n\}$ é usual representar a regra por

$$\frac{x_1 \ldots x_n}{u}.$$

Se $X = \emptyset$ é usual representar a regra simplesmente por

$$\frac{}{u}.$$

No que se segue supomos fixo um conjunto R de regras.

Definição A.2 *Fecho para regras.*
Um conjunto $I \subseteq U$ diz-se fechado para R quando, qualquer que seja a regra $\frac{X}{u} \in R$, se $X \subseteq I$ então também $u \in I$.

Definição A.3 *Definição indutiva por regras.*
O conjunto definido indutivamente por R é o resultado de intersectar todos os conjuntos $J \subseteq U$ fechados para R.

Lema A.4 *Definição indutiva versus fecho.*
Seja I o conjunto definido indutivamente por R. Então:

- *I é fechado para R;*
- *se $J \subseteq U$ é fechado para R então $I \subseteq J$.*

Logo, o conjunto definido indutivamente por R é caracterizado precisamente por ser o menor subconjunto de U fechado para R. Também se usa, por vezes, a designação de *conjunto definido indutivamente por R a partir de $I_0 \subseteq U$* para referir o menor conjunto fechado para R que contém I_0, e que pode ser alternativamente descrito como o conjunto definido indutivamente por $R \cup R_0$ onde $R_0 = \{\frac{}{i} : i \in I_0\}$.

É bem conhecida uma caracterização mais construtiva das definições indutivas no caso particular de as regras serem *finitárias*.

Definição A.5 *Regra finitária.*
Uma regra $\frac{X}{u}$ diz-se finitária se X é um conjunto finito.

Proposição A.6 *Definição indutiva por regras finitárias.*
Seja I o conjunto definido indutivamente por R. Se todas as regras de R são finitárias então $I = \bigcup_{n\in\mathbb{N}} I_n$ onde $I_0 = \emptyset$ e, para cada $n \in \mathbb{N}$, se define:

- $I_{n+1} = I_n \cup \{u : X \subseteq I_n$ *para alguma regra* $\frac{X}{u} \in R\}$.

Seja $\mathcal{P}$ uma propriedade de U, ou seja, uma função $\mathcal{P} : U \to \{0, 1\}$.

Definição A.7 *Propagação de propriedade.*
Diz-se que R propaga $\mathcal{P}$ quando, qualquer que seja a regra $\frac{X}{u} \in R$, se $\mathcal{P}(x) = 1$ para cada $x \in X$ então também $\mathcal{P}(u) = 1$.

Proposição A.8 *Princípio de Indução por Regras.*
Seja I o conjunto definido indutivamente por R. Se R propaga $\mathcal{P}$ então $\mathcal{P}(i) = 1$ para qualquer $i \in I$.

Note-se que uma regra $\frac{}{u}$ propaga $\mathcal{P}$ se e só se $\mathcal{P}(u) = 1$. Assim, é usual separar a verificação uma prova por indução nas regras em duas partes:

- *base de indução*: regras do tipo $\frac{}{u} \in R$;
- *passo de indução*: regras do tipo $\frac{X}{u}$ com $X \neq \emptyset$.

Exercícios

A. Seja U um conjunto. Recorde que uma relação binária $\Re \subseteq U \times U$ se diz:

- *reflexiva* se $\langle u, u\rangle \in \Re$ para qualquer $u \in U$;
- *simétrica* se $\langle u_1, u_2\rangle \in \Re$ implica $\langle u_2, u_1\rangle \in \Re$;
- *transitiva* se $\langle u_1, u_2\rangle, \langle u_2, u_3\rangle \in \Re$ implica $\langle u_1, u_3\rangle \in \Re$;
- *equivalência* se é reflexiva, simétrica e transitiva.

Considere as seguintes regras, onde $u, u_1, u_2, u_3 \in U$ são elementos arbitrários:

$$\text{reflexividade:} \quad \frac{}{\langle u,u\rangle}$$

$$\text{simetria:} \quad \frac{\langle u_1,u_2\rangle}{\langle u_2,u_1\rangle}$$

$$\text{transitividade:} \quad \frac{\langle u_1,u_2\rangle \quad \langle u_2,u_3\rangle}{\langle u_1,u_3\rangle}.$$

Mostre que:

(i) $\Re^{=}$, definida indutivamente pela regra da reflexividade a partir de $\Re$, é a menor relação reflexiva que estende $\Re$ (*fecho reflexivo* de $\Re$);

(ii) $\Re^{+}$, definida indutivamente pela regra da transitividade a partir de $\Re$, é a menor relação transitiva que estende $\Re$ (*fecho transitivo* de $\Re$);

(iii) $\Re^{*} = (\Re^{=})^{+}$ é a menor relação reflexiva e transitiva que estende $\Re$ (*fecho reflexivo e transitivo* de $\Re$);

(iv) a relação definida indutivamente pelas regras da reflexividade, simetria e transitividade a partir de $\Re$ é a menor relação de equivalência que estende $\Re$ (*relação de equivalência gerada por* $\Re$).

B. Seja $\Re$ uma relação binária e $\Re^*$ o seu fecho reflexivo e transitivo. Mostre que se $\langle u, u' \rangle \in \Re^*$ então existe uma sequência finita $u_1, \ldots, u_n \in U$ tal que:

- $u_1 = u$ e $u_n = u'$;
- $\langle u_1, u_2 \rangle, \langle u_2, u_3 \rangle, \ldots, \langle u_{n-1}, u_n \rangle \in \Re$.

C. O conjunto dos números naturais $\mathbb{N}$ é caracterizado, por exemplo sobre $U = \mathbb{R}$, por ser definido indutivamente pelas regras $\frac{}{0}$ e $\frac{x}{x+1}$. Mostre que os seguintes conjuntos de regras também definem indutivamente os naturais:

(i) $\frac{}{0}$, $\frac{}{1}$ e $\frac{x}{x+2}$;

(ii) $\frac{}{0}$ e $\frac{0 \quad 1 \quad \ldots \quad x-1 \quad x}{x+1}$.

D. Mostre que o usual princípio de indução matemática

$$\text{se}$$
$$\mathcal{P}(0) = 1$$
$$\text{e}$$
$$\mathcal{P}(n) = 1 \text{ implica } \mathcal{P}(n+1) = 1 \text{ para cada } n \in \mathbb{N}$$
$$\text{então}$$
$$\mathcal{P}(n) = 1 \text{ para qualquer } n \in \mathbb{N}$$

é um caso particular do princípio de indução por regras.

E. Mostre que o usual princípio de indução completa

$$\text{se}$$
$$\mathcal{P}(0) = 1$$
$$\text{e}$$
$$\mathcal{P}(0) = \ldots = \mathcal{P}(n) = 1 \text{ implica } \mathcal{P}(n+1) = 1 \text{ para cada } n \in \mathbb{N}$$
$$\text{então}$$
$$\mathcal{P}(n) = 1 \text{ para qualquer } n \in \mathbb{N}$$

é um caso particular do princípio de indução por regras.

F. Mostre por indução em $n \in \mathbb{N}$ as seguintes propriedades:

(i) $\sum_{k=0}^{n} k = \frac{n(n+1)}{2}$;

(ii) $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$, com $x \neq 1$;

(iii) $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$, onde $\binom{n}{k} = \frac{n!}{k!(n-k)!}$;

(iv) $(-1)^n = \begin{cases} 1 & \text{se } n \text{ é par} \\ -1 & \text{se } n \text{ é ímpar} \end{cases}$;

(v) se $n \geq 8$ então existem $u, v \in \mathbb{N}$ tais que $n = 3u + 5v$;

(vi) a sucessão de Fibonacci, definida indutivamente por $Fib(0) = 0$, $Fib(1) = 1$ e $Fib(k+2) = Fib(k) + Fib(k+1)$, é tal que $Fib(n) = \frac{\phi^n - \psi^n}{\sqrt{5}}$, onde $\phi = \frac{1+\sqrt{5}}{2}$ e $\psi = \frac{1-\sqrt{5}}{2}$ são as duas raízes da equação $x^2 = x + 1$.