11/94

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

NÚMEROS NATURAIS PARCIAIS

por

Martín Hötzel Escardó

Dissertação submetida como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dalcídio Moraes Cláudio
Orientador

Prof. Antônio Carlos da Rocha Costa
Co-orientador

Porto Alegre, 29 de janeiro de 1993

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Escardó, Martín Hötzel

Números naturais parciais/Martín Hötzel Escardó. - Porto Alegre: CPGCC da UFRGS, 1992.

161p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-graduação em Ciência da computação, Porto Alegre, 1992. Orientador: Dalcídio Moraes Cláudio. Co-orientador: Antônio Carlos da Rocha Costa.

Dissertação: objetos parciais e infinitos, teoria dos domínios de Scott, teoria da recursão, cálculo-λ tipado, processos perpétuos, interação, concorrência.

PREFÁCIO

Esta dissertação compreende uma parte da pesquisa relativa ao plano de mestrado em Ciência da Computação do autor. O projeto inicial foi o estudo da manipulação computacional de objetos infinitos (listas, conjuntos, números reais etc.), mas finitários. Cedo se reconheceu que a noção de objeto parcial serve de fundamentação para este estudo. A noção de algoritmo precisou ser generalizada, permitindo-se que algoritmos tivessem computações infinitas ao computar objetos infinitos ou parciais. Estas computações infinitas são significativas, no sentido que produzem resultados parciais, em quantidades finitas de passos, que convergem para o resultado que computam. Para que o trabalho pudesse ser desenvolvido no ámbito da teoria da recursão (também chamada teoria da computação), se decidiu primeiro generalizar a teoria da recursão através da noção de número natural parcial. Este trabalho aborda esta generalização. Também se reconheceu que objetos parciais e infinitos podem ser utilizados para modelar computações que envolvem interação, possivelmente perpétua, de um usuário com um agente computacional. Assim, esta generalização aproximaria a teoria da recursão da prática da Ciência da Computação.

Esta investigação não teria sido possível sem o estímulo e a colaboração de Vanderlei Rodrigues, A.C. da Rocha Costa, Dalcídio M. Cláudio e Benedito M. Acióly.

Este trabalho foi parcialmente financiado pela IBM do Brasil e pela minha esposa Daniele. Além do financiamento, a minha esposa forneceu as condições apropriadas para que o trabalho pudesse ser realizado.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Os números naturais parciais são informações parciais sobre números naturais e ω. As propriedades matemáticas do domínio de números naturais parciais e de funções que envolvem este domínio são estudadas via o uso da teoria dos domínios de Scott. A manipulação formal de naturais parciais é estudada mediante a utilização de um cálculo-λ tipado com constantes. Relações com a teoria da recursão são estudadas. É mostrado como funções contínuas entre naturais parciais podem representar processos interativos, possivelmente perpétuos.

PALAVRAS-CHAVE. Objetos parciais e infinitos. Teoria dos domínios de Scott. Teoria da recursão. Cálculo-λ tipado. Processos perpétuos. Interação. Concorrência.

TITLE: "Partial natural numbers"

ABSTRACT

Partial natural numbers are informations about natural numbers and ω. Mathematical properties of the domain of partial natural numbers and functions involving this domain are investigated with the aid of Scott domain theory. A typed λ-calculus is introduced for investigating formal manipulation of partial natural numbers. Relations with recursion theory are investigated. It is shown how continuous functions on natural numbers can represent (possibly perpetual) interactive processes.

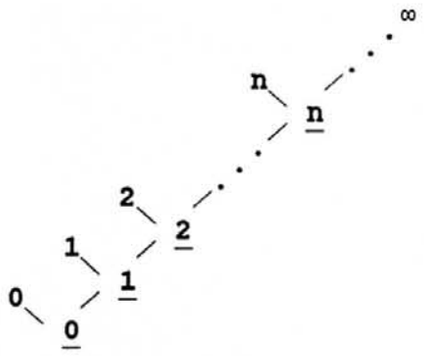
KEY-WORDS. Partial and infinite objects. Scott domain theory. Recursion theory. Typed λ-calculus. Perpetual processes. Interaction. Concurrence.

1 INTRODUÇÃO

Este trabalho estuda a noção de número natural parcial. Os números naturais parciais são às vezes referidos como *lazzy natural numbers* na literatura de língua inglesa [PAU 87] [GIR 89]. Os números naturais parciais podem ser vistos como casos particulares de reais parciais, definidos e estudados em [ACI 91] com o propósito de fundamentar computação com números reais.

As propriedades matemáticas dos números naturais parciais são analisadas. A teoria dos domínios de Scott [PLO 80] [SCO 82] [SCO 82*a*] é utilizada como ferramenta matemática para esta análise. A manipulação formal de números parciais é estudada através de um cálculo-λ tipado com constantes [PLO 77], e os resultados obtidos são comparados com os resultados da teoria da computação clássica [KLE 52] [ROG 67] [MAC 78]. Em particular, o uso de objetos parciais permite representar processos interativos, incluindo processos perpétuos, por funções, e representar operações entre processos, como interação, por operações entre funções [TUD 87].

O domínio de números parciais está representado na figura 1.1.



O domínio de naturais parciais

fig. 1.1

Os números parciais aparecem espontaneamente em diversos contextos, como teoria dos domínios de Scott [SCO 89], programação funcional [BIR 88], teoria dos tipos [GIR 89], lógica da computação [PAU 87], tipos de dados [SMY 82], análise dos algoritmos primitivos recursivos [COL 91]. Porém, até onde o autor tem conhecimento, o números parciais foram apenas apresentados, e nunca estudados. Suas potencialidades, aparentemente, nunca foram expostas. O propósito deste trabalho é justamente mostrar as potencialidades dos números parciais com relação ao estudo matemático de processos computacionais, principalmente em ocasiões onde o mais importante não é a determinação de computabilidade.

A teoria da computação não é um fundamento da Ciência da Computação, mas um ramo da lógica formal. De fato, a teoria da computação tem como objetivo formalizar a noção de decidibilidade [TUR 36], e, portanto, a noção de procedimento efetivo (ver [COS 91*a*]). Ainda, como é sabido, a teoria da computação tampouco é fundamento da *prática* da Ciência da Computação, onde ocorrem noções importantes como interação, que não estão no escopo da teoria da computação (ver [COS 91*b*] [ESC 91*a*]).

Os números parciais generalizam os números

naturais, estabelecendo um elo com a teoria da computação, e ao mesmo tempo possibilitando a noção de interação, estabelecendo um elo com a Ciência da Computação (ver [COS 91*a*]).

O capítulo **2** introduz informalmente e depois matematicamente a noção de número natural parcial, e estuda as suas propriedades. O capítulo **3**, analogamente, introduz informalmente e depois matematicamente a noção de função contínua de números parciais, e estuda as suas propriedades. O capítulo **4** estuda uma série de funções e classes de funções contínuas básicas necessárias para os demais capítulos. O capítulo **5** estuda a manipulação formal de números parciais. Uma linguagem formal, denominada Π, e uma noção de computabilidade para esta linguagem são introduzidas e estudadas. O capítulo **6** fornece ferramentas para a definição de funções nesta linguagem. O capítulo **7** mostra que as funções recursivas [DAV 58] [ROG 67] são definíveis em Π. O capítulo **8** aritmetiza a linguagem Π, permitindo que termos desta linguagem se refiram, via números de Gödel [ROG 67], a termos desta linguagem. O capítulo **9** apresenta relações com a teoria da recursão. O capítulo **10** apresenta as conclusões, problemas em aberto e trabalhos futuros.

▷ Definições e convenções são formadas por parágrafos precedidos pelo símbolo "▷". O início destes parágrafos é precedido por menos espaços que os demais, como é ilustrado pelo presente parágrafo. As definições não são numeradas, e são referidas, quando necessário, pelo número da seção ou subseção em que ocorrem. Os termos introduzidos são escritos em *itálico*.

▷ Lemas, teoremas e proposições que ocorrem na seção ou subseção x são numerados consecutivamente por $x.n$. Se um lema, teorema ou proposição que tem um corolário toma

numeração *x.n*, o corolário toma a mesma numeração *x.n*. Se há mais de um corolário, eles tomam consecutivamente as numerações *x.n.m*.

▷ O término de uma prova é assinalado pelo símbolo "□"

▷ *sss* abrevia "se e somente se".

2 NÚMEROS NATURAIS PARCIAIS

A seção 2.1 motiva a noção de número parcial, através de processos de contagem. A seção 2.2 discute o papel de números parciais em computações. A seção 2.3 introduz matematicamente a noção de número parcial. As seções 2.4 e 2.5 estudam as propriedades básicas dos números parciais. A seção 2.4 apresenta os números parciais como domínio de Scott. A seção 2.5 caracteriza os números parciais por axiomas semelhantes aos axiomas de Peano. Dois princípios de indução para números parciais são estabelecidos.

2.1 Números parciais e processos de contagem

Os números parciais estão associados a processos que se dão por passos cumulativos. A seguir, eles são introduzidos intuitivamente através de uma análise dos processos de contagem.

▷ Dado um aglomerado (conjunto, lista, etc.) de objetos A, uma regra de seleção de objetos de A e uma propriedade de objetos P, uma *contagem* de A é um processo que, por exame sucessivo e exaustivo dos objetos de A, via a regra de seleção, determina quantos objetos de A satisfazem a propriedade P.

▷ A regra de seleção é dita *justa* se cada objeto de A é selecionado após um número finito de seleções.

A regra de seleção pode ser justa somente se A tem uma quantidade enumerável, finita ou infinita, de objetos.

▷ Como uma contagem é um processo de enumeração, se assume que o aglomerado A é enumerável.

É implícito que o processo de contagem está associado a um método de verificar a propriedade P.

▷ Um objeto para o qual pode ser estabelecido que ele satisfaz a propriedade P é dito *P-objeto*, um objeto para o qual pode ser estabelecido que ele não satisfaz a propriedade P é dito um *¬P-objeto*, e um objeto para o qual não pode ser estabelecido se ele satisfaz ou não a propriedade (sendo irrelevante se de fato ele a satisfaz ou não) é dito *P?-objeto*.

▷ O processo de contagem procede por passos, do seguinte modo. Em cada passo, caso haja objetos em A, um objeto é selecionado e retirado, concreta ou virtualmente. Se um P-objeto é retirado, se *conta* "há mais um P-objeto". Se um ¬P-objeto ou P?-objeto é retirado, nenhuma ação é tomada. Se não há objetos em A, e se em nenhum passo anterior um P?-objeto foi retirado, se *conta* "não há mais P-objetos". O último passo ocorre, se ocorre, quando é constatado que A não contém (mais) objetos.

▷ "Há mais um P-objeto" e "não há mais P-objetos" são ditos *ações de contagem*, e são representados respectivamente pelos símbolos $\$$ e $\}$. A seqüência de ações de contagem *produzidas* pelo processo é dita *contagem* de A, onde a produção de uma ação de contagem é indicada pela palavra *contar* na descrição do processo de contagem.

Se A é finito com n P-objetos, e não há P?-objetos em A, então a contagem é, claramente, a seqüência $\$^n\}$. No mesmo caso, mas se há P?-objetos, a contagem é a seqüência

s^n. Se A contém infinitos objetos, mas uma quantidade finita n de P-objetos, e a regra de seleção é justa, então a contagem é a seqüência s^n, também. No mesmo caso, se a regra de seleção não é justa, a contagem é alguma seqüência s^m, para $m \leq n$. Se A contém infinitos objetos e infinitos P-objetos, a contagem é a seqüência infinita s^ω, se a regra de seleção é justa. Aqui ω é o primeiro cardinal infinito, e s^ω denota a seqüência infinita de ações s.

Destas contagens, as da forma $s^n\}$, correspondem, então, a processos de contagem de A que têm sucesso, em um número finito de passos, na determinação da quantidade de P-objetos de A. Uma contagem $s^n\}$ revela que A tem "exatamente n" P-objetos. Contagens da forma s^n correspondem a processos que têm um sucesso parcial nesta determinação. Uma contagem s^n revela que A tem "no mínimo n" P-objetos. A contagem s^ω corresponde a sucesso, mas em um número infinito de passos. Para cada natural n, é possível, mediante a observação de uma parte finita de s^ω, saber que a quantidade de P-objetos de A não é n. Porém, não é possível, mediante a observação de uma parte finita de s^ω, saber que a quantidade de P-objetos de A é ω. Na hipótese de que A é enumerável, a contagem completa s^ω revela que A tem "exatamente ω" P-objetos.

As observações acima motivam a seguinte definição provisória, a ser tornada rigorosa nas seções 2.3 e 2.4.

▷ A informação completa "exatamente n" é representada pelo número parcial $\mathbf{n}$. A informação parcial "no mínimo n" é representada pelo número parcial $\underline{\mathbf{n}}$. A informação completa "exatamente ω" é representada pelo número parcial ∞. Os números parciais são todos de uma das formas $\mathbf{n}, \underline{\mathbf{n}}$ ou ∞.

A contagem s^0 é a seqüência vazia, e está associada a processos de contagem que não produzem ações de contagem.

Esta contagem veicula a informação nula $\underline{\mathbf{0}}$. A nulidade desta informação vem do fato de que a sua interpretação, "no mínimo 0", não exclui nenhum natural nem ω.

▷ A *informação veiculada por uma contagem* c é *info*(c), onde a função *info* que mapeia contagens para informações é definida por:

- $info(s^n\wr) = \mathbf{n}$
- $info(s^n) = \underline{\mathbf{n}}$
- $info(s^\omega) = \infty$

Se $info(c) = \mathbf{x}$, também se diz que c é contagem de **x**. A função *info* é bijetiva. Sua inversa é escrita *cont*.

Uma contagem da forma s^n também pode ser vista como uma parte inicial de uma contagem $s^m\wr$, s^m ou s^ω, para $m \geq n$. Às vezes, uma parte inicial de uma contagem veicula informação suficiente para o propósito considerado, e a contagem não precisa ser desenvolvida integralmente. A relação de parte inicial é a relação de *prefixo* entre seqüências, e a ela corresponde uma relação entre informações. Quanto mais é desenvolvida uma contagem, mais informação ela veicula. A relação de "menos informação ou igual" entre números parciais é escrita $\leq$, e satisfaz:

- $\mathbf{x} \leq \mathbf{y}$ sss *cont*(**x**) é prefixo de *cont*(**y**)

Note que se considera que uma seqüência é prefixo de si mesma.

A relação $\leq$ é definida na seção 2.4. Ela é dita, também, relação de *aproximação*, por que o processo de expandir sucessivamente uma contagem em direção ao seu limite pode ser visto como um processo de aproximação.

2.2 Computações de números parciais

O conjunto de funções recursivas totais tem a propriedade de não poder ser enumerado por uma função recursiva total. Isto impede que se desenvolva um formalismo efetivo que consiga expressar as funções recursivas totais, e apenas elas. A solução adotada na teoria da recursão é a introdução da noção de função parcial [ROG 67]. Uma função parcial modela algoritmos que, para certas entradas, não produzem saída, por terem computação infinita.

A teoria dos domínios de Scott modela funções parciais $\mathbb{N}^k \to \mathbb{N}$ mediante o acréscimo de um valor $\perp$ aos naturais, que representa a indefinição, obtendo o domínio $\mathbb{N}_\perp$. Uma função parcial $f:\mathbb{N}^k \to \mathbb{N}$ é representada por uma função total $\hat{f}:\mathbb{N}_\perp^k \to \mathbb{N}_\perp$, definida por:

- $\hat{f}(x_1,\ldots,x_k)=f(x_1,\ldots,x_k)$ se $x_1,\ldots,x_k \in \mathbb{N}$ e f é definida para $(x_1,\ldots,x_k)$
- $\hat{f}(x_1,\ldots,x_k)=\perp$ caso contrário

O valor $\perp$ é interpretado como ausência de informação. A teoria da computação não permite manipular informação incompleta e, em particular, informação nula. Deste modo, as funções $\hat{f}:\mathbb{N}_\perp^k \to \mathbb{N}_\perp$ que representam funções parciais $f:\mathbb{N}^k \to \mathbb{N}$ são *estritas*, i.e.:

- $\hat{f}(x_1,\ldots,x_k)=\perp$ se $x_1=\perp$ ou ... ou $x_k=\perp$

Porém, a teoria dos domínios de Scott admite funções não estritas, como o *condicional* $cond:\mathbb{N}_\perp^3 \to \mathbb{N}_\perp$, definido por:

- $cond(0,y,z)=z$
- $cond(x+1,y,z)=y$
- $cond(\perp,y,z)=\perp$

O condicional é não estrito no segundo e terceiro argumento

porque $cond(0,\perp,z)=z$, e $cond(1,y,\perp)=y$. Ainda, é possível definir o condicional *paralelo* $pcond:\mathbb{N}_\perp^3\to\mathbb{N}_\perp$, não estrito no primeiro argumento (ver [PLO 77]), por:

- $pcond(0,y,z)=z$
- $pcond(x+1,y,z)=y$
- $pcond(\perp,y,z)=\perp$ se $y\neq z$
- $pcond(\perp,y,y)=y$

Em $\mathbb{N}_\perp$, as funções de um argumento não estritas são as funções constantes, pois as funções admitidas são as que satisfazem:

- $f(\perp)=y$ implica $f(x)=y$, exceto se $y=\perp$.

Como caso particular de função estrita, a soma satisfaz, $\perp+1=\perp$. Mas, sob certo ponto de vista, $\perp+1$ pode ser visto como a informação incompleta "diferente de 0" (que é equivalente a "no mínimo 1", e por sua vez é denotada por $\underline{1}$), pois qualquer número natural, mesmo que se tenha informação nula sobre ele, acrescido de 1 é distinto de 0. Sob este ponto de vista, funções estritas não são adequadas. Ainda, o domínio $\mathbb{N}_\perp$ não tem informações incompletas suficientes, como "diferente de 0", para dar conta destes casos.

Assim como processos de contagem infinitos podem produzir informação incompleta não *trivial*, i.e., não nula, computações infinitas também podem. Neste trabalho, computações infinitas (de números) computam um dos números parciais $\underline{0},\underline{1},\underline{2},\ldots,\underline{n},\ldots,\infty$. O número $\underline{0}$ corresponde a $\perp$. Intuitivamente, alguns dos passos de uma computação são indentificados como passos que contam "há mais um" ou "não há mais".

Do mesmo modo, algoritmos podem manipular informação de entrada incompleta, e a informação de entrada pode ser dada por um processo análogo ao de contagem. Assim, computações infinitas podem aparecer intermediariamente em

algoritmos que produzem computações finitas, como ilustrado a seguir.

Pode ocorrer o caso em que **f(x)** tenha computação infinita, mas **g(f(x))** tenha computação finita. Isto acontece, por exemplo, se **f(x)=ω** e **g** é a função que vale **1** se o seu argumento é distinto de **7**, e **0**, caso contrário. A informação incompleta $\underline{8} \leq$ **f(x)**, gerada em algum passo da computação de **f(x)**, é suficiente para realizar esta comparação, que produz a informação **1**.

Para computar **g(f(x))**, então, não é possível computar *primeiro* **y=f(x)** e *depois* **g(y)**. Isto ocorre por que **f(x)** pode ter computação infinita, e a computação de **g(y)** poderia ser postergada indefinidamente, impedindo que informação não nula fosse produzida. Explícita ou implicitamente, então, quando se computa **g(f(x))**, tem de haver alguma forma de *paralelismo* na computação de **y** e **g(y)**, e algum modo de *interação* entre estas computações. Assim, a operação de composição de funções pode ser interpretada como uma operação de *concorrência* de algoritmos.

Intuitivamente, a computação de **f(x)** pode ser vista como um processo que, dada uma contagem de **x**, produz uma contagem de **y**, e a computação de **g(y)**, como um processo que, dada uma contagem de **y**, produz uma contagem de **g(y)**. Neste modelo intuitivo, há tres contagens simultâneas na computação de **g(f(x))**: a de **x**, a de **f(x)** e a de **g(f(x))**. Os algoritmos que computam **y=f(x)** e **g(y)** interagem pela contagem de **y**.

2.3 O conjunto $\mathbb{P}$ de números parciais

Na seção 2.1, os números parciais foram vistos como informações. Uma informação corresponde a um predicado. Por

exemplo, a informação $\underline{\mathbf{n}}$ corresponde ao predicado ser maior ou igual a n. Um predicado, por sua vez, pode ser representado pela sua *extensão*, i.e., o conjunto de elementos para o qual ele é verdadeiro. Neste caso, uma informação completa corresponde a um conjunto unitário. Isto motiva as definições que seguem.

▷ O conjunto dos naturais 0,1,2,... é escrito $\mathbb{N}$. As variáveis n,m e às vezes as variáveis x,y,z, correm sobre $\mathbb{N}$. A cardinalidade de $\mathbb{N}$ é escrita ω. Os símbolos $<$, $\leq$, $>$, $\geq$ denotam as relações de ordem de magnitude usuais em $\mathbb{N}\cup\{\omega\}$.

▷ Dado um natural n, os *números parciais* $\mathbf{n}$, $\underline{\mathbf{n}}$ e ∞ são definidos por:

- $\mathbf{n}=\{n\}$
- $\underline{\mathbf{n}}=\{m\in\mathbb{N}\,|\,m\geq n\}\cup\{\omega\}$
- $\infty=\{\omega\}$

O número parcial $\mathbf{n}$ é lido "exatamente n". O número parcial $\underline{\mathbf{n}}$ é lido "no mínimo n".

▷ O conjunto $\mathbb{P}$ de números parciais é definido por:

- $\mathbb{P}=\mathbb{M}\cup\underline{\mathbb{M}}\cup\{\infty\}$

onde:

- $\mathbb{M}=\{\mathbf{n}\,|\,n\in\mathbb{N}\}$
- $\underline{\mathbb{M}}=\{\underline{\mathbf{n}}\,|\,n\in\mathbb{N}\}$

As variáveis $\mathbf{x},\mathbf{y},\mathbf{z}$ correm sobre $\mathbb{P}$. As variáveis $\mathbf{n},\mathbf{m}$ correm sobre $\mathbb{M}$.

▷ O símbolo $(\_)$ também é utilizado para denotar a função $\mathbb{P}\to\mathbb{P}$, definida por:

- $\underline{\mathbf{x}}=\mathbf{x}$ se $\mathbf{x}\in\underline{\mathbb{M}}\cup\{\infty\}$
  $=\underline{\mathbf{n}}$ se $\mathbf{x}=\mathbf{n}\in\mathbb{M}$

O conjunto de chegada desta função é escrito $\underline{\mathbb{P}}$. Este conjunto

satisfaz:

• $\underline{\mathbb{P}}=\underline{\mathbb{M}}\cup\{\infty\}$ • $\mathbb{P}=\mathbb{M}\cup\underline{\mathbb{P}}$

Se os elementos de $\mathbb{M}$ correspondem aos naturais, $\underline{\mathbb{P}}$ corresponde a números acrescentados aos naturais.

▷ O conjunto $\mathbb{F}$ de *números finitos* é definido por:

• $\mathbb{F}=\mathbb{M}\cup\underline{\mathbb{M}}$

Um conjunto $a\subseteq\mathbb{N}\cup\{\omega\}$ não vazio corresponde a uma informação, mas esta informação não necessariamente faz parte de $\mathbb{P}$ (por exemplo, $a=\{2,4\}$). Neste caso, é possível trabalhar com a menor informação em $\mathbb{P}$ que inclui a (para o exemplo acima esta informação é $\underline{2}=\{2,3,4,\ldots,\omega\}\supseteq\{2,4\}$). A operação de fecho definida a seguir determina tal elemento de $\mathbb{P}$.

▷ O *fecho*-$\mathbb{P}$ de um conjunto $a\subseteq\mathbb{N}\cup\{\omega\}$ não vazio é o elemento $\mathcal{C}_{\mathbb{P}}(a)\in\mathbb{P}$ definido por:

• $\mathcal{C}_{\mathbb{P}}(a)=\bigcap\{\mathbf{x}\in\mathbb{P}\,|\,\mathbf{x}\supseteq a\}$

Se $a\in\mathbb{P}$, é imediato que $\mathcal{C}_{\mathbb{P}}(a)=a$.

2.4 Estrutura domínio-teorética de $\mathbb{P}$

▷ A relação de *aproximação*, ou de *ordem de informação*, $\leq\subseteq\mathbb{P}\times\mathbb{P}$ é a relação de inclusão reversa, i.e.:

• $\mathbf{x}\leq\mathbf{y}$ sss $\mathbf{x}\supseteq\mathbf{y}$

Claramente $\leq$ é uma relação de ordem parcial. Esta ordem parcial está representada na figura 1.1.

Intuitivamente, quanto mais elementos uma informação contém, menos ela informa. O caso extremo é $\underline{0}$, que contém todos os números naturais e ω, e, portanto, inclui

todas as informações de $\mathbb{P}$, i.e., para todo $\mathbf{x}\in\mathbb{P}$, $\underline{0}\sqsubseteq x$. Assim, $\mathbf{x} \leq \mathbf{y}$ pode ser lido "**x** informa menos que **y**, ou tanto quanto **y**".

O símbolo $\leq$ não deve ser confundido com o símbolo $\leq$. O primeiro denota a relação de ordem (parcial) de informação em $\mathbb{P}$. O segundo denota a relação de ordem de magnitude em $\mathbb{N}$. A relação $\leq$ não é definida em $\mathbb{P}$, embora em certas notações (cf. seções 6.6 e 6.7) o símbolo $\leq$ seja utilizado para indicar a ordem de magnitude de elementos de $\mathbb{N}$ dados por informações em $\mathbb{P}$.

▷ Um número parcial é dito *total* sss ele é maximal para a relação de aproximação, e *propriamente parcial* sss ele não é total.

Os números totais correspondem a informações completas, e, conforme o lema 2.4.1 a seguir, são os da forma **n** e ∞, i.e., os números parciais que são conjuntos unitários.

Lema 2.4.1. Para todos **n**,**m**,**x**:

- $\mathbf{n} \leq \mathbf{x}$ sss $\mathbf{x}=\mathbf{n}$
- $\underline{\mathbf{n}} \leq \mathbf{x}$ sss $\mathbf{x}=\mathbf{m}$ e $n\leq m$ ou $\mathbf{x}=\underline{\mathbf{m}}$ e $n\leq m$ ou $\mathbf{x}=\infty$
- $\infty \leq \mathbf{x}$ sss $\mathbf{x}=\infty$

Prova. Direta da definição de $\mathbb{P}$ e $\leq$. □

Corolário 2.4.1. $\underline{\mathbf{0}} \leq \mathbf{x}$ para todo **x**. □

O lema 2.4.1 e o corolário que dele decorre nada mais são do que a expressão matemática da figura 1.1. mostrada na introdução.

▷ A expressão $\mathbf{x} \leq \mathbf{y}$ é lida, também, "**x** é uma parte de **y**".

Com esta leitura, um número é propriamente parcial quando ele é uma parte estrita de outro elemento. As partes de **n** são $\underline{0},\underline{1},\underline{2},\ldots,\underline{n},n$. As partes de $\underline{n}$ são $\underline{0},\underline{1},\underline{2},\ldots,\underline{n}$. As partes de ∞ são $\underline{0},\underline{1},\underline{2},\ldots,\underline{n},\ldots,\infty$. Logo, os números propriamente parciais são os da forma $\underline{n}$.

A seguir é introduzida a noção de domínio de Scott [PLO 80], e é mostrado que $\mathbb{P}$, com a relação de ordem $\leq$, é domínio de Scott. Isto permite que a teoria dos domínios de Scott seja utilizada como ferramenta matemática para o estudo de números parciais.

▷ Um subconjunto X de um conjunto D parcialmente ordenado por $\leq$ é *dirigido* sss ele é não vazio e todo par de elementos de X tem um majorante em X.

Intuitivamente, um conjunto dirigido é um conjunto de *partes* de um mesmo objeto. Partes que podem ser partes de um mesmo objeto são ditas *consistentes*. Por exemplo, $\underline{3}$ e $\underline{7}$, ou $\underline{3}$ e **7** são consistentes, mas **3** e **7**, ou **3** e $\underline{7}$ são inconsistentes.

▷ Uma ordem parcial $(D,\leq)$ é *completa* (cpo) sss D tem um menor elemento $\perp$ e todo subconjunto X de D dirigido tem um menor majorante, escrito $\bigsqcup X$, dito *supremo* de X.

Intuitivamente, a operação de supremo *junta* as partes de um mesmo objeto. Uma ordem parcial é cpo quando não *faltam* objetos, i.e., quando ela é fechada para o operação de juntar partes consistentes.

▷ Um elemento d de um cpo $(D,\leq)$ é *finito* sss sempre que $d \leq \bigsqcup X$ para um conjunto dirigido X, $d \leq e$ para algum $e \in X$.

Intuitivamente, um objeto é finito quando ele é uma

parte elementar. O elemento ω não é finito para $(\mathbb{P},\leq)$, pois $\omega=\bigsqcup\{\underline{n}\,|\,n\in\mathbb{N}\}$, e ω não é parte de $\underline{n}$ para $n\in\mathbb{N}$ algum.

▷ Um cpo $(D,\leq)$ é *algébrico* sss, para cada $x\in D$, o conjunto $\{d\leq x\,|\,d$ é finito$\}$ é dirigido e seu supremo é x.

Intuitivamente, um cpo é algébrico quando todo objeto, finito ou infinito, é formado apenas por partes finitas.

▷ Um cpo $(D,\leq)$ é *consistentemente completo* sss sempre que $x,y\in D$ têm um majorante, eles têm supremo.

Intuitivamente, se x e y têm majorante, eles são consistentes, por ambos serem parte de seu majorante. Um cpo é consistentemente completo quando quaisquer dois elementos consistentes formam um objeto, $x\sqcup y$, sem o auxílio de mais objetos. Neste caso, o objeto $x\sqcup y$ é parte de qualquer majorante de x e y, i.e., parte de qualquer objeto do qual x e y são parte.

▷ Um conjunto B é uma *base* para um cpo $(D,\leq)$ sss, para cada x em D, o supremo do conjunto $\{b\leq x\,|\,b\in B\}$ é x.

Intuitivamente, uma base de um cpo tem partes suficientes para formar todos os elementos do cpo, pela operação de supremo. É imediato que os elementos finitos de um cpo algébrico formam uma base, e que qualquer base de um cpo algébrico contém os elementos finitos do cpo.

▷ Um *domínio de Scott* é um cpo algébrico consistentemente completo com base enumerável. No que segue do texto, D,E correm sobre domínios de Scott.

Intuitivamente, computar um elemento de um domínio

de Scott é gerar, sucessivamente e em seqüência, partes do elemento, de modo que toda parte do elemento seja gerada em algum passo. A existência de uma base enumerável garante que isto seja possível.

Teorema 2.4.2. $(\mathbb{P},\leq)$ é domínio de Scott, com menor elemento $\underline{0}$, conjunto de elementos finitos $\mathbb{F}$ e base enumerável $\mathbb{F}$.

Prova. Os lemas de 2.4.3 a 2.4.7 a seguir estabelecem o teorema. □

Lema 2.4.3. $(\mathbb{P},\leq)$ é cpo.

Prova. O menor elemento de $\mathbb{P}$ é $\underline{0}$. Seja $X\subseteq\mathbb{P}$ um conjunto dirigido. (1) Se existe um elemento de X maximal em $\mathbb{P}$, então este elemento é o supremo de X. (2) Caso contrário, $X\subseteq\underline{\mathbb{M}}$, e há dois subcasos. (2.1) Existe $\underline{\mathbf{n}}\in X$ tal que, para todo $\mathbf{x}\in X$, $\mathbf{x}\leq\underline{\mathbf{n}}$. Neste caso, o supremo é $\underline{\mathbf{n}}$. (2.2) Caso contrário, para todo $\underline{\mathbf{n}}\in X$, existe $\underline{\mathbf{m}}\in X$ com $\underline{\mathbf{n}}\leq\underline{\mathbf{m}}$. Isto é, nenhum elemento de X majora X, e nenhum elemento de $\underline{\mathbb{M}}$ majora X. Analogamente, nenhum elemento de $\mathbb{M}$ majora X. Como todos os elementos de X estão em $\underline{\mathbb{M}}$, e ∞ majora $\underline{\mathbb{M}}$, então ∞ majora X. Conseqüentemente, ∞ é o único majorante de X, e portanto o seu supremo. Logo, X tem supremo. □

Lema 2.4.4. $\mathbb{F}$ é o conjunto de elementos finitos de $\mathbb{P}$.

Prova. O conjunto $\mathbf{x}^{\downarrow}=\{\mathbf{y}\,|\,\mathbf{y}\leq\mathbf{x}\}$, para $\mathbf{x}\in\mathbb{F}$, é finito (se $\mathbf{x}=\mathbf{n}$ então $\mathbf{x}^{\downarrow}$ tem $2n+2$ elementos, se $\mathbf{x}=\underline{\mathbf{n}}$ então $\mathbf{x}^{\downarrow}$ tem $2n+1$ elementos). Se o supremo de um conjunto X é $\mathbf{x}\in\mathbb{P}$, claramente $X\subseteq\mathbf{x}^{\downarrow}$. Logo, se o supremo de um conjunto X é $\mathbf{x}\in\mathbb{F}$, X é finito. Como o supremo de um conjunto finito X está em X, se $\mathbf{x}\in\mathbb{F}$ então $\mathbf{x}$ é finito. O único número parcial que não está em $\mathbb{F}$ é ∞. Este elemento é infinito, pois o supremo de $\underline{\mathbb{M}}$ é ∞ e não existe $\mathbf{x}\in\underline{\mathbb{M}}$ tal que $\infty\leq\mathbf{x}$. Logo, se $\mathbf{x}\notin\mathbb{F}$, então $\mathbf{x}$ não é finito. □

Lema 2.4.5. $(\mathbb{P},\leq)$ é algébrico.

Prova. Se $\mathbf{x}$ é finito, $\mathbf{x}\in\{d \leq \mathbf{x} \mid d \text{ é finito}\}$, e portanto o supremo deste conjunto é $\mathbf{x}$. Caso contrário, $\mathbf{x}=\infty$, e $\{d \leq \infty \mid d \text{ é finito}\}=\underline{\mathbb{M}}$, cujo supremo é ∞. □

Lema 2.4.6. $(\mathbb{P},\leq)$ é consistentemente completo.

Prova. As equações abaixo mostram como obter o supremo de dois números, para todos os casos possíveis, quando eles têm majorante:

- $\mathbf{n} \sqcup \mathbf{m}=\mathbf{n}$ sss $\mathbf{n}=\mathbf{m}$ (Se $\mathbf{n}\neq\mathbf{m}$ não existe majorante)
- $\mathbf{n} \sqcup \underline{\mathbf{m}}=\mathbf{n}$ sss $m\leq n$ (Se $m>n$ não existe majorante)
- $\mathbf{n} \sqcup \infty$ não existe ($\mathbf{n}$ e ∞ não têm majorante)
- $\underline{\mathbf{n}} \sqcup \mathbf{m}=\mathbf{m}$ sss $n\leq\mathbf{m}$ (Se $n>m$ não existe majorante)
- $\underline{\mathbf{n}} \sqcup \underline{\mathbf{m}}=\underline{\mathbf{k}}$ sss $k=max(n,m)$
- $\underline{\mathbf{n}} \sqcup \infty=\infty$
- $\infty \sqcup \mathbf{n}$ não existe (∞ e $\mathbf{n}$ não têm majorante)
- $\infty \sqcup \underline{\mathbf{n}}=\infty$
- $\infty \sqcup \infty=\infty$. □

Em termos de contagens, dois números são consistentes se a contagem de um é prefixo da contagem do outro, e o seu supremo tem a contagem mais longa das duas.

Lema 2.4.7. $\mathbb{F}$ é base de $\mathbb{P}$, e $\mathbb{F}$ é enumerável.

Prova. A enumerabilidade de $\mathbb{F}$ é imediata. Se $\mathbf{x}\in\mathbb{F}$, $\mathbf{x}=\bigsqcup\{b \leq \mathbf{x} \mid b\in\mathbb{F}\}$, pois um dos valores para b é $\mathbf{x}$. Caso contrário, $\mathbf{x}=\infty$, e $\{b \leq \infty \mid b\in\underline{\mathbb{F}}\}=\mathbb{M}$, cujo supremo é ∞. □

O lema acima conclui a prova do teorema 2.4.2. A seguinte proposição caracteriza a operação de supremo de um modo uniforme.

Proposição 2.4.8. Um conjunto não vazio $X\subseteq\mathbb{P}$ é dirigido sss $\bigcap X$ é não vazio, e, neste caso, o supremo de X é $\bigcap X$.

Prova. Similar à prova do lema 2.4.9 adiante. □

▷ Em uma ordem parcial $(D,\leq)$, o *ínfimo* de um conjunto $X\subseteq D$ é o seu maior minorante, escrito $\bigsqcap X$, quando existe.

Em termos de contagens, a operação de ínfimo de números parciais corresponde à operação de máximo prefixo comum. Assim, o ínfimo de um conjunto é a melhor informação compatível com todas as informações do conjunto.

Lema 2.4.9. Qualquer subconjunto X não vazio de $\mathbb{P}$ tem ínfimo, e este ínfimo é o fecho-$\mathbb{P}$ de $\bigcup X$.

Prova. A prova utiliza implicitamente o lema 2.4.1. Sejam $X_1=\{\mathbf{n}\in\mathbb{M} \mid \mathbf{n}\in X\}$, $X_2=\{\mathbf{x}\in\underline{\mathbb{P}} \mid \mathbf{x}\in X\}$. Então X é a união disjunta de X_1 e X_2. Se X_1 é vazio, então X_2 é não vazio, e tem um menor elemento $\mathbf{x}$, pois $\underline{\mathbb{P}}$ é uma cadeia. Logo, para todo $\mathbf{x}'\in X_2$, $\mathbf{x}\geq\mathbf{x}'$, e $\mathbf{x}=\bigcup X=\mathcal{C}_{\mathbb{P}}(\bigcup X)$. Se $X_1=\{\mathbf{n}\}$ e X_2 é vazio, então o ínfimo de X é $\mathbf{n}=\bigcup X=\mathcal{C}_{\mathbb{P}}(\bigcup X)$. Se X_1 tem mais de um elemento e X_2 é vazio ou $\{\infty\}$, seja o menor natural n (para a ordem dos naturais) tal que $\mathbf{n}\in X_1$. Então o ínfimo de X é $\underline{\mathbf{n}}=\mathcal{C}_{\mathbb{P}}(\bigcup X)\neq\bigcup X$. Caso contrário, X_1 e X_2 são não vazios, e $X_2\neq\infty$. Tome os menores naturais n e m tais que $\mathbf{n}\in X_1$, $\underline{\mathbf{m}}\in X_2$. Neste caso o ínfimo é $\underline{\mathbf{k}}$, onde $k=min(n,m)$, e $\underline{\mathbf{k}}=\mathcal{C}_{\mathbb{P}}(\bigcup X)\neq\bigcup X$, pois $X\neq\mathbb{P}$. □

O lema 2.4.10 a seguir mostra como obter o ínfimo de dois elementos por casos.

Lema 2.4.10. Se $k=min(n,m)$, então:

- $\mathbf{n}\sqcap\mathbf{m} = \mathbf{k}$ se $\mathbf{n}=\mathbf{m}=k$
  $= \underline{\mathbf{k}}$ se $\mathbf{n}\neq\mathbf{m}$
- $\underline{\mathbf{n}}\sqcap\mathbf{m} = \mathbf{m}\sqcap\underline{\mathbf{n}} = \underline{\mathbf{n}}\sqcap\underline{\mathbf{m}} = \underline{\mathbf{k}}$
- $\infty\sqcap\mathbf{x} = \mathbf{x}\sqcap\infty = \underline{\mathbf{x}}$

Prova. Aplicação direta do lema 2.4.1, ou do lema 2.4.9. □

▷ Dados domínios de Scott $D_1,\ldots,D_k$, ordenados respectivamente por $\leq_1,\ldots,\leq_k$, a ordenação *coordenada-a-coordenada* $\leq$ em $D_1\times\ldots\times D_k$ é definida por:

- $(x_1,\ldots,x_k) \leq (x'_1,\ldots,x'_k)$ sss $x_1 \leq_1 x_1',\ldots,x_k \leq_k x'_k$

Lema 2.4.11. Sejam $D_1,\ldots,D_k$ domínios de Scott, com menores elementos $\perp_1,\ldots,\perp_k$, conjuntos de elementos finitos $F_1,\ldots,F_k$ e bases $B_1,\ldots,B_k$ respectivamente. Então $D_1\times\ldots\times D_k$ ordenado coordenada-a-coordenada é domínio de Scott, com menor elemento $(\perp_1,\ldots,\perp_k)$, conjunto de elementos finitos $F_1\times\ldots\times F_k$ e base $B_1\times\ldots\times B_k$.

Prova. Ver [PLO 80]. □

▷ No que segue do texto, qualquer produto cartesiano de domínios é utilizado implicitamente com a ordenação coordenada-a-coordenada.

2.5 Estrutura número-teorética de $\mathbb{P}$

▷ A função *sucessor* **succ**:$\mathbb{P}\to\mathbb{P}$ é definida por:

- **succ(n)**=**m** se $m=n+1$
- **succ(**$\underline{\mathbf{n}}$**)**=$\underline{\mathbf{m}}$ se $m=n+1$
- **succ**$(\infty)=\infty$

A expressão **succ(x)** também é escrita **x+1.**

A função **succ** é a contrapartida natural, em um sentido matemático preciso definido na seção 3.4, da função sucessor $\mathbb{N}\to\mathbb{N}$, $n\mapsto n+1$. Intuitivamente, $\mathfrak{x}$ é uma contagem de **x** sss $s\mathfrak{x}$ é uma contagem de **succ(x)**, i.e., a contagem de A veicula informação **x** sss a contagem de $A\cup\{a\}$, para $a\notin A$, veicula informação **x+1**, quando a regra de seleção é justa (cf. seção 2.1).

Uma conseqüência imediata desta definição é que:

- $\mathbf{n}=\mathbf{succ}^n(\mathbf{0})$
- $\underline{\mathbf{n}}=\mathbf{succ}^n(\underline{\mathbf{0}})$
- $\infty=\mathbf{succ}^k(\mathbf{x})$ sss $\mathbf{x}=\infty$, para todo $k\in\mathbb{N}$

O seguinte teorema mostra como construir axiomaticamente os números parciais.

Teorema 2.5.1.

- $\mathbf{0},\underline{\mathbf{0}},\infty$ são elementos distintos de $\mathbb{P}$
- $\mathbb{P}$ é fechado para a operação sucessor, i.e.:
  - Se $\mathbf{x}\in\mathbb{P}$, então $\mathbf{succ}(\mathbf{x})\in\mathbb{P}$
- $\mathbf{0}$ e $\underline{\mathbf{0}}$ não são sucessores, i.e.:
  - $\mathbf{0}\neq\mathbf{succ}(\mathbf{x})\neq\underline{\mathbf{0}}$
- O único sucessor de si próprio é ∞, i.e.:
  - $\mathbf{succ}(\mathbf{x})=\mathbf{x}$ sss $\mathbf{x}=\infty$
- **succ** é injetiva, i.e.,
  - $\mathbf{succ}(\mathbf{x})=\mathbf{succ}(\mathbf{y})$ implica $\mathbf{x}=\mathbf{y}$
- $\mathbb{P}$ é o menor conjunto que contém $\mathbf{0},\underline{\mathbf{0}},\infty$ e é fechado para a operação sucessor (indução em $\mathbb{P}$, ou indução forte):
  - Se
    - $A\subseteq\mathbb{P}$
    - $\mathbf{0},\underline{\mathbf{0}},\infty\in A$
    - A é fechado para a operação sucessor,

    então
    - $A=\mathbb{P}$
- $\mathbf{0},\underline{\mathbf{0}}\in\mathbb{F}$ e $\mathbb{F}$ é fechado para a operação sucessor
- $\mathbb{F}$ é o menor conjunto que contém $\underline{\mathbf{0}},\mathbf{0}$ e é fechado para a operação sucessor (indução em $\mathbb{F}$ ou indução *fraca*)

Prova. A prova é rotineira, e é omitida. A propriedade de indução em $\mathbb{P}$ ($\mathbb{F}$) é provada por indução em $\mathbb{N}$ nos casos de

$\mathbb{P}$ ($\mathbb{F}$). □

A indução em $\mathbb{P}$ é utilizada para provar propriedades de números parciais do seguinte modo. Dada um propriedade P de números parciais, para provar que $P(\mathbf{x})$ para todo **x**, provar:

- $P(\mathbf{0})$, $P(\underline{\mathbf{0}})$, $P(\infty)$
- $P(\mathbf{x})$ implica $P(\mathbf{succ}(\mathbf{x}))$

A indução fraca é utilizada analogamente para provar propriedades em $\mathbb{F}$.

O lema 2.5.2 a seguir mostra que é possível definir funções e realizar provas por casos $\underline{\mathbf{0}}$,**0** e **x+1**, ao invés dos casos $\underline{\mathbf{n}}$,**n** e ∞.

Lema 2.5.2. Todo $\mathbf{x}\in\mathbb{P}$ é de uma das formas:

- $\underline{\mathbf{0}}$ ou
- **0** ou
- **succ**(**x**′) para um único $\mathbf{x}'\in\mathbb{P}$

Prova. Imediata. □

▷ A função *predecessor* **pred**:$\mathbb{P}\to\mathbb{P}$ é definida por:

- **pred**($\underline{\mathbf{0}}$)=$\underline{\mathbf{0}}$
- **pred**(**0**)=**0**
- **pred**(**succ**(**x**))=**x**

A expressão **pred**(**x**) também é escrita **x-1**.

É imediato que esta função satisfaz:

- **succ**(**pred**(**x**))=**x** sss $\mathbf{x}\neq\underline{\mathbf{0}}$ e **x**≠**0**

O lema 2.5.3 abaixo caracteriza a operação de ínfimo por casos $\underline{\mathbf{0}}$, **0**, **x+1**.

Lema 2.5.3. Para todos **x**,**y**:

- $\underline{0} \sqcap \mathbf{x} = \mathbf{x} \sqcap \underline{0} = \underline{0}$
- $\mathbf{0} \sqcap \mathbf{0} = \mathbf{0}$
- $\mathbf{0} \sqcap (\mathbf{x+1}) = (\mathbf{x+1}) \sqcap \mathbf{0} = \underline{0}$
- $(\mathbf{x+1}) \sqcap (\mathbf{y+1}) = (\mathbf{x} \sqcap \mathbf{y})\mathbf{+1}$

Prova. Aplicação direta do lema 2.4.10 e da definição da função sucessor, por casos $\mathbf{n}$,$\underline{\mathbf{n}}$ e ω em **x**. □

3 FUNÇÕES CONTÍNUAS

Este capítulo introduz e estuda a noção de função contínua entre domínios de Scott (em particular, $\mathbb{P}$ e $\mathbb{P}^k$). A seção 3.1 motiva a noção de continuidade para funções entre números parciais. A seção 3.2 apresenta a noção de continuidade e as propriedades fundamentais das funções contínuas. A seção 3.3 mostra que funções contínuas $\mathbb{P}^k \to \mathbb{P}$ podem ser vistas como informações sobre funções parciais $\mathbb{N}^k \to \mathbb{N}$. A seção 3.4 define modos de estender funções parciais $\mathbb{N}^k \to \mathbb{N}$ para funções contínuas $\mathbb{P}^k \to \mathbb{P}$.

3.1 Mapeamentos entre contagens

Esta seção motiva intuitivamente a noção matemática de continuidade para funções de números parciais, e continua a discussão iniciada na seção 2.1.

▷ Um *mapeamento entre contagens* é um processo que produz uma contagem em função de uma contagem observada. A contagem observada é dita contagem de *entrada*; a produzida, contagem de *saída*. Como contagens podem ser infinitas ou inacabadas, as ações de contagem de entrada e saída ocorrem simultaneamente, ou entrelaçadas. Considera-se que a contagem de saída depende apenas da contagem de entrada, não dependendo, por exemplo, do entrelaçamento particular entre as ações de entrada e saída. No caso geral, pode haver mais de uma contagem de entrada.

Por exemplo, o processo que multiplica por 2 a contagem de entrada pode ser descrito do seguinte modo. Para

cada s de entrada, produzir dois s de saída. Se ocorre ς na entrada, produzir ς na saída. Note que o entrelaçamento não é especificado.

▷ Uma função $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ *especifica abstratamente* um mapeamento entre contagens sss para toda contagem de entrada ξ que produz contagem de saída η, $\mathbf{f}(info(\xi))=info(\eta)$.

É imediato que a função que especifica abstratamente o processo descrito acima é $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ tal que:

- $\mathbf{f(n)=m}$ se $m=2n$
- $\mathbf{f}(\underline{\mathbf{n}})=\underline{\mathbf{m}}$ se $m=2n$
- $\mathbf{f}(\infty)=\infty$

Como outro exemplo, considere o processo descrito a seguir. Seja $k\in\mathbb{N}$ fixo. Se s^{k+1} é prefixo da contagem de entrada, ou a contagem de entrada é $s^{k}\varsigma$, produzir s na saída. Se a contagem de entrada é $s^{j}\varsigma$, para $j<k$, produzir ς na saída. A função $\mathbf{g}:\mathbb{P}\to\mathbb{P}$ que especifica este processo satisfaz:

- $\mathbf{g(n)}=\underline{\mathbf{1}}$ se $n\geq k$
  $=\mathbf{0}$ se $n<k$
- $\mathbf{g}(\underline{\mathbf{n}})=\underline{\mathbf{1}}$ se $n>k$
  $=\underline{\mathbf{0}}$ se $n\leq k$
- $\mathbf{g}(\infty)=\underline{\mathbf{1}}$

Intuitivamente, $\mathbf{g(x)}$ realiza a comparação $\mathbf{x\geq k}$. A resposta $\underline{\mathbf{1}}$ significa "sim", a resposta $\mathbf{0}$ significa "não", e a resposta $\underline{\mathbf{0}}$ significa "não sei". Mas, ao invés de dizer explicitamente "não sei", o processo se limita a não responder, causando o mesmo efeito. Assim, uma interpretação para $\underline{\mathbf{0}}$ é informação nula por ausência de resposta.

Estes dois processos, intuitivamente, podem ser realizados.

Uma questão importante é:

- Quais são as condições mínimas <u>necessárias</u> para que uma função $\mathbb{P}^k \to \mathbb{P}$ seja realizável, sem consideração das limitações que decorram de uma análise de processos de cálculo (computabilidade)?

Esta questão é discutida em detalhe em [ESC 91]. Tais condições são ditas *limitações externas*, uma vez que podem ser estabelecidas apenas mediante a análise dos modos de comunicação entre agentes computacionais. Se assume que em uma quantidade finita de ações de comunicação é comunicada uma quantidade finita de informação, onde a noção de quantidade finita de informação é formalizada pela noção de elemento finito de domínio de Scott.

A seguir é mostrada uma função que não é, intuitivamente, finitamente realizável. Seja $\mathbf{h}:\mathbb{P}\to\mathbb{P}$ definida por:

- $\mathbf{h}(\mathbf{n})=\underline{\mathbf{0}}$
- $\mathbf{h}(\underline{\mathbf{n}})=\underline{\mathbf{0}}$
- $\mathbf{h}(\infty)=\mathbf{0}$

Um processo que implementasse **h**, se calaria quando a entrada fosse uma contagem finita, e produziria $\mathfrak{z}$ quando a entrada fosse uma contagem infinita. Para realizar este processo, seria preciso examinar completamente uma contagem infinita, e *depois* produzir a saída. Logo, **h** não é finitamente realizável. Isto leva à seguinte condição necessária para realizabilidade finita de funções $\mathbb{P}^k \to \mathbb{P}$.

Um mapeamento entre contagens é finitamente realizável somente se:

- Partes finitas da contagem de saída dependem apenas de partes finitas das contagens de entrada.

Conseqüentente, uma função $\mathbf{f}:\mathbb{P}^k\to\mathbb{P}$ especifica

abstratamente um processo finitamente realizável somente se, para $\mathbf{x}\in\mathbb{P}^k$:

- Partes finitas de $\mathbf{f}(\mathbf{x})$ dependem apenas de partes finitas de $\mathbf{x}$

Na seção que segue, a noção de realizabilidade finita é formalizada pela noção de continuidade.

3.2 Continuidade

▷ Sejam $(D_1,\leq_1)$ e $(D_2,\leq_2)$ domínios de Scott.

▷ Uma função $f:D_1\to D_2$ é *contínua em* $x\in D_1$ sss, para todo $y\in D_2$ finito:

- $y\leq_2 f(x)$ sss existe $x'\leq_1 x$ finito tal que $y\leq_2 f(x')$

A função f é *contínua* sss ela é contínua em todos os pontos de D_1.

Uma função é contínua, então, quando partes finitas do seu valor dependem de partes finitas de seu argumento.

▷ Uma função $f:D_1\to D_2$ é *monotônica* sss para todos $x,y\in D_1$:

- $x\leq_1 y$ implica $f(x)\leq_2 f(y)$

Intuitivamente, uma função é monotônica se, quando lhe é fornecida mais informação, ela devolve mais informação, ou igual.

Lema 3.2.1. Funções contínuas são monotônicas.

Prova. Imediata da definição de continuidade. □

Lema 3.2.2. Qualquer função monotônica $f:D_1\to D_2$ é contínua em x para x finito.

Prova. Tome $x'=x$ na definição de continuidade. □

Corolário 3.2.2. f é contínua sss f é contínua em x para todo x infinito. □

O lema abaixo caracteriza funções contínuas como aquelas que preservam supremos de conjuntos dirigidos.

Lema 3.2.3. Uma função $f:D_1 \to D_2$ é contínua sss ela é monotônica e, para todo $X \subseteq D_1$ dirigido:

- $f(\bigsqcup_1 X) = \bigsqcup_2 \{f(x) \mid x \in X\}$

Prova. A premissa de monotonicidade serve para garantir que, se $X \subseteq D_1$ é dirigido, $\{f(x) \mid x \in X\} \subseteq D_2$ é dirigido e, portanto, tem supremo. Assumindo que o supremo existe, se prova que f é monotônica. Ver [PLO 80]. □

Lema 3.2.4. Seja F_1 o conjunto de elementos finitos de D_1. Uma função monotônica $f:F_1 \to D_2$ tem uma única extensão contínua $g:D_1 \to D_2$.

Prova. g é contínua sss $g(x) = \bigsqcup \{g(x') \mid x' \in F_1, x' \leq x\}$. □

Corolário 3.2.4. O valor de uma função contínua $\mathbb{P}^k \to \mathbb{P}$ para um argumento $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ em que um ou mais $\mathbf{x}_i$, $i=1,..,k$, são ∞ é determinado de modo único pelos valores da função para argumentos $(\mathbf{y}_1, \ldots, \mathbf{y}_k)$ para $\mathbf{y}_1, \ldots, \mathbf{y}_k$ na base de $\mathbb{P}$ ($\mathbb{F}=\mathbb{P}-\{\infty\}$).

Prova. Uma função $\mathbb{F}^k \to \mathbb{P}$ tem uma única extensão contínua para $\mathbb{P}^k \to \mathbb{P}$. □

O lema 3.2.5 abaixo mostra qual é a condição que a continuidade impõe para $\mathbf{f}(\infty)$, com $\mathbf{f}:\mathbb{P} \to D$.

Lema 3.2.5. Uma função $\mathbf{f}:\mathbb{P} \to D$ é contínua sss ela é monotônica e:

- $\mathbf{f}(\infty) = \bigsqcup \{\mathbf{f}(\underline{\mathbf{n}}) \mid n \in \mathbb{N}\}$

Prova. Aplicação do lema 3.2.3 com $X=\underline{M}$. □

O lema 3.2.6 abaixo permite aplicar o lema 3.2.5 acima para funções $\mathbf{f}:\mathbb{P}^k \to D$.

Lema 3.2.6. Uma função de mais de um argumento é contínua sss ela é contínua em cada argumento.

Prova. A ordenação em um produto cartesiano é coordenada-a-coordenada. □

▷ O conjunto de funções contínuas $D_1 \to D_2$ é dito *espaço de funções contínuas de* D_1 *para* D_2, e é escrito $[D_1 \to D_2]$.

▷ A ordenação *ponto-a-ponto* $\leq$ em $[D_1 \to D_2]$ é definida por:

· $f \leq g$ sss $f(x) \leq_2 g(x)$ para todo $x \in D_1$

Lema 3.2.7. Se D_1 e D_2 são domínios de Scott, então $[D_1 \to D_2]$ ordenado ponto-a-ponto é domínio de Scott, com menor elemento a função constante de valor $\perp_2$.

Prova. Ver [PLO 77]. □

Lema 3.2.8. Se F é um subconjunto dirigido de $[D_1 \to D_2]$, com supremo f, então $f(x) = \bigsqcup_2 \{g(x) \mid g \in F\}$.

Prova. A ordenação de $[D_1 \to D_2]$ é ponto-a-ponto. □

▷ Se d_1, d_2 são elementos finitos de D_1 e D_2 respectivamente, então $(d_1 \Longrightarrow d_2)$ é o elemento de $[D_1 \to D_2]$ definido por:

· $(d_1 \Longrightarrow d_2)(x) = d_2$ se $d_1 \leq x$
$= \perp_2$ caso contrário

Lema 3.2.9. Os elementos finitos de $[D_1 \to D_2]$ são todos supremos de conjuntos finitos de elementos da forma $(d_1 \Longrightarrow d_2)$,

com d_1, d_2 elementos finitos de D_1, D_2 respectivamente.

Prova. Ver [PLO 80]. □

A seguir é analisado em que sentido uma função $\mathbb{P}\to\mathbb{P}$ contínua é considerada finita, além do sentido abstrato definido na seção 2.4.

Em primeiro lugar, note que $(\mathbf{x}\Longrightarrow\mathbf{y})(\mathbf{x})=\mathbf{y}$. Para outros valores $\mathbf{x}'$ distintos de $\mathbf{x}$, esta função vale $\underline{\mathbf{0}}$, a informação nula, a menos que $\mathbf{x}\leq\mathbf{x}'$. No último caso a função também vale $\mathbf{y}$. A excessão é inevitável, pois, caso contrário, a função $(\mathbf{x}\Longrightarrow\mathbf{y})$ não seria monotônica. Assim, $(\mathbf{x}\Longrightarrow\mathbf{y})$ pode ser visto como um elemento do grafo de uma função. A operação de supremo, no lema acima, "junta" elementos de grafo. De acordo com o lema 3.2.9 acima, uma função é finita se ela é formada por uma quantidade finita de elementos de grafo. Como caso extremo, a função constante $\mathbf{0}$ é unicamente caracterizada pelo elemento de grafo $(\underline{\mathbf{0}}\Longrightarrow\mathbf{0})$, sendo finita.

Outro exemplo mais complicado é a função contínua $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ definida por:

- $\mathbf{f}(\mathbf{n})=\mathbf{1}$ se $n<2$
  $=\mathbf{0}$ caso contrário
- $\mathbf{f}(\underline{\mathbf{n}})=\underline{\mathbf{0}}$ se $n<2$
  $=\mathbf{0}$ caso contrário

Como $\mathbf{f}$ é contínua, $\mathbf{f}(\infty)=\mathbf{0}$, pelo lema 3.2.5. Uma vez que $\mathbf{f}=(\mathbf{0}\Longrightarrow\mathbf{1})\sqcup(\mathbf{1}\Longrightarrow\mathbf{1})\sqcup(\underline{\mathbf{2}}\Longrightarrow\mathbf{0})$, a função $\mathbf{f}$ é finita. Note, ainda, que $\mathbf{f}$ é uma função maximal. Logo, em $\mathbb{P}\to\mathbb{P}$ existem funções maximais finitas.

Lema 3.2.10. Uma função $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ é finita sss existe $\underline{\mathbf{n}}$ tal que $\mathbf{f}(\underline{\mathbf{n}})$ é finito, $\mathbf{f}(\mathbf{x})=\mathbf{f}(\underline{\mathbf{n}})$ para todo $\mathbf{x}$ tal que $\underline{\mathbf{n}}\leq\mathbf{x}$, e para todo $m\leq n$, $\mathbf{f}(\mathbf{m})$ é finito.

Prova. (Se) $\mathbf{f}=(\mathbf{0}\Longrightarrow\mathbf{f}(\mathbf{0}))\sqcup\ldots\sqcup(\mathbf{n-1}\Longrightarrow\mathbf{f}(\mathbf{n-1}))\sqcup(\underline{\mathbf{n}}\Longrightarrow\mathbf{f}(\underline{\mathbf{n}}))$.

(Somente se) Caso contrário, **f** não poderia ser o supremo de um conjunto finito de elementos da forma $(\mathbf{x}\Longrightarrow\mathbf{y})$. □

Corolário 3.2.10. Existem $\mathbf{f},\mathbf{g}:\mathbb{P}\to\mathbb{P}$ tais que **f** é infinita, **g** é finita e $\mathbf{f}\leq\mathbf{g}$.

Prova. Tome $\mathbf{f}=\bigsqcup\{(\mathbf{n}\Longrightarrow\mathbf{0})\,|\,n\in\mathbb{N}\}$ e $\mathbf{g}=(\underline{\mathbf{0}}\Longrightarrow\mathbf{0})$. □

Logo, em domínios de Scott, existem partes que são mais complexas que o todo.

Lema 3.2.11. Uma função contínua $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ é maximal sss:

- $\mathbf{f}(\mathbf{n})$ é maximal
- $\mathbf{f}(\underline{\mathbf{n}})=\bigsqcap\{\mathbf{f}(\mathbf{m})\,|\,\underline{\mathbf{n}}\leq\mathbf{m}\}$

Prova. Se a primeira afirmação não valesse, $\mathbf{f}(\mathbf{n})$ poderia ser aumentado, sem prejuízo de continuidade, o que contradiria a maximalidade de **f**. A segunda afirmação vale pela monotonicidade de **f**. O ínfimo acima é o valor mais definido **y** tal que $\mathbf{y}\leq\mathbf{f}(\mathbf{m})$ para todo **m** tal $\underline{\mathbf{n}}\leq\mathbf{m}$. Para que **f** seja monotônica, é necessário que $\mathbf{f}(\underline{\mathbf{n}})\leq\mathbf{y}\leq\mathbf{f}(\mathbf{m})$ para $\underline{\mathbf{n}}\leq\mathbf{m}$. Logo, o valor mais definido que $\mathbf{f}(\underline{\mathbf{n}})$ pode assumir é **y**. O valor de $\mathbf{f}(\infty)$ é determinado pelos demais valores de $\mathbf{f}(\mathbf{x})$ e pela continuidade de **f**, conforme o lema 3.2.5. □

Corolário 3.2.11. Os valores de uma função contínua maximal $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ dependem apenas dos valores que a função assume para $\mathbf{n}\in\mathbb{M}$. □

Lema 3.2.12. Uma função $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ finita é maximal sss existe $\underline{\mathbf{n}}$ tal que $\mathbf{f}(\underline{\mathbf{n}})$ é finito maximal, e para todo $m\leq n$, $\mathbf{f}(\mathbf{m})$ é finito maximal.

Prova. Aplicação dos lemas 3.2.10 e 3.2.11. □

3.3 Funções contínuas como informações

Assim como um número parcial é visto como uma informação sobre um natural ou ω, uma função $\mathbf{f}:\mathbb{P}^k\to\mathbb{P}$ pode ser vista como uma informação sobre uma função total ou parcial $f:\mathbb{N}^k\to\mathbb{N}$.

▷ Uma função $\mathbf{f}:\mathbb{P}^k\to\mathbb{P}$ é dita *informação sobre* uma função $f:\mathbb{N}^k\to\mathbb{N}$, escrito $f\in\mathbf{f}$, se, para todos $n_1,\ldots,n_k\in\mathbb{N}$, $\mathbf{x}_1,\ldots,\mathbf{x}_k\in\mathbb{P}$:

- $n_1\in\mathbf{x}_1,\ldots,n_k\in\mathbf{x}_k$ e f é definida para $(n_1,\ldots,n_k)$ implica $f(n_1,\ldots,n_k)\in\mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k)$

Isto é, se $\mathbf{x}_1,\ldots,\mathbf{x}_k$ são informações sobre $n_1,\ldots,n_k$ respectivamente, então $\mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k)$ é informação sobre $f(n_1,\ldots,n_k)$.

A notação $f\in\mathbf{f}$ é utilizada por analogia à notação $x\in\mathbf{x}$, que significa que $\mathbf{x}$ é uma informação sobre x. De imediato, a função $\mathbb{P}^k\to\mathbb{P}$ de valor constante $\underline{0}$ é informação sobre qualquer função $\mathbb{N}^k\to\mathbb{N}$, mas esta informação é nula.

Suponha que $\mathbf{f}(\mathbf{3})=\mathbf{0}$ e $\mathbf{f}(\underline{\mathbf{3}})=\mathbf{1}$, e que $f\in\mathbf{f}$. Então, necessariamente $f(3)=0$, pela primeira equação. Mas, pela segunda equação, $f(n)=1$ se $n\geq 3$, o que é uma contradição. Isto ocorre porque $\mathbf{f}$ não é monotônica.

Conforme a discussão acima, a monotonicidade de $\mathbf{f}$ pode ser vista como a sua *coerência*. Uma função $\mathbb{P}^k\to\mathbb{P}$ não monotônica é uma informação contraditória.

▷ Deste ponto em diante, a notação $f\in\mathbf{f}$ é utilizada apenas quando $\mathbf{f}$ é contínua.

Suponha que f é definida por:

- $f(n)$ é indefinida se $n<3$

- $f(n)=1$ se $n\geq 3$

e que **f** é definida por:

- $\mathbf{f}(\mathbf{n})=\mathbf{1}$
- $\mathbf{f}(\underline{\mathbf{n}})=\underline{\mathbf{0}}$

Se $n<3$, então a afirmação $n\in\mathbf{x}$ e f é definida para n é falsa, e, portanto, implica na afirmação $f(n)\in\mathbf{f}(\mathbf{x})$. Por outro lado, quando $n\geq 3$, a afirmação $n\in\mathbf{x}$ e f é definida para n vale quando $\underline{\mathbf{3}}\ \mathbf{x}$. Neste caso, $\mathbf{f}(\mathbf{x})=\mathbf{1}$ ou $\mathbf{f}(\mathbf{x})=\underline{\mathbf{0}}$. Como $1\in\mathbf{1}$ e $1\in\underline{\mathbf{0}}$, $f(n)\in\mathbf{f}(\mathbf{x})$. Logo, $f\in\mathbf{f}$. Mas a informação que **f** fornece sobre f não é a melhor possível. A informação $\mathbf{f}(\underline{\mathbf{3}})=\underline{\mathbf{0}}$ não é incorreta, pois $f(x)\in\underline{\mathbf{0}}$ para todo $x\in\underline{\mathbf{3}}$. Mas a informação $\mathbf{f}(\underline{\mathbf{3}})=\mathbf{1}$ seria melhor, porque $f(x)\in\mathbf{1}$ para todo $x\in\underline{\mathbf{3}}$. A informação $\mathbf{f}(\mathbf{0})=\mathbf{1}$ também não é incorreta, porque ela indica que se f *fosse* definida para 0, o seu valor estaria em $\mathbf{1}=\{1\}$. Mas f não é definida para 0. Se $\mathbf{f}(\mathbf{1})$ fosse ∞, o fato de que f não é definida para 0 seria indicado por **f**, pois uma tentativa de estimar o valor de $f(0)$, via **f**, levaria à contradição (especificamente, $n\in\infty$), mostrando que este valor não existe, pois nenhum natural está em $\infty=\{\omega\}$. De acordo com este ponto de vista, a função que melhor informaria sobre f seria **g** tal que:

- $\mathbf{g}(\mathbf{n})=\infty$ se $n<3$
  $=\mathbf{1}$ se $n\geq 3$
- $\mathbf{g}(\underline{\mathbf{n}})=\underline{\mathbf{0}}$ se $n<3$
  $=\mathbf{1}$ se $n\geq 3$

Tal função **g** satisfaz $\mathbf{f}\ \mathbf{g}$.

A propriedade que **g** tem que a distingue das demais **f** tais que $f\in\mathbf{f}$ é que **g** é uma função maximal.

Assim, a maximalidade de uma função **f** pode ser vista como a sua *completeza*. Uma função $\mathbb{P}^k\to\mathbb{P}$ não maximal é uma informação que pode ser melhorada.

3.4 Extensões de funções parciais $\mathbb{N}^k \to \mathbb{N}$ para funções contínuas $\mathbb{P}^k \to \mathbb{P}$

▷ A *restrição* para $\mathbb{N}^k \to \mathbb{N}$ de uma função contínua $\mathbf{f}:\mathbb{P}^k \to \mathbb{P}$ é a função parcial $f:\mathbb{N}^k \to \mathbb{N}$ definida por:

- f é indefinida para $(x_1,\ldots,x_k)$ se $\mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k) \in \underline{\mathbb{P}}$
- $f(x_1,\ldots,x_k)=y$ se $\mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k)=\mathbf{y} \in \mathbb{M}$

Por exemplo, a restrição da função $\mathbf{x} \mapsto \underline{\mathbf{x}}$ é a função parcial $\mathbb{N} \to \mathbb{N}$ totalmente indefinida. Em geral, se $\mathbf{f}:\mathbb{P} \to \mathbb{P}$ é qualquer, a restrição da função $\mathbf{x} \mapsto \underline{\mathbf{f}(\mathbf{x})}$ é a função totalmente indefinida. A restrição da função **succ** é a função $x \mapsto x+1$.

▷ Uma *extensão* para $\mathbb{P}^k \to \mathbb{P}$ de uma função parcial $f:\mathbb{N}^k \to \mathbb{N}$ é qualquer função contínua $\mathbf{f}:\mathbb{P}^k \to \mathbb{P}$ cuja restrição é f.

O exemplo acima revela que a função parcial totalmente indefinida $\mathbb{N} \to \mathbb{N}$ tem infinitas extensões para $\mathbb{P} \to \mathbb{P}$ (não enumeráveis).

Proposição 3.4.1. Se $\mathbf{f}:\mathbb{P}^k \to \mathbb{P}$ é uma extensão de $f:\mathbb{N}^k \to \mathbb{N}$, então $f \in \mathbf{f}$.

Prova. A prova é feita para $k=1$. Se $\mathbf{f}$ é uma extensão de f e $\mathbf{f}(\mathbf{n})=\mathbf{m}$, então $f(n)=m$. Como $n \in \mathbf{n}$ e $m \in \mathbf{m}$, $f \in \mathbf{f}$. □

O primeiro modo considerado de estender $\mathbb{N}^k \to \mathbb{N}$ para uma função $\mathbb{P}^k \to \mathbb{P}$ consiste em atribuir a informação nula aos novos pontos da função e aos pontos em que ela é indefinida.

▷ A extensão *trivial* de uma função $f:\mathbb{N}^k \to \mathbb{N}$ é a função $\mathbf{f}:\mathbb{P}^k \to \mathbb{P}$ tal que:

$\cdot \mathbf{f}(\mathbf{n}_1,\ldots,\mathbf{n}_k)=\mathbf{m}$ se $f(n_1,\ldots,n_k)=m$

$=\underline{\mathbf{0}}$ se f é indefinida para $(n_1,\ldots,n_k)$

$\cdot \mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k)=\underline{\mathbf{0}}$ se $\mathbf{x}_i \in \underline{\mathbb{P}}$ para algum $i=1,\ldots,k$

De imediato, uma extensão trivial é contínua. A extensão trivial **f** de uma função f é a extensão minimal, no sentido que, para toda extensão **f'** de f, $\mathbf{f} \leq \mathbf{f}'$. Então:

· $\mathbf{f}=\bigsqcap\{\mathbf{f}' \mid \mathbf{f}'$ é contínua e $\mathbf{f}'$ é extensão de $f\}$

A extensão trivial da função indefinida é a função constante $\underline{\mathbf{0}}$. A extensão trivial da função identidade é a função **f** tal que:

· $\mathbf{f}(\mathbf{n})=\mathbf{n}$

· $\mathbf{f}(\underline{\mathbf{n}})=\underline{\mathbf{0}}$

· $\mathbf{f}(\infty)=\underline{\mathbf{0}}$,

que não é uma função identidade.

O segundo modo considerado de estender funções $f:\mathbb{N}^k\to\mathbb{N}$ para $\mathbf{f}:\mathbb{P}^k\to\mathbb{P}$ consiste em atribuir a melhor informação possível para os novos pontos, e ∞ nos pontos em que f é indefinida (vide **g** na seção anterior 3.3 e teorema 3.4.4 adiante). Note que, nos pontos em que f é indefinida, é possível apenas pôr um valor de $\underline{\mathbb{P}}$ para **f**, pois, caso contrário, a restrição de **f** não seria f.

▷ A extensão *natural* de uma função f é sua extensão contínua maximal. A extensão natural **f** de f satisfaz:

· $\mathbf{f}=\bigsqcup\{\mathbf{f}' \mid \mathbf{f}'$ é contínua e $\mathbf{f}'$ é extensão de $f\}$

O lema 3.4.2 abaixo mostra que **f** é bem definida.

Lema 3.4.2. Se **F** é um conjunto de extensões contínuas de uma função $f:\mathbb{N}^k\to\mathbb{N}$, então **F** é dirigido e seu supremo é uma extensão de f.

Prova. A prova é desenvolvida para o caso k=1. Se f é definida para n e $f(n)=m$, então $\mathbf{f}(\mathbf{n})=\mathbf{m}$ para toda $\mathbf{f}\in\mathbf{F}$. Caso contrário, $\mathbf{f}(\mathbf{n})\in\underline{\mathbb{P}}$ para todo $\mathbf{f}\in\mathbf{F}$. Como qualquer subconjunto de $\underline{\mathbb{P}}$ tem supremo, o supremo de **F** existe. Se f é definida para n e $f(n)=m$, então:

- $(\bigsqcup\mathbf{F})(\mathbf{n})=\bigsqcup\{\mathbf{f}(\mathbf{n})\mid\mathbf{f}\in\mathbf{F}\}$ (lema 3.2.8)
  $=\bigsqcup\{\mathbf{m}\}=\mathbf{m}$

Pela definição de extensão, o fato de **f** ser ou não uma extensão de f depende apenas dos valores que **f** assume para $\mathbf{n}\in\mathbb{M}$ tal que f é definida para n. Logo, o supremo de **F** é uma extensão de f. □

A extensão natural da função identidade $\mathbb{N}\to\mathbb{N}$ é a identidade $\mathbb{P}\to\mathbb{P}$. A extensão natural da função totalmente indefinida $\mathbb{N}\to\mathbb{N}$ é a função constante ω.

Proposição 3.4.3. As extensões naturais das funções $succ:x\mapsto x+1$ e $pred:0\mapsto 0, x+1\to x$ são as funções **succ** e **pred** respectivamente.

Prova. succ é uma extensão de $succ$. **succ** é maximal pois (1) para pontos $\mathbf{n}\in\mathbb{M}$, $\mathbf{succ}(\mathbf{n})\in\mathbb{M}$, e portanto não pode ser aumentada, pois os valores em $\mathbb{M}$ são maximais, (2) para pontos $\underline{\mathbf{n}}\in\underline{\mathbb{M}}$ **succ** não pode ser aumentada, pois caso contrário **succ** se tornaria não monotônica e (3) para o ponto ω **succ** não pode ser aumentada, por continuidade. A prova para **pred** é similar. □

O seguinte teorema caracteriza explicitamente a extensão natural para o caso $\mathbb{N}\to\mathbb{N}$.

Teorema 3.4.4. $\mathbf{f}:\mathbb{P}\to\mathbb{P}$ é a extensão natural de $f:\mathbb{N}\to\mathbb{N}$ sss:

- $\mathbf{f}(\mathbf{n})=\mathbf{m}$ se f é definida para n e $f(n)=m$
  $=\infty$ se f é indefinida para n
- $\mathbf{f}(\underline{\mathbf{n}})=\bigsqcap\{\mathbf{f}(\mathbf{m})\,|\,\underline{\mathbf{n}}\leq\mathbf{m}\}$

Prova. O lema é estabelecido pelo lema 3.2.11. □

Um número parcial $\underline{\mathbf{n}}$ é um conjunto de *possibilidades* $\{n,n+1,\ldots\}$. Dada $f:\mathbb{N}\to\mathbb{N}$, é natural pensar em extendê-la para $\mathbf{f}:\mathbb{P}\to\mathbb{P}$, de tal modo que $\mathbf{f}(\underline{\mathbf{n}})=\mathbf{f}(\{n,n+1,\ldots\})=\{f(n),f(n+1),\ldots\}$, i.e., aplicando a função f a cada possibilidade em $\underline{\mathbf{n}}$. Porém, pode ocorrer que o resultado desta operação não esteja em $\mathbb{P}$. Neste caso, é possível tomar o fecho-$\mathbb{P}$ deste resultado. O teorema a seguir mostra que este procedimento leva à extensão natural de $\mathbf{f}$.

Teorema 3.4.5. $\mathbf{f}:\mathbb{P}^k\to\mathbb{P}$ é a extensão natural de $f:\mathbb{N}^k\to\mathbb{N}$ sss, para todos $\mathbf{x}_1,\ldots,\mathbf{x}_k\in\mathbb{F}$:

- $\mathbf{f}(\mathbf{x}_1,\ldots,\mathbf{x}_k)=\mathcal{C}_{\mathbb{P}}(f[\mathbf{x}_1,\ldots,\mathbf{x}_k])$ se $f[\mathbf{x}_1,\ldots,\mathbf{x}_k]\neq\emptyset$
  $=\infty$ caso contrário

onde:

- $f[\mathbf{x}_1,\ldots,\mathbf{x}_k]=\{f(n_1,\ldots,n_k)\,|\,f$ é definida para $(n_1,\ldots,n_k)$, $n_1\in\mathbf{x}_1$, $i=1,\ldots,k\}$

Prova. A prova é desenvolvida para o caso $k=1$. Há dois casos para $\mathbf{f}(\mathbf{x})$. (1) Se $\mathbf{x}$ é $\mathbf{n}\in\mathbb{M}$, há dois subcasos. (1.1) Se f é definida para n e $f(n)=m$, então $f[\mathbf{x}]=\mathbf{m}$, e $\mathbf{f}(\mathbf{n})=\mathbf{m}$. Pelo teorema 3.4.4 $\mathbf{f}$ é a extensão natural de f para o ponto $\mathbf{x}$. (1.2) Se f é indefinida para n, então $f[\mathbf{x}]=\emptyset$, e $\mathbf{f}(\mathbf{x})=\infty$. Pelo mesmo lema, $\mathbf{f}$ é a extensão natural de f para o ponto $\mathbf{x}$. (2) Se $\mathbf{x}=\underline{\mathbf{n}}\in\mathbb{M}$, então há dois subcasos. (2.1) Se para todo $m\geq n$ f é definida, então:

- $\mathbf{f}(\mathbf{x})=\mathcal{C}_{\mathbb{P}}\{f(m)\,|\,m\geq n\}$
  $=\mathcal{C}_{\mathbb{P}}\big(\bigcup\{\{f(m)\}\,|\,m\geq n\}\big)=\mathcal{C}_{\mathbb{P}}\big(\bigcup\{\mathbf{f}(\mathbf{m})\,|\,\underline{\mathbf{n}}\leq\mathbf{m}\}\big)$
  $=\bigsqcap\{\mathbf{f}(\mathbf{m})\,|\,\mathbf{n}\leq\underline{\mathbf{m}}\}$ (lema 2.4.9)

Pelo mesmo teorema, $\mathbf{f}$ é a extensão natural para o ponto $\mathbf{x}$.

(2.2) Se f é indefinida para algum $m \geq n$, há dois subsubcasos. (2.2.1) Se f é indefinida para todo $m \geq n$, então $f[\mathbf{x}]=\emptyset$ e $\mathbf{f}(\mathbf{x})=\infty$, e $\mathbf{f}$ é a extensão natural de f para $\mathbf{x}$. (2.2.2) Se existe $m \geq n$ para o qual f é definida, então $f[\mathbf{x}] \neq \emptyset$, e:

- $\mathbf{f}(\mathbf{x}) = \mathcal{C}_{\mathbb{P}}\{f(m) \mid m \geq n,\ f \text{ d.p. } m\}$ (d.p.="é definida para")
  $= \mathcal{C}_{\mathbb{P}}\left(\cup(\{f(m)\} \mid m \geq n,\ f \text{ d.p. } m\} \cup \{\infty\}) \right)$
  $= \mathcal{C}_{\mathbb{P}}\left(\cup\{\mathbf{f}(\mathbf{m}) \mid \underline{\mathbf{n}} \leq \mathbf{m}\} \right)$ (pois $\mathbf{f}(\mathbf{m})=\infty$ se f não d.p. m)
  $= \bigsqcap \{\mathbf{f}(\mathbf{m}) \mid \underline{\mathbf{n}} \leq \mathbf{m}\}$ (lema 2.4.9)

Pelo teorema 3.4.4, $\mathbf{f}$ é a extensão natural para $\mathbf{x}$. A segunda linha da dedução é justificada por que $\mathbf{f}(\mathbf{x})$ necessariamente está em $\underline{\mathbb{P}}$, pois $\mathbf{x} \sqcap \infty = \underline{\mathbf{x}}$ e $\mathbf{f}(\mathbf{m})=\infty$ para algum $m \geq n$. Deste modo, $\infty \leq \mathbf{f}(\mathbf{x})$, pois $\mathbf{f}(\mathbf{x}) \leq \infty$. Logo, a inclusão de ∞ não altera o resultado. □

Embora os números naturais não possuam um valor "indefinido", é possível acomodar este "valor" via o uso de funções parciais $\mathbb{N}^0 \to \mathbb{N}$. Uma função $f:\mathbb{N}^0 \to \mathbb{N}$ costuma ser pensada como uma constante. Quando f é indefinida, é possível pensar f como uma constante indefinida u. Deste modo, é possível pensar os números parciais como extensões dos números naturais e u:

- $\mathbf{n}$ é a única extensão de n
- $\underline{\mathbf{n}}$ e ∞ são extensões de u
- a extensão trivial de u é $\underline{\mathbf{0}}$
- a extensão natural de u é ∞

4 FUNÇÕES CONTÍNUAS PARTICULARES

Este capítulo introduz funções particulares que são usadas nos demais capítulos. A seção 4.1 discute definição por casos. A seção 4.2 define operações lógicas correspondentes aos conectivos proposicionais. A seção 4.3 define a noção de função $\mathbb{P}$-característica e, em particular, estuda a função $\mathbb{P}$-característica do predicado de igualdade. A seção 4.4 define operações lógicas que correspondem a quantificadores. A seção 4.5 introduz o operador de ponto fixo. A seção 4.6 estuda as extensões naturais de algumas operações aritméticas.

4.1 Definição por casos

▷ O operador de *análise de casos* $\mathbf{cond}_D:\mathbb{P}\times D^2\to D$ é definido por:

- $\mathbf{cond}_D(\underline{0},\mathbf{y},\mathbf{z})=\mathbf{y}\sqcap\mathbf{z}$
- $\mathbf{cond}_D(\mathbf{0},\mathbf{y},\mathbf{z})=\mathbf{z}$
- $\mathbf{cond}_D(\mathbf{x+1},\mathbf{y},\mathbf{z})=\mathbf{y}$

Quando o subscrito D é omitido, se subentende que $D=\mathbb{P}$.

▷ Pelo uso que é dado a este operador na seção 4.2, ele também é dito *condicional (não estrito)*.

Proposição 4.1.1. **cond** é a extensão natural da função $cond:\mathbb{N}^3\to\mathbb{N}$ tal que:

- $cond(0,y,z)=z$
- $cond(x+1,y,z)=y$

Prova. Seja $\mathbf{f}:\mathbb{P}^3\to\mathbb{P}$ a extensão natural de *cond*. Pelo teorema

3.4.5, a função **f** satisfaz, para $\mathbf{x},\mathbf{y},\mathbf{z}\in\mathbb{F}$:

- $\mathbf{f}(\underline{\mathbf{0}},\mathbf{y},\mathbf{z})=\mathcal{C}_{\mathbb{P}}\{cond(x,y,z)\,|\,x\in\underline{\mathbf{0}},\ y\in\mathbf{y},\ z\in\mathbf{z},\ x,y,z\in\mathbb{N}\}$
  $=\mathcal{C}_{\mathbb{P}}(\{cond(0,y,z)\,|\,y\in\mathbf{y},\ z\in\mathbf{z},\ y,z\in\mathbb{N}\}$
  $\cup\{cond(x+1,y,z)\,|\,y\in\mathbf{y},\ \mathbf{z}\in\mathbf{z},\ x,y,z\in\mathbb{N}\})$
  $=\mathcal{C}_{\mathbb{P}}(\{z\,|\,\mathbf{z}\in\mathbf{z},\ z\in\mathbb{N}\}\cup\{y\,|\,y\in\mathbf{y},\ y\in\mathbb{N}\})$
  $=\mathcal{C}_{\mathbb{P}}((\mathbf{y}-\{\omega\})\cup(\mathbf{z}-\{\omega\}))=_{\mathbb{P}}\mathcal{C}\ (\mathbf{y}\cup\mathbf{z})$
  $=\mathbf{y}\sqcap\mathbf{z}$ (lema 2.4.9)

- $\mathbf{f}(\mathbf{0},\mathbf{y},\mathbf{z})=\mathcal{C}_{\mathbb{P}}\{cond(x,y,z)\,|\,x\in\mathbf{0},\ y\in\mathbf{y},\ z\in\mathbf{z},\ x,y,z\in\mathbb{N}\}$
  $=\mathcal{C}_{\mathbb{P}}\{cond(0,y,z)\,|\,y\in\mathbf{y},\ z\in\mathbf{z},\ y,z\in\mathbb{N}\}$
  $=\mathcal{C}_{\mathbb{P}}\{z\,|\,\mathbf{z}\in\mathbf{z},\ z\in\mathbb{N}\}=\mathcal{C}_{\mathbb{P}}(\mathbf{z}-\{\omega\})$
  $=\mathbf{z}$

- $\mathbf{f}(\mathbf{x+1},\mathbf{y},\mathbf{z})=\mathcal{C}_{\mathbb{P}}\{cond(x,y,z)\,|\,x\in\mathbf{x+1},\ y\in\mathbf{y},\ z\in\mathbf{z},\ x,y,z\in\mathbb{N}\}$
  $=\mathcal{C}_{\mathbb{P}}\{cond(x+1,y,z)\,|\,x\in\mathbf{x},\ y\in\mathbf{y},\ z\in\mathbf{z},\ x,y,z\in\mathbb{N}\}$
  $=\mathcal{C}_{\mathbb{P}}\{y\,|\,y\in\mathbf{y},\ y\in\mathbb{N}\}=\mathcal{C}_{\mathbb{P}}(\mathbf{y}-\{\omega\})$
  $=\mathbf{y}$

A igualdade $\mathcal{C}_{\mathbb{P}}\{\mathbf{a}-(\omega)\}=\mathbf{a}$, $\mathbf{a}\in\mathbb{P}$, depende de que $\mathbf{a}\neq\infty$, i.e., $\mathbf{a}\in\mathbb{F}$. Como **f** e **cond** são iguais na base, e ambos são contínuos, tem-se que **f=cond**, pelo lema 3.2.4 e corolário. □

Corolário 4.1.1. **cond** é função contínua e maximal. □

▷ O operador de *análise de casos fraca* $\mathbf{scond}_D:\mathbb{P}\times D^2\to D$ é definido por:

- $\mathbf{scond}_D(\underline{\mathbf{0}},\mathbf{y},\mathbf{z})=\perp_D$
- $\mathbf{scond}_D(\mathbf{0},\mathbf{y},\mathbf{z})=\mathbf{z}$
- $\mathbf{scond}_D(\mathbf{x+1},\mathbf{y},\mathbf{z})=\mathbf{y}$

O operador de análise de casos fraca devolve informação nula para informação nula no argumento em que se faz análise de casos. O operador fraco é definível explicitamente em termos do forte por:

- $\mathbf{scond}_D(\mathbf{x},\mathbf{y},\mathbf{z})=\mathbf{cond}_D(\mathbf{x},\mathbf{cond}_D(\mathbf{x},\mathbf{y},\perp_D),\mathbf{cond}_D(\mathbf{x},\perp_D,\mathbf{z}))$

É imediato que:

- $\mathbf{scond}_D \leq \mathbf{cond}_D$

▷ O operador de análise de casos fraca também é dito *condicional estrito*.

▷ O operador de *definição por casos* $\mathbf{cases}_D : \mathbb{P} \to D^3 \to \mathbb{P}$ é definido para todos $\mathbf{x} \in \mathbb{P}$, $\mathbf{a}, \mathbf{b}, \mathbf{c} \in D$, por:

- $\mathbf{cases(x,a,b,c)} = \mathbf{cond(x,cond(x,c,a),b)}$

onde o subscrito é omitido quando pode ser inferido do contexto.

Lema 4.1.2. Sejam $\mathbf{f}, \mathbf{g} : \mathbb{P} \to D$ contínuas, $\mathbf{a}, \mathbf{b} \in D$, com $\mathbf{f}$ definida a partir de $\mathbf{g}$ por:

- $\mathbf{f}(\underline{0}) = \mathbf{a}$
- $\mathbf{f}(0) = \mathbf{b}$
- $\mathbf{f}(x+1) = \mathbf{g}(x)$

Então $\mathbf{f}$ tem a definição explícita:

- $\mathbf{f}(x) = \mathbf{cases}(x, a, b, g(x-1))$

Prova. Como $\mathbf{f}$ é contínua, $\mathbf{a} \leq \mathbf{b}$, pois $\mathbf{f}(\underline{0}) \leq \mathbf{f}(0)$, e $\mathbf{a} \leq \mathbf{g}(\underline{0})$, pois $\mathbf{f}(\underline{0}) \leq \mathbf{f}(\underline{1})$. Então $\mathbf{a} \sqcap \mathbf{b} \sqcap \mathbf{g}(\underline{0}) = \mathbf{a}$. Logo:

- $\begin{aligned} \mathbf{f}(\underline{0}) &= \mathbf{cases}(\underline{0}, a, b, g(\underline{0}-1)) \\ &= \mathbf{cond}(\underline{0}, \mathbf{cond}(\underline{0}, g(\underline{0}), a), b) \\ &= (g(\underline{0}) \sqcap a) \sqcap b \\ &= a. \end{aligned}$

Os demais casos são imediatos. □

Lema 4.1.3. O condicional **cond** satisfaz:

- $\mathbf{cond}(x, 0, 0) = 0$
- $\mathbf{cond}(x, y+1, z+1) = \mathbf{cond}(x, y, z) + 1$

Prova. A primeira equação é imediata, pois $0 \sqcap 0 = 0$. A segunda é provada por casos para $\mathbf{x}$:

- se $\mathbf{x=\underline{0}}$, $\mathbf{cond(x,y+1,z+1)=(y+1)\sqcap(z+1)}$
  $\mathbf{=(y\sqcap z)+1}$ (lema 2.5.3)
  $\mathbf{=cond(x,y,z)+1}$
- se $\mathbf{x=0}$, $\mathbf{cond(x,y+1,z+1)=z+1=cond(x,y,z)+1}$
- se $\mathbf{x=x'+1}$, $\mathbf{cond(x,y+1,z+1)=y+1=cond(x,y,z)+1}$ □

Lema 4.1.4. $\mathbf{cond}_{D\to E}(\mathbf{x},f,g)(y)=\mathbf{cond}_{E}(\mathbf{x},f(y),g(y))$.

Prova. $(f\sqcap g)(y)=f(y)\sqcap g(y)$. □

4.2 Operações lógicas

▷ As *informações de verdade* "é verdadeiro", "é falso", e "é verdadeiro ou falso" são representadas por t, f e u respectivamente. A informação de verdade "é verdadeiro ou falso" é nula, assim como a informação de quantidade $\underline{\mathbf{0}}$. Por conveniência, t,f,u são respectivamente $\mathbf{1},\mathbf{0},\underline{\mathbf{0}}$. O *domínio de informações de verdade* é $\mathbb{B}=\{t,f,u\}\subseteq\mathbb{P}$, com a relação de ordem de $\mathbb{P}$ restrita a $\mathbb{B}$. As variáveis $\mathbf{p},\mathbf{q},\mathbf{r}$ correm sobre $\mathbb{B}$.

A relação de ordem $\leq$ de $\mathbb{B}$ satisfaz:

- $\mathbf{p}\leq\mathbf{q}$ sss $\mathbf{p}=u$ ou $\mathbf{p}=\mathbf{q}$
- $u=t\sqcap f$

Lema 4.2.1. $\mathbb{B}$ é domínio de Scott com menor elemento u, base $\mathbb{B}$ e conjunto de elementos finitos $\mathbb{B}$.

Prova. Ver prova do teorema 2.4.2. □

Pelo lema 4.2.1 acima e o lema 3.2.2, qualquer função monotônica $\mathbb{B}\to D$ é contínua.

▷ O *condicional* $\mathbf{bcond}:\mathbb{B}^3\to\mathbb{B}$ é definido por:

- **bcond**(t,**p**,**q**)=**p**
- **bcond**(f,**p**,**q**)=**q**
- **bcond**(t$\sqcap$f,**p**,**q**)=**p**$\sqcap$**q**

Este operador é definível pelo operador de análise de casos:

- **bcond**(**p**,**q**,**r**)=**cond**(**p**,**q**,**r**)

▷ O primeiro argumento do condicional é dito *condição*, e o segundo e o terceiro, *ramos verdadeiro* e *falso*, respectivamente.

Se a condição é a informação nula u, o resultado é a melhor informação consistente com ambos os ramos. Por exemplo, **bcond**(u,**4**,**5**)=**4**. Um caso particular ocorre quando ambos os ramos são iguais: **bcond**(u,**x**,**x**)=**x**.

▷ As operações lógicas sobre informações de verdade $\neg:\mathbb{B}\to\mathbb{B}$, $\wedge,\vee,\to:\mathbb{B}^2\to\mathbb{B}$ são respectivamente definidas por:

- $\neg$**p** = **bcond**(**p**,f,t)
- **p**$\wedge$**q** = **bcond**(**p**,**q**,f)
- **p**$\vee$**q** = **bcond**(**p**,t,**q**)
- **p**$\to$**q** = **bcond**(**p**,**q**,t)

Estas operações satisfazem, para **p**,**q**$\in\mathbb{B}$:

- $\neg$t=f
- $\neg$f=t
- $\neg$u=u
- **p**$\wedge$f = f$\wedge$**p** = f
- t$\wedge$t=t
- t$\wedge$u = u$\wedge$t = u$\wedge$u = u
- **p**$\vee$t = t$\wedge$**p** =t
- f$\vee$f=f
- f$\vee$u = u$\vee$f = u$\vee$u = u
- f$\to$**p** = **p**$\to$t = t
- t$\to$f=f
- u$\to$f = t$\to$u = u$\to$u = u
- **p**$\wedge\neg$**p**=f sss **p**$\neq$u
- **p**$\vee\neg$**p**=t sss **p**$\neq$u
- $\neg\neg$**p**=**p**
- **p**$\to$**q** = $\neg$**p**$\vee$**q** = $\neg$**q**$\to\neg$**p**
- $\neg$(**p**$\wedge$**q**)=$\neg$**p**$\vee\neg$**p**
- $\neg$(**p**$\vee$**q**)=$\neg$**p**$\wedge\neg$**p**

Note que as equações valem em particular quando **p**=u ou **q**=u. Assim, u$\vee$t = t$\vee$u = t, por exemplo. Se **bcond** fosse estrito na condição, i.e., **bcond**(u,**p**,**q**)=u, esta operação satisfaria u = u$\vee$t $\neq$ t$\vee$u = t, e não seria comutativa.

É fácil mostrar que estas operações são as extensões naturais das correspondentes nos naturais. O caso para $\wedge$ é exemplificado. A prova é omitida, por ser análoga à prova da proposição 4.1.1.

Os naturais 1 e 0 são escritos t e ℓ. O operador lógico $\wedge$ é a extensão natural de $e:\{t,\ell\}^2\to\{t,\ell\}$ tal que:

- $e(\ell,x) = e(x,\ell) = \ell$ para todo $x\in\{t,\ell\}$
- $e(t,t) = t$

4.3 Funções características, igualdade

Como na seção 4.2 anterior, $\mathfrak{t}=\mathbf{1}$, $\mathfrak{f}=\mathbf{0}$, $\mathfrak{u}=\underline{\mathbf{0}}$, $\mathbb{B}=\{\mathfrak{t},\mathfrak{f},\mathfrak{u}\}$ e $t=1$, $\ell=0$.

▷ A *função característica* de um conjunto $A\subseteq\mathbb{N}^k$ é a função $\aleph_A:\mathbb{N}^k\to\mathbb{N}$ tal que:

- $\aleph_A(n_1,\ldots,n_k)=t$ se $n_1,\ldots,n_k\in A$
  $=\ell$ caso contrário

É imediato que uma função caracteristíca $\aleph_A$ determina unicamente o conjunto A:

- $A=\{(n_1,\ldots,n_k)\in\mathbb{N}^k \mid \aleph_A(n_1,\ldots,n_k)=t\}$

▷ Uma *função $\mathbb{P}$-característica parcial* (*f.c.p*) de um conjunto $A\subseteq\mathbb{N}^k$ é uma função contínua $\mathbf{p}:\mathbb{P}^k\to\mathbb{B}$ tal que:

- $\mathbf{p}(\mathbf{n}_1,\ldots,\mathbf{n}_k)=\mathfrak{t}$ implica $(n_1,\ldots,n_k)\in A$
- $\mathbf{p}(\mathbf{n}_1,\ldots,\mathbf{n}_k)=\mathfrak{f}$ implica $(n_1,\ldots,n_k)\notin A$

O prefixo $\mathbb{P}$ é omitido quando não há ambigüidade possível.

Assim como os números parciais são pensados como informações sobre números naturais, funções características

parciais são pensadas como informações sobre conjuntos (k=1) ou relações (k>1) de números naturais. Isto motiva a notação introduzida a seguir (cf. seção 3.3).

▷ Escreve-se $A \in \mathbf{p}$ sss $\mathbf{p}$ é função característica parcial de A.

Lema 4.3.1. Para toda função contínua $\mathbf{p}:\mathbb{P}^k \to \mathbb{B}$ existe $A \in \mathbf{p}$.

Prova. Tome $A=\{(n_1,\ldots,n_k) \in \mathbb{N}^k \mid \mathbf{p}(\mathbf{n}_1,\ldots,\mathbf{n}_k)=\mathrm{t}\}$. □

▷ O conjunto exibido na prova é dito o *núcleo* de $\mathbf{p}$.

Corolário 4.3.1. Se $A \in \mathbf{p}$, então A contém o núcleo de $\mathbf{p}$. □

▷ O *conúcleo* de uma função característica parcial $\mathbf{p}:\mathbb{P}^k \to \mathbb{P}$ é o conjunto $\{(n_1,\ldots,n_k) \in \mathbb{N}^k \mid \mathbf{p}(\mathbf{n}_1,\ldots,\mathbf{n}_k)=\mathrm{f}\}$.

Lema 4.3.2. Se $A \in \mathbf{p}$, então A é disjunto do conúcleo de $\mathbf{p}$.

Prova. Imediata. □

▷ Uma função característica parcial $\mathbf{p}:\mathbb{P}^k \to \mathbb{P}$ é *completa* sss a união de seu núcleo com o seu conúcleo é $\mathbb{N}^k$.

Corolário 4.3.2. Se $\mathbf{p}$ é completa então $\{A \mid A \in \mathbf{p}\}$ é unitário, e é formado pelo núcleo de $\mathbf{p}$. □

Aparentemente, uma função característica completa forneceria a melhor informação possível sobre um conjunto. Mas este não é o caso.

Lema 4.3.3. Existem funções características completas, distintas, e com o mesmo núcleo.

Prova. Seja a família de funções $\varnothing_q:\mathbb{P} \to \mathbb{B}$, para $q \in \mathbb{N} \cup \{\omega\}$, tal que:

- $\varnothing_q(\mathbf{n})=f$
- $\varnothing_q(\underline{\mathbf{n}})=f$ se $n\geq q$
  $=u$ se $n<q$
- $\varnothing_q(\infty)=f$ se $q<\omega$
  $=u$ se $q=\omega$

Claramente, para cada q, $\varnothing_q$ é contínua, o núcleo de $\varnothing_q$ é o conjunto vazio, e o conúcleo de $\varnothing_q$ é $\mathbb{N}$. Logo, $\varnothing_q$ é completa. □

De acordo com a discussão a seguir, $\varnothing_\omega$ não fornece informação tão boa quanto $\varnothing_0$ a respeito do conjunto vazio.

▷ Um subconjunto de $\mathbb{N}^k$ é dito *cofinito* sss o seu complemento com relação a $\mathbb{N}^k$ é finito. Um subconjunto de $\mathbb{N}^k$ é dito *propriamente infinito* sss ele não é finito nem cofinito.

O lema abaixo pode ser facilmente generalizado para $A\subseteq\mathbb{N}^k$.

Lema 4.3.4. Se $A\in\mathbf{p}$, para $A\in\mathbb{N}$, então:

- $\mathbf{p}(\underline{\mathbf{n}})=f$ implica $m\notin A$ para todo $m\geq n$, e portanto que A é finito
- $\mathbf{p}(\underline{\mathbf{n}})=t$ implica $m\in A$ para todo $m\geq n$, e portanto que A é cofinito

Prova. É provada a primeira afirmação. Como $\mathbf{p}$ é monotônica, $\mathbf{p}(\underline{\mathbf{n}})=f$ implica $\mathbf{p}(\mathbf{x})=f$ para todo $\mathbf{x}$ tal que $\underline{\mathbf{n}}\leq\mathbf{x}$. Em particular, isto vale para $\mathbf{x}=\mathbf{m}$ para todo $m\geq n$. □

Corolário 4.3.4. Nas condições do lema 4.3.4:

- $\mathbf{p}(\infty)=f$ implica A é finito
- $\mathbf{p}(\infty)=t$ implica A é cofinito

Prova. É provada a primeira afirmação. Como $\mathbf{p}$ é contínua, $\mathbf{p}(\infty)=f$ implica existe $\mathbf{x}\leq\infty$ finito tal que $\mathbf{p}(\mathbf{x})=f$. Tal $\mathbf{x}$ tem

que ser da forma $\underline{\mathbf{n}}$. □

A interpretação deste lema é que o valor $\mathbf{p}(\mathbf{n})$ informa sobre a presença individual de n em A, e que o valor $\mathbf{p}(\underline{\mathbf{n}})$ informa sobre a presença coletiva em A dos $m \geq n$. Isto é, $\mathbf{p}(\underline{\mathbf{n}})$ deve ser lido como $\mathbf{p}$("números que são no mínimo n"). O valor de $\mathbf{p}(\infty)$ é a acumulação dos valores $\mathbf{p}(\underline{\mathbf{n}})$.

Lema 4.3.5. Nas condições do lema anterior:

- se $n \in A$, $m \notin A$, $k \leq n$ e $k \leq m$, então $\mathbf{p}(\underline{\mathbf{k}}) = \mathbf{u}$

Prova. Se $\mathbf{p}(\mathbf{n})$ ou $\mathbf{p}(\mathbf{m})$ valem u, então, como $\underline{\mathbf{k}} \leq \mathbf{n},\mathbf{m}$ e $\mathbf{p}$ é monotônica, $\mathbf{p}(\underline{\mathbf{k}}) = \mathbf{u}$. Caso contrário, $\mathbf{p}(\mathbf{n}) = \mathbf{t}$ e $\mathbf{p}(\mathbf{m}) = \mathbf{f}$. De novo por monotonicidade, $\mathbf{p}(\underline{\mathbf{k}}) \leq \mathbf{t}$ e $\mathbf{p}(\underline{\mathbf{k}}) \leq \mathbf{f}$. Logo, $\mathbf{p}(\underline{\mathbf{k}}) = \mathbf{u}$.

Corolário 4.3.5. Se A é propriamente infinito, então $\mathbf{p}(\underline{\mathbf{k}}) = \mathbf{u}$ para todo $\underline{\mathbf{k}}$, e, conseqüentemente, $\mathbf{p}(\infty) = \mathbf{u}$. □

A interpretação do lema 4.3.5 é que, para k nas condições do lema:

- $\mathbf{p}(\underline{\mathbf{k}}) = \mathbf{p}$("números que são no mínimo k")
  $= \mathbf{u}$
  = "t para alguns, f para outros"

A definição a seguir seleciona a função $\mathbb{P}$-característica parcial que fornece mais informação como a função $\mathbb{P}$-característica.

▷ *A função $\mathbb{P}$-característica* de um conjunto $A \subseteq \mathbb{N}^k$ é a sua função $\mathbb{P}$-característica parcial $\mathbf{X}_A : \mathbb{P}^k \to \mathbb{B}$ tal que, para todos $\mathbf{x}_1, \ldots, \mathbf{x}_k$ finitos:

· $\mathbf{X}_A(\mathbf{x}_1,\ldots,\mathbf{x}_k)=$t se para todos $x_1,\ldots,x_k \in \mathbb{N}$ tais que $x_1\in\mathbf{x}_1,\ldots,x_k\in\mathbf{x}_k$, $(x_1,\ldots,x_k)\in A$

=f se para todos $x_1,\ldots,x_k \in \mathbb{N}$ tais que $x_1\in\mathbf{x}_1,\ldots,x_k\in\mathbf{x}_k$, $(x_1,\ldots,x_k)\notin A$

=u caso contrário.

A unicidade de $\mathbf{X}_A$ sai do lema 3.2.4 e corolário.

Proposição 4.3.6. A função $\mathbb{P}$-característica de um conjunto é a extensão natural da sua função característica.

Prova. Teorema 3.4.5. □

Corolário 4.3.6. A função $\mathbb{P}$-característica de um conjunto é maximal. □

Mas:

Proposição 4.3.7. A função $\mathbb{P}$-característica de um conjunto propriamente infinito é a extensão trivial da sua função característica.

Prova. Corolário 4.3.5. □

Corolário 4.3.7. A função característica $\mathbb{N}^k\to\mathbb{N}$ de um conjunto propriamente infinito tem uma única extensão para $\mathbb{P}^k\to\mathbb{B}$. □

O lema 4.3.8 a seguir determina funções $\mathbb{P}$-características para o caso k=1.

Lema 4.3.8. A função $\mathbb{P}$-característica de um conjunto $A\subseteq\mathbb{N}$ é a sua função característica parcial $\mathbf{X}_A:\mathbb{P}\to\mathbb{B}$ tal que:

- $\mathbf{X}_A(\mathbf{n})=$t se $n\in A$
  $=$f se $n\notin A$
- $\mathbf{X}_A(\underline{\mathbf{n}})=$t se para todo $m\geq n$, $m\in A$
  $=$f se para todo $m\geq n$, $m\notin A$
  $=$u se existem $m,m'\geq n$ tal que $m\in A$, $m'\notin A$
- $\mathbf{X}_A(\infty)=$t se A é cofinito
  $=$f se A é finito
  $=$u se A é propriamente infinito

Prova. Imediata da definição e do lema 4.3.4 e corolário. □

A seguinte definição está baseada na identificação de $\mathbb{M}$ com $\mathbb{N}$, e é utilizada no capítulo **8**.

▷ A função $\mathbb{P}$-característica de um conjunto $\mathbf{A}\subseteq\mathbb{M}^k$ é a função $\mathbb{P}$-característica do conjunto $A\subseteq\mathbb{N}^k$ definido por:

- $A=\{(n_1,\ldots,n_k)\mid(\mathbf{n}_1,\ldots,\mathbf{n}_k)\in\mathbf{A}\}$

▷ A função $\mathbb{P}$-característica da relação de igualdade de naturais é escrita **==**.

Lema 4.3.9. Para todos **n**,**m**:

- $(\mathbf{n}==\mathbf{m})$ = t se $n=m$
  = f caso contrário
- $(\mathbf{n}==\underline{\mathbf{m}})$ = $(\underline{\mathbf{m}}==\mathbf{n})$ = f se $n<m$
  = u caso contrário
- $(\infty==\mathbf{n})$ = $(\mathbf{n}==\infty)$ = f
- $(\underline{\mathbf{n}}==\underline{\mathbf{m}})$ = $(\underline{\mathbf{n}}==\infty)$ = $(\infty==\underline{\mathbf{n}})$ = $(\infty==\infty)$ = u

Prova. Os casos que não envolvem ∞ saem direto da definição de função $\mathbb{P}$-característica. Os casos que envolvem ∞ são obtidos por continuidade. □

A função $\mathbf{p}:\mathbb{P}^2\to\mathbb{B}$ tal que:

- $\mathbf{p}(\mathbf{x},\mathbf{y})=t$ se $\mathbf{x}=\mathbf{y}$
  $=f$ caso contrário

não é monotônica. Se **p** fosse monotônica, $\mathbf{p}(\mathbf{x},\mathbf{y})$ deveria valer t para todos $\mathbf{x},\mathbf{y}\in\mathbb{P}$, pois $\mathbf{p}(\underline{0},\underline{0})=t$, e $(\underline{0},\underline{0}) \leq (\mathbf{x},\mathbf{y})$, o que contradiz a definição de **p**. Isto motiva a seguinte definição.

▷ Uma função $\mathbf{p}:\mathbb{P}^2\to\mathbb{B}$ é dita uma *função de igualdade* se:

- $\mathbf{p}(\mathbf{x},\mathbf{y})=t$ implica $\mathbf{x}=\mathbf{y}$
- $\mathbf{p}(\mathbf{x},\mathbf{y})=f$ implica $\mathbf{x}\neq\mathbf{y}$

Claramente, == é uma função de igualdade. Como == é maximal, por ser uma extensão natural, == é a função de igualdade mais definida, e, portanto, o supremo das funções de igualdade. Que a igualdade contínua mais definida se comporte do modo exposto no lema é extremamente razoável. O fato de duas informações serem a mesma não quer dizer que o objeto sobre o qual informam seja o mesmo. Por exemplo, que $\underline{3}=\underline{3}$ (igualdade de informações) não quer dizer que quaisquer dois números maiores que 3 sejam iguais. Portanto, não vale que $(\underline{3}==\underline{3})=t$ (igualdade de objetos, via informações sobre eles), e vale que $(\underline{3}==\underline{3})=u$.

O seguinte lema é utilizado em um capítulo adiante para computar ==.

Lema 4.3.10. Para todos **x,y**:

- **(0==0)** = t
- **(0==y+1)** = f
- **(x+1==0)** = f
- **(x+1==y+1)** = **(x==y)**

Ainda, == é a única função contínua que satisfaz a equação:

- **(x==y)** = **cond(x,cond(y,(x-1==y-1),f),**
  cond(y,f,t))

Prova. Primeira parte, por casos $\mathbf{n}, \underline{\mathbf{n}}, \infty$ para $\mathbf{x}$ e $\mathbf{y}$, utilizando lema 4.3.9. Segunda parte, por indução fraca, uma vez que a função é contínua (lema 3.2.4 e corolário). □

O seguinte lema também é utilizado em um capítulo adiante.

Lema 4.3.11. A função $\mathbb{P}$-característica de um conjunto finito $A=\{n_1,\ldots,n_k\}\subseteq\mathbb{N}$ satisfaz:

- $\mathbf{X}_A(\mathbf{x}) = \mathbf{x}==\mathbf{n}_1 \vee \ldots \vee \mathbf{x}==\mathbf{n}_k$

Prova. Por indução em $\mathbb{N}$ na cardinalidade de A. □

4.4 Quantificadores existencial e universal

▷ Uma função contínua $\mathbf{e}:(\mathbb{P}\to\mathbb{B})\to\mathbb{B}$ é dita um *quantificador existencial* sss para toda $\mathbf{p}:\mathbb{P}\to\mathbb{B}$ contínua:

- $\mathbf{e}(\mathbf{p})=t$ implica existe $\mathbf{n}$ tal que $\mathbf{p}(\mathbf{n})=t$
- $\mathbf{e}(\mathbf{p})=f$ implica para todo $\mathbf{n}$, $\mathbf{p}(\mathbf{n})=f$

A função $\mathbf{p}$ é pensada como uma função $\mathbb{P}$-característica parcial. Intuivamente, um quantificador existencial quantifica existencialmente sobre o núcleo de $\mathbf{p}$, e universalmente sobre o conúcleo de $\mathbf{p}$.

Lema 4.4.1. O conjunto $\mathbf{E}$ de quantificadores existenciais é dirigido e seu supremo e é um quantificador existencial.

Prova. A função $(\mathbb{P}\to\mathbb{B})\to\mathbb{P}$ constante u é um quantificador existencial, logo $\mathbf{E}$ é não vazio. Basta, então, mostrar diretamente que o supremo de $\mathbf{E}$ existe. Sejam $\mathbf{e},\mathbf{f}\in\mathbf{E}$, e $\mathbf{p}\in[\mathbb{P}\to\mathbb{B}]$. Se $\mathbf{e}(\mathbf{p})=t$, então $\mathbf{f}(\mathbf{p})\neq f$, pois $\mathbf{e}(\mathbf{p})=t$ implica existe $\mathbf{n}$ tal que $\mathbf{p}(\mathbf{n})=t$, e $\mathbf{f}(\mathbf{p})=f$ implicaria para todo $\mathbf{n}$, $\mathbf{p}(\mathbf{n})=f$. Analogamente, se $\mathbf{e}(\mathbf{p})=f$, então $\mathbf{f}(\mathbf{p})\neq t$. Logo, a função **exist** definida a seguir é bem definida:

· **exist**(**p**)=t se para algum **e**∈**E**, **e**(**p**)=t
=f se para algum **e**∈**E**, **e**(**p**)=f
=u se para todo **e**∈**E**, **e**(**p**)=u

Evidentemente, **exist** é um quantificador existencial, e **exist** satisfaz:

· **exist**(**p**)=$\bigsqcup\{\mathbf{e}(\mathbf{p}) \mid \mathbf{e}\in\mathbf{E}\}$

Pelo lema 3.2.8, **exist** é o supremo de **E**. □

Assim, **exist** é o quantificador existencial mais definido. O lema a seguir caracteriza **exist** explicitamente.

Lema 4.4.2. O quantificador existencial **exist** satisfaz:

· **exist**(**p**)=t se existe **n** tal que **p**(**n**)=t
=f se para todo **n**, **p**(**n**)=f, mas **p**≠$\varnothing_\omega$
=u caso contrário

Prova. (Ver lema 4.3.3 para definição de $\varnothing_\omega$). Uma função que satisfaz as equações acima, se for contínua, é um quantificador existencial. Se não fosse pela excessão **p**≠$\varnothing_\omega$, seria evidente que este seria o quantificador existencial mais definido. Esta excessão é necessária para que **exist** seja contínuo. Se existe **n** tal que **p**(**n**)=t, então **p**'=(**n**⟹t) é uma parte finita de **p**, tal que existe **n** com **p**'(**n**)=t. Logo, **exist** é contínuo para tal **p**. Se para todo **n**, **p**(**n**)=f, então **p** ≤ $\varnothing_0$ e, pelo lema 3.2.10, **p** é finito sss **p**≠$\varnothing_\omega$. Se **p**=$\varnothing_\omega$, nenhuma parte finita de **p** satisfaz a condição da primeira ou segunda equações, logo, se exist(**p**)=f, então **exist** não seria contínuo para este **p**. Para o último caso, se **exist**(**p**)=u, então qualquer parte finita de **p** recai neste caso. □

O corolário a seguir mostra como **exist** funciona operacionalmente.

Corolário 4.4.2. O quantificador existencial **exist** satisfaz:

· **exist**(**p**)=t se existe $n\in\mathbb{N}$ tal que **p**(**n**)=t

=f se existe $n\in\mathbb{N}$ tal que **p**(**n**)=f
e para todo $m\leq n$, **p**(**m**)=f

=u caso contrário

Prova. Lema 3.2.12. □

A noção de quantificador universal pode ser definida meramente trocando t por f e f por t da definição de quantificador existencial.

▷ Uma função contínua **a**:(ℙ→𝔹)→𝔹 é dita um *quantificador universal* sss para toda **p**:ℙ→𝔹 contínua:

· **a**(**p**)=f implica existe **n** tal que **p**(**n**)=f
· **a**(**p**)=t implica para todo **n**, **p**(**n**)=t

Assim, a discussão sobre quantificação existencial é dual à discussão sobre quantificação universal. Em particular, **e** é um quantificador existencial sss a função **a** definida por:

· **a**(**p**)=¬**e**(¬**p**) onde (¬**p**)(**x**)=¬**p**(**x**)

é um quantificador universal. Isto ocorre porque o núcleo de **p** é o conúcleo de ¬**p**, e o conúcleo de **p** é o núcleo de ¬**p**. Assim, o quantificador universal **all** definido abaixo é facilmente provado maximal:

· **all**(**p**)=¬**exist**(¬**p**)

4.5 Operador de ponto fixo

▷ Um *ponto fixo* de uma função $f:D\to D$ é um $x\in D$ tal que:

· $f(x)=x$

Lema 4.5.1. Toda função contínua $f:D\to D$ tem um ponto fixo x

tal que:

- $x=\bigsqcup\{f^n(\bot)\,|\,n\in\mathbb{N}\}$
- x é o menor ponto fixo de f, i.e.:
  - x' é ponto fixo de f implica $x \leq x'$

Este ponto fixo é escrito $fix_D\, f$. Ainda, a função $fix_D:(D\to D)\to D$ é contínua.

Prova. Ver [PLO 80]. □

4.6Funções aritméticas

▷ Sejam **+,-** as respectivas extensões naturais das funções +,- de adição e subtração, onde se entende que $n-m=0$ se $n\leq m$.

Lema 4.7.2. Se $m+n=k$, então:

- $\mathbf{n+m} = \mathbf{k}$
- $\underline{\mathbf{n}}+\mathbf{m} = \mathbf{n}+\underline{\mathbf{m}} = \underline{\mathbf{n}}+\underline{\mathbf{m}} = \underline{\mathbf{k}}$
- $\mathbf{x}+\infty = \infty+\mathbf{x} = \infty$

Prova. São provados os casos $\underline{\mathbf{n}}+\mathbf{m}=\underline{\mathbf{k}}$ e $\infty+\mathbf{x}=\infty$. Pelo teorema 3.4.5:

- $\underline{\mathbf{n}}+\mathbf{m}=\mathcal{C}_{\mathbb{P}}\{n+m,\ (n+1)+m,\ (n+2)+m,\ \ldots\}$
  $=\mathcal{C}_{\mathbb{P}}\{k,\ k+1,\ k+2,\ \ldots\}$
  $=\underline{\mathbf{k}}.$

Pelo lema 3.2.5, para $\mathbf{x}\in\mathbb{F}$, $\infty+\mathbf{x}=\bigsqcup\{\underline{\mathbf{n}}+\mathbf{x}\,|\,n\in\mathbb{N}\}=\infty$. Para $\mathbf{x}=\infty$, utiliza-se de novo o lema 3.2.5. □

A interpretação deste lema é:

- "exatamente n" + "exatamente m" = "exatamente $n+m$"
- "no mínimo n" + "exatamente m" = "no mínimo $n+m$"
- etc.

Lema 4.7.2. A função de adição **+** é unicamente caracterizada

pelas equações:

- $\mathbf{x+\underline{0} = \underline{x}}$
- $\mathbf{x+0 = x}$
- $\mathbf{x+succ(y) = succ(x+y)}$

Prova Por indução fraca em **x**, utilizando o lema 4.7.1. □

Lema 4.7.3 Se $n-m=k$, então:

- $\mathbf{n-m=k}$
- $\mathbf{n-\underline{m}=0}$ se $n \leq m$
  $\mathbf{=\underline{0}}$ se $n > m$
- $\mathbf{n-\infty=0}$
- $\mathbf{\underline{n}-\underline{m}=\underline{k}}$
- $\mathbf{\underline{n}-\underline{m} = \underline{n}-\infty = \infty-\underline{n} = \infty-\infty = \underline{0}}$
- $\mathbf{\infty-n=\infty}$

Prova. Como no lema 4.7.1, utiliza-se o teorema 3.4.5 para provar casos envolvendo $\underline{\mathbf{n}},\underline{\mathbf{m}}$, e o lema 3.2.5 para provar casos envolvendo ∞. □

5 MANIPULAÇÃO FORMAL DE NÚMEROS PARCIAIS E FUNÇÕES

Este capítulo define uma linguagem, denominada Π, que permite expressar formalmente certas funções contínuas de números parciais, bem como computar formalmente estas funções. Esta linguagem é um cálculo-λ tipado com constantes [COS 90], no estilo da linguagem **PCF** [PLO 77]. A seção 5.1 apresenta a noção geral de cálculo-λ tipado com constantes. A sintaxe e a semântica destes cálculos são estudadas. A seção 5.2 apresenta a linguagem Π. A seção 5.3 estuda computabilidade formal em Π. A seção 5.4 estuda programas e computações. A seção 5.5 estuda brevemente a execução interativa de programas Π.

5.1 Cálculo-λ tipado com constantes

5.1.1 Sintaxe

▷ Dado um alfabeto de *tipos básicos*, o conjunto de *tipos* é o menor conjunto que contém os tipos básicos, e que contém $(\sigma\to\tau)$ quando contém σ e τ. As letras gregas σ, τ correm sobre tipos. Os parênteses são omitidos com a convenção de que $\to$ associa à direita.

▷ A partir de uma coleção $\mathcal{K}$ de *constantes*, cada uma associada a um único tipo, e uma quantidade enumerável de variáveis α_i^σ, $i\in\mathbb{N}$, para cada tipo σ, os *termos-*$\mathcal{K}$ são dados pelas regras:

- toda constante de tipo σ é um termo-$\mathcal{K}$ de tipo σ
- toda variável α_i^σ é um termo-$\mathcal{K}$ de tipo σ
- se M e N são termos-$\mathcal{K}$ de tipo $(\sigma\to\tau)$ e σ respectivamente, então (MN) é um termo-$\mathcal{K}$ de tipo τ
- se M é um termo-$\mathcal{K}$ de tipo τ, então $(\lambda\alpha_i^\sigma.M)$ é um termo-$\mathcal{K}$ de tipo $(\sigma\to\tau)$

▷ A expressão M é *um termo-$\mathcal{K}$ do tipo* τ é abreviada por $M{:}\tau$ quando $\mathcal{K}$ é subentendido do contexto. Na mesma situação, o sufixo $\mathcal{K}$ é omitido. O conjunto de termos-$\mathcal{K}$ é dito *linguagem* $\mathcal{K}$. As letras M,N, e outras maiúsculas em itálico, correm sobre os termos-$\mathcal{K}$, c corre sobre $\mathcal{K}$ e α corre sobre as variáveis. Os índices de tipo são omitidos quando não há ambigüidade possível.

▷ A relação de *ocorrência livre de variáveis em termos* é a menor relação tal que:

- α ocorre livre em α
- α ocorre livre em (MN) se α ocorre livre em M ou N
- α ocorre livre em $(\lambda\alpha'.M)$ se $\alpha\neq\alpha'$ e α ocorre livre em M

▷ Termos da forma (MN) são ditos *combinações*. Os parênteses de combinações podem ser omitidos, com a convenção de que elas associam à esquerda. Termos da forma $(\lambda\alpha.M)$ são ditos *abstrações*. Termos sem ocorrências de variáveis livres são ditos *termos fechados* ou *combinadores*.

5.1.2 Semântica

▷ Uma *interpretação* de uma linguagem $\mathcal{K}$ é uma coleção $\{\mathcal{C}_\sigma\}$ de conjuntos, um para cada tipo σ, junto com uma função $\mathcal{A}{:}\mathcal{K}\to\bigcup\{\mathcal{C}_\sigma\}$, tais que:

- $\mathcal{C}_{(\sigma\to\tau)}$ é subconjunto de $\mathcal{C}_{\tau}^{\mathcal{C}_{\sigma}}$, o conjunto de funções com conjunto de partida $\mathcal{C}_{\sigma}$ e conjunto de de chegada $\mathcal{C}_{\tau}$
- $\{\mathcal{C}_{\sigma}\}$ é fechada para aplicação definida, i.e., se $f\in\mathcal{C}_{(\sigma\to\tau)}$ e $x\in\mathcal{C}_{\sigma}$, então $f(x)\in\mathcal{C}_{\tau}$
- $\mathcal{A}$ preserva tipos, i.e., se $c:\tau$ então $\mathcal{A}(c)\in\mathcal{C}_{\tau}$

▷ O conjunto $\mathcal{C}=\cup\{\mathcal{C}_{\sigma}\}$ é dito *universo de discurso de* $\mathcal{K}$. A operação de *aplicação* $\mathcal{C}^2\to\mathcal{C}$ toma argumentos f e x, e tem valor $f(x)$. Esta operação é parcial, e é definida para f e x sss existem σ e τ tais que $f\in\mathcal{C}_{\sigma\to\tau}$ e $x\in\mathcal{C}_{\sigma}$. Se a aplicação não é definida para f e x, então $f(x)$ não tem significado, assim como 3(4) e 1/0 não têm significado na aritmética. A aplicação $f(x)$ é escrita simplesmente fx, e se assume que ela associa à esquerda. Por exemplo, $fxyz$ e $f(x)(y)(z)$ abreviam $((fx)y)z$.

▷ Uma interpretação $\mathcal{I}=\langle\{\mathcal{C}_{\sigma}\},\mathcal{A}\rangle$ de $\mathcal{K}$ *induz* uma *semântica* $\hat{\mathcal{A}}$ para a linguagem $\mathcal{K}$, do seguinte modo.

▷ Um *ambiente* é uma função que preserva tipos, do conjunto de variáveis para $\mathcal{C}$. Se $x\in\mathcal{C}_{\sigma}$, e ρ é um ambiente, então $\rho[x/\alpha]$ é o ambiente ρ' tal que:

- $\rho'(\alpha')=x$ se $\alpha'=\alpha$
  $=\rho(\alpha')$ caso contrário

▷ A *semântica* $\hat{\mathcal{A}}$:termo→ambiente→$\mathcal{C}$ é dada indutivamente abaixo, onde termos são escritos entre $[\![$ e $]\!]$:

- $\hat{\mathcal{A}}[\![c]\!](\rho)=\mathcal{A}[\![c]\!]$
- $\hat{\mathcal{A}}[\![\alpha]\!](\rho)=\rho(\alpha)$
- $\hat{\mathcal{A}}[\![MN]\!](\rho)=\hat{\mathcal{A}}[\![M]\!](\rho)(\hat{\mathcal{A}}[\![N]\!](\rho))$
- $\hat{\mathcal{A}}[\![\lambda\alpha.M]\!](\rho)(x)=\hat{\mathcal{A}}[\![M]\!](\rho[x/\alpha])$, para $x\in\mathcal{C}_{\sigma}$ e $\alpha:\sigma$

▷ Se a semântica de M, relativa a um dado ambiente, é x, se diz que *M denota x* neste ambiente. Se M não tem variáveis livres, a sua semântica não depende do ambiente. Neste caso, se omite a referência ao ambiente. Se M e N têm a mesma denotação, para cada ambiente, em uma interpretação $\mathcal{I}$, escreve-se $M\equiv_{\mathcal{I}}N$. É imediato que $\equiv_{\mathcal{I}}$ é relação de equivalência. Se para toda interpretação $\mathcal{I}$, $M\equiv_{\mathcal{I}}N$, escreve-se $M\equiv N$

Lema 5.1.2.1. Para todos termos M,N,M',N':

- $\lambda\alpha.M\equiv\lambda\alpha.M'$ sss $M\equiv M'$ (extensionalidade)
- $(MN)\equiv(M'N')$ se $M\equiv M'$ e $N\equiv N'$ (substitutividade)

Prova. $M\equiv M'$ sss $\mathcal{A}(M)(\rho)=\mathcal{A}(M')(\rho)$ para todos $\mathcal{A}$ e ρ sss $\mathcal{A}(M)=\mathcal{A}(M')$para todo $\mathcal{A}$.

(extensionalidade) $M\equiv M'$ sss:

- $\mathcal{A}(\lambda\alpha.M)(\rho)(x)=\hat{\mathcal{A}}(M)(\rho[x/\alpha])=\hat{\mathcal{A}}(M')(\rho[x/\alpha])$
  $=\mathcal{A}(\lambda\alpha.M')(\rho)(x)$,

e $\lambda\alpha.M\equiv\lambda\alpha.M'$.

(substitutividade) Se $M\equiv M'$ e $N\equiv N'$, então:

- $\mathcal{A}(MN)(\rho)=\mathcal{A}(M)(\rho)(\hat{\mathcal{A}}(N)(\rho))=\hat{\mathcal{A}}(M')(\rho)(\hat{\mathcal{A}}(N')(\rho))$
  $=\mathcal{A}(M'N')(\rho)$ □

5.1.3 Convertibilidade

▷ A *susbtituição de α por N em M*, escrita $M[N/\alpha]$, é definida indutivamente por:

- $c[N/\alpha]=c$
- $\alpha[N/\alpha]=N$
- $\alpha[N/\alpha']=\alpha$, se $\alpha\neq\alpha'$
- $(MM')[N/\alpha]=(M[N/\alpha])\ (M'[N/\alpha])$
- $(\lambda\alpha.M)[N/\alpha]=\lambda\alpha.M$
- $((\lambda\alpha^{\sigma}_{j}.M)[N/\alpha^{\tau}_{i}]=(\lambda\alpha^{\sigma}_{k}.M[\alpha^{\sigma}_{k}/\alpha^{\sigma}_{j}][N/\alpha^{\tau}_{i}])$ se $\alpha^{\sigma}_{j}\neq\alpha^{\tau}_{i}$, para o menor k tal que α^{σ}_{k} que não ocorre livre em N nem em M

Lema 5.1.3.1 . Para todos termos M,N,M',N':

- $(\lambda\alpha.M) \equiv \lambda\alpha'.M[\alpha'/\alpha]$ se α' não ocorre livre em M
- $((\lambda\alpha.M)N)\equiv M[N/\alpha]$
- $\lambda\alpha.(M\alpha)\equiv M$ se α não ocorre livre em M

Prova. As duas primeiras afirmações são provadas por indução na regra de formação dos termos. A terceira é provada como o lema 5.1.2.1. □

▷ As relações de *redução imediata* α, β e η, escritas, $\Rightarrow_{\alpha'}$ $\Rightarrow_{\beta}$ e $\Rightarrow_{\eta}$ são definidas por:

- $((\lambda\alpha.M) \Rightarrow_{\alpha} \lambda\alpha'.M[\alpha'/\alpha]$ se α' não ocorre livre em M
- $((\lambda\alpha.M)N)\Rightarrow_{\beta} M[N/\alpha]$
- $(\lambda\alpha.M\alpha)\Rightarrow_{\eta} M$ se α não ocorre livre em M

▷ A relação de redução imediata $\alpha\beta\eta$, escrita $\Rightarrow_{\alpha\beta\eta}$, é a união das relações de redução imediata α,β e η. M é dito um *$\alpha\beta\eta$-redex* se existe M' tal que $M\Rightarrow_{\alpha\beta\eta}M'$.

▷ Uma relação binária R entre termos *preserva denotação* sss:

- M denota x e $M\ R\ M'$ implica M' denota x

Pelo lema 5.1.3.1, é imediato que $\Rightarrow_{\alpha\beta\eta}$ preserva

denotação.

▷ Uma relação binária $\Leftrightarrow$ entre termos é dita *relação de convertibilidade* sss:

- $\Leftrightarrow$ preserva denotação
- $\Leftrightarrow$ é relação de equivalência
- $\lambda\alpha.M \Leftrightarrow \lambda\alpha.M'$ quando $M \Leftrightarrow M'$
- $MN \Leftrightarrow M'N'$ quando $M \Leftrightarrow M'$ e $N \Leftrightarrow N'$

▷ O *fecho de convertibilidade* de uma relação binária R entre termos, quando existe, é a menor relação $\Leftrightarrow$ que contém R e é relação convertibilidade.

Pelo lema 5.1.2.1, este fecho existe sss R preserva denotação.

5.2 A linguagem Π

▷ A linguagem Π é dada pelo alfabeto de tipos básicos $\{\pi\}$ e pelo conjunto de constantes:

- $0:\pi$
- $SUCC:\pi\rightarrow\pi$
- $PRED:\pi\rightarrow\pi$
- $COND:\pi^{(3)}\rightarrow\pi$
- $Y_\sigma:(\sigma\rightarrow\sigma)\rightarrow\sigma$ para cada tipo σ

onde expressão $\sigma^{(n)}\rightarrow\tau$, para cada n, abrevia o tipo definido indutivamente por:

- $\sigma^{(0)}\rightarrow\tau=\tau$
- $\sigma^{(n+1)}\rightarrow\tau=\sigma\rightarrow(\sigma^{(n)}\rightarrow\tau)$

5.2.1 Interpretação padrão de Π, Π-definibilidade

▷ A interpretação *padrão* de Π é $\langle \mathcal{A}, \{\mathcal{C}_\sigma\}\rangle$ tal que:

- $\mathcal{C}_\pi = \mathbb{P}$
- $\mathcal{C}_{(\sigma\to\tau)} = [\mathcal{C}_\sigma \to \mathcal{C}_\tau]$, o conjunto de funções contínuas de $\mathcal{C}_\sigma$ para $\mathcal{C}_\tau$.
- $\mathcal{A}(0) = \mathbf{0}$
- $\mathcal{A}(SUCC) = \mathbf{succ}$
- $\mathcal{A}(PRED) = \mathbf{pred}$
- $\mathcal{A}(COND)(\mathbf{x})(\mathbf{y})(\mathbf{z}) = \mathbf{cond}(\mathbf{x},\mathbf{y},\mathbf{z})$
- $\mathcal{A}(Y_\sigma) = \mathbf{fix}_{\mathcal{C}_\sigma}$

▷ O elemento menos definido de $\mathcal{C}_\sigma$ é escrito $\perp_\sigma$. Os elementos $\perp_\sigma$ satisfazem:

- $\perp_\pi = \underline{\mathbf{0}}$
- $\perp_{\sigma\to\tau}(x) = \perp_\tau$ para todo $x \in \mathcal{C}_\sigma$

▷ A semântica induzida pela interpretação padrão é dita padrão. A semântica padrão de um termo é dita sua denotação padrão. A relação de igualdade de denotação padrão para qualquer ambiente é escrita $\equiv_\Pi$.

No que segue do texto, a linguagem Π é estudada relativamente à sua interpretação padrão.

▷ Um elemento $x \in \mathcal{C}_\sigma$ é dito Π-*definível* sss existe um combinador de tipo σ cuja denotação padrão é x.

5.2.2 Convertibilidade-Π

▷ As relações de redução imediata *p*, *c* e *y* entre termos, escritas respectivamente $\Rightarrow_p$, $\Rightarrow_c$ e $\Rightarrow_y$, são definidas por:

- $PRED\ 0 \Rightarrow_p 0$
- $PRED\ (SUCC\ M) \Rightarrow_p M$
- $COND\ 0\ M\ N \Rightarrow_c N$
- $COND\ (SUCC\ P)\ M\ N \Rightarrow_c M$
- $COND\ P\ 0\ 0 \Rightarrow_c 0$
- $COND\ P\ (SUCC\ M)\ (SUCC\ N) \Rightarrow_c (SUCC\ (COND\ P\ M\ N))$
- $YM \Rightarrow_y M(YM)$

Lema 5.2.2.1. As relações de redução p, c e y preservam denotação padrão.

Prova. O único caso não imediato é o da relação de redução c, que é estabelecido pelo lema 3.1.3. □

▷ M é dito um Π-*redex* sss existe M' tal que $M \Rightarrow M'$ para alguma das relações de redução p, c e y. M é dito um *redex* sss M é um $\alpha\beta\eta$-redex ou M é um Π-redex. A relação de redução imediata é a união das relações de redução imediata α, β, η, p, c e y, escrita $\Rightarrow_\Pi$.

▷ Pelo lema 5.2.2.1, o fecho de convertibilidade da relação de redução imediata existe. Ele é escrito $\Leftrightarrow_\Pi$ e é denominado *convertibilidade*-Π. Dois termos M e N tais que $M \Leftrightarrow_\Pi N$ são ditos Π-*interconvertíveis*. O subscrito Π é omitido dos símbolos de relação $\Rightarrow$, $\Leftrightarrow$ e $\equiv$ quando o contexto não permite ambigüidade.

Nem todos os termos que têm a mesma denotação são interconvertíveis. $(Y(\lambda\alpha.\alpha))$ e $(Y\ PRED)$ ambos denotam $\underline{0}$ e não são interconvertíveis. Na seção 8.3.4 é mostrado que a equivalência de denotação padrão é indecidível, mesmo para termos de tipo π.

5.3 Computabilidade

▷ Um *numeral* é um termo da forma $(SUCC^n 0)$, $n \in \mathbb{N}$, onde $(M^n N)$ é definido indutivamente por:

- $(M^0 N) = N$
- $(M^{n+1} N) = (M(M^n N))$

Claramente, o numeral $(SUCC^n 0)$ denota o número parcial **n**.

▷ Os numerais são ditos, também, *formas normais fortes*. Uma *forma normal fraca* é um termo de uma das formas 0 ou $(SUCC\ M)$.

É imediato que uma forma normal forte é uma forma normal fraca, e que uma forma normal fraca denota um número parcial distinto de $\underline{\mathbf{0}}$.

▷ Um termo M *tem forma* N sss M é interconvertível com N.

Por exemplo, $(Y\ SUCC)$ tem forma $(SUCC^2(Y\ SUCC))$. Não se deve confundir uma afirmação "M tem forma N" com uma afirmação "M é da forma N". A primeira indica que M é interconvertível com N, como foi dito. A segunda indica que M e N são o mesmo termo.

▷ Um termo $M{:}\pi$ *exibe explicitamente informação* **x**, ou simplesmente *exibe* **x**, para $\mathbf{x} \in \mathbb{F}$, sss uma das afirmações vale:

- $\mathbf{x} = \mathbf{n}$ e M é da forma $(SUCC^n 0)$
- $\mathbf{x} = \underline{\mathbf{n}}$ e M é da forma $(SUCC^n M')$ para algum M'

Por exemplo, a única informação que $(Y\ SUCC)$ exibe é $\underline{\mathbf{0}}$, pois $(Y\ SUCC) = (SUCC^0(Y\ SUCC))$. O termo $(SUCC^2(Y\ SUCC))$ exibe as informações $\underline{\mathbf{0}}, \underline{\mathbf{1}}, \underline{\mathbf{2}}$, e mais nenhuma outra.

▷ A relação de *geração* entre termos de tipo π e números parciais finitos é definida por:

- M gera **x** sss M tem forma que exibe **x**

Claramente, todo termo de tipo π gera $\underline{0}$. Conseqüentemente, o conjunto de informações que um termo de tipo π gera é não vazio. Como a relação de convertibilidade preserva denotação, e 0 e $(SUCC\ M')$ têm denotação distinta, não é possível que M gere **0** e M gere **x+1**.

Por exemplo, $(Y\ SUCC)$ gera $\underline{\mathbf{n}}$ para todo n, pois $(Y\ SUCC)$ tem forma $(SUCC^n(Y\ SUCC))$ para todo n, que exibe $\underline{\mathbf{n}}$.

Lema 5.3.1. Para todos M,**x**,**y**:

- Se M exibe **x** e M denota **y**, então **x** $\leq$ **y**
- Se M gera **x** e M denota **y**, então **x** $\leq$ **y**

Prova. A relação de convertibilidade preserva denotação. □

▷ A relação de *computabilidade*-Π entre termos e elementos de $\mathcal{C}$ é definida por indução nos tipos dos termos:

- $M{:}\pi$ computa $\mathbf{x}\in\mathbb{P}$ sss:
  - M gera as partes finitas de **x**, e apenas elas
- $M{:}\sigma\to\tau$ computa $f{:}\mathcal{C}_\sigma\to\mathcal{C}_\tau$ sss:
  - para todo $N{:}\sigma$ que computa $x\in\mathcal{C}_\sigma$, (MN) computa $f(x)\in\mathcal{C}_\tau$

▷ Um elemento $x\in\mathcal{C}_\sigma$ é Π-*computável* sss existe um termo $M{:}\sigma$ que computa x.

De imediato, todo número parcial é Π-computável. **n** é computável por $(SUCC^n 0)$. $\underline{\mathbf{n}}$ é computável por $(SUCC^n(Y(\lambda\alpha.\alpha)))$. ∞ é computável por $(Y\ SUCC)$. Porém, existe

$x \in \mathcal{C}_{\sigma\to\tau}$ que não é computável, uma vez que os espaços de funções contínuas são não enumeráveis e existe uma quantidade enumerável de termos. A não enumerabilidade do espaço de funções contínuas $\mathbb{P}\to\mathbb{P}$ sai do fato que a extensão trivial de uma função $\mathbb{N}\to\mathbb{N}$ é contínua, e que há uma quantidade não enumerável de funções $\mathbb{N}\to\mathbb{N}$.

Lema 5.3.2. $M{:}\pi$ computa $\mathbf{x}$ sss $\mathbf{x}=\bigsqcup\{\mathbf{x}' \mid M \text{ gera } \mathbf{x}'\}$.

Prova. Se M gera $\mathbf{x}$ e M gera $\mathbf{y}$, então $\mathbf{x}$ e $\mathbf{y}$ são consistentes. □

Conseqüentemente, se M computa $\mathbf{x}$ e M computa $\mathbf{y}$, então $\mathbf{x}=\mathbf{y}$.

Corolário 5.3.2.

- M computa $\mathbf{n}$ sss M gera $\mathbf{n}$
- M computa $\underline{\mathbf{n}}$ sss M gera $\underline{\mathbf{n}}$ e M gera $\mathbf{x}$ implica $\mathbf{x} \leq \underline{\mathbf{n}}$
- M computa ∞ sss M gera $\underline{\mathbf{n}}$ para todo $n\in\mathbb{N}$
- M computa $\underline{\mathbf{n}}$ sss M tem forma $(SUCC^{n}M')$ para M' que não tem forma normal fraca. □

Em particular, M computa $\underline{\mathbf{0}}$ sss M não tem forma normal fraca.

Lema 5.3.3. Se M fechado computa $\mathbf{x}$ e M denota $\mathbf{y}$, então $\mathbf{x} \leq \mathbf{y}$.

Prova. Lemas 5.3.1 e 5.3.2. □

Teorema 5.3.4. M fechado computa x sss M denota x.

Prova. A prova deste teorema é dada pelos lemas a seguir. O método de prova consiste em mostrar que uma parte da relação de convertibilidade é capaz de realizar a computação da denotação de M. □

▷ A função de *passo de computação* *passo*:termos→termos é

definida indutivamente por:

- $passo(c)=c$
- $passo(\alpha)=\alpha$
- $passo(\lambda\alpha.M)=\lambda\alpha.passo(M)$
- $passo(MN)=(passo\ M)(passo\ N)$ se (MN) não é um redex
- $passo(M)=M'$ se $M \Rightarrow_{det} M'$

onde a relação *redução imediata determinística* $\Rightarrow_{det} \subseteq \Rightarrow$ é a menor relação tal que:

- $M \Rightarrow_{det} M'$ se $M \Rightarrow M'$, e M é da forma $(COND\ P\ N\ N')$ implica P é da forma 0 ou $(SUCC\ P')$
- $COND\ P\ 0\ 0 \Rightarrow_{det} 0$ se P não é da forma 0 ou $(SUCC\ P')$
- $COND\ P\ (SUCC\ M)\ (SUCC\ N) \Rightarrow_{det} SUCC\ (COND\ P\ M\ N)$ se P não é da forma 0 ou $(SUCC\ P')$

Claramente, a função *passo* está incluída propriamente na relação de convertibilidade.

▷ A ordem de redução de redexes que a aplicação sucessiva da função *passo* determina é denominada *ordem Π-normal* de redução.

A ordem Π-normal de redução imposta por *passo* pode ser descrita informalmente por: em cada passo,

- reduzir simultaneamente todos os redexes mais externos
- no caso de que um termo possa ser um redex de mais de uma forma, escolher a primeira forma da lista:
  - *COND* 0 *M N*
  - *COND* (*SUCC P*) *M N*
  - *COND P* 0 0
  - *COND P* (*SUCC M*) (*SUCC N*)

(Estes são os únicos casos de ambigüidade)

A solução da ambigüidade é arbitrária. Qualquer ordem funciona. Porém, a simultaneidade de redução dos redexes mais externos é importante. Isto se deve às regras de redução de *COND*, que permitem que um termo da forma (*COND P M N*) tenha forma normal fraca em casos que *P*, *M* ou *N* não têm forma normal fraca. Isto acontece por que **cond(x,y,z)** pode ter um valor distinto de $\underline{\mathbf{0}}$ em casos que **x**, **y** ou **z** valem $\underline{\mathbf{0}}$. Assim, tentar achar a eventual forma normal fraca de qualquer de *P*, *M* e *N* primeiro pode fazer com que a forma normal fraca de (*COND P M N*) não seja achada em casos que existe.

Lema 5.3.5

- Se *M* exibe **x**, então *passo*(*M*) exibe **x**.
- Se *M* exibe **x**, *M'* resulta de substituir alguns subtermos N_i de *M* por *passo*(N_i), então *M'* exibe **x**.

Prova. As duas asserções decorrem do fato de que *passo*(0)=0 e *passo*(*SUCC P*)=(*SUCC* (*passo P*)). A primeira decorre da segunda. Para a segunda, se **x=n**, então $M=(SUCC^n 0)$, e $M'=M$; se $\mathbf{x}=\underline{\mathbf{n}}$, então $M=(SUCC^n N)$, e $M'=(SUCC^n N')$ que exibe **x**, para certos *N* e *N'*. □

▷ *M gera* **x** *pela função passo* sss existe $i \in \mathbb{N}$ tal que $passo^i(M)$ exibe **x**. Quando não há ambigüidade possível, a

referência à função *passo* é omitida.

Lema 5.3.6. Para todos $P,M,N:\pi$ fechados que geram respectivamente **x,y,z** pela função de passo, $L=(COND\ P\ M\ N)$ gera **cond(x,y,z)** pela função *passo*.

Prova. Por indução fraca em **cond(x,y,z)**. (1) (base da indução) Se **cond(x,y,z)=0**, então L gera **cond(x,y,z)**, por que qualquer termo gera **0**. Se **cond(x,y,z)=0**, então L gera **cond(x,y,z)**. Seja o menor i tal que $passo^i(L)$ é um redex. Então $passo^i(L)$ é de uma das formas:

- (1.1) $(COND\ (SUCC\ P')\ passo^i(M)\ passo^i(N))$
- (1.2) $(COND\ 0\ passo^i(M)\ passo^i(N))$
- (1.3) $(COND\ passo^i(P)\ 0\ 0)$

(1.1) $passo^{i+1}(L)=passo^i(M)$. Seja d tal que $passo^{i+d}(M)$ exibe **y**. Então $passo^{i+1+d}(L)$ exibe **y=cond(x,y,z)**. (1.2) $passo^{i+1}(L)=passo^i(N)$. O argumento é análogo, tomando-se d tal que $passo^{i+d}(N)$ exibe **z**. (1.3) $passo^{i+1}(L)=0$, que exibe **0=cond(x,y,z)**. (2) (passo de indução) Se o lema vale para **cond(x,y,z)**, então ele vale para **cond(x,y,z)+1**. Sejam $\mathbf{x,y,z}\in\mathbb{P}$ e termos fechados L,P,M,N tais que P,M,N têm denotação exibível e $L=(COND\ P\ M\ N)$ denota **cond(x,y,z)+1**. Tome o menor i tal que $passo^i(L)$ é um redex. Então $passo^i(L)$ é de uma das formas:

- (2.1) $(COND\ (SUCC\ P')\ passo^i(M)\ passo^i(N))$
- (2.2) $(COND\ 0\ passo^i(M)\ passo^i(N))$
- (2.3) $(COND\ passo^i(P)\ (SUCC\ M')\ (SUCC\ N'))$

Para os casos (2.1) e (2.2), a hipótese de indução não é utilizada, e os argumentos são como em (1.1) e (1.2). (2.3) $passo^{i+1}(L)=(SUCC\ L')$ onde $L'=(COND\ passo^i(P)\ M'\ N'))$. Neste caso, P,M',N',L' denotam respectivamente **x,y,z** e **cond(x,y,z)**. Pela hipótese de indução, L' tem denotação exibível, e, portanto, L tem denotação exibível. □

▷ A propriedade de termos de *computabilidade de*

denotação (pela função passo) é definida indutivamente por:

- $M:\pi$ fechado, de denotação **x**, tem denotação computável sss para todo $\mathbf{x}' \leq \mathbf{x}$ finito M gera $\mathbf{x}'$ pela função *passo*
- $M:\sigma\to\tau$ fechado tem denotação computável sss $N:\sigma$ fechado tem denotação computável implica (MN) tem denotação computável
- $M:\sigma\to\tau$ com variáveis livres $\alpha_1:\sigma_1,\ldots,\alpha_k:\sigma_k$ tem denotação computável sss $N_1:\sigma_1,\ldots,N_k:\sigma_k$ fechados têm denotação computável implica $M[N_1/\alpha_1]\ldots[N_k/\alpha_k]$ tem denotação computável

A prova do lema 5.3.8 a seguir é baseada na prova do teorema 3.1 de [PLO 77]. Analogamente ao que ocorre em [PLO 77], a única dificuldade da prova é causada pelos operadores de ponto fixo.

▷ Os termos Ω_σ são definidos indutivamente por:

- $\Omega_\pi = Y(\lambda\alpha_0^\pi.\alpha_0^\pi)$
- $\Omega_{\sigma\to\tau} = \lambda\alpha_0^\sigma.\Omega_\tau$

Claramente, Ω_σ denota $\perp_\sigma$.

▷ Os termos $Y_\sigma^{(n)}$ são definidos indutivamente por:

- $Y_\sigma^{(0)} = \Omega_{(\sigma\to\sigma)\to\sigma}$
- $Y_\sigma^{(n+1)} = \lambda\alpha_0^{(\sigma\to\sigma)}.(\alpha_0^{(\sigma\to\sigma)}(Y_\sigma^{(n)}\alpha_0^{(\sigma\to\sigma)}))$

Estes termos aproximam Y_σ, no sentido que o supremo das denotações de $Y_\sigma^{(n)}$, para $n\in\mathbb{N}$, é a denotação de Y_σ.

Lema 5.5.7. Se $passo^1(Y_\sigma^{(n)}N_1\ldots N_k)$ exibe **x**, então $passo^1(Y_\sigma N_1\ldots N_k)$ exibe **x**.

Prova. Por indução em n. Note que $passo(YM)=M(YM)$ e que

$passo(Y^{(n+1)}M)=M(Y^{(n)}M)$. □

Lema 5.4.8. Todo termo tem denotação computável pela função *passo*.

Prova. A prova é por indução nas regras de formação de termos.

(1) Toda variável α tem denotação computável, uma vez que para todo N fechado com denotação computável, $\alpha[N/\alpha]=N$ tem denotação computável.

(2) Todas as constantes distintas de Y_σ têm denotação computável. Para 0 isto é evidente, pois 0 denota **0** e exibe **0**. Para *PRED*, basta mostrar que (*PRED M*) tem denotação computável se M tem denotação computável, para $M:\pi$ fechado que denota **x**. Sejam $\mathbf{x}' \leq \mathbf{x}$ finito e $i\in\mathbb{N}$ tal que $passo^i(M)$ exibe **x**'. Se $\mathbf{x}'=\mathbf{n}$, então $passo^i(M)=(SUCC^n 0)$, e:

- $passo^{i+1}(PRED\ M)=(SUCC^n 0)$ se $n=0$
- $passo^{i+1}(PRED\ M)=(SUCC^{n-1} 0)$ se $n>0$

que exibem **pred**(**n**). Se $\mathbf{x}'=\underline{\mathbf{n}}$, então $passo^i(M)=(SUCC^n M')$ e:

- $passo^{i+1}(PRED\ M)=passo(M')$ se $n=0$
- $passo^{i+1}(PRED\ M)=(SUCC^{n-1} M')$ se $n>0$

que exibem **pred**($\underline{\mathbf{n}}$). Para *SUCC*, a prova é análoga. Para *COND*, basta mostrar que $L=(COND\ P\ M\ N)$ tem denotação computável se P,M,N fechados têm denotação computável. Isto é estabelecido pelo lema 5.3.6.

(3) Se M e N têm denotação computável, então (MN) tem denotação computável. Se (MN) é fechado, então M e N são fechados, e a computabilidade da denotação de (MN) sai da segunda cláusula da definição de computabilidade de denotação. Se (MN) têm variáveis livres, qualquer substituição L das suas variáveis livres por termos fechados com denotação computável tem a forma $(M'N')$ com M' e N' fechados e com denotação computável, pela terceira cláusula. Logo, L tem denotação computável e (MN) por sua vez, pela

terceira cláusula.

(4) Se M tem denotação computável, então $(\lambda\alpha.M)$ tem denotação computável. Basta mostrar que o termo de tipo π $LN_1...N_k$ tem denotação computável quando $N_1,...,N_k$ são termos fechados com denotação computável e L é uma substituição das variáveis livres de $(\lambda\alpha.M)$ por termos de denotação computável. Tal L tem que ter a forma $(\lambda\alpha.M')$, onde M' é uma substituição das variáveis livres de M exceto α, por termos fechados de denotação computável. Como $M'[N_1/\alpha]$ tem denotação computável, $M'[N_1/\alpha]N_2...N_k$ também tem. Se $passo^i(M'[N_1/\alpha]N_2...N_k)$ exibe $\mathbf{x}$, então:

- $passo^{i+1}(LN_1...N_k)=passo^i(M'[N_1/\alpha]passo(N_2)...passo(N_k))$

Pelo pelo lema 5.3.5, os termos da equação exibem $\mathbf{x}$, como requerido.

(5) Cada Y_σ tem denotação computável. Basta mostrar que $Y_\sigma N_1...N_k$ do tipo π tem denotação computável se $N_1,...,N_k$ fechados têm denotação computável. Ω_π denota $\underline{0}$ e $passo^0(\Omega_\pi)$ exibe $\underline{0}$, portanto Ω_π tem denotação computável; por (1), (3) e (4), Ω_σ e $Y_\sigma^{(n)}$ têm denotação computável. Sejam M_n o termo $Y^{(n)}N_1...N_k$ e M o termo $YN_1...N_k$. Se M_n denota $\mathbf{x}_n$ e M denota $\mathbf{x}$, então $\mathbf{x}=\bigsqcup\{\mathbf{x}_n \mid n\in\mathbb{N}\}$. Como $Y^{(n)}$ tem denotação computável, e, por hipótese, $N_1,...,N_k$ têm denotação computável, M_n tem denotação computável. Seja i tal que $passo^i(M_n)$ exibe $\mathbf{x}' \leq \mathbf{x}_n$. Pelo lema 5.3.7, $passo^i(M)$ exibe $\mathbf{x}'$. Logo, para todo $\mathbf{x}' \leq \mathbf{x}$, existe i tal que $passo^i(M)$ exibe $\mathbf{x}'$, e M tem denotação computável. Portanto, Y_σ tem denotação computável. □

O teorema 5.3.4 é estabelecido pelos lemas 5.3.3 e 5.3.8.

Corolário 5.3.4.1. Uma função é Π-definível sss ela é Π-computável. □

Corolário 5.3.4.2. $M{:}\pi$ computa **x** sss:

· $\mathbf{x}=\bigsqcup\{\mathbf{x}' \mid \text{existe } i \text{ t.q. } passo^i(M) \text{ exibe } \mathbf{x}'\}$. □

O corolário 5.3.4.2 acima mostra como obter o valor computado por $M{:}\pi$.

5.4 Programas e computações

▷ Um termo fechado de tipo π é dito *programa*.

Intuitivamente, mediante o uso da função *passo*, o valor denotado por um programa P pode ser *contado* (cf. seção 2.1) pelo seguinte algoritmo, dado de modo informal em linguagem procedural:

```
var x: programa
x:=P
enquanto x≠0 fazer:
     se x é da forma (SUCC P'),
          então contar ;
               e x:=P',
          senão x:=passo(x)
contar ;
```

Se e quando x atinge um termo sem forma normal fraca (i.e., que denota $\underline{0}$), a execução do algoritmo "entra em laço infinito", e termina de contar. Quando P denota ∞, a execução também entra em laço, mas nunca termina de contar.

▷ Um programa P *pára em n passos* sss n é o menor i tal que $passo^i(P)=passo^{i+1}(P)$.

Por exemplo, o programa $(SUCC^2 0)$ para em zero passos, e o programa $(Y\ SUCC)$ não para em n passos, para qualquer n.

▷ Um programa *P pára com resultado* **x** sss *P* pára em *n* passos, $passo^n(P)=M$, e **x** é a melhor informação que *M* exibe.

Por exemplo, $(SUCC^2 0)$ pára com resultado **2**, pois $(SUCC^2 0)$ exibe $\underline{0},\underline{1},\underline{2}$, e **2**, sendo que **2** é a melhor informação que $(SUCC^2 0)$ exibe.

▷ A *computação* de um programa *P* é a seqüência $\{passo^i(P)\}$, com $i\in\mathbb{N}$, e $i\leq n$ se *P* pára em *n* passos.

Claramente, a computação de *P* é finita sss *P* pára, e um programa pára sss ele denota um número finito total. Neste caso, o último termo da computação do programa é o numeral que denota este número.

▷ Um programa *P pára fracamente* sss existe $i\in\mathbb{N}$ tal que $passo^i(P)$ é forma normal fraca.

É imediato que um programa pára fracamente sss ele denota um número parcial distinto de $\underline{0}$, e que se um programa pára, ele pára fracamente. Intuitivamente, um programa pára fracamente no passo de sua computação em que *produz* informação não nula. Esta definição se deve à identificação implícita de parada com produção de resultados feita na teoria da computação [ROG 67].

▷ Um programa *termina*, tenha ele computação finita ou infinita, sss existe *i* tal que o número parcial mais definido exibido por $passo^j(P)$ se mantém constante para $j>i$.

Conseqüentemente, todo programa que pára termina, e um programa termina sss ele denota um número distinto de ∞. Intuitivamente, um programa termina no passo de sua computação em que deixa de produzir informação.

Um exemplo de programa que não pára mas termina é o programa (*SUCC* (*SUCC* (*Y* $\lambda\alpha.\alpha$))). Um exemplo de programa que nem pára e nem termina é (*Y SUCC*).

Como foi dito, parada implica terminação. Mas o contrário não é verdadeiro. A parada ocorre no passo em que a computação não pode mais ser desenvolvida. A terminação ocorre no passo em que a denotação do programa é exibida. A parada é finitamente observável [VIC 89], no sentido que, se ela ocorre, isto pode ser constatado em um número finito de passos. A terminação não é finitamente observável, no caso geral, por que ela exige o exame completo de computações infinitas, i.e., a execução de um número infinito de passos.

5.5 Computação interativa de programas-Π

O propósito desta seção é mostrar de modo informal como termos-Π podem modelar programas interativos, e como estes programas interativos podem ser computados.

▷ Um *programa interativo de k entradas*, ou simplesmente *k-programa*, é um termo de tipo π com variáveis livres incluídas em $\{\alpha_0^{\pi},\ldots,\alpha_{k-1}^{\pi}\}$.

A seguir, as expressões em *itálico* não definidas são tomadas como primitivas, com o seu significado intuitivo.

▷ A computação de um programa interativo envolve um *agente computacional* e um *agente usuário*. A computação resultante depende de ambos. O agente computacional *pratica ações de computação*, sobre o programa, e ações de *saída*. O agente usuário pratica ações de *entrada* e *observa* as ações de saída. Ambos agentes *interagem* e se *comunicam* pelas ações de

entrada e saída. *Internamente*, o agente usuário pode realizar ações de computação, e, em princípio, as ações que o agente usuário pratica podem ser modeladas por um programa interativo também. O tratamento é mantido assimétrico, mas isto não implica em perda de generalidade.

▷ O agente usuário de um k-programa P pode praticar as ações de entrada $\$_i$ e $?_i$, para $i=0,\ldots,k-1$. Uma ação $\$_i$ *transforma* o programa P em $P[(SUCC\alpha_i)/\alpha_i]$. Uma ação $?_i$ transforma o programa P em $P[0/\alpha_i]$. Como o programa P também é transformado pelo agente computacional, deve haver algum *mecanismo de sincronização*, não especificado aqui, que regula a prática de ações sobre o programa.

▷ O agente computacional *de* um k-programa P realiza a função denotada por $\lambda\alpha_0\ldots\lambda\alpha_{k-1}\,P$, por passos seqüenciais. Em cada passo, o agente procede da seguinte maneira. Se P é da forma 0, ele pratica a ação $?$, e pára. Se P é da forma $(SUCC\ P')$, ele pratica a ação $\$$ e transforma P em P'. Caso contrário, o agente transforma P em $passo(P)$.

A prova de que o processo acima descrito realiza o mapeamento entre contagens especificado abstratamente pela função denotada por $\lambda\alpha_0\ldots\lambda\alpha_{k-1}\,P$ é omitida.

Note que o *meio* onde se realiza a interação é o programa. As variáveis livres do programa cumprem o papel de *portas* de entrada. O próprio programa cumpre o papel de porta de saída. Em um caso mais geral, o tipo do programa e das suas variáveis livres pode ser outro que π. Também poderiam ocorrer ações de entrada que criassem portas, ao substituir uma variável livre por um termo que contivesse variáveis livre novas.

De um ponto de vista antropomórfico, este modo de

interação parece razoável. A maneira pela qual um agente se comporta dependeria de um "programa" que regularia as ações do agente. Mas também, este programa dependeria das ações que o agente observasse. Assim, um processo adaptativo tanto seria ditado pelo programa do agente, como alteraria o programa do agente. Isto mostraria explicitamente de que maneira o programa adaptativo de um agente não seria fixo.

6 Π-DEFINIBILIDADE

Para mostrar efetivamente que uma função é Π-definível, é preciso exibir um termo-Π que a denota, e provar que este termo a denota. Na maioria das situações, porém, é impraticável exibir explicitamente tal termo, uma vez que ele pode ser excessivamente complicado, e ocupar páginas inteiras. Este capítulo desenvolve técnicas que permitem mostrar efetivamente Π-definibilidade sem exibir explicitamente os termos necessários para tal. Estas técnicas permitem que estes termos sejam exibidos explicitamente, caso requerido. A partir destas técnicas, é possível fornecer caracterizações matemáticas não formais de Π-definibilidade.

A técnica mais freqüente consiste em utilizar modos de definir funções que preservam Π-definibilidade, i.e., que produzem funções Π-definíveis a partir de funções Π-definíveis. Este capítulo cataloga alguns deste métodos. Também são introduzidas notações para simplificar a definição de funções. Algumas funções básicas são mostradas Π-definíveis, a título de ilustração dos modos de definir funções.

Neste capítulo, os símbolos de domínio D,E correm sobre $\{\mathcal{C}_\sigma\}$.

As funções **succ**, **pred** são Π-definíveis, uma vez que os termos *SUCC*, *PRED*, respectivamente, as denotam, por definição. Como foi mostrado na seção 5.3, cada número parcial é Π-definível.

A seção 6.1 introduz a noção de definição

explícita. A seção 6.2 introduz a notação-λ para definições explícitas. A seção 6.3 apresenta a noção de currificação. A seção 6.4 mostra que a definição por casos de funções contínuas preserva Π-definibilidade. A seção 6.5 introduz a noção de definição por recursão geral. A seção 6.6 definine minimização limitada. A seção 6.7 define quantificação limitada. A seção 6.8 introduz a noção de função de pareamento. A seção 6.9 define a operação de seleção. A seção 6.10 fornece caracterizações matemáticas não formais dos elementos Π-definíveis.

6.1 Definição explícita

▷ O modo principal de definir funções é a *definição explícita*. Suponha que $y_1,\ldots,y_j \in \mathcal{C}$ são elementos provados Π-definíveis, que $x_1 \in D_1,\ldots,x_k \in D_k$ são quaisquer e que $\mathcal{E}[y_1,\ldots,y_j,x_1,\ldots,x_k] \in E$ é um valor construído finitamente a partir de $y_1,\ldots,y_j,x_1,\ldots,x_k$ por sucessivas aplicações da operação de aplicação (cf. seção 5.1.2). Uma definição explícita de $f:D_1\to\ldots\to D_k\to E$ é:

· $f\ x_1\ldots x_k = \mathcal{E}[y_1,\ldots,y_j,x_1,\ldots,x_k]$

Um termo-Π que denota f pode ser construído a partir de termos-Π $M_1,\ldots,M_j$ que denotam respectivamente $y_1,\ldots,y_j$, do seguinte seguinte modo. Suponha que $D_i = \mathcal{C}_{\sigma_i}$. No processo de construção de $\mathcal{E}$, se substitui cada valor y_i por M_i, cada valor $x_i \in D_i$ pelo termo $\alpha_i^{\sigma_i}$, e a operação de aplicação pela operação de formação de combinação. Deste modo, ao invés de se obter um valor $\mathcal{E}$, se obtém um termo N. Por indução na formação de N, e pela definição de f, se prova que o termo $(\lambda\alpha_1^{\sigma_1}.\ldots\lambda\alpha_k^{\sigma_k}.N)$ denota f. Detalhes formais são omitidos.

6.2 Notação-λ para definições explícitas

▷ A função f definida na seção anterior 6.1 também é escrita:

• $\lambda x_1 \ldots x_k . \mathcal{E}[y_1, \ldots, y_j, x_1, \ldots, x_k]$

O símbolo λ é utilizado aqui em um sentido distinto que na linguagem Π. Na linguagem Π, λ é um símbolo formal, que faz parte de um *cálculo*. No uso acima, λ é um símbolo não formal, que faz parte de uma *notação*. A sua função é dar um nome, especificamente $\lambda x.\mathcal{E}[x]$, à função que $\mathcal{E}[x]$ é de x. Ao utilizar esta notação, é preciso especificar, implícita ou explicitamente, o domínio sobre o qual x corre. Se o domínio é D, isto pode ser especificado explicitamente escrevendo-se $\lambda x \in D.\mathcal{E}[x]$.

Deve se ter cuidado com a notação-λ e definições explícitas em geral. Por exemplo, a expressão $\lambda x.x(x)$ não tem significado, pois a aplicação $x(x)$ não é definida para $x \in \mathcal{C}$ algum (cf. seção 5.1.2).

6.3 Currificação

▷ A *currificação* de uma função $f: D_1 \times \ldots \times D_k \to E$ é a função $\lambda x_1 \ldots x_k . f(x_1, \ldots, x_k) : D_1 \to \ldots \to D_k \to E$, escrita também *curry* f.

Uma função f e sua currificação são essencialmente a mesma, uma vez que $f(x_1, \ldots, x_k) = (curry\ f)x_1 \ldots x_k$. A diferença é que f toma os seus argumentos "ao mesmo tempo", e sua currificação toma os seus argumentos "um de cada vez".

▷ Uma vez que em Π não há produto cartesiano, se convenciona que uma função $D_1 \times \ldots \times D_k \to E$ é Π-definível sss a sua currificação o é. A notação $D^{(k)} \to E$ abrevia $D \to \ldots \to D \to E$,

onde em $D\to\ldots\to D$ há k ocorrências de D, de tal modo que, se $f:D^k\to E$, então $(curry\ f):D^{(k)}\to E$.

▷ Para cada D, o *condicional* $\mathbf{if}_D:\mathbb{P}\to D^{(2)}\to D$ é a currificação de $\mathbf{cond}_D:\mathbb{P}\times D^2\to D$. O subscrito D é omitido quando pode ser inferido do contexto. A seguinte notação é utilizada:

- **if x then y else z=if x y z**
- **if x then y=if x y ⊥**

onde ⊥ é o menor elemento de D.

$\mathbf{if}_{\mathbb{P}}$ é Π-definível, pois a denotação de *COND* é $\mathbf{if}_{\mathbb{P}}$, por definição. Se $\mathbf{if}_E$ é Π-definível, então $\mathbf{if}_{D\to E}$ é Π-definível, pois, pelo lema 4.1.4:

- $\mathbf{if}_{D\to E}$**x then** *f* **else** *g*=λ*y*.$\mathbf{if}_E$**x then** *fy* **else** *gy*

Logo, por indução nas regras de formação de $\{\mathcal{C}_\sigma\}$, para todos D,E, $\mathbf{if}_{D\to E}$ é Π-definível.

▷ Na seção 4.2 as informações de verdade foram representadas por t=**1**, f=**0**, e u=$\underline{\mathbf{0}}$. É conveniente generalizar esta representação, de tal modo que todos os valores que são sucessores representem a informação de verdade "é verdadeiro". Esta decisão é tomada porque os sucessores são *verdadeiros perante o condicional*, no seguinte sentido:

- **if x+1 then y else z=y** para todo **x**

Como exemplo de uso do condicional, a função Π-definível **ézero**:$\mathbb{P}\to\mathbb{B}$, que determina se seu argumento é nulo, é definida por:

- **ézero x=if x then** f **else** t

(Pronuncie "é zero", e não como uma palavra proparoxítona.)

Como as operações lógicas da seção 4.2 são definidas explicitamente a partir do condicional, elas são Π-definíveis.

O operador **cases** da seção 4.1 também é Π-definível, uma vez que **ccases**=*curry* **cases** é definível explicitamente em função do condicional.

6.4 Definição por casos

▷ Sejam $\mathcal{E}_i[y_1,\ldots,y_j,x_1,\ldots,x_k]\in E$, para $i=1,2,3$, definidos como na seção 6.1, a partir de elementos Π-definíveis $y_1,\ldots,y_j\in\mathcal{C}$, e $x_1\in D_1,\ldots,x_k\in D_k$ quaisquer. Uma definição *por casos* de $f:\mathbb{P}\to D_1\to\ldots\to D_k\to E$ é dada por:

- $f\ \underline{\mathbf{0}}\ x_1\ldots x_k=\mathcal{E}_1[y_1,\ldots,y_j,x_1,\ldots,x_k]$
- $f\ \mathbf{0}\ x_1\ldots x_k=\mathcal{E}_2[y_1,\ldots,y_j,x_1,\ldots,x_k]$
- $f\ (\mathbf{x+1})\ x_1\ldots x_k=\mathcal{E}_3[y_1,\ldots,y_j,x_1,\ldots,x_k]$

Tal definição é equivalente a:

- $f\ \underline{\mathbf{0}}=g_1$
- $f\ \mathbf{0}=g_2$
- $f\ (\mathbf{x+1})=g_3$

onde:

- $g_i x_1\ldots x_k=\mathcal{E}_i[y_1,\ldots,y_j,x_1,\ldots,x_k]$

Se f é contínua, tem-se que, pelo lema 4.1.2:

- $f=\mathbf{cases}(\mathbf{x},g_1,g_2,g_3)=\mathbf{ccases}\ \mathbf{x}\ g_1g_2g_3$

Logo, f é Π-definível sss f é contínua.

▷ Se o caso para $\underline{\mathbf{0}}$ é omitido, se assume implicitamente que $\mathcal{E}_1[y_1,\ldots,y_j,x_1,\ldots,x_k]=\perp$.

Como exemplo, a função **ézero** definida na seção 6.3 é definível por casos:

- **ézero 0**=t
- **ézero x+1**=f

A função **édefinido**:$\mathbb{P}\to\mathbb{B}$ tal que:

- **édefinido x**=t se **x**$\neq\underline{\mathbf{0}}$
  =u se **x**=$\underline{\mathbf{0}}$

pode ser definida por:

- **édefinido** $\underline{\mathbf{0}}$=u
- **édefinido 0**=t
- **édefinido (x+1)**=t

6.5 Definição por recursão geral

▷ Outro modo importante de definir funções é a *definição por recursão geral*. Seja $\mathcal{E}[y_1,\ldots,y_j,h,x_1,\ldots,x_k]\in E$ definido como na seção 6.1, a partir de elementos Π-definíveis $y_1,\ldots,y_j\in\mathcal{C}$, e $h:E\to E$, $x_1\in D_1,\ldots,x_k\in D_k$ quaisquer. Uma definição recursiva geral de $f:D_1\to\ldots\to D_k\to E$ é dada pela definição implícita:

- $f\ x_1\ldots x_k=\mathcal{E}[y_1,\ldots,y_j,f,x_1,\ldots,x_k]$

que abrevia a definição explícita:

- $f\ x_1\ldots x_k=\mathbf{fix}_E(\lambda h.\mathcal{E}[y_1,\ldots,y_j,h,x_1,\ldots,x_k])$

Seja σ tal que $E=\mathcal{C}_\sigma$. O operador de ponto fixo $\mathbf{fix}_E$ é Π-definível, pois Y_σ denota $\mathbf{fix}_E$, por definição. Como **f** tem definição explícita a partir de $\mathbf{fix}_E$, $y_1,\ldots,y_j$ Π-definíveis, **f** é Π-definível.

Para exemplificar o uso da definição por recursão geral, é mostrado abaixo, em virtude do lema 4.3.10, que a função $\mathbb{P}$-característica da igualdade é Π-definível **==**:

- **x==y=if x then (if y then x-1==y-1 else** f **else** t)

else (if y then f else t)

Uma definição recursiva geral mais legível para **==** é:

- **x==y=ézero x** $\wedge$ **ézero y**
  $\vee$ $\neg$**ézero x** $\wedge$ $\neg$**ézero y** $\wedge$ **x-1==y-1**

▷ Definição por casos pode ser combinada com recursão geral. A definição de $f:\mathbb{P}\to D$ abaixo, com $a,b,\mathcal{E}[f,\mathbf{x}]\in D$:

- $f\ \underline{\mathbf{0}}=a$
- $f\ \mathbf{0}=b$
- $f\ (\mathbf{x+1})=\mathcal{E}[f,\mathbf{x}]$

abrevia a definição recursiva geral:

- $f\ \mathbf{x}=$**ccases x** $a\ b\ (\mathcal{E}[f,$**x-1**$])$

▷ Como caso particular, quando $\mathcal{E}[f,\mathbf{x}]=\mathcal{E}'[f(\mathbf{x})]$, para algum $\mathcal{E}'[\mathbf{x}]$, tal definição recursiva geral é dita *definição iterativa*.

O operador de *parcialização* **parcial**$=\lambda$**x.**$\underline{\mathbf{x}}$ é Π-definível, pois ele tem definição iterativa:

- **parcial 0=**$\underline{\mathbf{0}}$
- **parcial (x+1)=(parcial x)+1**

Pelo lema 4.6.2, que provê uma definição iterativa para a extensão natural da operação de soma, ela é Π-definível.

A título de ilustração do papel do condicional não estrito, com relação ao estrito (cf. seções 4.1 e 4.2), é mostrada a seguinte definição possível para a extensão natural da soma:

- **x+y=if y==0 then x else succ(x+pred y)**

A prova é omitida. Se o condicional fosse substituído pelo condicional estrito, então a função definida não seria a extensão natural da soma, mas seria uma extensão intermediária entre a trivial e a natural. Tal função não seria sequer comutativa. Por exemplo, valeria que $\mathbf{x}+\underline{\mathbf{0}}=\underline{\mathbf{0}}$ e $\underline{\mathbf{0}}+\mathbf{x}=\underline{\mathbf{x}}$.

6.6 Minimização limitada

▷ O operador de *minimização limitada* $\mathbf{min}:\mathbb{P}\to(\mathbb{P}\to\mathbb{P})\to\mathbb{P}$ é definido abaixo, onde $\mu\mathbf{x}<\mathbf{k}.\mathcal{E}[\mathbf{x}]$ abrevia $\mathbf{min}\ \mathbf{k}\ (\lambda\mathbf{x}.\mathcal{E}[\mathbf{x}])$:

- $\mu\mathbf{x}<\mathbf{0}.\mathbf{px}=\mathbf{0}$
- $\mu\mathbf{x}<\mathbf{k+1}.\mathbf{px}=\mathbf{if\ p0\ then\ 0\ else\ 1}+\mu\mathbf{x}<\mathbf{k}.\mathbf{p(x+1)}$

A variável **k** é dita *limite* da minimização. A notação $\mu\mathbf{x}\leq\mathbf{k}.\mathcal{E}[\mathbf{x}]$ abrevia $\mu\mathbf{x}<\mathbf{k+1}.\mathcal{E}[\mathbf{x}]$.

O operador de minimização limitada realiza, como caso particular, a operação de minimização ilimitada, quando o limite é ∞. A notação $\mu\mathbf{x}.\mathcal{E}[\mathbf{x}]$ abrevia $\mu\mathbf{x}<\infty.\mathcal{E}[\mathbf{x}]$. Pela última equação, tem-se que:

- $\mu\mathbf{x}.\mathbf{px}=\mathbf{if\ p0\ then\ 0\ else\ 1}+\mu\mathbf{x}.\mathbf{p(x+1)}$

Intuitivamente, (1) $(\mu\mathbf{x}<\mathbf{k}.\mathbf{px})=\mathbf{n}$ para o menor $n<k$, tal que **pn** é verdadeiro, se tal n existe, e (2) $\mu\mathbf{x}<\mathbf{k}.\mathbf{px}=\mathbf{k}$ quando não existe $n<k$ tal que **pn** é verdadeiro. A igualdade (1) somente vale se não existe $m<n$ tal que $\mathbf{pm}=\mathfrak{u}$. Se tal m existe, (1′) $\mu\mathbf{x}<\mathbf{k}.\mathbf{px}=\underline{\mathbf{m}}$ para o menor m. A igualdade (2) somente vale se não existe $m<k$ tal que $\mathbf{pm}=\mathfrak{u}$. Se tal m existe (2′) $\mu\mathbf{x}<\mathbf{k}.\mathbf{px}=\underline{\mathbf{m}}$ para o menor m, também. As provas destas asserções são omitidas.

Pela discussão acima, $(\mu\mathbf{n}.\neg\mathbf{n}==\mathbf{n})==\mathbf{0}$ vale f, pois $(\mu\mathbf{n}.\neg\mathbf{n}==\mathbf{n})$ vale ∞. Na teoria da recursão, o operador de minimização ilimitada não produz resultado algum em tais

casos. Este é mais uma situação em que indefinição é representada por ∞ e não por $\underline{0}$ (cf. seções 3.3 e 3.4).

6.7 Quantificação limitada

▷ O *quantificador existencial limitado* **ex**:ℙ→(ℙ→ℙ)→ℙ é definido abaixo, onde ∃**x**<**k**.$\mathcal{E}$[**x**] abrevia **ex k** (λ**x**.$\mathcal{E}$[**x**]):

- ∃**x**<**0**.**px**=f
- ∃**x**<**k+1**.**px**=**p0** ∨ ∃**x**<**k**.**p**(**x+1**)

A variável **k** é dita *limite* do quantificador. A notação ∃**x**≤**k**.$\mathcal{E}$[**x**] abrevia ∃**x**<**k+1**.$\mathcal{E}$[**x**].

É imediato que se tem, devido às propriedades da função ∨ (cf. seção 4.2):

- ∃**n**<**k**.**pn**=t se existe $n<k$ tal que **pn** é verdadeiro
- ∃**n**<**k**.**pn**=f se para todo $n<k$, **pn** é falso e **k** é finito total

O quantificador existencial limitado realiza um modo de quantificação existencial ilimitada, quando o limite é ∞, uma vez que a função **exist'** tal que:

- **exist'p**=∃**x**<∞.**px**

é um quantificador existencial (cf. seção 4.4). A notação ∃**x**.**px** abrevia ∃**x**<∞.**px**. De todo modo, **exist'**≠**exist**, uma vez que, para nenhum **p**, **exist'**(**p**)=f, mas **exist**(λ**x**.f)=f. Assim, ∃**x**<**k**.**px** pode valer f somente se o limite não é ∞. A Π-definibilidade de **exist** é considerada na seção 9.4.

Analogamente, pode ser definido um quantificador universal limitado. A definição é omitida.

6.8 Funções de pareamento

A noção de função de pareamento apresentada a seguir [MAC 78] é utilizada para dois propósitos distintos. Um dos propósitos é definir a função de seleção, na seção 6.9. O outro, é construir numerações de Gödel, na seção 8.1.

▷ Uma *função de pareamento* é qualquer função bijetiva $\langle -,- \rangle : \mathbb{N}^2 \to \mathbb{N}$ estritamente crescente em cada argumento, i.e., tal que:

- $x<x'$ ou $y<y'$ implica $\langle x,y \rangle < \langle x',y' \rangle$

▷ As *funções de pareamento inverso* de uma função de pareamento $\langle -,- \rangle$ são $fst, snd : \mathbb{N} \to \mathbb{N}$ tais que:

- $fst\langle x,y \rangle = x$
- $snd\langle x,y \rangle = y$

Estas funções são bem definidas, uma vez que $\langle -,- \rangle$ é bijetiva.

Tais funções de pareamento inverso satisfazem:

- $fst\ z = \mu x \leq z.\ \exists y \leq z.\ z = \langle x,y \rangle$
- $snd\ z = \mu y \leq z.\ \exists x \leq z.\ z = \langle x,y \rangle$

onde $\mu x \leq z$ e $\exists x \leq z$ são os operadores de minimização limitada e quantificação existencial limitada da teoria da recursão, e não os definidos nas seções 6.7 e 6.8 anteriores. Assim, se $\langle -,- \rangle$ é recursiva, fst e snd também são.

As funções de pareamento $\langle -,- \rangle$ enumeram $\mathbb{N}^2$, no sentido que:

- $\mathbb{N}^2 = \{(x,y) \mid \text{existe } z \text{ tal que } z = \langle x,y \rangle\}$
  $= \{(fst\ z, snd\ z) \mid z \in \mathbb{N}\}$

▷ No que segue do texto, $\langle -,- \rangle$ é uma função de pareamento recursiva qualquer, mas fixa. Suas inversas são *fst* e *snd*. A função $\langle -,- \rangle : \mathbb{P}^{(2)} \to \mathbb{P}$ é a currificação da extensão trivial de $\langle -,- \rangle$. As funções $\mathbf{fst}, \mathbf{snd} : \mathbb{P} \to \mathbb{P}$ são as extensões triviais de *fst*,*snd* respectivamente. Para outras aplicações, seria conveniente trabalhar com as extensões naturais destas funções. A escolha da extensão trivial é feita apenas por simplicidade, uma vez que, de acordo com o teorema 7.1, as currificações das extensões triviais de $\langle -,- \rangle$, *fst* e *snd* são Π-definíveis.

É imediato que $\langle -,- \rangle$ enumera $\mathbb{M}^2$, no mesmo sentido exposto acima para $\langle -,- \rangle$ em relação a $\mathbb{N}^2$:

- $\mathbb{M}^2 = \{ (\mathbf{x},\mathbf{y}) \mid \text{existe } \mathbf{z} \in \mathbb{M} \text{ tal que } \mathbf{z} = \langle \mathbf{x},\mathbf{y} \rangle \}$
  $= \{ (\mathbf{fst}\ \mathbf{z}, \mathbf{snd}\ \mathbf{z}) \mid \mathbf{z} \in \mathbb{M} \}$

▷ A expressão $\langle \mathbf{x}_1, \ldots, \mathbf{x}_k \rangle$ abrevia:

- $\langle \mathbf{x}_1, \langle \mathbf{x}_2 \ldots, \langle \mathbf{x}_{k-1}, \mathbf{x}_k \rangle \rangle \ldots \rangle$

6.9 Seleção

Às vezes é necessário obter um $\mathbf{x} \in \mathbb{M}$ para o qual existe $\mathbf{y} \in \mathbb{M}$ tal que $\mathbf{p}(\mathbf{x},\mathbf{y})$ é verdadeiro, para uma dada função característica parcial $\mathbf{p} : \mathbb{P}^2 \to \mathbb{B}$. Pode ser que mais de um $\mathbf{x}$ ou que nenhum $\mathbf{x}$ satisfaça tal especificação. A operação de seleção definida a seguir determina um de tais $\mathbf{x}$, no casos em que isto é possível. Esta determinação é feita por enumeração sucessiva dos pares $(\mathbf{x},\mathbf{y}) \in \mathbb{M}^2$, mediante o uso da função de pareamento fixada na seção 6.8.

▷ O operador de *seleção* $\mathbf{sel} : (\mathbb{P}^{(2)} \to \mathbb{B}) \to \mathbb{P}$ é definido por:

- $\mathbf{sel}\ \mathbf{p} = \mathbf{fst}(\mu \mathbf{z}.\mathbf{p}(\mathbf{fst}\ \mathbf{z})(\mathbf{snd}\ \mathbf{z}))$

A notação $\mathbf{x}/\exists \mathbf{y}.\mathcal{E}[\mathbf{x},\mathbf{y}]$ abrevia $(\mathbf{sel}\ \lambda \mathbf{xy}.\mathcal{E}[\mathbf{x},\mathbf{y}])$.

6.10 Caracterização não formal dos elementos Π-definíveis

A noção de Π-definibilidade é formal, porque é relativa a um sistema formal, a linguagem Π. Nesta seção é dada uma caracterização matemática não formal dos elementos Π-definíveis. Note que por "não formal" não se entende "informal", mas "desprovisto do uso de sistemas formais", no sentido de [KLE 52] de sistema formal.

▷ Para cada $A,B,C\in\mathcal{C}$, as funções:

- $\mathbf{K}_{A,B}: A\to B\to A$
- $\mathbf{S}_{A,B,C}: (A\to B\to C)\to(A\to B)\to(A)\to C$

são definidas para todos $f:A\to B\to C$, $g:A\to B$, $x\in A$, $y\in B$ por:

- $\mathbf{K}\ x\ y=x$
- $\mathbf{S}\ f\ g\ x=f\ x\ (g\ x)$

onde os subscritos são omitidos quando inferíveis do contexto.

Teorema 6.10.1. O conjunto de elementos Π-definíveis é o menor conjunto $\mathcal{R}$ que contém:

- **0, succ, pred, if,** $\mathbf{fix}_D$ para cada $D\in\mathcal{C}$
- $\mathbf{K}_{A,B}$, $\mathbf{S}_{A,B,C}$ para cada $A,B,C\in\mathcal{C}$

e é fechado para a aplicação definida, i.e.:

- $f,x\in\mathcal{R}$, $f:D\to E$, $x\in D$ implica $fx\in R$

Prova. Os combinadores:

- $K_{\sigma,\tau}=\lambda\alpha_0^{\sigma}.\lambda\alpha_1^{\tau}.\alpha_0^{\sigma}$
- $S_{\sigma,\tau,\upsilon}=\lambda\alpha_0^{\sigma\to\tau\to\upsilon}.\lambda\alpha_1^{\sigma\to\tau}.\lambda\alpha_2^{\sigma}.\alpha_0^{\sigma\to\tau\to\upsilon}\alpha_2^{\sigma}(\alpha_1^{\sigma\to\tau}\alpha_2^{\sigma})$

denotam as funções $\mathbf{K}_{A,B}$ e $\mathbf{S}_{A,B,C}$ para $A=\mathcal{C}_{\sigma}$, $B=\mathcal{C}_{\tau}$ e $C=\mathcal{C}_{\upsilon}$.

Utilizando os métodos expostos em [HIN 86], é possível transformar qualquer termo fechado em um termo fechado de mesma denotação, formado a partir das constantes $0, SUCC, PRED, COND, Y_\sigma$ e dos combinadores $K_{\sigma,\tau}$ e $S_{\sigma,\tau,\upsilon}$ por sucessivas aplicações da operação de formação de combinação bem tipada. Como a operação de formação de combinação bem tipada é interpretada como a operação de aplicação definida, tem-se o resultado desejado. □

Utilizando este teorema e os resultados das seções anteriores, é possível dar uma caracterização mais operacional de Π-definibilidade.

Teorema 6.10.2. O conjunto de funções Π-definíveis é o menor conjunto $\mathcal{R}$ que contém:

- **0,succ**

e é fechado para:

- definição explícita
- definição por casos de funções contínuas
- definição por recursão geral

Prova. Definição explícita e definição por recursão geral sempre introduzem funções contínuas. A função **pred** é definível por casos. As funções **fix** são definíveis por recursão geral:

- **fix** f=f(**fix** f)

Para mostrar que **if**$\in\mathcal{R}$, basta mostrar que **inf**:$\mathbb{P}^{(2)}\to\mathbb{P}$ tal que **inf x y**=$\mathbf{x}_{\sqcap}\mathbf{y}$ está em $\mathcal{R}$, e definir **if** por casos por:

- **if** $\underline{0}$ **y z=inf y z**
- **if 0 y z=z**
- **if (x+1) y z=y**

A definição de **inf** é por casos e por recursão geral, utilizando o lema 2.5.3. Note que é necessário definir funções auxiliares, pois a recursão é nos dois argumentos. Se

$\mathcal{R}$ é fechado para definição explícita, então $\mathcal{R}$ é fechado para a aplicação, pois $y=fx$ é uma definição explícita. As funções **S** e **K** são definíveis explicitamente. Então, pelo teorema 6.10.1, $\mathcal{R}$ contém os elementos Π-definíveis. Pelos resultados das seções 6.1, 6.4 e 6.5, $\mathcal{R}$ está contido no conjunto de elementos Π-definíveis. □

7 Π-DEFINIBILIDADE DAS FUNÇÕES RECURSIVAS

Este capítulo mostra que a linguagem Π tem ao menos o mesmo poder de decisão que as funções parciais recursivas, no sentido do teorema 7.6. As noções de conjunto Π-definível e Π-enumerável são introduzidas.

▷ Uma função parcial $f:\mathbb{N}^k\to\mathbb{N}$ é *Π-definível* sss sua extensão trivial é Π-definível.

A idéia intuitiva é que, na teoria recursão, indefinição é representada por computações infinitas que não produzem resultados, e que, na linguagem Π, computações infinitas que não produzem informação denotam $\underline{0}$.

Teorema 7.1. As funções parciais recursivas são Π-definíveis.

Prova. (▷ A formulação da noção de *função parcial recursiva* adotada para provar o teorema é a de [DAV 58]). A prova é por indução nas regras de formação do conjunto de funções parciais recursivas. O lema 7.2 prova o caso base, e os lemas 7.3-7.5 provam os passos de indução. Os métodos do capítulo **6** são utilizados implicitamente. □

A função **fintot**:$\mathbb{P}\to\mathbb{B}$ tal que, para todo **x**:

- **fintot x**=t se **x** é finito total
  =u caso contrário,

é Π:definível, pois:

- **fintot** $\underline{\mathbf{0}}$=u
- **fintot 0**=t
- **fintot (x+1)=fintot x**

As funções $\mathbf{TRIVIAL}_k:(\mathbb{P}^{(k)}\to\mathbb{P})\to(\mathbb{P}^{(k)}\to\mathbb{P})$, $k\in\mathbb{N}$, tais que, se $\mathbf{f}:\mathbb{P}^{(k)}\to\mathbb{P}$ é a currificação de uma extensão de uma função parcial $f:\mathbb{N}^k\to\mathbb{N}$, então $\mathbf{TRIVIAL}_k\mathbf{f}$ é a currificação da extensão trivial de f, são Π-definíveis, pois:

- $(\mathbf{TRIVIAL}_k\,\mathbf{f})\mathbf{x}_1\ldots\mathbf{x}_k=$
  $\mathbf{if\ fintot}\ \mathbf{x}_1\wedge\ldots\wedge\mathbf{fintot}\ \mathbf{x}_k\wedge\mathbf{fintot}(\mathbf{f}\ \mathbf{x}_1\ldots\mathbf{x}_k)$
  $\mathbf{then}\ \mathbf{f}\ \mathbf{x}_1\ldots\mathbf{x}_k$
  $\mathbf{else}\ \underline{\mathbf{0}}$

O subscrito k é omitido quando pode ser inferido do contexto.

As funções recursivas base são Π-definíveis:

Lema 7.2. As funções sucessor $S:\mathbb{N}\to\mathbb{N}$, constante zero $Z:\mathbb{N}\to\mathbb{N}$, e projeções $P^k_j:\mathbb{N}^k\to\mathbb{N}$, $j,k\in\mathbb{N}$, $1\le j\le k$, tais que $S:n\mapsto n+1$, $Z:n\mapsto 0$, $P^k_j:x_1,\ldots,x_k\mapsto x_j$ são Π-definíveis.

Prova. As extensões triviais de S, Z e P^k_j são Π-definíveis, pois são, respectivamente:

- $\mathbf{TRIVIAL\ succ}$
- $\mathbf{TRIVIAL}(\lambda\mathbf{x}.\mathbf{0})$
- $\mathbf{TRIVIAL}(\lambda\mathbf{x}_1\ldots\lambda\mathbf{x}_k.\mathbf{x}_j)$ □

A substituição de funções Π-defíniveis é Π-defínivel:

Lema 7.3. Se $g:\mathbb{N}^j\to\mathbb{N}$ e $h_1,\ldots,h_j:\mathbb{N}^k\to\mathbb{N}$ são Π-definíveis, então $f:\mathbb{N}^k\to\mathbb{N}$ tal que:

- $f(x_1,\ldots,x_k)=g(h_1(x_1,\ldots,x_k),\ldots,h_j(x_1,\ldots,x_k))$

é Π-definível.

Prova. Sejam $\mathbf{f},\mathbf{g},\mathbf{h}_1,\ldots,\mathbf{h}_j$ as currificações das extensões triviais de $f,g,h_1,\ldots,h_j$ respectivamente. Então **f** é Π-definível, pois satisfaz a definição explícita:

- $\mathbf{f}\ \mathbf{x}_1\ldots\mathbf{x}_k=\mathbf{g}(\mathbf{h}_1\mathbf{x}_1\ldots\mathbf{x}_k)\ldots(\mathbf{h}_j\mathbf{x}_1\ldots\mathbf{x}_k)$ □

A recursão primitiva sobre funções Π-definíveis é Π-definível:

Lema 7.4. Se $b:\mathbb{N}^k\to\mathbb{N}$, e $g:\mathbb{N}^{k+2}\to\mathbb{N}$ são Π-definíveis, então $f:\mathbb{N}^{k+1}\to\mathbb{N}$ tal que:

- $f(0,x_2,\ldots,x_{k+1})=b(x_2,\ldots,x_{k+1})$
- $f(x_1+1,x_2,\ldots,x_{k+1})=g(x_1,x_2,\ldots,x_{k+1},f(x_1,x_2,\ldots,x_{k+1}))$

é Π-definível.

Prova. Sejam **f,b,g** as currificações das extensões triviais de f,b,g respectivamente. Então **f** é Π-definível, pois satisfaz a definição recursiva geral:

- $\mathbf{f} = \mathbf{TRIVIAL\ f'}$

onde

- $\mathbf{f'x_1\ldots x_{k+1}}=\mathbf{if\ x_1==0\ then\ b\ x_2\ldots x_{k+1}\ else\ g(x_1-1)x_2\ldots x_{k+1}(f'(x_1-1)x_2\ldots x_{k+1})}$ □

A minimização de uma função total Π-definível é Π-definível:

Lema 7.5. Se $g:\mathbb{N}^{k+1}\to\mathbb{N}$ total é Π-definível, então $f:\mathbb{N}^k\to\mathbb{N}$ tal que $f(x_1,\ldots,x_k)$ é o menor y tal que $g(x_1,\ldots,x_k,y)=0$ se tal y existe, e é indefinida, caso contrário, é Π-definível.

Prova. Sejam **f,g** as currificações das extensões triviais de f,g respectivamente. Então **f** é Π-definível, pois satisfaz a definição explícita:

- $\mathbf{f=TRIVIAL}(\lambda\mathbf{x_1}\ldots\lambda\mathbf{x_k}.\mu\mathbf{y}<\infty.\mathbf{g\ x_1\ldots x_k\ y==0})$ □

Isto conclui a prova do teorema 7.1.

▷ Um conjunto $A\subseteq\mathbb{N}^k$ é dito Π-*definível* sss a sua função $\mathbb{P}$-característica é Π-definível. Um conjunto $A\subseteq\mathbb{N}^k$ é dito Π-enumerável sss existe uma função característica parcial cujo núcleo é A.

Teorema 7.6.

- Se um conjunto é recursivo, então ele é Π-definível.
- Se um conjunto é recursivamente enumerável, então ele é Π-enumerável.

Prova. Pelo lema 4.3.7, a extensão natural da função característica de um conjunto A propriamente infinito coincide com a sua extensão trivial. Logo, se A é propriamente infinito e A é recursivo, então A é Π-definível, pelo teorema 7.1. Se A é finito, pelo lema 4.3.11 e pelos resultados do capítulo **6**, a sua função $\mathbb{P}$-característica é $\mathbb{P}$-definível. Se A é cofinito, o seu complemento $\overline{A}$ é finito. Seja **p** a função $\mathbb{P}$-característica de $\overline{A}$. Então $\lambda x.\neg px$ é a função $\mathbb{P}$-característica de A. A segunda parte é provada de modo análogo, e utiliza propriedades básicas dos conjuntos recursivamente enumeráveis, que podem ser achadas em [ROG 67] ou [MAC 78]. □

Na teoria da recursão, um problema de decisão é representado por um conjunto de naturais, via uma codificação, quando o domínio do problema não é formado por naturais [ROG 67].

▷ Um problema é dito *recursivamente decidível* sss o conjunto que o representa é recursivo. Um problema é dito *recursivemente semidecidível* sss o conjunto que o representa é recursivamente enumerável. Um problema é dito Π-*decidível* sss o conjunto que o representa é Π-definível. Um problema é dito Π-*semidecidível* sss o conjunto que o representa é Π-enumerável.

Corolário 7.6.

- Se um problema é recursivamente decidível, então ele é Π-decidível.
- Se um problema é recursivamente semidecidível, então ele é Π-semidecidível. □

O uso dado neste trabalho ao corolário 7.6 é de concluir que um problema não é recursivamente decidível quando ele não é Π-decidível.

De acordo com a discussão nas seções 3.3 e 3.4, a extensão natural veicula informação melhor que a veiculada pela extensão trivial. Cabe indagar, então, se as extensões naturais das funções parciais recursivas são Π-definíveis. Na seção 9.5 é mostrado, que, mesmo que a extensão natural de uma função parcial recursiva f seja Π-definível, ela não pode ser obtida efetivamente a partir de uma definição de f. Ainda, é mostrado que há funções recursivas (primitivas) cujas extensões naturais não podem ser Π-definíveis. Por outro lado, mostra-se que há funções Π-definíveis sobre conjuntos, que não são extensões de funções recursivas.

8 ARITMETIZAÇÃO DA LINGUAGEM Π

A linguagem Π, relativamente à sua interpretação padrão, tem como universo de discurso números parciais e funções que envolvem direta ou indiretamente números parciais. O objetivo deste capítulo é desenvolver meios de esta linguagem se referir, indiretamente, a termos dela mesma. A cada termo-Π é associado um único número, e a cada operação sobre termos é associada uma operação sobre números. A noção de numeração Π-efetiva de termos e tipos é definida.

É mostrado que, para qualquer numeração Π-efetiva, o problema de existência de forma normal fraca para programas-Π não é Π-decidível (teorema da parada fraca), mas que é Π-semidecidível. A partir das técnicas desenvolvidas neste capítulo, é possível reconstruir a teoria da recursão para funções Π-definíveis, obtendo diversos teoremas similares, porém com diferenças importantes. Este desenvolvimento é iniciado neste capítulo, e levado até um certo ponto no capítulo **9**.

A seção 8.1 define numerações de Gödel de termos e tipos, e introduz a noção de numeração Π-efetiva. A seção 8.2 numera os elementos Π-definíveis através de numerações de termos e tipos. A seção 8.3 apresenta o teorema da parada fraca. A seção 8.4 mostra que a função de passo de computação sobre números de Gödel é Π-definível para numerações efetivas. A seção 8.5 mostra que o problema da parada fraca é Π-semidecidível

8.1 Numerações de Gödel de termos e tipos

Esta seção define meios de codificar termos-Π por números parciais, e de manipular termos-Π via os seus códigos. Para codificar termos é preciso codificar tipos também. Os números finitos totais $\mathbb{M}=\{\mathbf{0},\mathbf{1},\mathbf{2},\ldots\}$ são utilizados como códigos de termos e tipos, e os demais números são utilizados eventualmente como informações parciais sobre códigos. A razão desta escolha é apresentada após a introdução da noção de numeração Π-efetiva. Por simplicidade das definições (especificamente, para que seja possível utilizar a noção de função $\mathbb{P}$-característica), a determinação de uso de números finitos totais é mantida arbitrária, por enquanto.

▷ Uma *numeração de Gödel* de termos e tipos é qualquer função injetiva $\mathbf{ng}:\mathbb{T}\to\mathbb{P}$, onde $\mathbb{T}$=termos ∪ tipos, com conjunto de chegada incluído em $\mathbb{M}$. Um número $\mathbf{x}\in\mathbb{P}$ é dito *o número de Gödel de* $t\in\mathbb{T}$, com relação a **ng**, sss $\mathbf{x}=\mathbf{ng}\,t$. O símbolo de função **ng** corre sobre numerações de Gödel de termos e tipos.

▷ Uma função $\mathbf{f}:\mathbb{P}^{(k)}\to\mathbb{P}$ *codifica uma função* $f:\mathbb{T}^{(k)}\to\mathbb{T}$, com relação a uma numeração **ng**, sss para todos $t_1,\ldots,t_k\in\mathbb{T}$:

- $\mathbf{f}\ (\mathbf{ng}\ t_1)\ldots(\mathbf{ng}\ t_k)=\mathbf{ng}(f\ t_1\ldots t_k)$

Note que pode haver mais de uma função **f** que codifica f, uma vez que os valores de **f** são especificados apenas para argumentos que são números de Gödel.

▷ Uma função $\mathbf{f}:\mathbb{P}^{(k)}\to\mathbb{B}$ *codifica um conjunto* $A\subseteq\mathbb{T}^k$, com relação a uma numeração **ng**, sss **f** é a função $\mathbb{P}$-característica do conjunto:

- $\{(\mathbf{ng}\ t_1,\ldots,\mathbf{ng}\ t_k)\in\mathbb{M}^k \mid (t_1,\ldots,t_k)\in A\}$

Um dos objetivos desta seção é definir o conceito de numeração efetiva de termos e tipos. O problema enfrentado é que, no sistema formal disponível, não há como expressar operações sobre termos e tipos, de modo que não há como mostrar, ao menos diretamente, que uma dada função **ng**:$\mathbb{T}\to\mathbb{P}$ é efetivamente calculável. Ainda, é justamente para definir calculabilidade formal sobre termos e tipos que se quer numerá-los via uma função **ng**:$\mathbb{T}\to\mathbb{P}$.

As operações primitivas sobre tipos são as de formação de tipos básicos e de tipos de função. As operações primitivas sobre termos são as de formação de constantes, variáveis, combinações e abstrações. Estas operações induzem uma partição de $\mathbb{T}$ em classes sintáticas de tipos básicos, tipos de função, constantes, variáveis, combinações e abstrações. A operação de reconhecer qual é a classe de um elemento de $\mathbb{T}$ também deve ser considerada primitiva.

As observações acima motivam a seguinte definição de efetividade para numerações de termos e tipos, que depende de uma noção de efetividade de computações com números.

▷ Uma numeração de Gödel de termos e tipos é efetiva sss:

- dados os números de Gödel de σ e τ, α, M, N, é possível obter efetivamente o número de Gödel do tipo $(\sigma\to\tau)$, e de cada termo Y_σ, α_i^σ, (MN), $(\lambda\alpha.M)$
- dado **i**$\in\mathbb{M}$, é possível decidir efetivamente se **i** é número de Gödel de tipo básico, tipo de função, de constante, de variável, de combinação ou de abstração

Se uma numeração de Gödel de termos e tipos é efetiva, e se são conhecidos inicialmente os números de Gödel de π, 0, *SUCC*, *PRED* e *COND*, então é possível construir efetivamente o número de Gödel de qualquer termo, operando

sucessivamente sobre os números iniciais.

▷ Funções **seta**, **alfa**, **comb**, **abs**:$\mathbb{P}^{(2)}\to\mathbb{P}$, **Y**:$\mathbb{P}\to\mathbb{P}$ são ditas *construtores* de uma numeração de termos e tipos com relação a **ng** sss:

- **seta** codifica a operação de formação de $(\sigma\to\tau)$ em função de σ e τ
- **Y** codifica a formação de Y_σ a partir de σ
- **alfa i** codifica a formação de α_i^σ a partir de σ
- **comb** codifica a formação (MN) a partir de M e N
- **abs** codifica a formação $(\lambda\alpha.M)$ a partir de α e M

▷ Os *discrimimadores* de uma numeração de termos e tipos **ng** são as funções que codificam, com relação a **ng**, os conjuntos de tipos de função, de termos, de constantes Y_σ, de variáveis e de combinações respectivamente. Os símbolos de função **éseta**, **éY**, **éalfa**, **écomb**, **éabs** correm sobre os respectivos discriminadores de **ng**.

▷ Uma numeração de Gödel de termos e tipos é Π-*efetiva* sss ela tem construtores e discriminadores Π-definíveis.

Neste ponto, é possível justificar a determinação de que **ng** tenha conjunto de chegada incluído em $\mathbb{M}$, i.e., que os números de Gödel sejam finitos totais. Se houvessem termos com números de Gödel propriamente parciais, os discriminadores não poderiam ser monotônicos, e, portanto, não poderiam ser Π-definíveis. A título de contradição, suponha que uma combinação (MN) tenha número de Gödel $\underline{\mathbf{n}}$. Então **écomb** $\underline{\mathbf{n}}=\mathsf{t}$. Por monotonicidade, **écomb** $\mathbf{x}=\mathsf{t}$ se $\underline{\mathbf{n}}\leq\mathbf{x}$. Logo, o conjunto:

- $\{\mathbf{x}\in\mathbb{P}\,|\,\textbf{écomb}\ \mathbf{x}=\mathsf{f}\}\subseteq\{\mathbf{x}\in\mathbb{P}\,|\,\text{não}\ \underline{\mathbf{n}}\leq\mathbf{x}\}=\{\underline{\mathbf{0}},\mathbf{0},\underline{\mathbf{1}},\mathbf{1},\ldots,\underline{\mathbf{n-1}}\}$

seria finito. Isto é impossível, pois há um número infinito

de termos que não são combinações. Um raciocínio análogo pode ser feito assumindo que (MN) tenha número de Gödel ∞. Neste, caso, a continuidade dos discriminadores seria violada.

▷ É conveniente ter disponíveis, também, discriminadores **étipo**, **étermo** e **éconst**, que codificam respectivamente os conjuntos de tipos, termos e constantes. Eles pode ser definidos a partir dos demais por:

- **étipo x=x==ng** π ∨ **éseta x**
- **éconst x=x==ng** 0 ∨ **x==ng** *SUCC*
  ∨ **x==ng** *PRED* ∨ **x==ng** *COND*
  ∨ **éY x**
- **étermo x=éconst x** ∨ **éalfa x** ∨ **écomb x** ∨ **éabs x**

É possível mostrar que, se uma numeração tem construtores Π-definíveis e a função que codifica o conjunto $\mathbb{T}$ é Π-definível, então os seus discriminadores são Π-definíveis. Assim, uma numeração **ng** é Π-efetiva sss ela tem construtores Π-definíveis e a função que codifica $\mathbb{T}$ com relação a **ng** é Π-definível. Por simplicidade, porém, é conveniente trabalhar com a definição menos compacta e elegante de Π-efetividade.

Pela definição de codificação com relação a uma numeração, as especificações dos construtores de uma numeração **ng** são equivalentes às especificações abaixo. Quando os números de Gödel dos tipos e constantes iniciais são fornecidos, estas especificações podem ser lidas como uma definição indutiva de **ng**:

- **ng**$(\sigma\to\tau)$=**seta** (**ng** σ) (**ng** τ)
- **ng** Y_σ=**Y** (**ng** σ)
- **ng** α_i^σ=**alfa i** (**ng** σ)
- **ng**(MN)=**comb** (**ng** M) (**ng** N)
- **ng**$(\lambda\alpha.M)$=**abs** (**ng** α) (**ng** M)

Teorema 8.1.1. Existe uma numeração de Gödel de termos e tipos **ng** Π-efetiva.

Prova. Tome:

- **classe <x,y>=x**
- **ng** π=**<0,0>**
- **seta x y=<1,x,y>**
- **contradomínio <x,y,z>=z**
- **étipo x=x==ng** π
  $\vee$ $\exists$**y**$\leq$**x.**$\exists$**z**$\leq$**x.x=seta y z** $\wedge$ **étipo y** $\wedge$ **étipo z**
- **éseta x=étipo x** $\wedge$ **classe x==0**
- **tipo <x,y,z>=y**
- **ng** 0=**<2,ng** π**,1>**
- **ng** *SUCC*=**<2,ng** $\pi\to\pi$**,2>**
- **ng** *PRED*=**<2,ng** $\pi\to\pi$**,3>**
- **ng** *COND*=**<2,ng** $\pi^{(3)}\to\pi$**,4>**
- **Y x=<3,x,5>**
- **alfa x y=<4,y,x>**
- **comb x y=<5,(contradomínio (tipo x),x,y>**
- **abs x y=<6,seta (tipo x) (tipo y),x,y>**

A primeira componente de um código codifica a classe sintática da entidade codificada. Ela garante, junto com as propriedades de <–,–>, que os construtores tenham conjuntos de chegada disjuntos para números de Gödel, de modo a que **ng** seja injetiva, como requerido. A segunda componente de um código de termo codifica o seu tipo, uma vez que, por indução nas regras de formação de termos, se mostra que, se **i=ng**$(M{:}\sigma)$, então **tipo**(**i**)=**ng**(σ). A sua finalidade é evitar a

necessidade de inferir tipos na verificação de boa tipagem, requerida para a definição de **étermo**. Tome:

- **étermo z=z==ng** 0
  - ∨ **z==ng** *SUCC*
  - ∨ **z==ng** *PRED*
  - ∨ **z==ng** *COND*
  - ∨ **∃x≤z.z==Y x ∧ étipo x**
  - ∨ **∃i≤z.∃x≤z.z==alfa i x ∧ étipo x**
  - ∨ **∃m≤z.∃n≤z.z==comb m n ∧ étermo m ∧ étermo n**
    ∧ tipo m == seta (tipo n) (tipo z)
  - ∨ **∃x≤z.∃m≤z.z==abs x m ∧ étermo x ∧ étermo m**
    ∧ tipo z == seta (tipo x) (tipo m)

- **éY z=étermo z ∧ classe x==3**
- **éalfa z=étermo z ∧ classe x==4**
- **écomb z=étermo z ∧ classe x==5**
- **éabs z=étermo z ∧ classe x==6**

As funções dadas são Π-definíveis, de acordo com os resultados do capítulo **5**, e, portanto, **ng** é Π-efetiva. □

▷ Funções **domínio**, **contradomínio**, **Ytipo**, **alfatipo**, **alfaind**, **fun**, **arg**, **par**, **corpo** que satisfazem as propriedades abaixo, para uma numeração de termos e tipos **ng**, são ditas *destrutores* de **ng**:

- **domínio** (**ng**$(\sigma\to\tau)$)=**ng** σ
- **contradomínio** (**ng**$(\sigma\to\tau)$)=**ng** τ
- **Ytipo** (**ng** Y_σ)=**ng** σ
- **alfatipo** (**ng** α_ι^σ)=**ng** σ
- **alfaind** (**ng** α_ι^σ)=**i**
- **fun** (**ng**(MN))=**ng** M
- **arg** (**ng**(MN))=**ng** N
- **par** (**ng**$(\lambda\alpha.M)$)=**ng** α
- **corpo** (**ng**$(\lambda\alpha.M)$)=**ng** M

A numeração **ng** construída no teorema 8.1.1 acima tem destrutores Π-definíveis, como pode facilmente ser mostrado.

Lema 8.1.2. Toda numeração Π-efetiva **ng** tem destrutores Π-definíveis.

Prova. Todos os destrutores são definíveis de maneira similar. É mostrado o caso de **domínio:**

- **domínio z=x/∃y.étipo x ∧ étipo y ∧ z==seta x y** □

▷ Uma função **tipo** é dita *extrator de tipo* sss:

- **tipo (ng** M**)=ng** σ se $M:\sigma$

Lema 8.1.3. Toda numeração Π-efetiva **ng** tem extrator de tipo Π-definível.

Prova. tipo é um extrator de tipo sss **tipo** satisfaz:

- **tipo(ng** 0**)=ng** π
  tipo(ng *SUCC***)=ng**$(\pi\to\pi)$
  tipo(ng *PRED***)=ng**$(\pi\to\pi)$
  tipo(ng *COND***)=ng**$(\pi^{(3)}\to\pi)$
  tipo(Y x)=x
  tipo(alfa i x)=x
  tipo(comb x y)=z se **tipo x=seta z t** para algum **t**
  tipo(abs x y)=seta (tipo x) (tipo y)

Conseqüentemente, a seguinte definição para **tipo**, que utiliza os resultados do lema 8.1.2, é correta:

- **tipo(x)=if x==ng** 0 **then ng** π
  else if x==ng *SUCC* **then ng**$(\pi\to\pi)$
  else if x==ng *PRED* **then ng**$(\pi\to\pi)$
  else if x==ng *COND* **then ng**$(\pi^{(3)}\to\pi)$
  else if éY x then Ytipo x
  else if éalfa x then alfatipo x
  else if écomb x then contradomínio (tipo (fun x))
  else if éabs x then seta (tipo (par x))
  (tipo (corpo x)) □

O método utilizado no lema 8.1.3 acima de transformar uma especificação indutiva que utiliza construtores em uma definição recursiva que utiliza discriminadores e destrutores é utilizado implicitamente em muitas provas.

O seguinte teorema mostra que duas numerações Π-efetivas distintas são essencialmente a mesma, uma vez que a tradução entre elas é Π-definível.

Teorema 8.1.4. Para quaisquer numerações Π-efetivas **ng** e **ng'**, existe uma função Π-definível **trad**:$\mathbb{P}\to\mathbb{P}$ tal que **ng'=trad∘ng**.

Prova. ng'=trad∘ng sss:

- **trad(ng** π**)=ng'** π
- **trad(seta x y)=seta'(trad x) (trad y)**
- **trad(ng** 0**)=ng'** 0
- **trad(ng** *SUCC***)=ng'** *SUCC*
- **trad(ng** *PRED***)=ng'** *PRED*
- **trad(ng** *COND***)=ng'** *COND*
- **trad(Y x)=Y'(trad x)**
- **trad(alfa x y)=alfa'x (trad y)**
- **trad(comb x y)=comb'(trad x) (trad y)**
- **trad(abs x y)=abs'(trad x) (trad y)** □

▷ No que segue do texto, **ng** é uma numeração de Gödel qualquer, e Π-efetiva, a menos que indicado em contrário, com construtores, discriminadores, destrutores e extrator de tipo Π-definíveis escritos de acordo com as convenções acima. A omissão de uma referência obrigatória a uma numeração é considerada implicitamente relativa a **ng**.

8.2 Números de Gödel de elementos Π-definíveis

▷ O conjunto de *números de Gödel de elementos de* $\mathcal{C}_\sigma$ é definido por:

· $\mathbf{NG}_\sigma = \{\mathbf{ng}\ M \mid M{:}\sigma \text{ é fechado}\}$

▷ A função de *valoração de números de Gödel de elementos de* $\mathcal{C}_\sigma$ $\mathbf{val}_\sigma{:}\mathbf{NG}_\sigma \to \mathcal{C}_\sigma$ é definida por:

· $\mathbf{val}_\sigma(\mathbf{ng}\ M) = x$ se M denota x

▷ *Um número de Gödel de* $x \in \mathcal{C}_\sigma$ é qualquer $\mathbf{i} \in \mathbf{NG}_\sigma$ tal que $\mathbf{val}_\sigma \mathbf{i} = x$. Tal x é dito o *valor* de **i**. A expressão $\mathbf{ng}_\sigma x$ corre sobre números de Gödel de $x \in \mathcal{C}_\sigma$.

É imediato que $x \in \mathcal{C}_\sigma$ tem números de Gödel sss x é Π-definível, e que para qualquer número de Gödel $\mathbf{ng}_\sigma x$ de x, $\mathbf{val}_\sigma(\mathbf{ng}_\sigma x) = x$.

8.3 Π-indecidibilidade do problema da parada fraca

▷ O *problema da parada fraca* é o problema de decidir, para qualquer programa P, se P pára fracamente ou não (cf. seção 5.4).

▷ Uma função $\mathbf{h}{:}\mathbb{P} \to \mathbb{B}$ *resolve o problema da parada fraca via uma numeração de Gödel* **ng** sss **h** codifica, com relação a

ng, o conjunto de programas que param fracamente.

Pela definição de programa, de parada fraca e pelo corolário 5.3.2, **h** satisfaz, para todo **x**∈ℙ e para todo número de Gödel $\mathbf{ng}_\pi\mathbf{x}$ de **x**:

- $\mathbf{h}(\mathbf{ng}_\pi\mathbf{x})=\mathbf{1}$ se $\mathbf{x}\neq\underline{\mathbf{0}}$
  $=\mathbf{0}$ se $\mathbf{x}=\mathbf{0}$

Note que <u>há</u> numerações de Gödel **ng** para as quais existe **h** Π-definível que resolve o problema da parada fraca via **ng**. Considere a numeração de Gödel especificada a seguir. Ordene lexicograficamente os termos-Π. Reserve os números pares para os termos fechados de tipo π que têm forma normal fraca, e os ímpares para os demais. Atribua os números pela ordem lexicográfica. A função ℙ-característica do conjunto dos pares é Π-definível e resolve o problema da parada fraca via tal numeração. Pelo teorema 8.3.3 abaixo, tal numeração não pode ser Π-efetiva.

Lema 8.3.1. Existe **num**:ℙ→ℙ Π-definível tal que, para todo **n**∈𝕄 $\mathbf{val}_\pi(\mathbf{num}\ \mathbf{n})=\mathbf{n}$.

Prova. Tome:

- **num 0**=**ng** 0
- **num**(**x**+**1**)=**comb** (**ng** *SUCC*) (**num x**)

Claramente, para todo **n**∈𝕄, **num n**=**ng** ($SUCC^n0$), que é um número de Gödel de **n**. □

Lema 8.3.2. Para toda f:ℙ→$\mathcal{C}_\sigma$ Π-definível, e todo **n**∈𝕄:

- $\mathbf{val}_\sigma(\mathbf{comb}\ (\mathbf{ng}_{\pi\to\sigma}f)\ (\mathbf{num}\ \mathbf{n}))=\mathbf{val}_\sigma(\mathbf{ng}_\sigma(f\ \mathbf{n}))$

Prova. Pela definição de $\mathbf{val}_\sigma$ e **comb**, para todos $\mathbf{i}\in\mathbf{NG}_{\tau\to\sigma}$ e $\mathbf{j}\in\mathbf{NG}_\tau$:

- $\mathbf{val}_\sigma(\mathbf{comb}\ \mathbf{i}\ \mathbf{j})=(\mathbf{val}_{\tau\to\sigma}\mathbf{i})(\mathbf{val}_\tau\mathbf{j})$

Logo:

· $\mathbf{val}_{\sigma}(\mathbf{comb}\ (\mathbf{ng}_{\pi\to\sigma} f)\ (\mathbf{num}\ \mathbf{n})) = (\mathbf{val}_{\pi\to\sigma}(\mathbf{ng}_{\pi\to\sigma} f))(\mathbf{val}_{\pi}(\mathbf{num}\ \mathbf{n}))$
$= f\ \mathbf{n}$
$= \mathbf{val}_{\sigma}(\mathbf{ng}_{\sigma}(f\ \mathbf{n}))$ □

Teorema 8.3.3. Para toda numeração de Gödel **ng**, se **h** resolve o problema da parada fraca via **ng**, então:

· **h** é Π-definível implica **ng** não é Π-efetiva

Prova. Suponha que **h** é Π-definível e que **ng** é Π-efetiva. Então **f**:ℙ→ℙ definida abaixo é Π-definível, por que **comb** é Π-definível:

· **f x=if h(comb x (num x)) then** $\underline{\mathbf{0}}$ **else 0**

Como **f** é Π-definível, **f** tem um número de Gödel $\mathbf{ng}_{\pi\to\pi}\mathbf{f}$. Tome $\mathbf{y}=\mathbf{f}(\mathbf{ng}_{\pi\to\pi}\mathbf{f})$. Então, substituindo **x** por $\mathbf{ng}_{\pi\to\pi}\mathbf{f}$ na definição de **f x:**

· $\mathbf{y}=$**if h(comb** $(\mathbf{ng}_{\pi\to\pi}\mathbf{f})$ **(num** $(\mathbf{ng}_{\pi\to\pi}\mathbf{f}))$ **then** $\underline{\mathbf{0}}$ **else 0**

Pelo lema 8.3.2:

· $\mathbf{y}=$**if h(**$\mathbf{ng}_{\pi}(\mathbf{f}(\mathbf{ng}_{\pi\to\pi}\mathbf{f}))$ **then** $\underline{\mathbf{0}}$ **else 0**

Uma vez que, por definição, $\mathbf{y}=\mathbf{f}(\mathbf{ng}_{\pi\to\pi}\mathbf{f})$, tem-se que:

· $\mathbf{y}=$**if h(**$\mathbf{ng}_{\pi}\mathbf{y}$**) then** $\underline{\mathbf{0}}$ **else 0**

Claramente, **y** pode apenas ser $\underline{\mathbf{0}}$ ou **0**, pelas propriedades do condicional. Se **y=0**, então $\mathbf{h}(\mathbf{ng}_{\pi}\mathbf{y})=\mathbf{1}$, pela definição de **h**, e:

· **y=if 1 then** $\underline{\mathbf{0}}$ **else 0=**$\underline{\mathbf{0}}$

Então **y** não pode ser **0**. Se $\mathbf{y}=\underline{\mathbf{0}}$, então $\mathbf{h}(\mathbf{ng}_{\pi}\mathbf{y})=\mathbf{0}$, pela definição de **h**, e:

· **y=if 0 then** $\underline{\mathbf{0}}$ **else 0=1**

Então **y** não pode ser $\underline{\mathbf{0}}$. De tal contradição se conclui que **h** não é Π-definível ou **ng** não é Π-efetiva. □

O argumento é semelhante ao utilizado originalmente por [TUR 36] para provar o teorema da parada. O seguinte corolário enuncia o teorema de um modo mais usual.

Corolário 8.3.3. Para toda numeração de Gödel efetiva **ng**, o problema da parada fraca é Π-indecidível. □

Mas este problema é Π-semidecidível para numerações Π-efetivas, como mostrado no teorema 8.5.2.

Teorema 8.3.4. A equivalência de denotação é Π-indecidível para numerações Π-efetivas.

Prova. Para termos de tipo π, isto é imediato do teorema da parada fraca. Para termos de tipo σ qualquer, basta notar que, para $\mathbf{x},\mathbf{y}\in\mathbb{P}$ e $f,g\in\mathcal{C}_\sigma$, com $f=\lambda z_1\ldots z_k.\mathbf{x}$ e $g=\lambda z_1\ldots z_k.\mathbf{y}$, onde k e os domínios de $z_1,\ldots,z_k$ dependem de σ, vale que $\mathbf{x}=\mathbf{y}$ sss $f=g$. □

8.4 Codificação da função de passo de computação

Esta seção mostra que, para qualquer numeração Π-efetiva **ng**, a função de passo de computação *passo* tem codificação Π-definível.

Lema 8.4.1. As função **livre**:$\mathbb{P}^{(2)}\to\mathbb{B}$ e **éfechado**:$\mathbb{P}\to\mathbb{B}$ que codificam respectivamente os conjuntos:

- $\{(\alpha,M)\,|\,\alpha$ ocorre livre em $M\}$
- $\{M\,|\,M$ é fechado$\}$

são Π-definíveis.

Prova. Tome:

- **livre x y =**
  éalfa x ∧ **étermo y**
  ∧ (**éalfa y** ∧ **x==y**
  ∨ **écomb y** ∧ (**livre x (fun y)** ∨ **livre x (arg y)**)
  ∨ **éabs y** ∧ ¬**x==par y** ∧ **livre x (corpo y)**)

A construção de **éfechado** é omitida. □

Lema 8.4.2. A função de substituição tem codificação **subst** Π-definível.

Prova. **subst** codifica a subtituição sss:

- **subst** (**ng** *M*) (**ng** *N*) (**ng** α)=**ng**($M[N/\alpha]$)

A definição da substituição para o caso $((\lambda\alpha_j^\sigma.M)[N/\alpha_i^\tau]$, $\alpha_j^\sigma \neq \alpha_i^\tau$ é repetida abaixo:

- $((\lambda\alpha_j^\sigma.M)[N/\alpha_i^\tau]=(\lambda\alpha_k^\sigma.M[\alpha_k^\sigma/\alpha_j^\sigma][N/\alpha_i^\tau])$

para o menor k tal que α_k^σ que não ocorre livre em N nem em M

Tome:

- **subst c n a=c** se **c** codifica constante
- **subst a n a=n** se **a** codifica variável
- **subst a'n a=a** se **a** e **a'** codificam variáveis, **a≠a'**
- **subst (comb m m') n a=comb (subst m n a) (subst m'n a)**
- **subst (abs a m) n a=abs a m**
- **subst (abs j m) n i=abs k (subst (subst m k j) n i)**

se **j≠i**, onde:

- **s=alfatipo j,**
- **k=alfa (μx.¬livre (alfa x s) n) s** □

Lema 8.4.3. O conjunto de redexes que são combinações tem codificação **éredex** Π-definível.

Prova. Um redex que é uma combinação é de uma das formas:

- $(\lambda\alpha.M)N$
- *YM*
- *PRED* 0
- *PRED* (*SUCC M*)
- *COND* 0 *M N*
- *COND* (*SUCC P*) *M N*
- *COND P* 0 0
- *COND P* (*SUCC M*) (*SUCC N*)

Tal função **éredex** pode ser construída mediante o uso do condicional e dos discriminadores e destrutores de **ng** de modo óbvio, embora trabalhoso. □

Lema 8.4.4. Existe **red**:$\mathbb{P}\to\mathbb{P}$ Π-definível tal que, para toda combinação M:

- **red** (**ng** M)=**ng** N se $M\Rightarrow_{det} N$

Prova. **red** satisfaz a especificação do teorema sss:

- **red** (**ng**$((\lambda\alpha.M)N)$)=**ng**$(M[N/\alpha])$
  =**subst** (**ng** M) (**ng** N) (**ng** α)
- **red** (**ng**(*PRED* 0)) = **ng** 0
- **red** (**ng**(*PRED* (*SUCC* M))) = **ng** M
- **red** (**ng**(*COND* 0 M N)) = **ng** N
- **red** (**ng**(*COND* (*SUCC* P) M N)) = **ng** M
- **red** (**ng**(*COND* P 0 0)) = **ng** 0
- **red** (**ng**(*COND* P (*SUCC* M) (*SUCC* N)))= **ng**(*SUCC* (*COND* P M N))
- **red** (**ng**(*YM*)) = **ng**$(M(YM))$

E, portanto, **red** pode ser definida utilizando o condicional, os discriminadores e os destrutores de **ng**. □

Teorema 8.4.5. A função de passo de computação *passo* tem codificação Π-definível **passo**.

Prova. **passo** codifica *passo* sss:

- **passo**(**ng** M)=**ng**(*passo* M)

sss:

- **passo c**=**c** se **c** codifica constante
- **passo a**=**a** se **a** codifica variável
- **passo** (**comb m m′**)=**comb** (**passo m**) (**passo m′**)
  se **comb m m′** não codifica redex
- **passo** (**comb m m′**)=**red** (**comb m m′**)
  se **comb m m′** codifica redex
- **passo** (**abs a m**)=**abs a** (**passo m**) □

8.5 Π-semidecidibilidade do problema da parada fraca

▷ As funções de valoração $\mathbf{val}_\sigma$:$\mathbf{NG}_\sigma\to\mathscr{C}_\sigma$ definidas na seção

8.2 podem ser extendidas para uma função contínua $\mathbb{P}\to\mathcal{C}_\sigma$ pela atribuição do valor $\perp_\sigma$ a elementos de $\mathbb{P}$ que não são números de Gödel:

- $\mathbf{val}_\sigma\mathbf{i}=\perp_\sigma$ se $\mathbf{i}\notin\mathbf{NG}_\sigma$

Teorema 8.5.1. A função de valoração $\mathbf{val}_\pi:\mathbb{P}\to\mathbb{P}$ é Π-definível.

Prova. Tome:

- $\mathbf{val}_\pi\mathbf{i}=$**if** étermo **i** $\wedge$ **tipode i==ng** π $\wedge$ **éfechado i** **then v i**

onde $\mathbf{v}:\mathbb{P}\to\mathbb{P}$ é tal que:

- **v(ng** 0**)=0**
- **v(ng(***SUCC M***))=succ(v(ng** *M***))**
- **v(ng** *M***)=v(ng(***passo M***))** se *M* não é f.n.fraca

e é definida por:

- **v x=if x==ng** 0 **then 0**
  else if écomb x $\wedge$ **fun x==ng** *SUCC* **then succ(v(arg x))**
  else v(passo x) □

Compare a função **v** definida na prova do teorema 8.5.1 acima com o algoritmo procedural descrito na seção 5.4.

Teorema 8.5.2. O problema da parada fraca é Π-semidecidível para numerações Π-efetivas.

Prova. A função $\mathbf{h}_t:\mathbb{P}\to\mathbb{P}$ definida abaixo tem como núcleo os códigos de programas que param fracamente:

- $\mathbf{h}_t\mathbf{i}=$**édefinido(val**$_\pi$**i)**.

onde a função **édefinido** foi definida na seção 6.4. □

9 RELAÇÕES COM A TEORIA DA RECURSÃO

A seção 9.1 estuda a noção de função universal externa e funções universais externas particulares. A seção 9.2 define uma função universal interna particular. A seção 9.3 estuda termos universais e manipulação de seqüências infinitas. A seção 9.4 estuda certas operações Π-definíveis sobre conjuntos, via índices de funções $\mathbb{P}$-características, que não são recursivas via índices de funções características. A seção 9.5 mostra que existem funções recursivas que não têm extensão natural Π-definível, e que, mesmo para funções recursivas que têm extensão natural Π-definível, tal extensão não pode ser encontrada por uma função Π-definível.

9.1 Funções universais externas

▷ Uma função $\mathbf{univ}^{\sigma}:\mathbb{P}\to\mathcal{C}_{\sigma}$ é dita *universal externa para* $\mathcal{C}_{\sigma}$ sss $\mathbf{univ}^{\sigma}$ é Π-definível e para todo $x\in\mathcal{C}_{\sigma}$ Π-definível existe $\mathbf{i}\in\mathbb{P}$ tal que:

· $x=\mathbf{univ}^{\sigma}\mathbf{i}$

A denominação *universal* é devida a Turing [TUR 36]. A denominação *externa* é devida ao fato que $\mathbf{univ}^{\sigma}\notin\mathcal{C}_{\sigma}$.

Funções de valoração $\mathbf{val}_{\sigma}:\mathbb{P}\to\mathcal{C}_{\sigma}$ Π-definíveis são funções universais para $\mathcal{C}_{\sigma}$. Mas, ao contrário da definição de função de valoração, a definição de função universal não faz referência a números de Gödel de termos. Para se provar que funções universais existem, porém, são utilizadas numerações de Gödel de termos.

As funções predecessor e identidade $\mathbb{P}\to\mathbb{P}$ são funções universais para $\mathcal{C}_\pi$.

Os elementos Π-definíveis podem ser caracterizados via funções universais, do seguinte modo:

- Um elemento $x\in\mathcal{C}_\sigma$ é Π-definível sss existe **i** tal que $x=\mathbf{univ}^\sigma\mathbf{i}$, com $\mathbf{univ}^\sigma$ função universal para $\mathcal{C}_\sigma$.

Para os propósitos deste trabalho, interessam os casos $\sigma=\pi^{(k)}\to\pi$, $k\in\mathbb{N}$, e $\sigma=(\pi\to\pi)\to\pi$. A notação $\mathbf{univ}^{(k)}$ abrevia $\mathbf{univ}^\sigma$ para $\sigma=\pi^{(k)}\to\pi$, e **univ** abrevia $\mathbf{univ}^{(\pi\to\pi)\to\pi}$.

Assim como uma função $\mathbb{P}^{(k)}\to\mathbb{P}$ é pensada como uma função de k argumentos, uma função $(\mathbb{P}\to\mathbb{P})\to\mathbb{P}$ é pensada como uma função de infinitos argumentos. Assim, **univ** poderia ter sido escrita $\mathbf{univ}^\omega$.

▷ Para cada D, o domínio $(\mathbb{P}\to D)$ é escrito D^ω, e é dito domínio de D-seqüências. O elemento $x=[x_0,\ldots,x_{k-1}]\in D^\omega$, com $x_0,\ldots,x_{k-1}\in D$, é definido por, para $\mathbf{i}\in\mathbb{P}$:

- $x\mathbf{i}=$ **if i==0 then** x_0
  else ...
  else if i==k-1 then x_{k-1}

Assim, $x\mathbf{i}$ é escrito x_i.

Claramente, $[x_0,\ldots,x_{k-1}]$ é Π-definível sss $x_0,\ldots,x_{k-1}$ são Π-definíveis. Para $i<k$, $[x_0,\ldots,x_{k-1}]_i=x_i$, e para $i\geq k$, $[x_0,\ldots,x_{k-1}]_i=\perp$, o elemento menos definido de D. Em particular, se permite a seqüência [] para $k=0$.

Lema 9.1.1. Se existe uma função universal **univ**, então para cada k existe uma função universal $\mathbf{univ}^{(k)}$.

Prova. Seja $\mathbf{f}:\mathbb{P}^{(k)}\to\mathbb{P}$ Π-definível. Então $\mathbf{g}:\mathbb{P}^\omega\to\mathbb{P}$ tal que:

· $\mathbf{g\ x = f\ x_1 \ldots x_k}$

é Π-definível, e:

· $\mathbf{f\ x_1 \ldots x_k = g\ [x_1, \ldots, x_k]}$

Seja **i** tal que **g=univ i**. Então:

· $\mathbf{f\ x_1 \ldots x_k = univ\ i\ [x_1, \ldots, x_k]}$

Logo, se $\mathbf{univ}^{(k)}$ é definida por:

· $\mathbf{univ}^{(k)}\mathbf{i\ x_1 \ldots x_k = univ\ i\ [x_1, \ldots, x_k]}$

tem-se que $\mathbf{univ}^{(k)}$ é Π-definível e que $\mathbf{f = univ}^{(k)}\mathbf{i}$, como requerido. □

Para construir funções universais, é introduzida a noção de função de valoração com ambiente. Funções de valoração com ambiente são análogas à função semântica da seção 5.1.2.

▷ Um *σ-ambiente* é uma função $\mathbb{P} \to \mathcal{C}_\sigma$. O símbolo de função ρ corre sobre σ-ambientes. Um σ-ambiente é uma $\mathcal{C}_\sigma$-seqüência, e as notações de seqüência são utilizadas para σ-ambientes.

σ-ambientes são utilizados para atribuir valores a variáveis formais de Π, como na seção 5.1.2. Especificamente, um σ-ambiente ρ atribui o valor ρ_i à variavel α_i^σ.

▷ Um termo é dito *σ-aberto de aridade k* sss as suas variáveis livres estão incluídas em $\{\alpha_0^\sigma, \ldots, \alpha_{k-1}^\sigma\}$. O conjunto $\mathbf{NG}_\tau^\sigma$ é definido por:

· $\mathbf{NG}_\tau^\sigma = \{M{:}\tau \mid \text{existe } k \text{ tal que } M \text{ é } \sigma\text{-aberto de aridade } k\}$

O *fecho-σ-k* de um termo M é $(\lambda\alpha_0^\sigma \ldots \lambda\alpha_{k-1}^\sigma . M)$.

▷ A *função de valoração para τ com σ-ambiente* $\mathbf{val}_\tau^\sigma : \mathbf{NG}_\tau^\sigma \to (\mathbb{P} \to \mathcal{C}_\sigma) \to \mathcal{C}_\tau$ é definida por:

· $\mathbf{val}_\tau^\sigma(\mathbf{ng}\ M)[x_0, \ldots, x_{k-1}] = f\ x_0 \ldots x_{k-1}$

onde $f:\mathcal{C}_\sigma^{(k)}\to\mathcal{C}_\tau$ é a função denotada pelo fecho-σ-k de M.

A função $\mathbf{val}_\tau^\sigma:\mathbf{NG}_\tau^\sigma\to(\mathbb{P}\to\mathcal{C}_\sigma)\to\mathcal{C}_\tau$ pode extendida para uma função contínua $\mathbb{P}\to(\mathbb{P}\to\mathcal{C}_\sigma)\to\mathcal{C}_\tau$ pela atribuição de valores arbitrários apropriados a $(\mathbf{val}_\tau^\sigma\mathbf{i})$ para $\mathbf{i}\notin\mathbf{NG}_\tau^\sigma$. Estes valores não são importantes, uma vez que é possível decidir se $\mathbf{i}\in\mathbf{NG}_\tau^\sigma$ ou não, por uma função Π-definível, se a numeração é Π-efetiva.

Lema 9.1.2. Se existe $\mathbf{val}_\tau^\sigma:\mathbb{P}\to(\mathbb{P}\to\mathcal{C}_\sigma)\to\mathcal{C}_\tau$ Π-definível, então existe $\mathbf{univ}^{\sigma\to\tau}$.

Prova. Seja $f:\mathcal{C}_\sigma\to\mathcal{C}_\tau$ Π-definível. Então existe $M:\sigma\to\tau$ fechado que denota **f**. Tome $\mathbf{i}=\mathbf{ng}(M\alpha_0^\sigma)$. Então, por definição de $\mathbf{val}_\tau^\sigma$:

- $\mathbf{f}\ x=\mathbf{val}_\tau^\sigma\mathbf{i}\ [x]$

Logo, se $\mathbf{univ}^{\sigma\to\tau}$ é definida por:

- $\mathbf{univ}^{\sigma\to\tau}\mathbf{i}=\lambda x.\mathbf{val}_\tau^\sigma\mathbf{i}\ [x]$

tem-se que $\mathbf{univ}^{\sigma\to\tau}$ é Π-definível e que $\mathbf{f}=\mathbf{univ}^{\sigma\to\tau}\mathbf{i}$, como requerido. □

Então, para construir **univ**, é suficiente que haja $\mathbf{val}_\pi^{\pi\to\pi}$ Π-definível.

Lema 9.1.3. Existe $\mathbf{val}_\pi^{\pi\to\pi}$ Π-definível.

Prova. $\mathbf{v}=\mathbf{val}_\pi^{\pi\to\pi}$ é função de valoração para π com $\pi\to\pi$-ambiente sss:

- $\mathbf{v}(\mathbf{ng}(Y_\pi\alpha_i^{\pi\to\pi}))\ \rho=\mathbf{fix}_{\mathbb{P}}\rho_i$
- $\mathbf{v}(\mathbf{ng}(\alpha_i^{\pi\to\pi}M))\ \rho=\rho_i\ (\mathbf{v}(\mathbf{ng}\ M)\ \rho)$
- $\mathbf{v}(\mathbf{ng}\ 0)\ \rho=\mathbf{0}$
- $\mathbf{v}(\mathbf{ng}(SUCC\ M))\ \rho=\mathbf{succ}(\mathbf{v}(\mathbf{ng}\ M)\ \rho)$
- $\mathbf{v}(\mathbf{ng}\ M)\ \rho=\mathbf{v}(\mathbf{ng}(passo\ M))\ \rho$

  para M que não recai em um caso anterior

Tal função **v** pode ser construída utilizando os discriminadores e destrutors de **ng**, e a função **passo** definida no teorema 8.4.5. □

Teorema 9.1.4. Existe uma função universal **univ**, e existem funções universais $\mathbf{univ}^{(k)}$ para cada k.

Prova. Aplicação direta dos lemas 9.1.2, 9.1.3 e 9.1.1. □

9.2 Funções universais internas

Pela definição de função universal, para todo k existe $\mathbf{u}_k \in \mathbb{P}$ tal que:

- $\mathbf{univ}^{(k)} = \mathbf{univ}^{(k+1)}\,\mathbf{u}_k$

Assim, $\mathbf{univ}^{(k)}$ engloba $\mathbf{univ}^{(j)}$ para todo $j<k$. Pelo teorema 9.1.1, **univ** engloba $\mathbf{univ}^{(j)}$ para todo j. Em um sentido, **univ** engloba a si mesma, também, embora não possa haver $\mathbf{u}_\omega \in \mathbb{P}$ tal que:

- $\mathbf{univ} = \mathbf{univ}\ \mathbf{u}_\omega$

pois **univ** e ($\mathbf{univ}\ \mathbf{u}_\omega$) têm funcionalidades diferentes:

- $\mathbf{univ} : \mathbb{P} \to \mathbb{P}^\omega \to \mathbb{P}$
- $\mathbf{univ}\ \mathbf{u}_\omega : \mathbb{P}^\omega \to \mathbb{P}$

onde por funcionalidade se entende, aqui, a contrapartida semântica da noção formal de tipo, i.e., ▷ *x tem funcionalidade* $\mathcal{C}_\sigma$ sss $x \in \mathcal{C}_\sigma$.

▷ Uma função $f:\mathbb{P}\to\mathbb{P}^\omega\to\mathbb{P}$ pode ser vista como uma função que toma primeiro um argumento e depois infinitos argumentos. Esta função pode ser transformada em uma função $g:\mathbb{P}^\omega\to\mathbb{P}$, equivalente, que toma o primeiro e os demais argumentos de uma só vez:

- $g\ x = f\ x_0\ x'$

onde o operador $(')$:$\mathbb{P}^{\omega}\to\mathbb{P}^{\omega}$ tal que:

- $[x_0,x_1,\ldots,x_{k-1}]'=[x_1,\ldots,x_{k-1}]$

é definido por:

- $(x')_i=x_{i+1}$

Esta dependência entre f e g é escrita g=**uncurry** f, para **uncurry**:$(\mathbb{P}\to\mathbb{P}^{\omega}\to\mathbb{P})\to(\mathbb{P}^{\omega}\to\mathbb{P})$ definida por:

- **uncurry** $f=\lambda x.f\ x_0\ x'$

A equivalência de f e g consiste em que, para todo $x\in\mathbb{P}^{\omega}$:

- $f\ x_0\ x'=g\ x$

Isto é, para todos $d\in\mathbb{P}, x\in\mathbb{P}^{\omega}$:

- $f\ d\ x=g$(**cons** $d\ x$)

onde o operador **cons**:$\mathbb{P}\to\mathbb{P}^{\omega}\to\mathbb{P}^{\omega}$ tal que:

- (**cons** $d\ x$)$_0=d$
- (**cons** $d\ x$)$'=x$
- **cons** $x_0[x_1,\ldots,x_{k-1}]=[x_0,x_1,\ldots,x_{k-1}]$

é definido por:

- (**cons** $d\ x$)$_i$=**if i==0 then** d **else** x_{i-1}

Assim, tem-se que f=**curry** g, onde **curry**:$(\mathbb{P}^{\omega}\to\mathbb{P})\to(\mathbb{P}\to\mathbb{P}^{\omega}\to\mathbb{P})$ é definida por:

- **curry** $g=\lambda d.\lambda x.g$(**cons** $d\ x$)

E, então:

- f=**curry**(**uncurry** f)

Mas pode ser provado que a igualdade g=**uncurry**(**curry** g) não vale, e que apenas vale que:

- $g\leq$ **uncurry**(**curry** g)

A igualdade vale somente para g tal que:

- $g\ x{=}g(\mathbf{cons}\ x_0 x')$

Isto ocorre por que:

- $x_i{=}(\mathbf{cons}\ x_0 x')_i$ sss $\mathbf{i{\neq}\underline{0}}$
- $x_{\underline{0}} \leq (\mathbf{cons}\ x_0 x')_{\underline{0}}{=}x_0 \sqcap x_{\underline{1}}$

Conseqüentemente, tais g são caracterizados como os que não dependem do valor $x_{\underline{0}}$. Ainda, a função **uncurry∘curry** transforma g em uma função que não depende de tal valor, e é um operador idempotente. Especificamente, se g depende de $x_{\underline{0}}$ então **uncurry(curry** g**)** mantém a mesma relação de dependência com $x_0 \sqcap x_{\underline{1}}$.

A função **univ** engloba a si mesma, no sentido que há um $\mathbf{u}_\omega{\in}\mathbb{P}$ tal que:

- $\mathbf{univ{=}curry(univ}\ \mathbf{u}_\omega\mathbf{)}$

Isto é possível, pois **univ** e **curry(univ** $\mathbf{u}_\omega$**)** têm funcionalidades iguais:

- **univ** $:\mathbb{P}{\to}\mathbb{P}^\omega{\to}\mathbb{P}$
- **curry(univ** $\mathbf{u}_\omega$**)** $:\mathbb{P}{\to}\mathbb{P}^\omega{\to}\mathbb{P}$

Teorema 9.2.1. Existe $\mathbf{u}_\omega{\in}\mathbb{P}$ tal que **univ=curry(univ** $\mathbf{u}_\omega$**)**.

Prova. As funções **curry** e **uncurry** são Π-definíveis. Então a função **int=(uncurry univ)** $:\mathbb{P}^\omega{\to}\mathbb{P}$ é Π-definível. Pela definição de função universal, existe **i** tal que:

- **int=univ i**

A discussão sobre **uncurry∘curry** a seguir pode ser omitida sem prejuízo. Ela consta apenas por completeza.

Logo, aplicando **curry** a ambos membros da igualdade e simplificando-a:

· **univ=curry(univ i)**

Então, tome $\mathbf{u}_{\omega}$=**i**. □

Mais importante que o teorema é a sua conseqüência imediata. Pela definição de **curry**, tem-se que:

· **int(cons i** x**)=univ i** x

Pela definição de **u**, vale que:

· **int(cons u** x**)=int** x

E, de novo pela definição de **curry**, obtem-se que:

· **curry int u=int**

▷ Assim, A função **int** pode ser considerada uma função *universal interna* para a classe de funções $\mathcal{F}$ definida por:

· $\mathcal{F}=\{f:\mathbb{P}^{\omega}\rightarrow\mathbb{P} \mid f$ é Π-definível$\}$

no sentido que:

· **int**$\in\mathcal{F}$

· para toda $f\in\mathcal{F}$ existe **i**$\in\mathbb{P}$ tal que f=**curry int i**

A última afirmação é equivalente a:

· para toda $f\in\mathcal{F}$ existe **i**$\in\mathbb{P}$ tal que $f\ x$=**int(cons i** x**)**

A função **cons** é dita função de *entrada para* **int**, de acordo com a nomenclatura de [SAN 88]. Uma generalização do conceito de função universal interna para outras classes de funções Π-definíveis que $\mathcal{F}$ é omitida, mas pode ser facilmente obtida, caso desejado.

A função **cons** cumpre o papel que as funções de pareamento cumprem na teoria da recursão, de passar mais de um argumento a uma função que em princípio recebe somente um. O papel de **curry** é aumentar de um o número de argumentos da

função à qual se aplica, mediante o uso de **cons**. A aplicação k argumentos $x_0,x_1,\ldots,x_{k-1}$ a uma função $f:\mathbb{P}^\omega\to\mathbb{P}$ que recebe um único argumento x, é representada por:

- ((**curry** ... (**curry** (**curry** f x_0) x_1) ... x_{k-1}) []

que é o mesmo que, expandindo a definição de **curry**:

- f (**cons** x_0(**cons** x_1... (**cons** x_{k-1}[])...))

e que:

- f $[x_0,x_1,\ldots,x_{k-1}]$

como já tinha sido feito no teorema 9.1.1.

É necessário utilizar tal função **cons** de ordem superior, pois não existem funções de primeira ordem $\mathbb{P}^2\to\mathbb{P}$ bijetivas (ou injetivas), e contínuas. Se existissem, $\mathbb{P}^2$ seria *isomorfo a (ou incluído em)* $\mathbb{P}$. Ainda, os domínios $\mathbb{P}^{(k)}\to\mathbb{P}$ e $\mathbb{P}^{(j)}\to\mathbb{P}$ não são isomorfos para $k\neq j$, de tal modo que uma única função $\mathbb{P}\to\mathbb{P}$ não poderia cumprir o papel que a função **int**:$\mathbb{P}^\omega\to\mathbb{P}$ cumpre, do modo que o cumpre.

9.3 Termos universais, seqüências infinitas

No trabalho de Turing [TUR 36], a uma função universal corresponde uma máquina universal, que simula qualquer outra máquina de Turing. Neste trabalho, tem-se termos universais, com comportamento análogo, de acordo com a discussão a seguir.

Como **int** e **curry** são Π-definíveis, existem termos fechados *INT* e *CURRY:*

- $INT:\pi^\omega\to\pi$
- $CURRY:(\pi^\omega\to\pi)\to(\pi\to\pi^\omega\to\pi)$

onde $\pi^\omega=\pi\to\pi$, tais que, para todo termo fechado $F:\pi^\omega\to\pi$, existe um numeral $N:\pi$ tal que:

· *CURRY INT N* computa a mesma função que *F*

Isto vale inclusive para *F=INT*, pelo teorema 9.2.1. Assim, *INT* é um termo universal para $(\pi\to\pi)\to\pi$, com termo de entrada *CURRY*, no sentido análogo de máquina universal. O termo *CURRY* corresponde à operação implícita que tem que ser feita na fita de uma máquina universal, de acréscimo do numeral da máquina a ser simulada. A rigor, *CURRY* transforma *INT* em um termo, especificamente (*CURRY INT*), que primeiro aceita *N* e depois simula o termo *F* de numeral *N* (vide o tipo de *CURRY*).

Como **cons** e [] são Π-definíveis, existem termos fechados *CONS* e *NIL*:

· $CONS:(\pi\to\pi^{\omega}\to\pi^{\omega})$

· $NIL:\pi^{\omega}$

tais que, para todos termos fechados $F:\pi^{(k)}\to\pi$, $X_1,\ldots,X_k:\pi$, existe um numeral $N:\pi$ para o qual:

· *CURRY INT N* (*CONS* X_1...(*CONS* X_K *NIL*)...)
computa o mesmo número parcial que $F\ X_1\ldots X_k$

Nesta situação, o termo (*CONS* X_1...(*CONS* X_K *NIL*)...) cumpre o papel análogo de configuração inicial da fita de uma máquina de Turing. Um termo $F:\pi^{\omega}\to\pi$ corresponde, então, a uma máquina com fita de entrada infinita. A situação em que entrada e saída são infinitas pode ser obtida por um termo de tipo $\pi^{\omega}\to\pi^{\omega}$.

Dada uma função $g:\mathbb{P}^{\omega}\to\mathbb{P}^{\omega}$, é possível obter uma função $h:\mathbb{P}^{\omega}\to\mathbb{P}$ equivalente, definida por:

· $h\ y=(g\ y')_i$ onde $i=y_0$

pois, para todo *x*:

· $g\ x=\lambda\mathbf{i}.h(\mathbf{cons}\ \mathbf{i}\ x)$

Assim:

· $g=\mathbf{B}(\mathbf{curry}\ h)$

onde **B**: $(\mathbb{P}\to\mathbb{P}^{\omega}\to\mathbb{P})\to(\mathbb{P}^{\omega}\to\mathbb{P}^{\omega})$ é definida por:

· **B** *f d x*=*f x d*

Então, todo termo $G:\pi^{\omega}\to\pi^{\omega}$ é equivalente ao termo:

· *B* (*CURRY* (*CURRY INT N*))

para um numeral $N:\pi$ apropriado, onde $B:(\pi\to\pi^{\omega}\to\pi)\to(\pi^{\omega}\to\pi^{\omega})$ é um termo fechado que denota a função **B**.

Assim, a linguagem Π, em princípio projetada para manipular números parciais, permite manipular, também, seqüências finitas e infinitas, mediante os operadores (′) e **cons**.

A seqüência constante **n** pode ser definida por (**K n**), onde **K**:$\mathbb{P}\to\mathbb{P}^{\omega}$ é definida por:

· **K n**=**cons n** (**K n**)

Como **K n i**=**n**, **K** corresponde ao operador de formação de função constante da lógica de combinadores (cf. seção 6.10 e [HIN 86]). A seqüência dos números a partir de *n* pode ser definida por (**from n**), onde **from**:$\mathbb{P}\to\mathbb{P}^{\omega}$ é definida por:

· **from n**=**cons n** (**from** (**n+1**))

A função **from** nada mais é do que o operador de soma **soma**:$\mathbb{P}\to\mathbb{P}\to\mathbb{P}=\lambda$**xy.x+y**, como pode ser facilmente verificado. Defina **map**: $(\mathbb{P}\to D)\to\mathbb{P}^{\omega}\to D^{\omega}$ por:

· **map** *f x*=**cons** $(f\ x_0)$ (**map** *f x*′)

de tal modo que:

· **map** *f* $[x_0,x_1,\ldots,x_{k-1}]=[f\ x_0,f\ x_1,\ldots,f\ x_{k-1}]$

O operador **map** funciona como um operador de composição. O operador **delta**:$\mathbb{P}\to\mathbb{P}^{\omega}$ tal que:

· **delta x**=[0,**x**,**x+x**,**x+x+x**,...]

pode ser definido por:

· **delta x=cons 0 (map (soma x) (delta x))**

O operador **delta** corresponde ao operador de multiplicação.

Assim como funções contínuas $\mathbb{P}^k \to \mathbb{P}$ podem ser vistas como informações sobre funções parciais $\mathbb{N}^k \to \mathbb{N}$ (cf. seção 3.3), D-seqüênciais podem ser vistas como informações sobre seqüências de elementos de D. Para uma D-seqüência x e $\mathbf{i}=\underline{0}$, x_i é uma informação comum a todos os elementos de x, devido à monotonicidade de x. Esta informação é capturável finitariamente, pela aplicação de $\underline{0}$ a x. Em geral, para $\mathbf{i}=\underline{n}$, x_i é uma informação comum a todos os elementos a partir de x_n, e para $\mathbf{i}=\infty$, x_i é a acumulação das informações x_j para $\mathbf{j}=\underline{0},\underline{1},\ldots,\underline{n},\ldots$ (cf. lema 3.2.5).

Por exemplo, quando se obtém $x_i=\mathbf{m}$, para $\mathbf{i}=\underline{0}$, se obtém que a seqüência sobre a qual x informa é a seqüência constante **m**, e que a informação x é:

· $x=\lambda\mathbf{i}.\mathbf{m}=\mathbf{K\ m}$

sem a execução de um exame infinitário de x. É claro que existem informações mais precárias sobre a seqüência constante **m**, como:

· $y=\lambda\mathbf{i}.$**if édefinido i then m else** $\underline{0}$

Para tal D-seqüência e $\mathbf{i}=\underline{0}$, tem-se que $y_i=\underline{0}$, a informação nula. Mesmo assim, para $\mathbf{i}=\underline{1}$, $y_i=\mathbf{m}$, o que estabelece a constância da seqüência, junto com o fato de que $y_0=\mathbf{m}$. Uma informação mais precária ainda sobre esta seqüência é:

· $z=\lambda\mathbf{i}.$**if fintot i then m else** $\underline{0}$

Para tal informação sobre a seqüência constante **m**, não é possível determinar finitariamente a sua constância. Isto ocorre porque, para todo $\mathbf{i}=\underline{0},\underline{1},\ldots,\underline{n},\ldots,\infty$, vale que $z_i=\underline{0}$, a informação nula.

Um dos objetivos da presente investigação é estudar métodos para a obtenção de informações de boa qualidade, na atividade da programação, sobre números, funções, seqüências, conjuntos. Pelo teorema 9.5.2, tais métodos não podem ser automáticos, i.e., Π-definíveis.

9.4 Operações efetivas com conjuntos

▷ Um *índice característico* de um conjunto A Π-definível é qualquer número de Gödel da função $\mathbb{P}$-característica de A.

▷ Um *índice característico parcial* de um conjunto Π-enumerável A é qualquer número de Gödel de uma função $\mathbb{P}$-característica parcial de A.

▷ Para cada **i**, o conjunto W_i é definido por:

· $W_i = \bigcap\{A \mid \mathbf{val}_{\pi\to\pi}\mathbf{i}$ é função $\mathbb{P}$-característica parcial de $A\}$

É imediato que:

· W_i = núcleo de $\mathbf{val}_{\pi\to\pi}\mathbf{i}$

▷ **v**:$\mathbb{P}\to\mathbb{B}$ é uma *função de teste de conjunto vazio* sss:

· **v i**=f implica W_i é vazio
  =t implica W_i é não vazio

Se **p** é a função $\mathbb{P}$-característica de um conjunto $A\subseteq\mathbb{N}$, então **exist p**=f sss A é vazio, **exist p**=t sss A é não vazio. Ainda, se **p** é funções $\mathbb{P}$-característica parcial de A, então:

· **exist p**=f implica A é vazio
  =t implica A é não vazio

Assim, o quantificador **exist** corresponde ao teste de conjunto

vazio, via funções $\mathbb{P}$-características parciais, e a melhor função de teste de conjunto vazio é **évazio** tal que:

- $\mathbf{évazio\ i}=\neg\mathbf{exist}(\mathbf{val}_{\pi\to\pi}\mathbf{i})$

Lema 9.4.1. Existem funções $\mathbf{exist}_t,\mathbf{exist}_f:(\mathbb{P}\to\mathbb{P})\to\mathbb{P}$ Π-definíveis tais que:

- $\mathbf{exist}_t$ e $\mathbf{exist}_f$ são consistentes
- $\mathbf{exist}_t \sqcup \mathbf{exist}_f=\mathbf{exist}$
- $\mathbf{exist}_t\mathbf{p}=t$ sss $\mathbf{exist\ p}=t$
- $\mathbf{exist}_f\mathbf{p}=f$ sss $\mathbf{exist\ p}=f$

Prova. Pela definição de $\exists$ (cf. seção 6.7) e pelo lema 4.4.2 e corolário, as funções definidas abaixo satisfazem a primeira e as duas últimas equações:

- $\mathbf{exist}_t\mathbf{p}=\exists\mathbf{n}.\mathbf{pn}$
- $\mathbf{exist}_f\mathbf{p}=\exists\mathbf{n}.\mathbf{p}(\underline{\mathbf{n}})==f \wedge \forall\mathbf{m}\leq\mathbf{n}.\mathbf{p}(\mathbf{m})==f$

O fato de que $\mathbf{exist}_t \sqcup \mathbf{exist}_f=\mathbf{exist}$ é conseqüência direta das demais equações. □

Lema 9.4.2. Existe $\mathbf{sup}:\mathbb{P}^{(2)}\to\mathbb{P}$ Π-definível tal que, se $\mathbf{x},\mathbf{y}$ são consistentes e $\mathbf{i},\mathbf{j}$ são números de Gödel de $\mathbf{x},\mathbf{y}$ respectivamente, então:

- $\mathbf{sup\ i\ j}=\mathbf{x}\sqcup\mathbf{y}$

Prova. A construção de **sup** é uma modificação da construção de $\mathbf{val}_\pi$ dada no teorema 8.5.1, que simula a valoração de **i** e **j** em paralelo, utilizando o lema 2.4.6 indutivamente:

- **sup** (**ng**0) (**ng** *N*)=**0**
- **sup** (**ng***M*) (**ng** 0)=**0**
- **sup** (**ng**(*SUCC M*)) (**ng** *N*))=**succ** (**sup** (**ng** *M*) (**ng** (*PREDN*))
- **sup** (**ng***M*) (**ng** (*SUCC N*))=**succ** (**sup** (**ng** (*PRED M*)) (**ng***N*))
- **sup** (**ng***M*) (**ng** *N*)=**sup** (**ng** (*passo M*)) (**ng** (*passo N*))
  se *M* ou *N* não são formas normais fracas □

Note que se o supremo não existe, o valor de **sup i j** é arbitrário, e não depende apenas de **x** e **y**, mas também dos termos escolhidos para obter os números de Gödel **i,j**. Compare a construção de **sup** com o mapeamento entre contagens de duas contagens de entrada e uma de saída, descrito a seguir. A cada vez que uma contagem de entrada produz a ação $\alpha\in\{s,z\}$, produzir α na saída, desconsiderando a eventual próxima ação α da outra contagem de entrada. Se as duas contagens são consistentes, a saída produzida é o seu supremo. Mas se elas não são, a saída resultante não depende apenas das contagens de entrada, mas também da velocidade relativa de ocorrências de ação na entrada. Por exemplo, o par de entradas (s,z) pode produzir tanto s como z na saída, de acordo com qual ação ocorre primeiro.

Teorema 9.4.3. Existe **ngexist**:$\mathbb{P}\to\mathbb{P}$ Π-definível tal que para todo número de Gödel **i** de **p**:$\mathbb{P}\to\mathbb{B}$:

- **ngexist i**=**exist**($\mathbf{val}_{\pi\to\pi}$**i**)

Prova. Sejam **m,n** números de Gödel de $\mathbf{exist}_t$,$\mathbf{exist}_f$ respectivamente. Defina **évazio** por:

- **ngexist i**=(**sup** (**comb m i**) (**comb n j**)) □

Corolário 9.4.3. **évazio** é Π-definível.

Prova. évazio i=¬(**ngexist i**) □

Utilizando **évazio** é possível definir uma série de

operações em conjuntos, via índices característicos, parciais ou não.

▷ Nas definições a seguir, a ocorrência de $\mathbf{ng}_\sigma x$ no lado esquerdo é imprópria, pois $\mathbf{ng}_\sigma x$ corre sobre todos os números de Gödel de x, e é um abuso de notação. Entende-se que $\mathbf{ng}_\sigma x$ é o número de Gödel do termo M obtido em função de definição de x, e não de x, pelos métodos do capítulo **6**. Nestas definições A, B correm sobre índices característicos de conjuntos, de modo a tornar a notação mais legível. Ainda as notações de conjunto são utilizadas para denotarem índices característicos, parciais ou não, conforme o caso, dos conjuntos de mesma notação. A primeira série de definições introduz as notações básicas:

- $n\in A=\mathbf{val}_\pi^{\pi\to\pi}(\mathbf{comb}\ A\ (\mathbf{ng}\ \alpha_0^{\pi\to\pi}))[\lambda\mathbf{x}.\mathbf{n}]$ · $n\notin A=\neg(n\in A)$
- $\{n\in\mathbb{N}\,|\,\mathcal{E}[\mathbf{n}]\}=\mathbf{ng}(\lambda\mathbf{x}.\mathcal{E}[\mathbf{x}])$

a definição mais direta abaixo de (ϵ) não foi utilizada porque não se mostrou $\mathbf{val}_{\pi\to\pi}$ Π-definível:

- $n\in A=(\mathbf{val}_{\pi\to\pi}A)\mathbf{n}$

A segunda série de definições introduz novas notações em função das básicas:

- $f[A]=\{n\in\mathbb{N}\,|\,\exists\mathbf{m}.m\in A\wedge f(\mathbf{m})==\mathbf{n}\}$
- $f^{-1}[A]=\{n\in\mathbb{N}\,|\,f(\mathbf{n})\in A\}$
- $\varnothing=\{n\in\mathbb{N}\,|\,\mathrm{f}\}$
- $\{m\}=\{n\in\mathbb{N}\,|\,\mathbf{n}==\mathbf{m}\}$
- $A\cup B=\{n\in\mathbb{N}\,|\,n\in A\ \vee\ n\in B\}$
- $A\cap B=\{n\in\mathbb{N}\,|\,n\in A\ \wedge\ n\in B\}$
- $\overline{A}=\{n\in\mathbb{N}\,|\,n\notin A\}$
- $A-B=A\cap\overline{B}$
- $A\subseteq B=\mathbf{évazio}(A-B)$
- $\{n_1,\ldots,n_k\}=\{n_1\}\cup\ldots\cup\{n_k\}$
- $|\mathrm{A}|=\mathbf{if}\ \mathbf{évazio}\ A\ \mathbf{then}\ \mathbf{0}$
  $\mathbf{else}\ (\mathbf{if}\ \mathbf{0}\in A\ \mathbf{then}\ \mathbf{1}\ \mathbf{else}\ \mathbf{0})+|\mathbf{succ}^{-1}[A]|$

É implícito que nas notações $f[A]$ e $f^{-1}[A]$ se passa como argumento um número de Gödel de f, e não f. É por esta razão que a definição de $|A|$ é uma definição recursiva geral correta.

Note que $\mathbf{succ}^{-1}[A] \neq \mathbf{pred}[A]$, pois:

- $\mathbf{succ}^{-1}[\{0\}]=\varnothing$
- $\mathbf{pred}[\{0\}]=\{0\}$

de tal modo que, se A é índice característico de um conjunto finito com maior elemento k, então:

- $\mathbf{succ}^{-(k+1)}[A]=\varnothing$

e para todo elemento n do conjunto:

- $n \in A$ sss $\mathbf{0} \in (\mathbf{succ}^{-n}[A])$

como pode ser facilmente provado por indução na cardinalidade do conjunto. Isto estabelece o teorema 9.4.4 abaixo.

Uma definição mais óbvia para $|A|$ seria:

- $\mu A = \mu \mathbf{n}.\mathbf{n} \in A$
- $|A| = \mathbf{if\ évazio}\ A\ \mathbf{then\ 0\ else\ 1+}|A-\{\mu A\}|$

mas tal definição não provê a melhor informação possível para funções características parciais.

Teorema 9.4.4. Para todos $n_1,\ldots,n_k \in \mathbb{N}$, $\{n_1,\ldots,n_k\}$ é um índice característico do conjunto de mesma notação, e:

- $|\{n_1,\ldots,n_k\}| = \mathbf{k}$

Ainda, W_i é infinito sss:

- $|\mathbf{i}| = \infty$

e W_i é finito de cardinalidade k sss:

- $|\mathbf{i}| \in \{\underline{\mathbf{k}}, \mathbf{k}\}$

onde o valor $\underline{\mathbf{k}}$ ocorre somente se i é um índice característico

propriamente parcial.

Prova. A primeira parte é estabelecida pelo lema 4.3.11 e pelas discussões que sucedem a definição de $|A|$. A prova dos demais casos é omitida. □

Note que o valor **k** para o último caso seria inadequado se **i** fosse índice característico propriamente parcial, pois neste caso **i** seria índice característico de infinitos conjuntos, que são os conjuntos que incluem W_i. A única informação compatível com todos estes conjuntos é que eles têm no mínimo k elementos.

Este teorema deve ser comparado com o teorema 5.6.**XV**(b) em [ROG 67], cujo enunciado é transcrito a seguir:

- "*There is no partial recursive ψ such that for all x, if φ_x is a characteristic function for a finite set A, then $\psi(x)$ is the size of A.*"

Pelo teorema 9.4.4 acima, se **i** é um índice da extensão natural de φ_x, então $|\mathbf{i}|$=cardinalidade de A. Logo:

- Existe ψ Π-definível tal que para todo **x**, se $\mathbf{val}_{\pi\to\pi}\mathbf{x}$ é uma função $\mathbb{P}$-característica de um conjunto finito A, então $\psi(\mathbf{x})=\mathbf{n}$ sss n é a cardinalidade de A.

Em compensação, a obtenção de extensões naturais não é efetiva, de acordo com o teorema 9.5.2.

9.5 Efetividade da obtenção de extensões naturais

Chame de p a restrição para $\mathbb{N}\to\mathbb{N}$ da função **passo**:$\mathbb{P}\to\mathbb{P}$ definida no teorema 8.4.5. Embora seja trabalhoso, não é difícil argumentar que, para uma numeração Π-efetiva apropriada (por exemplo, a dada no teorema 8.1.1), p é

recursiva primitiva. A argumentação é padrão (ver Kleene 52]), e é omitida. Para um dado código de programa **i**, considere o conjunto:

· $A_i=\{n \mid p^n(i)$ codifica uma forma normal fraca$\}$

É possível, também, mostrar que existe $f:\mathbb{N}^2\to\mathbb{N}$ recursiva primitiva tal que $f(i,n)=1$ sss $n\in A_i$, $f(i,n)=0$ sss $n\notin A_i$. O conjunto A_i é vazio sss o programa codificado codificado por **i** não pára fracamente. A currificação $\mathbf{f}:\mathbb{P}^{(2)}\to\mathbb{P}$ da extensão natural de f não pode ser Π-definível, pois **h** definida abaixo resolveria o problema da parada fraca:

· **h i=évazio(f i)**

Teorema 9.5.1. Existem funções recursivas primitivas cuja extensão natural não é Π-definível.

Prova. A discussão que precede o teorema o estabelece. □

Pela discussão acima, qualquer sistema formal que permitisse computar a extensão natural de cada função recursiva primitiva poderia ser utilizado efetivamente para resolver o problema da parada fraca. Assim, o teorema acima não é um defeito da linguagem Π, mas uma limitação intrínseca do tratamento formal de extensões naturais.

Para cada i, a função g_i definida por:

· $g_i(n)=f(i,n)$

é primitiva recursiva e satisfaz:

· $g_i(n)=0$ para todo n

ou existe m tal que

· $g_i(n)=0$ para $n<m$
$=1$ para $n\geq m$

Claramente, a extensão natural de g_i é Π-definível. Considere uma numeração φ_n das funções parciais recursivas.

Teorema 9.5.2. Não existe **f**:$\mathbb{P}\to\mathbb{P}$ Π-definível tal que se φ_n tem extensão natural Π-definível, então **f**(**n**) é um código da extensão natural de φ_n.

Prova. De novo por redução ao problema da parada fraca, em função dos comentários que precedem o teorema. □

10 CONCLUSÕES E PROBLEMAS EM ABERTO

A seção 10.1 tira conclusões do trabalho desenvolvido. A seção 10.2 expõe problemas a serem resolvidos.

Este capítulo é mais longo que o usual, e deve ser considerado como parte integrante do trabalho. O primeiro parágrafo de cada seção a resume.

10.1 Conclusões

Os números parciais permitem o desenvolvimento de uma teoria da computação com as seguintes características, com relação à teoria da recursão:

- Expressões *não denotativas* são eliminadas.
- O domínio de computação é naturalmente *fechado* para as operações de computação.
- Algoritmos tomam e produzem *informações* sobre dados, e não *dados* em si.
- Computações que *não terminam* são *significativas*.
- A noção de *parada* é desvinculada da noção de *produção de resultados*, via as noções de informação *nula*, *incompleta* e *infinita*.
- A inspeção *completa* de certos objetos *infinitos* é realizável mediante processos *finitos*.
- Computações *interativas* são modeláveis por *funções*.

Estas afirmações são elaboradas a seguir. Uma conseqüência destas características é, de novo com relação a teoria da

recursão:

- O escopo de tal teoria para o estudo de computadores reais é maior.

Expressões não denotativas são eliminadas (cf. [CON 85]). Funções parciais $A \to B$ são definidas como relações $R \subseteq A \times B$ tais que $(a,b) \in R$ e $(a,b') \in R$ implica $b=b'$ (cf. [ROG 67]). Equivalentemente, f é uma função parcial $A \to B$ sss f é uma função $A' \to B$ para algum $A' \subseteq A$. Tal A' é o domínio de definição f. A notação $f(x)$, para uma função parcial $A \to B$, faz sentido (i.e., tem denotação) sss x está no domínio de definição de f. Assim, por exemplo, a expressão "$f(x)$ é indefinido" não diz respeito a $f(x)$, que não existe, mas significa que x não está no domínio de definição de f. Como os domínios das funções parciais recursivas não são recursivos (são apenas recursivamente enumeráveis) não é possível decidir recursivamente se uma expressão tem ou não tem denotação.

Pode se dizer que, com a introdução de números parciais, as expressões não denotativas são substituídas por expressões que denotam um dos números parciais $\underline{0},\underline{1},\ldots,\underline{n},\ldots,\infty$, e que expressões que denotam n são substituídas por expressões que denotam o número parcial $\mathbf{n}$.

O domínio de computação é naturalmente fechado para as operações de computação. Por uma *operação de computação* se entende uma regra de computação. A operação básica de busca ilimitada que computa a minimização de uma função, da teoria da recursão, não produz resultados para certos argumentos. Assim, os naturais não são fechados para as operações de computação, no mesmo sentido que os racionais não são fechados para a operação de supremo. Por exemplo, o conjunto $\{x \in \mathbb{Q} \mid x^2<2\}$ não tem supremo em $\mathbb{Q}$, mas tem supremo $\sqrt{2}$ em $\mathbb{R}$, de tal modo que $\mathbb{R}$ é um fecho de $\mathbb{Q}$ para supremos.

O simples acréscimo de $\perp$ a $\mathbb{N}$ para realizar o fecho para operações de computação é uma solução artificial. Suponha que fosse acrescentado um elemento $\perp$ a $\mathbb{Q}$, com a propriedade $x<\perp$ para todo $x\in\mathbb{Q}$. Deste modo, o supremo em $\mathbb{Q}\cup\{\perp\}$ de um subconjunto de $\mathbb{Q}$ sem supremo em $\mathbb{Q}$ seria $\perp$. Então, $\mathbb{Q}\cup\{\perp\}$ também é um fecho de $\mathbb{Q}$ para supremos.

As operações de computação básicas da teoria da recursão são aquelas utilizadas para computar funções definidas por recursão primitiva e por minimização. A operação que leva para fora de $\mathbb{N}$ é a operação de busca ilimitada, utilizada para computar minimizações. Na seção 6.6 foi definido um operador de minimização μ que, para predicados P dados por funções $\mathbb{P}$-características $\mathbf{p}:\mathbb{P}\to\mathbb{B}$, o resultado da busca de um $x\in\mathbb{N}$ tal que $P(x)$ é expressa pelo valor $(\mu\mathbf{x}\leq\infty.\mathbf{px})$. Esta busca falha quando não existe x tal que $P(x)$. Neste caso, $(\mu\mathbf{x}\leq\infty.\mathbf{px})$ vale ∞. Conforme argumentado a seguir, este valor é natural sob diversos aspectos.

A informação nula não é adequada como valor, pois ela informa que $x\geq 0$, mas x não existe.

O número ω é caracterizado como o único número $x\in\mathbb{N}\cup\{\omega\}$ tal que $\forall n\in\mathbb{N}.n\leq x$. Em contrapartida, $\infty=\bigsqcup\{\underline{\mathbf{n}}\mid n\in\mathbb{N}\}$, lembrando que $\underline{\mathbf{n}}$ é lido "no mínimo n". Assim, a resposta ∞ é interpretada como a resposta "aquele $x\in\mathbb{N}$ tal que $\forall n\in\mathbb{N}.n\leq x$", i.e., "nenhum natural". Por exemplo, como foi mencionado na seção 6.6, a expressão $(\mu\mathbf{x}<\infty.\neg\mathbf{x}==\mathbf{x})==\mathbf{n}$ vale f (falso) para todo $n\in\mathbb{N}$.

Por último, assim como a minimização $\mu\mathbf{x}<\mathbf{k}.\mathbf{px}$ produz $\mathbf{k}$ para $\mathbf{k}$ finito, para indicar falha na busca, ela também produz $\mathbf{k}$ para $\mathbf{k}=\infty$, com o mesmo objetivo.

A operação de minimização não mostra explicitamente como os números parciais $\underline{0},\underline{1},\ldots,\underline{n},\ldots$ surgem. Ainda, esta não é uma operação de computação primitiva da linguagem Π, que é o sistema formal utilizado neste trabalho para expressar computações de naturais parciais. As operações de computação básicas para Π são a regra de substituição, as regras de redução das constantes, e, como caso particular, a regra de recursão geral. A seguir a naturalidade do fechamento de $\mathbb{N}$ para as operações de computação, que resulta no domínio de naturais parciais $\mathbb{P}$, é analisada sob este ponto de vista.

Considere a seguinte interpretação não padrão de Π (cf. seções 5.1.2 e 5.2.1):

- $\mathcal{C}_{\pi}=\mathbb{N}$
- $\mathcal{C}_{\sigma\to\tau}$=funções parciais $\mathcal{C}_{\sigma}\to\mathcal{C}_{\tau}$
- *SUCC*, *PRED*, *COND*, *Y* são respectivamente as funções sucessor, predecessor, *cond*, e algum operador de ponto fixo.

A função *cond* é definida na proposição 4.1.1. Os conjuntos de funções parciais, ordenados pela relação de inclusão de grafo, são domínios de Scott. Com esta interpretação, deliberadamente alguns termos não tem denotação, e a função semântica é parcial.

As relações de redução α, β e η valem para qualquer interpretação. Mas as relações de redução para as constantes são dependentes da interpretação. Para todos $x,y,z\in\mathbb{N}$, vale que:

- $(x+1)-1=x$
- $cond(0,y,z)=z$ · $cond(x+1,y,z)=y$
- $cond(x,0,0)=0$ · $cond(x,y+1,z+1)=cond(x,y,z)+1$

de tal modo que as regras de redução de Π para constantes (cf. seção 5.2.2) parecem coerentes com a interpretação não

padrão dada acima. Mas este não é caso. Por exemplo, se $P=(Y\ SUCC)$, P não tem denotação, pois a função sucessor dos naturais não tem pontos fixos. Logo, $(COND\ P\ 0\ P)$ não tem denotação. Mas:

$$\begin{aligned} \cdot\ (COND\ P\ 0\ P) &= (COND\ (Y\ SUCC)\ 0\ P) \\ &\Leftrightarrow (COND\ (SUCC(Y\ SUCC))\ 0\ P) \\ &\Leftrightarrow (COND\ (SUCC\ P)\ 0\ P) \\ &\Leftrightarrow 0 \end{aligned}$$

As regras de Π não são coerentes com esta interpretação, porque, apesar de preservarem denotação, não preservam ausência de denotação, como o exemplo mostra.

Estas regras funcionam se é acrescentada a ressalva de que se aplicam somente a termos com denotação. Isto implica na necessidade de avaliar totalmente os termos M,N,P antes de aplicar as regras (chamada por valor), obtendo os numerais equivalentes. Se tais numerais não existem, porque um dos termos não tem denotação, se incorre numa computação infinita, e a aplicação de tais regras é postergada indefinidamente.

A causa real do problema é a operação de recursão geral, que introduz termos sem denotação, e não as demais regras de redução de Π, que preservam denotação.

O completamento de $\mathbb{N}$ pelo acréscimo de $\underline{0},\underline{1},\ldots,\underline{n},\ldots,\infty$ pode ser caracterizado como o completamento mínimo tal que as regras de redução das constantes *SUCC*, *PRED*, *COND* continuam preservando denotação, com relação a interpretação não padrão acima, e a recursão geral deixa de introduzir termos sem denotação. O modo padrão de extender funções em $\mathbb{N}$ para o domínio que resulta de tal completamento é denominado extensão natural, e é definido na seção 3.4.

O completamento de $\mathbb{N}$ pelo acréscimo de um elemento

$\perp$, tal que os termos que não tem denotação passam a ter denotação $\perp$, não tem a propriedade do completamento anterior, pois, no raciocínio acima, *P* denotaria $\perp$ e (*COND P* 0 *P*) denotaria $cond(\perp,0,\perp)=\perp\neq 0$, para qualquer extensão monotônica de *cond* para o domínio $\mathbb{N}\cup\{\perp\}$, onde se entende que $x \leq y$ sss $x=y$ ou $x=\perp$, como é usual.

As operações utilizadas para construir as funções parciais recursivas podem ser classificadas em duas espécies: as que garantidamente produzem funções totais quando aplicadas a funções totais, e as que não necessariamente produzem funções totais quando aplicadas a funções totais. O operador de recursão primitiva é da primeira espécie, e o operador de minimização é da segunda espécie. Do ponto de vista operacional, os operadores de primeira espécie correspondem a processos computacionais que realizam uma tarefa um número de vezes que é conhecido antes que a computação comece, e os da segunda espécie correspondem a processos que realizam uma tarefa um número de vezes que vem a ser conhecido apenas depois que a computação termine, caso seja finito. As funções parciais surgem porque este número pode ser infinito. O conjunto dos naturais é fechado para funções produzidas por operadores da primeira espécie, mas não é fechado para funções produzidas por operadores da segunda espécie.

Via o uso de números parciais, todos os operadores podem ser colocados em uma única espécie, por parametrização do número máximo de vezes que uma tarefa pode ser realizada. Quando este limite é dado por um número parcial finito, se obtém os operadores da primeira espécie, e quando ele é dado pelo número parcial ∞, se obtém os operadores da segunda espécie.

Por exemplo, como foi mostrado na seção 6.6, o

operador de minimização ilimitada para números parciais é um caso particular do operador de minimização limitada, quando o limite é ω.

No domínio de naturais parciais, recursão geral é um caso particular de recursão primitiva (vide [GIR 89]). Seja $f:\mathbb{P}\to D$ contínua definida a partir de $b\in D$ e $g:\mathbb{P}\to D\to D$ por:

- $f(\underline{\mathbf{0}})=\perp$
- $f(\mathbf{0})=b$
- $f(\mathbf{x+1})=g\ \mathbf{x}\ (f\ \mathbf{x})$

Suponha que g é definida para uma dada $h:D\to D$ por:

- $g\ \mathbf{x}\ y=h\ y$

de tal modo que:

- $f(\mathbf{x+1})=h(f\ \mathbf{x})$

Como $\omega=\omega+\mathbf{1}$, tem-se que $f(\omega)$ é um ponto fixo de h:

- $f(\omega)=h(f(\omega))$

Ainda, este é o menor ponto fixo de h, pois:

- $f(\underline{\mathbf{n}})=h^n(\perp)$

e, pela continuidade de f e pelo lema 3.2.5:

- $f(\omega)=\bigsqcup\{f(\underline{\mathbf{n}})\mid n\in\mathbb{N}\}$
  $=\bigsqcup\{h^n(\perp)\mid n\in\mathbb{N}\}$
  $=\mathbf{fix}_D h$

Se a dependência entre f e b,g é escrita:

- $f=\mathbf{primrec}_D b\ g$

então:

- $\mathbf{fix}_D=\lambda h.(\mathbf{primrec}_D\perp\ (\lambda xy.h\ y)\ \omega)$

Deste modo, é possível eliminar as constantes Y_σ e acrescentar constantes $INF:\pi$, que denota ω, e $PRIMREC_\sigma$, que

denotam $\mathbf{primrec}_{\mathcal{C}_\sigma}$, com regras de redução:

- $INF \Rightarrow SUCC\ INF$
- $PRIMREC\ B\ G\ 0 \Rightarrow B$
- $PRIMREC\ B\ G\ (SUCC\ N) \Rightarrow G\ N\ (PRIMREC\ B\ G\ N)$

A constante *PRED* também é eliminável, pois o termo abaixo denota a função predecessor:

- $PRIMREC_\pi 0(\lambda\alpha_0^\pi.\lambda\alpha_1^\pi.\lambda\alpha_0^\pi)$

Ainda, se o operador de recursão primitiva fosse definido de tal modo que $f=\mathbf{primrec}'_D(b,g)$ sss:

- $f\ \underline{\mathbf{0}}=b \sqcap \mathbf{fix}\ \lambda x.(g \circ x)$
- $f\ \mathbf{0}=b$
- $f\ (\mathbf{x+1})=g\ \mathbf{x}\ (f\ \mathbf{x})$

então a constante *COND* também seria eliminável, pois:

- $\mathbf{cond(x,y,z)=primrec'z}\ (\lambda ab.\mathbf{y})\ \mathbf{x}$

O valor $b \sqcap \mathbf{fix}\ \lambda x.(g \circ x)$ é o mais definido que $\mathbf{f}(\underline{\mathbf{0}})$ assumir sem que f deixe de ser monotônica. Com esta alteração, o valor de $f(\infty)$ permanece o mesmo, como pode ser provado sem dificuldade. As constantes *PRIMREC'* que denotam **primrec'** tem regras de redução complicadas, que englobam as regras de redução de *PRED* e *COND*.

Assim, as constantes abaixo são suficientes para expressar todas as funções Π-definíveis:

- $0:\pi$
- $SUCC:\pi\to\pi$
- $PRIMREC'_\sigma:\sigma\to(\pi\to\sigma\to\sigma)\to\pi\to\sigma$
- $INF:\pi$

Pode ser provado que todo termo fechado de tipo π que não tem como subtermo a constante *INF* denota e computa um natural finito total. Assim, é somente pelo acréscimo de ∞ que surgem os demais números parciais $\underline{\mathbf{0}},\underline{\mathbf{1}},\ldots,\underline{\mathbf{n}},\ldots$ como resultados de computações, e que surgem funções que tem valor não finito

total para argumentos finitos totais.

A linguagem Π′ que resulta das constantes $0, SUCC, PRIMREC'_{\sigma}$ e *INF* é uma generalização do sistema T de Gödel definido em [GIR 89], e tem mais poder de expressão que o sistema T, mesmo sem a constante *INF*. Por exemplo, um termo *G* tal que:

$$\cdot\ G\ T\ X \Rightarrow T \qquad \cdot\ G\ X\ T \Rightarrow T \qquad \cdot\ G\ F\ F \Rightarrow F$$

onde *T* e *F* são termos que representam respectivamente t e f, não é definível no sistema T, mas é definível em Π′, sem o uso de *INF*:

$$\cdot\ G = \lambda\alpha_0 . \lambda\alpha_1 . COND\ \alpha_0\ T\alpha_1$$

Algoritmos tomam e produzem informações sobre dados, e não dados em si. Na teoria da recursão, algoritmos produzem dados. Deste modo, quando um algoritmo não pode produzir o dado, não produz nada. Via números parciais, um algoritmo que não pode produzir informação finita e completa (a contrapartida de dado) produz informação incompleta (como caso limite, a informação nula) ou infinita e completa, por aproximação sucessiva.

O número ω não pode ser produzido por um algoritmo da teoria da recursão, pois se exige que algoritmos produzam resultados em um número finito de passos. O número ∞ (a informação "exatamente ω") pode ser produzido por um algoritmo que manipula informações. O número ω é caracterizado como o único $x \in \mathbb{N} \cup \{\omega\}$ tal que $n \leq x$ para todo $n \in \mathbb{N}$, como foi mencionado acima. Um algoritmo que produz a seqüência infinita de informações $\underline{0}, \underline{1}, \ldots, \underline{n}, \ldots$, computando ∞, produz todas as informações a respeito de ω. Para se poder ter ∞, então, é indispensável ter os números propriamente parciais $\underline{0}, \underline{1}, \ldots, \underline{n}, \ldots$.

Computações que não terminam são significativas. Por terminação se entende, na teoria da recursão, finitude. Computações infinitas, de algoritmos que manipulam dados, são não significativas, no sentido que não produzem resultados. Pela mesma razão que não é possível excluir as expressões não denotativas por um processo efetivo, não é possível determinar se um algoritmo tem computação infinita para uma certa entrada. Computações infinitas de algoritmos que manipulam informação são sempre significativas.

Nos casos em que algoritmos que manipulam dados não produzem resultado, os algoritmos que manipulam informação produzem informação incompleta ou infinita. Como caso limite, a informação nula é produzida. O acréscimo da informação nula é analogo ao acréscimo da quantidade nula 0 aos números positivos.

A noção de parada é desvinculada da noção de produção de resultados, via as noções de informação nula, incompleta e infinita. É supreendente que processos que procedem por passos produzam um resultado apenas no último passo, como os processos descritos por algoritmos da teoria da recursão. Se um processo procede por passos, é porque cada passo realiza uma parte da tarefa completa. É natural que o resultado seja parcialmente visível antes que a execução do processo termine, e, como caso particular, quando ainda há infinitos passos a realizar. Assim, números propriamente parciais fazem sentido mesmo em computações de números finitos totais. Neste caso, o seu papel é informar o que foi determinado a respeito do resultado, antes que o processo termine.

A inspeção completa de certos objetos infinitos é realizável mediante processos finitos. Isto foi exemplificado

nas seções 9.2, 4.3, 9.4. Estes objetos infinitos são finitos do ponto de vista da teoria dos domínios (cf. seções 2.4 e 3.2).

Computações interativas são modeláveis por funções. Este fato é bem conhecido pela comunidade de programação funcional [TUD 85]. A vantagem, em tal caso, é que se mostra que as linguagens de programação funcionais puras são suficientes para programar computadores. [TUD 87] discute como programar um sistema operacional completo em linguagem funcional. A vantagem, do ponto de vista de uma teoria da computação, é que se tem um modelo matemático simples de processos interativos e concorrentes [KAH 74] [KAH 77].

Os objetos parciais servem para permitir modos de interação mais complicados que o modo de interação de uma máquina de Turing. A interação de uma máquina de Turing consiste em primeiro receber um dado, depois processá-lo, e depois entregar o resultado. Certamente este é o modo mais simples possível de interação. Um processo interativo tem uma componente de *troca* de informação e uma componente de *transformação* de informação (cálculo). As máquinas de Turing possuem o modo de troca mais simples possível porque foram pensadas para estudar o processo de cálculo, e não para estudar processo mais geral de interação, que está fora do escopo da lógica, dentro da qual o trabalho de Turing se insere. Como é sabido, a teoria da computação teve origem na lógica formal, muito antes da existência dos computadores.

As linguagens de programação tem o mesmo poder de expressão que as máquinas de Turing, com relação a *computabilidade*. Mas, certamente, muito mais poder de expressão que as máquinas de Turing no que diz respeito a *interatibilidade*, por assim dizer. Na prática da Ciência da Computação, na maioria da vezes, a componente mais

complicada, e que exige mais esforço do programador, é a componente de troca. É fundamental que uma teoria da computação tenha, ou permita expressar de modo direto, uma noção de computação interativa.

Modelos tradicionais como CCS [MIL 80], CSP [HOA 85] dão ênfase aos aspectos *operacionais* de processos concorrentes, embora, é claro, tenham semântica. Este trabalho mostra que processos interativos *podem* ser modelados por funções, mas não desenvolve sistematicamente uma teoria matemática de processos interativos baseada nesta modelagem, e nem aponta como isto poderia ser feito.

10.2 Problemas em aberto

Os problemas importantes que ainda não foram investigados ou que o autor não conseguiu resolver estão nas seguintes categorias:

- Expressibilidade da linguagem Π.
- Determinação da existência de funções universais para todos os tipos.
- Complexidade de funções Π-definíveis.
- Matematização e formalização da noção de interação.
- Programação procedural com números parciais.

Os problemas que se encontram dentro das categorias são abordados a seguir.

Expressibilidade da linguagem Π. São Π-definíveis todos os elementos de $\mathfrak{C}$ que deveriam ser tidos como computáveis, no sentido intuitivo e irrestrito de computabilidade? Esta pergunta pode ser reduzida à tese de Church utilizando os métodos utilizados em [PLO 77].

Os conceitos de conjunto recursivo e conjunto Π-definível coincidem? Os conceitos de conjunto recursivamente enumerável e conjunto Π-enumerável coincidem? Vide captítulo **6**. O autor acredita que sim, embora não tenha tentado uma prova de tal proposição importante. Um problema equivalente a este seria o de determinar se a restrição de uma função Π-definível $\mathbb{P}^k \to \mathbb{P}$ para uma função parcial $\mathbb{N}^k \to \mathbb{N}$ é recursiva.

O objetivo de uma teoria da computação baseada em números parciais não é alterar a noção de computabilidade, mas aproximar a teoria da computação da Ciência da computação, como foi salientado na introdução, aumentando seu escopo na tarefa de programação de computadores. Como foi dito na seção anterior, os computadores, além de computarem, interagem, e esta capacidade de interagir é tão importante quanto a capacidade de calcular. Assim, dentro do questionamento sobre a expressibilidade de Π, cabe perguntar: quais são os modos de interação expressíveis em Π?

Utilizando o conceito de determinismo exposto logo adiante, quando é discutida a matematização e formalização da noção de interação, é possível dizer que Π permite expressar uma parte dos processos de interação determinísticos. Será que Π pode expressar todos os processos de interação determinísticos? Via números de Gödel, é possível simular certos processos não determinísticos (cf. comentário após lema 9.4.2).

Outro problema em aberto é a possibilidade de um análogo da tese de Church para processos interativos [BOU 88]. A resposta para estes problemas depende de uma matematização destas noções, que por sua vez requer uma análise das noções concretas de processo interativo.

Os problemas em aberto na categoria a seguir também recaem na categoria de Π-definibilidade.

Determinação da existência de funções universais para todos os tipos. Existe uma função universal para cada tipo de Π? Esta questão é respondida, pelo uso do lema 9.1.1, se o seguinte problema é resolvido. Existe uma função de valoração Π-definível $\mathbf{val}_{\tau}^{\sigma}$, para cada σ e τ?

A função **exist**:$(\mathbb{P}\to\mathbb{P})\to\mathbb{P}$ definida na seção 4.4 é definível via números de Gödel, como mostrado na seção 9.4. É um problema que o autor não conseguiu resolver se **exist** é Π-definível ou não (cf. [PLO 77]. Se **exist** não é Π-definível, então isto pode ser consertado pelo acréscimo de uma constante $EXIST:(\pi\to\pi)\to\pi$. Existem regras de redução efetivas para esta constante, uma vez que **e** é Π-definível.

Complexidade de funções Π-definíveis. Como seria uma teoria de complexidade para funções Π-definíveis? Tal teoria deve ser mais forte que a teoria para funções recursivas, pois não é suficiente estudar quantos passos são necessários para computar $f(x)$, que inclusive podem ser infinitos em quantidade. É preciso, também, estudar quantos passos são necessários para computar cada parte finita de $f(x)$.

Por exemplo, pode ser que a computação de $f(x)$, supondo que $f(x)=\mathbf{3}$, produza em poucos passos a informação $\underline{\mathbf{1}}$, a seguir em muitos passos a informação $\underline{\mathbf{2}}$, e depois em poucos passos informação **3**. Se a informação $\underline{\mathbf{1}}$ basta para calcular $g(f(x))$, porque $g(\underline{\mathbf{1}})=g(f(x))$, então a complexidade do cálculo completo de $g(f(x))$ não depende da complexidade do cálculo completo de $f(x)$. O que se tem é que a complexidade da computação de cada parte de $g(f(x))$ depende da complexidade da computação de cada parte de $f(x)$.

Matematização e formalização da noção de interação. Em processos interativos triviais, como aqueles que podem ser executados por máquinas de Turing, o rumo da computação depende apenas da entrada. Quando o processo de troca envolvido na interação é mais complicado, o rumo da computação pode depender não apenas da informação trocada, mas também do modo particular que a troca é feita. Quando as computações são estudadas em função de suas entradas possíveis, desconsiderando-se a influência dos distintos modos de fornecimento que uma mesma entrada pode ter, surge não determinismo. Os objetos computados por tais computações, com domínio de entrada D e domínio de saída E, podem ser tidos como funções $f:D\rightarrow\mathcal{P}(E)$, onde $y\in f(x)$ significa que y é uma saída possível para a entrada x. São os domínios potência [PLO 76] [SMY 78] adequados para modelar tais computações?

Programação procedural com números parciais. Como seria uma linguagem de programação procedural que manipulasse números parciais?

BIBLIOGRAFIA

[ACI 91] ACIÓLY, B.M. *Fundamentação computacional da matemática intervalar*. Porto Alegre: CPGCC da UFRGS, 1991. 253p. Tese de doutorado.

[BIR 88] BIRD, R.; WADLER, P. *Introduction to functional programming*. New York: Prentice-Hall, 1988. 430p.

[BOU 88] BOUDOL, G.; CASTELLANI, I. Concurrency and atomicity. *Theoretical Computer Science*, Amsterdam: Elsevier, p.25-84, 1988.

[COL 91] COLSON, L. About primitive recursive algorithms. *Theoretical Computer Science*, Amsterdam: Elsevier, v.83, n.1, p.57-69, 1991.

[CON 85] CONSTABLE, R. Partial functions and constructive formal theories. In: THEORETICAL COMPUTER SCIENCE GI-CONFERENCE, 6., 1985, Berlin. *Proceedings...* Berlin: Springer-Verlag, 1985. p.1-18. (Lecture Notes in Computer Science 145).

[COS 90] COSTA, A.C.R. *Teoria dos tipos*. Porto Alegre: CPGCC da UFRGS, 1990. 98p.

[COS 91] COSTA, A.C.R. *Continuous predicates and logical reflexivity*. Porto Alegre: CPGCC da UFRGS, 1991. 22p. (Relatório de pesquisa, n.156)

[COS 91a] COSTA, A.C.R. *Efetividade abstrata e efetividade concreta*. Porto Alegre: CPGCC da UFRGS, 1991.

11p.

[COS 91*b*] COSTA, A.C.R. *A fundamentação da inteligência artificial.* Porto Alegre: CPGCC da UFRGS, 1991. 20p. Proposta de tese de doutorado.

[DAV 58] DAVIS, M. *Computability and Unsolvability.* New York: McGraw-Hill, 1958. 230p.

[ESC 91] ESCARDÓ, M.H. *Uma teoria da computação com não terminação significativa: aspectos externos dos computadores*. Porto Alegre: CPGCC da UFRGS, 1991. 43p. (Trabalho individual, n.227).

[ESC 91*b*] ESCARDÓ, M.H. *Agentes, objetos, interação e comportamento*. Porto Alegre: CPGCC da UFRGS, 1991. Trabalho da disciplina "Sistemas Teleonômicos". 12p.

[GIR 89] GIRARD, J.Y. et al. *Proofs and types*. Cambridge: Cambridge University Press, 1989. 176p.

[HIN 86] HINDLEY, J.R.; SELDIN, J.P. *Introduction to Combinators and λ-calculus*. Cambrigde: Cambridge University Press, 1986.

[HOA 85] HOARE, C.A.R. *Communicating Sequential Processes.* Englewood Cliffs: Prentice-Hall, 1985.

[KAH 74] KAHN, G. The semantics of a simple language for parallel processing. In: IFIP CONGRESS 1974, Amsterdam. *Proceedings...* Amsterdam: North-Holland, 1974. p.471-475. (Information Processing 74, v.6)

[KAH 77] Kahn, G.; MacQueen, D. Coroutines and networks of parallel processes. In: IFIP CONGRESS 1977, Amsterdam. *Proceedings...* Amsterdam: North-Holland 1977. 998p. p.994-998. (Information Processing 77, v.7)

[KLE 52] KLEENE, S.C. *Introduction to Metamathematics.* Amsterdam: North-Holland, 1952. 550p.

[MAC 78] MACHTEY, M.; YOUNG, P. *An Introduction to the General Theory of Algorithms.* New-York: North-Holland, 1978. 264p.

[MIL 80] MILNER, R. *A calculus of communicating systems.* Berlin: Springer-Verlag, 1980. (Lecture Notes in Computer Science 92).

[PAU 87] PAULSON, L.C. *Logic and Computation, Interative Proof with Cambridge LCF.* Cambridge: Cambridge University Press, 1987. 302p.

[PLO 76] PLOTKIN, G. A power domain construction. *SIAM Journal of Computing* v.5, p.452-486, 1976.

[PLO 77] PLOTKIN, G. LCF considered as a programming language. *Theoretical Computer Science,* Amsterdam: Elsevier, v.5, n.1, p.223-255, Mar. 1977.

[PLO 80] PLOTKIN, G. *Post-Graduate Lecture Notes in Advanced Domain Theory.* Edinburgh: Department of Computer Science/University of Edinburgh, 1980. 297p.

[ROG 67] ROGERS, H. *Theory of Recursive Functions and Effective Computability.* New York: McGraw-Hill,

1967. 482p.

[SCO 76] SCOTT, D.S. Data Types as Lattices. *SIAM Journal of Computing*, v.5, n.3., p.522-587, 1976.

[SCO 82] SCOTT, D.S. Domains for Denotational Semantics. In: COLLOQUIUM ON AUTOMATA, LANGUAGES AND PROGRAMMING, 9., July 12-16, 1982, Aarthus. *Proceedings...* Berlin: Springer-Verlag, 1982. 613p. p.231-252. (Lecture Notes in Computer Science 140)

[SCO 82*a*] SCOTT, D.S. Lectures on a Mathematical Theory of Computation. In: THEORETICAL FUNDATIONS OF PROGRAMMING METHODOLOGY. Berlin: Springer-Verlag, 1982. p.145-292.

[SCO 89] SCOTT, D.S. Six Lectures on Domains & Logic. International Summer School on Logic, Algebra & computation. Marktobedorf, BRD, July 25 - August 6, 1989. Transparências.

[SMY 78] SMYTH, M.B. Power domains. *Journal of Computer and System Sciences*, v.16, p.23-26, 1978.

[SMY 82] SMYTH , M.B. & PLOTKIN, G.D. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, v.11, n.4, 1982.

[TUR 36] TURING, A.M. On computable numbers, with an application to the Entscheidungsproblem, In: The London Mathematical Society, ser. 2, 1936, London. *Proceedings...* London: [s.n.], 1936. v.42 p.230-265.

[TUD 85] TURNER, D.A. Miranda: A non-strict functional programming language with polymorphic types, In: FUNCTIONAL PROGRAMMMING LANGUAGES AND COMPUTER ARCHITECTURE, 1985, Berlin. *Proceedings...* Berlin: Springer-Verlag, 1985. p. 1-16. (Lecture Notes in Computer Science 201)

[TUD 87] TURNER, D. Functional programming and communicating processes. In: PARLE CONFERENCE ON PARALLEL ARCHITECTURES AND LANGUAGES EUROPE, 1987, Eindhoven, The Netherlands. *Proceedings...* Berlin: Springer-Verlag, v.2, p.54-74. (Lecture Notes in Computer Science 259)

[VIC 89] VICKERS, S. *Topology via Logic*. Cambrigde: Cambridge University Press, 1989. 235p.

Informática
UFRGS

"Números Naturais Parciais"

Dissertação apresentada aos Srs.:

Prof. Dr. Benedito Melo Acióly
(Universidade Federal de Pernambuco)

O Examinador enviou parecer por escrito.
Prof. Dr. Daltro José Nunes

Prof. Dr. Paulo Alberto de Azeredo

Prof. Antônio Carlos da Rocha Costa

Vista e permitida a impressão.
Porto Alegre, 14/04/93.

Prof. Dr. Dalcídio Moraes Cláudio
Orientador.

Prof. Dr. Ricardo A. da L. Reis,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação.