Capítulo 2

O Cálculo λ sem Tipos

2.1 Síntaxe e Redução

Por volta de 1930 o cálculo lambda sem tipos foi introduzido como uma fundação para a lógica e a matemática. Embora este objectivo não tenha sido cumprido devido à ocorrência de paradoxos, uma parte significativa desta teoria pôde ser usada com muito sucesso como uma teoria da computação [2].

A linguagem do cálculo λ contém os seguintes símbolos:

- $v_0, v_1, v_2, \ldots$ um conjunto enumerável de variáveis;
- λ a letra grega lambda;
- $\cdot$ o ponto;
- $(,)$ parênteses.

Definição 2.1 (Termo λ)**.** *Os termos de cálculo λ são construídos na seguinte forma:*

- *Uma variável v_i é um termo λ*
- *Se M e um termo λ, então $(\lambda v_i.M)$ é um termo λ* (abstracção)
- *Se M e N são termos λ, então (MN) e um termo λ* (aplicação)

Denotamos termos λ com M, N, K e variáveis com x, y, z (às vezes com índices).

Mais compacto podemos definir também os termos λ pela gramática seguinte:

$$M, N, K ::= x \mid (MN) \mid (x.M)$$

Exemplo 2.1. Exemplos de termos λ são:

$$(\lambda x.((xy)z)), \quad ((\lambda y.y)(\lambda x.(xy))), \quad (x(\lambda x.(\lambda x.x))), \quad (\lambda x.(yz)).$$

Nestes exemplos (e exemplos semelhantes) assumimos sempre que x, y e z designam variáveis diferentes, por exemplo v_0, v_1 e v_2.

Fazemos os seguintes convenções para poupar parênteses:

- Os parêntesis exteriores não se escrevem.

- A aplicação é mais forte do que a abstracção, por exemplo, $\lambda x.MN$ denota $(\lambda x.(MN))$ e não $((\lambda x.M)N)$.
- A aplicação associa à esquerda, por exemplo, $M_1M_2M_3$ denota $((M_1M_2)M_3)$.
- A abstracção associa à direita, por exemplo, $\lambda x_1x_2\ldots x_n.M$ denota $(\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M)\ldots)))$.

Definição 2.2 (Variáveis Livres e Variáveis Mudas)**.** *Uma variável x ocorre livre num termo lambda M se x não está dentro do alcance de λx, No caso contrário diz-se que a variável é muda (ou ligada).*

Exemplo 2.2. *Nos seguintes termos as variáveis sublinhadas ocorrem livres no termo lambda.*

$$\lambda x.x\underline{yz},\quad (\lambda y.y)(\lambda x.x\underline{y}),\quad \underline{x}(\lambda xx.x),\quad \lambda x.\underline{yz}.$$

Uma variável pode ocorrer livre e muda no mesmo termo lambda, por exemplo $y(\lambda y.y)$.

Definição 2.3 (Conjunto das Variáveis Livres)**.** *Denota-se por $FV(M)$ o conjunto das variáveis livres de M, sendo que a sua definição recursiva é dada por:*

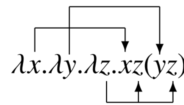
- $FV(x) = \{x\}$,
- $FV(\lambda x.M) = FV(M) \setminus \{x\}$,
- $FV(MN) = FV(M) \cup FV(N)$.

Definição 2.4 (Combinador)**.** *Um termo M é fechado, ou é um combinador, se $FV(M) = \emptyset$.*

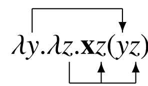
Temos as seguintes notações

- $\Lambda^0 = \{M \in \Lambda \mid M \text{ é fechado}\}$
- $\Lambda^0(\overline{x}) = \{M \in \Lambda \mid \mathrm{FV}(M) \subseteq \{\overline{x}\}\}$,
- O fecho de $M \in \Lambda$ é o lambda termos $\lambda(\overline{x}).M$, com $\{\overline{x}\} = \mathrm{FV}(M)$.

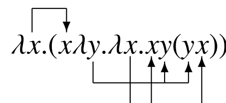
O âmbito de uma abstracção lambda é toda a expressão lambda (até à ocorrência de uma outra abstracção na mesma variável). As variáveis livres são aquelas que não estão no âmbito de nenhuma abstracção lambda. Por exemplo no termo lambda $\lambda x.\lambda y.\lambda z.xz(yz)$ temos:



Enquanto no termo lambda $\lambda y.\lambda z.xz(yz)$ temos:



Para o termo lambda $\lambda x.(x\lambda y.\lambda x.xy(yx))$ temos:



Definição 2.5 (Comprimento de um Termo Lambda)**.** O comprimento de um termo lambda M, $||M||$, é definido por:

- $||x|| := 0$,
- $||MN|| := \max(||M||, ||N||) + 1$,
- $||\lambda x.M|| := ||M|| + 1$.

As demonstrações *com indução sobre M* de uma dada propriedade, são de facto demonstrações *por indução sobre o comprimento de M*.

Os símbolos escolhidos para as variáveis mudas não são importantes, podendo ser trocados sem que isso afecte o significado do termo lambda, tais termos, iguais a menos de uma mudança nas variáveis mudas, podem ser identificados. Por exemplo $\lambda x.\lambda y.\lambda z.xz(yz)$ é identico ao termo $\lambda z.\lambda x.\lambda y.zy(xy)$. Tais termos designam-se por termos *α-congruentes*

Antes de definirmos formalmente a noção de mudança de variáveis mudas, vamos definir a noção de sub-termo.

Definição 2.6 (Sub-termo)**.**

- *M é um sub-termo de N (notação, $M \subset N$) se $M \in \mathrm{Sub}(N)$, aonde $\mathrm{Sub}(N)$, a colecção de sub-termos de N, é definida de forma inductiva da seguinte forma:*
  - $\mathrm{Sub}(x) = \{x\}$;
  - $\mathrm{Sub}(\lambda x.N_1) = \mathrm{Sub}(N_1) \cup \{\lambda x.N_1\}$;
  - $\mathrm{Sub}(N_1N_2) = \mathrm{Sub}(N_1) \cup \mathrm{Sub}(N_2) \cup \{N_1N_2\}$.
- *Um sub-termo pode ocorrer mais do que uma vez num dado termo, por exemplo $M \equiv \lambda x.xI(xI)$ possuí duas ocorrências do sub-termos $I = \lambda y.y$.*
- *Sejam N_1, N_2 dois sub-termos em M. Então N_1 e N_2 são disjuntos se N_1 e N_2 não têm nenhum símbolo em comum.*
- *Uma ocorrência de um termo N de M é dita* activa *se N ocorre como $(NZ) \subset M$ para algum Z; no caso contrário é dita* passiva.

Definição 2.7 (Substituição de Variáveis Mudas)**.**

- *Uma substituição de variáveis mudas em M é a substituição da parte $\lambda x.N$ de M por $\lambda y.(N[x := y])$, sendo que y não ocorre (de todo) em N.*
- *M é α-*congruente *com N, $M \equiv_\alpha N$, se N resulta de M por uma série de mudanças de variáveis mudas.*

Exemplo 2.3. Eis alguns exemplos de termos α-congruentes, e alguns que o não são:

$$\begin{array}{rclrcl}
\lambda x.x & \equiv_\alpha & \lambda y.y, & \lambda x.y & \not\equiv_\alpha & \lambda x.z, \\
\lambda x.xy & \equiv_\alpha & \lambda z.zy, & \lambda x.xy & \not\equiv_\alpha & \lambda y.yy, \\
\lambda x.(\lambda x.x)x & \equiv_\alpha & \lambda x.(\lambda z.z)x, & \lambda x.(\lambda x.x)x & \not\equiv_\alpha & \lambda x.(\lambda z.z)z, \\
\lambda yz.xz(yz) & \equiv_\alpha & \lambda zy.xy(zy), & \lambda yz.xz(yz) & \not\equiv_\alpha & \lambda xy.zy(xy).
\end{array}$$

Podemos adoptar alguma convenções que nos permitem facilitar a forma como lidamos com a substituição de variáveis mudas em termos lambda com.

Nota 1 (Convenção α-congruência)**.** *Termos que sejam α-congruentes são identificados como sendo iguais, por exemplo, $\lambda x.x \equiv \lambda y.y$.*

Nota 2 (Convenção de Variáveis)**.** *Se $M_1, \ldots, M_n$ ocorrem num dado contexto (definição, demonstração, etc.), então, em todos esses termos as variáveis mudas são escolhidas de forma a serem diferentes das variáveis livres.*

Esta última convenção permite-nos manipular os termos lambda de uma forma simplificada.

Definição 2.8 (Substituição)**.** *O resultado de substituir as ocorrências livres de x em M por N, $M[x := N]$, é definido de seguinte forma:*

- $x[x := N] \equiv N$;
- $y[x := N] \equiv y$, *se* $x \not\equiv y$;
- $(\lambda y.M_1)[x := N] \equiv \lambda y.(M_1[x := N])$;
- $(M_1 M_2)[x := N] \equiv (M_1[x := N])(M_2[x := N])$.

Na terceira cláusula não é necessário dizer, "sempre que $y \not\equiv x$ e $y \notin FV(N)$", desde que se respeite a convenção das variáveis.

Lema 2.1 (Lema da Substituição)**.** *Se $x \not\equiv y$ e $x \notin FV(L)$ então*

$$M[x := N][y := L] \equiv M[y := L][x := N[y := L]]$$

A demonstração é feita por indução na estrutura de M [2].

Definição 2.9 (Redex β)**.** *Um termo da forma $(\lambda x.M)N$ diz-se um* redex β *(do inglês "reducible expression"). O termo $M[x := N]$ é designado o seu* contractum.

Definição 2.10 (Passo de redução β, $\triangleright_{1\beta}$)**.** *Se um termo K contém uma ocorrência dum redex $(\lambda x.M)N$ e se substituirmos esta ocorrência por $M[x := N]$ com o termo K' como resultado, então dizemos que K se reduz em um passo a K' (em símbolos: $K \triangleright_{1\beta} K'$).*

Definição 2.11 (Redução β, $\triangleright_\beta$)**.** *Dizemos que M se reduz a N (em símbolos: $M \triangleright_\beta N$), se existe uma sequência finita (possivelmente vazia) de passos de redução β de M até N.*

Definição 2.12 (Conversão β, $=_\beta$)**.** *Dizemos que M e N são* convertíveis β *(em símbolos: $M =_\beta N$), se existe uma sequência finita (possivelmente vazia) de passos de redução β e passos inversos de redução β de M a N.*

A Redução β, $\triangleright_\beta$, é o fecho transitivo e reflexivo de $\triangleright_{1\beta}$, e $=_\beta$ é o fecho simétrico de $\triangleright_\beta$.

Definição 2.13 (Forma normal β)**.** *Um termo diz-se na forma normal β, se não contém nenhum redex β.*

Exemplo 2.4.

1. $(\lambda x.x(xy))N \triangleright_{1\beta} N(Ny)$.
2. $(\lambda x.y)N \triangleright_{1\beta} y$.
3. $(\lambda x.(\lambda y.yx)z)u \triangleright_{1\beta} (\lambda y.yu)z \triangleright_{1\beta} zu$.

4. $(\lambda x.xx)(\lambda x.xx) \triangleright_{1\beta} (\lambda x.xx)(\lambda x.xx) \triangleright_{1\beta} (\lambda x.xx)(\lambda x.xx) \triangleright_{1\beta} \ldots$.

5. $(\lambda x.xx)(\lambda xy.xy) \triangleright_{1\beta} (\lambda xy.xy)(\lambda xy.xy) \triangleright_{1\beta} \lambda y.(\lambda xy.xy)y \triangleright_{1\beta} \lambda y.(\lambda z.yz)$.

Nota 3. As relações $\triangleright_{1\beta}$, $\triangleright_{\beta}$ e $=_{\beta}$ podem também ser definidas na seguinte forma:

$$\frac{}{(\lambda x.M)N \triangleright_{1\beta} M[N/x]} \quad \frac{M \triangleright_{1\beta} N}{(MK) \triangleright_{1\beta} (NK)} \quad \frac{M \triangleright_{1\beta} N}{(KM) \triangleright_{1\beta} (KN)} \quad \frac{M \triangleright_{1\beta} N}{(\lambda x.M) \triangleright_{1\beta} (\lambda x.N)}$$

$$\frac{}{M \triangleright_{\beta} M} \quad \frac{M \triangleright_{1\beta} N}{M \triangleright_{\beta} N} \quad \frac{M_1 \triangleright_{\beta} M_2 \quad M_2 \triangleright_{\beta} M_3}{M_1 \triangleright_{\beta} M_3}$$

$$\frac{M \triangleright_{\beta} N}{M =_{\beta} N} \quad \frac{M =_{\beta} N}{N =_{\beta} M} \quad \frac{M_1 =_{\beta} M_2 \quad M_2 =_{\beta} M_3}{M_1 =_{\beta} M_3}$$

Lema 2.2. *As relações $\triangleright_{1\beta}$, $\triangleright_{\beta}$ e $=_{\beta}$ têm as seguintes propriedades:*

1. $M \triangleright_{1\beta} N \Longrightarrow M[y := K] \triangleright_{1\beta} N[y := K]$.

2. $M \triangleright_{\beta} N \Longrightarrow (KM) \triangleright_{\beta} (KN)$ e $(MK) \triangleright_{\beta} (NK)$.

3. $M_1 \triangleright_{\beta} N_1$ e $M_2 \triangleright_{\beta} N_2 \Longrightarrow (M_1M_2) \triangleright_{\beta} (N_1N_2)$.

4. $M \triangleright_{\beta} N \Longrightarrow (\lambda x.M) \triangleright_{\beta} (\lambda x.N)$.

5. $K \triangleright_{\beta} K' \Longrightarrow M[y := K] \triangleright_{\beta} M[y := K']$.

6. $M \triangleright_{\beta} N \Longrightarrow M =_{\beta} N$.

7. $M =_{\beta} N \Longrightarrow N =_{\beta} M$.

8. $M =_{\beta} N \Longrightarrow (KM) =_{\beta} (KN)$ e $(MK) =_{\beta} (NK)$.

9. $M_1 =_{\beta} N_1$ e $M_2 =_{\beta} N_2 \Longrightarrow (M_1M_2) =_{\beta} (N_1N_2)$.

10. $M =_{\beta} N \Longrightarrow (\lambda x.M) =_{\beta} (\lambda x.N)$.

Dem.: Exercício. □

2.1.1 Estratégia de redução e Church-Rosser

A definição de redução β não determina nenhuma *estratégia de redução*. Isto significa, quando um termo tem dois redexes β, a ordem pela qual a redução deve ser efectuada não é fixa.

Exemplo 2.5. $(\lambda x.(\lambda y.xy)x)z$

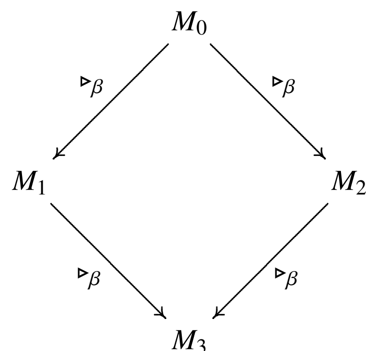
1. $(\lambda x.(\lambda y.xy)x)z \triangleright_{1\beta} (\lambda y.zy)z$

2. $(\lambda x.(\lambda y.xy)x)z \triangleright_{1\beta} (\lambda x.xx)z$

Mas temos a *propriedade Church-Rosser*:

Teorema 2.1 (Confluência de $\triangleright_{\beta}$)**.** *Se $M_0 \triangleright_{\beta} M_1$ e $M_0 \triangleright_{\beta} M_2$, então existe um termo M_3, tal que $M_1 \triangleright_{\beta} M_3$ e $M_2 \triangleright_{\beta} M_3$.*

Ilustrando:



Aqui não vamos demonstrar "Church-Rosser" dado que esta demonstração é muito técnica (e não trivial!) [2].

No exemplo temos imediatamente:

1. $(\lambda x.(\lambda y.xy)x)z \triangleright_{1\beta} (\lambda y.zy)z \triangleright_{1\beta} zz$
2. $(\lambda x.(\lambda y.xy)x)z \triangleright_{1\beta} (\lambda x.xx)z \triangleright_{1\beta} zz$

A estratégia de redução corresponde à ordem em que calculamos uma dada expressão:

Exemplo 2.6. $2 \cdot (3 + 5)$

$2 \cdot (3 + 5) = (3 + 5) + (3 + 5) = 8 + (3 + 5) = 8 + 8 = 16.$

$2 \cdot (3 + 5) = 2 \cdot 8 = 8 + 8 = 16.$

A ordem utilizada nas reduções tem uma grande relevância para a complexidade de um dado cálculo. No entanto pode-se demonstrar que para as estratégias habituais a complexidade é igual à complexidade dos piores casos ("worst-case complexity").

Exemplo 2.7.

1. $(\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta} (\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta}$
$(\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta} \ldots$

2. $(\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta} z$

3. $(\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta} (\lambda y.z)((\lambda x.xx)(\lambda x.xx)) \triangleright_{1\beta} z$

Essencialmente temos duas estratégias de redução interessantes (e implementadas):

1. *Leftmost-Innermost*
2. *Leftmost-Outermost*

No caso "Leftmost-Innermost" reduzimos sempre o redex que está no lado mais esquerdo e que não contém nenhum outro redex.

No caso "Leftmost-Outermost" reduzimos o redex mais exterior e mais esquerdo dum termo.

1. "Leftmost-Innermost":

   $(\lambda x.(\lambda y.yx)z)u \triangleright_{1\beta} (\lambda x.zx)u$

   $(\lambda x.y)((\lambda z.zzz)u) \triangleright_{1\beta} (\lambda x.y)(uuu) \triangleright_{1\beta} y$ $(\lambda x.y)((\lambda x.xx)(\lambda x.xx))$
   $f(x) = 0, g(y) = \frac{1}{y}$
   $f(g(y))$ Argumentos são reduzidos antes da aplicação: "*strict evaluation*".

2. "Leftmost-Outermost":

   $(\lambda x.(\lambda y.yx)z)u \triangleright_{1\beta} (\lambda y.yu)z$

   $(\lambda x.y)((\lambda z.zzz)u) \triangleright_{1\beta} y$

   Aplicações são reduzidas antes dos argumentos: "*lazy evaluation*".

2.1.2 Completude de Turing do cálculo λ e pontos fixos

Até neste momento só considerámos o cálculo λ *puro*. Neste cálculo temos só variáveis como termos atómicos. Mas pode ser demonstrado que este cálculo é *completo no sentido de Turing*, i.e., podemos computar todos a funções computáveis com uma máquina de Turing (e estes são consideradas como todas as funções *intuitivamente* computáveis: ver a *Tese de Church*). De forma mais rigorosa: Podemos representar qualquer função computável Turing por um termo λ puro e podemos calcular os seus valores por β reduções.

Aqui não vamos apresentar esta codificação.

Mas dizemos só que os números naturais podem ser codificado por *numerais de Church*: n pode ser representado por $\lambda f, x.f^n x$ com $f^n x \equiv \underbrace{f(f(\ldots(f}_{n \text{ vezes}}x)\ldots))$. No caso limite de $n = 0$ temos $f^0 x \equiv x$. A função de sucessor (i.e., $f(n) = n + 1$), por exemplo, pode ser definido por $\lambda n.\lambda f, x.f(nfx)$.

A poder de expressão do cálculo λ puro é baseada na possibilidade ilimitada a definir pontos fixos:

> Para cada termo M existe um termo N tal que $MN =_\beta N$.

De facto, existe um termo (fechado) Y que calcula para todo o M um ponto fixo YM de M:

Teorema 2.2 (Ponto fixo)**.**

$$\forall_F \exists_X FX = X$$

Dem.: Sejam $W = \lambda x.F(xx)$ e $X = WW$ então tem-se:

$$\begin{array}{rcl} X & \equiv & WW \\ & \equiv & (\lambda x.F(xx))W \\ & \triangleright_\beta & F(WW) \\ & \equiv & FX \end{array}$$

□

Pontos fixos são usados muitas vezes em definições matemáticas mesmo quando isto não é explicitamente indicado. Em particular, definições por recursão são casos especiais de procura dum ponto fixo.

Exemplo 2.8. *1. Podemos considerar a definição duma função que calcula os números de* FIBONACCI*:*

$$f(x) = \begin{cases} 0, & se\ x = 0, \\ 1, & se\ x = 1, \\ f(x-1) + f(x-2), & se\ x \geq 2. \end{cases}$$

Nesta definição de f usamos na equação da definição *obviamente a própria função f. O que procuramos é que um ponto fixo desta equação.*

Podemos rescrever a definição na seguinte forma:

$$f(x) = \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \textit{ if } x = 1 \textit{ then } 1 \textit{ else } f(x-1) + f(x-2)$$

Na notação λ temos:

$$f(x) = \lambda x. \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \textit{ if } x = 1 \textit{ then } 1 \textit{ else } f(x-1) + f(x-2) \qquad (\star)$$

Vamos definir *f a partir do seguinte funcional (que não é recursivo!):*

$$g := \lambda f.\lambda x. \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \textit{ if } x = 1 \textit{ then } 1 \textit{ else } f(x-1) + f(x-2)$$

Então f deve ser um ponto fixo deste funcional:

$$f := Yg$$

Que f satisfaz a equação $(\star)$ pode ser facilmente calculado:

$$\begin{aligned} f &= Yg \\ &= g(Yg) \quad \textit{pelo teorema do ponto fixo} \\ &= g(f) \\ &= (\lambda f.\lambda x. \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \textit{ if } x = 1 \textit{ then } 1 \textit{ else } f(x-1) + f(x-2))(f) \\ &= \lambda x. \textit{ if } x = 0 \textit{ then } 0 \textit{ else } \textit{ if } x = 1 \textit{ then } 1 \textit{ else } f(x-1) + f(x-2) \quad (\star) \end{aligned}$$

2. $f(x) = f(x+1)$.

$$f(0) = f(1) = f(2) = f(3) = \cdots$$

Como em cima podemos definir f como ponto fixo:

$$\begin{aligned} f(x) &= f(x+1) \\ f &= \lambda x.f(x+1) \qquad (\star) \\ g &:= \lambda f.\lambda x.f(x+1) \\ f &:= Yg \end{aligned}$$

A equação $(\star)$ é verificada na mesma maneira:

$$\begin{aligned} f &\equiv Yg \\ &\triangleright_\beta g(Yg) \\ &\equiv g(f) \\ &\equiv (\lambda f.\lambda x.f(x+1))f \\ &\triangleright_\beta \lambda x.f(x+1) \qquad (\star) \end{aligned}$$

Quando aplicamos a função a um argumento obtemos a seguinte sequência de reduções β:

$$\begin{aligned} f(0) &\triangleright_\beta f(0+1) \\ &\text{“}\triangleright_\beta\text{”} f(1) \\ &\triangleright_\beta f(1+1) \\ &\text{“}\triangleright_\beta\text{”} f(2) \\ &\vdots \end{aligned}$$

Nota se, que qualquer função constante $f(x) = k$ satisfaz a equação de definição de f. Mas, o ponto fixo construído com Y não devolve uma função constante, em vez, a função que é indefinida para todos os argumentos.

3. *O exemplo seguinte é (em certo sentido) “ainda pior”:*

$$f(x) = f(x) + 1.$$

Como nos casos anteriores podem definir f como ponto fixo dum certo funcional. Mas na aplicação desta função a um argumento recebemos a seguinte sequência de reduções β:

$$f(0) \triangleright_\beta f(0) + 1 \triangleright_\beta f(0) + 2 \triangleright_\beta f(0) + 3 \triangleright_\beta \cdots$$

Este função tem só a função totalmente indefinida como ponto fixo.

Nestes exemplos o teorema de ponto fixo só “construa” as *funções*. O teorema diz nada sobre os valores destas funções.

As funções dos últimos exemplos podem ser também implementados (mais ou menos sem problemas) em quase todas as outras linguagens de programação, por exemplo em Pascal:

```
function f(x: integer): integer;
begin
  y := f(x+1);
  f := y;
end;

function f(x: integer): integer;
begin
  y := f(x)+1;
  f := y;
end;
```

Este funções vão resulta em “ciclos infinitos” quando aplica-mo-los a argumentos. Mas, um qualquer compilador de Pascal não tem quaisquer problemas conseguindo compilar as funções acima definidas.