Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
Bacharelado em Tecnologia da Informação
Fundamentos Matemáticos da Computação II

Estudo dirigido do conteúdo da Unidade 2

Autor: Yuri Alessandro Martins

Natal/RN
13 de maio de 2016

O que é esse documento?

Esse é um documento que visa resumir o conteúdo apresentado em sala de aula na disciplina **Fundamentos Matemáticos da Computação** da **Universidade Federal do Rio Grande do Norte**, durante o descorrer da segunda unidade, matéria essa da grade curricular do curso de **Bacharelado em Tecnologia da Informação**.

Tenha em vista que esse documento não server como base concreta/completa de estudo. Ele, na verdade, visa resumir e direcionar o estudo da disciplina. Dessa forma, cabe ao aluno buscar formas complementares de entender o que está sendo dito aqui, como, por exemplo, o material referência utilizado.

Saiba também que esse é um material *open-source*, que pode ter sido alterado por diversas pessoas (você pode conferir isso em 3 - Política de Colaboração), e portanto pode não refletir um conteúdo totalmente "padronizado".

Sumário

1 Teoria dos Conjuntos

1.1 Os Axiomas de Zermelo-Frankel

1.1.1 Extensionalidade

Para quaisquer conjuntos A, B:

$$A = B \iff (\forall x)(x \in A \iff x \in B)$$

1.1.2 Emptyset

Garante que existe um conjunto vazio ($\emptyset$).

$$\exists x \forall y \ (y \notin x)$$

1.1.3 Pairset

Para todo a e b, existe o conjunto $\{a, b\}$.

$$\forall a \forall b \ \exists w \forall x (x \in w \iff x = a \ \vee \ x = b)$$

1.1.4 Separation

Para cada condição $P(x)$,

$$\forall a \exists w \forall x (x \in w \iff x \in a \ \wedge \ P(x))$$

Um problema de usar somente esses últimos três axiomas é que só somos capazes de formar conjuntos com cardinalidade ≤ 2.

- ZF4 (1.1.4) é um axiom-scheme. Isto é, possui infinitos axiomas dentro dele, já que para cada P(x) estamos formando um novo axioma.

- Usando os axiomas anteriores, é possível representarmos algumas coisas como conjuntos:

  - (x, y) $\triangleq$ { {x}, {x, y}}
  - A \ B $\triangleq$ { x ∈ A | x ∉ B }
  - $A \cap B \triangleq \{x \in A \mid x \in B\}$

1.1.5 Powerset $\wp$

Para cada conjunto a, existe um conjunto b, onde os elementos de b são subconjuntos de a.

$$\forall a \exists p \forall b (b \in p \iff \forall x (x \in a \implies x \in b))$$

Esse é o conjunto $\wp$(a).

Aqui $x \in a$ é uma abreviação de $(\forall t)[t \in x \implies t \in a]$. O Axioma da Extensionalidade (1.1.1) implica que para cada a, apenas um conjunto b pode satisfazer a definição do Powerset; Nós podemos chamar **Conjunto Potência** de a e denotá-lo como:
$\wp(a) \triangleq \{x \mid Set(x) \& x \subseteq a\}$

Algumas propriedades interesantes:
$\wp(\emptyset) = \{\emptyset\}$
$\wp(\{\emptyset\}) = \{\emptyset\ , \{\emptyset\}\}$

Exercício: Para cada conjunto A, existe um conjunto B cujo membros são exatamente singletons dos membros de A:

$$x \in B \iff (\exists t \in A)[x = \{t\}]$$

1.1.6 Unionset

Corresponde ao conjunto $\cup a$.

$$\forall a \exists u \forall x (x \in u \iff (\exists e \in a)[x \in e])$$

Ex: $\cup\emptyset = \cup\{\emptyset\} = \emptyset$

- $a \cup b = \cup\{a, b\}$

  Usando os axiomas ZF2 (1.1.2) e ZF5 (1.1.6)

$$t \in A \cup B \iff (\exists X \in \{A, B\})[t \in X]$$
$$t \in A \cup B \iff t \in A \vee t \in B$$

- $a \times b \triangleq \{w \in S \mid \exists x \exists y (w = (x, y) \wedge x \in a \wedge x \in b)\}$
  Onde $S = \wp(\wp(a \cup b))$
- sigletonset $\triangleq \{x \in \wp a \mid (\exists t \in a)[x = \{t\}]\}$
- $\cap a \triangleq \{x \in \cup a \mid (\forall e \in a)[x \in e]\}$

1.1.7 Infinity Axiom

$\exists I(\emptyset \in I \wedge \forall x(x \in I \implies \{x\} \in I))$ ou
$\exists I(\emptyset \in I \wedge \forall x(x \in I \implies x \cup \{x\} \in I))$

Esse axioma é garantido pois

$$\{x\} \neq x$$
$$x \cup \{x\} \neq x$$

Com ele, somos capazes de montar os seguintes conjuntos:

$I = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \cdots\}$ *ou*
$I = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \cdots\}$

1.1.8 Definindo um par ordenado

Consideraremos primeiro a operação básica de par (ordenado). Intuitivamente, o par (x, y) de dois objetos x e y é a "coisa" que tem o "primeiro membro" x e o "segundo membro" y, e é diferente do par não-ordenado $\{x, y\}$ desde que (por exemplo) $\{0, 1\} = \{1, 0\}$ enquanto $(0, 1) \neq (1, 0)$. Assim, a primeira caracteristica de um par ordenado seria:

$$(x, y) = (x', y') \iff x = x' \ \& \ y = y' \qquad (OP1)$$

Existe um segundo, provavelmente uma propriedade caracteristica dos pares menos óbvia, que torna possível definir Produtos Cartesianos: para dois conjunto A e B:

$$\text{A classe } A \times B =_{\text{df}} \{(x, y) \mid x \in A \ \& \ y \in B\} \text{é conjunto.} \qquad (OP2)$$

Então, o problema de representar a noção de "par" na teoria dos conjunto é que: é preciso definir uma operação (x, y), tal que OP1 e OP2 respeitam os axiomas de Zermelo.

A operação de par do **Kuratowski**[1]:

$$(x, y) = \{\{x\}, \{x, y\}\} \qquad \text{Respeita OP1 e OP2}$$

[1]Como vimos anteriormente, apenas exemplificando como um Axioma da Separação poderia ser utilizado, na seção 1.1.4.

Demonstração. (OP1) Provando a direção $\implies$ [2], vamos distinguir dois casos.

Se $x = y$, então $\{x, y\} = \{x, x\} = \{x\}$, o conjunto $(x, x) = \{\{x\}, \{x\}\} = \{\{x\}\}$ é um singleton, consequentemente o conjunto (x', y') , que se presume ser igual a ele também é um singleton, de modo que $x' = y'$ e $(x', y') = \{\{x'\}\}$; e desde que este último singleton seja igual a $\{\{x\}\}$, nos temos $x = x'$, consequentemente, também $y = x = x' = y'$.

Agora, se $x \neq y$, então os membros de (x, y) são singletons de $\{x\}$ e o doubleton $\{x, y\}$, e estes devem corresponder aos membros $\{x'\}$ e $\{x', y'\}$ do conjunto (x', y'), e que nos devemos ter $\{x\} = \{x'\}, \{x, y\} = \{x', y'\}$, e, então, imediatamente, $x = x'$ e $y = y'$.

(OP2) É suficiente provar que para dois conjunto A e B, existe um conjunto C tal que:

$$x \in A \ \& \ y \in B \implies \{\{x\}, \{x, y\}\} \in C$$

Porque então:

$$A \times B = \{z \in C \mid (\exists x \in A)(\exists y \in B)[z = (x, y)\}$$

Seguindo pelo x:

$$\begin{aligned} x \in A, y \in B &\implies \{x\}, \{x, y\} \subseteq (A \cup B) \\ &\implies \{x\}, \{x, y\} \in \wp(A \vee B) \\ &\implies \{\{x\}, \{x, y\}\} \subseteq \wp(A \cup B) \\ &\implies \{\{x\}, \{x, y\}\} \in \wp(\wp(A \cup B)) \end{aligned}$$

então nós podemos ter $C = \wp\wp(A \cup B)$. □

Nós agora definimos a operação (x, y) que satisfaz (**OP1**) e (**OP2**), talvez o **Kuratowski**, provado anteriormente, ou talvez algum outro: não importa mais a definição escolhida, a única coisa que conta é a operação de par que satisfaça (**OP1**) e (**OP2**)

1.1.9 União Disjunta

- União Disjunta: $A \uplus B = (\{0, a\} \times A) \cup (\{1, b\} \times B)$

[2] A direção oposta é muito trivial, segundo [3]

1.2 Relações e Funções Parciais

1.2.1 Relações

Def: Sejam A, B conjuntos, R é uma relação entre A e B se $R \subseteq A \times B$.

Dessa forma,

- $f(a) = b \rightsquigarrow (a, b) \in f$

Sendo R uma relação sobre o conjunto $\mathbb{N}(R \subseteq A \times A), R$ pode ser:

$$\begin{aligned} & xRx : Reflexiva \rightsquigarrow \text{“} =,\ \leq,\ \geq,\ \subseteq'' \\ xRy \implies & yRx : Simetrica \rightsquigarrow \text{“} ='' \\ xRy \wedge yRz \implies & xRz : Transitiva \rightsquigarrow \text{“} =,\ \leq,\ \geq,\ <,\ >,\ \subseteq'' \end{aligned}$$

Ainda existem outras propriedas como essas, como a Antireflexiva ou Antisimétrica.

1.2.2 Relações de Equivalência

Uma relação sobre um conjunto A é chamada **relação de equivalência** se ela for reflexiva, simétrica e transitiva.

O conjunto de todos os elementos que são relacionados a um elemento a de A é chamado de classe de equivalência de a. Isso implica que:

$$\cup[a] = A$$

- $[a] \cap [b] = \emptyset \ quando \ [a] \ \neq [b]$

Uma partição de um conjunto S é uma coleção de subconjuntos disjuntos não vazios de S. A união de todas as partições resulta, portanto, em S. Em outras palavras, os subconjuntos A_i formam partições de S se e somente se

$$\begin{aligned} & A_i \neq \emptyset \\ & A_i \cap A_j = \emptyset, quando\ i \neq j \\ \cap & A_i = S \end{aligned}$$

Podemos definir classes de equivalência como:

$$\begin{aligned} [x/\backsim] &\triangleq \{a \in A \mid x \backsim a\} \\ [A/\backsim] &\triangleq \{c \in \wp(A) \mid \exists x\ C = [x/\backsim]\} \end{aligned}$$

Seja $x, y \in A$, e $\backsim$ uma relação de equivalência no A:

$$[x/\backsim] = [y/\backsim] \iff x \backsim y$$

$$[x/\backsim] = [y/\backsim] \iff \begin{cases} [x/\backsim] & \text{se } x \backsim y \\ \emptyset & \text{se não} \end{cases}$$

$$\cup\{[x/\backsim] \mid x \in a\} = A$$

1.2.3 Funções parciais na ZFC

O conceito de funções parciais remete a ideia de uma função em que nem todos os x possuem uma $f(x)$ (Usaremos a notação $\rightharpoonup$).

$f : \mathbb{N} \rightharpoonup \mathbb{N}$ Domínio da função

$f(x) = \sqrt[2]{x}$ $x = 3$ não possui uma saída bem definida nesse domínio.

1.3 Currying

Dada uma f do tipo $f : (X \times X) \to Z$, então a técnica de **currying** a torna $(f) : X \to (Y \to Z)$. Isto é, currying torna um paramêtro do tipo X e retorna uma função do tipo $Y \to Z$.

Achar um $\phi((x, y) \to A \rightarrowtail\!\!\!\!\!\rightarrow (x \to (y \to A)))$
$\phi(F) = G$, onde G é definida pela,
$G(x)(y) = g$, onde g é definida pela,
$g(y), f(x, y)$

1.4 Cardinais

Seja A um conjunto. O que é $|A|$?

c1. $A =_c |A|$

c2. $A =_c B \iff |A| = |B|$

c3. para todo conjunto de conjuntos ϵ,
$\{|x| \mid x \in \epsilon\}$ é conjunto.

1.4.1 Números Cardinais

Uma (fraca) **atribuição de cardinalidade** é qualquer operação definida nos conjuntos $A \mapsto |A|$ que satisfaz (c1) e (c3). Uma **atribuição de cardinal forte** também satisfaz o (c2). Iremos utilizar a definição **fraca**, que pode ser

ajustada ao (c2) como:

c2. $A =_c B \iff |A| =_c |B|$

Os **números cardinais** (relativo a uma dada atribuição de cardinalidade) são os valores:

$$\mathrm{Card}(\kappa) \iff \kappa \in \mathrm{Card} \iff_{\mathrm{df}} (\exists A)[\kappa = |A|]$$

Vamos definir uma especifica (provavelmente fraca) atribuição de cardinalidade e vamos definir as operações aritiméticas nos cardinais como: Sejam κ, λ, μ números cardinais:

$$\begin{aligned} \kappa + \lambda &\triangleq_c \kappa \uplus \lambda \\ \kappa.\lambda &\triangleq_c \kappa \times \lambda \\ \kappa^{\lambda} &\triangleq_c (\kappa \to \lambda) \end{aligned}$$

Atente ao fato que existe apenas uma escolha para $|\emptyset|$,

$$0 =_{\mathrm{df}} |\emptyset| = \emptyset$$

uma vez que apenas $|\emptyset| = \emptyset$ satisfaz $\emptyset =_c |\emptyset|$. Isso é também conveniente para o conjunto:

$$1 =_{\mathrm{df}} |\{\emptyset\}|, \ 2 =_{\mathrm{df}} |\{0, 1\}|$$

1.5 Os Axiomas de Peano

Structed set: $(\mathbb{N}; 0; S)$, onde $0 \in \mathbb{N}$ e $S : \mathbb{N} \to \mathbb{N}$

$$\begin{aligned} 0 &\in \mathbb{N} \\ S &: \mathbb{N} \to \mathbb{N} \\ S &: \mathbb{N} \rightarrowtail \mathbb{N} \\ (\forall x \in \mathbb{N})&[S_n \neq 0] \\ (\forall x \subseteq \mathbb{N})&[[0 \in X \wedge (\forall n \in \mathbb{N})[n \in X \implies S_n \in X]] \implies X = \mathbb{N}] \end{aligned}$$

O axioma de peano 5 é o que nos permite realizar indução matemática. Observe:

$$\begin{aligned} &\forall x \subseteq \mathbb{N} \textit{ corresponde a } \textbf{base}. \\ (\forall n \in \mathbb{N})[n \in X \implies S_n \in X] &\textit{ corresponde ao } \textbf{passo indutivo}. \\ &n \in X \textit{ corresponde a } \textbf{hipotése indutiva}. \end{aligned}$$

1.6 Teorema da Recursão

Theorem. *Sejam:* $(\mathbb{N}, 0, S)$ *um sistema de naturais conjunto E.*
$a \in E$
$h : E \to E$
Então existe $f : \mathbb{N} \to E$
tal que: $f(0) = aef(S_n) = h(f(n))$.

Agora, precisamos provar que f existe e é bem definida no domínio escolhido.

Demonstração.

Sendo $f_0 \triangleq \{0\} \to E$ $\qquad f_0 \ : \ \{\emptyset\} \to E$

Sendo $f_1 \triangleq \{(0, a)\} \to E$ $\qquad f_1 \ : \ \{0, 1\} \to E$

Sendo $f_2 \triangleq \{(0, a), (1, h(a))\}$ $\qquad f_2 \ : \ \{0, 1, 2\} \to E$

$$\vdots$$

$$f \triangleq \bigcup_i f_i$$

$$\begin{aligned} p \in \mathscr{A} \iff & \\ & \text{Function}(p) \\ & \wedge \, \text{domain}(p) \\ & \wedge \, \text{image}(p) \\ & \wedge \, 0 \in \text{domain}(p) \\ & \wedge \, p(0) = a \\ & \wedge \, S_n \in \text{domain}(p) \implies (n \in \text{domain}(p) \wedge p(S_n) = h(p(n))) \end{aligned}$$

Seja $f = \bigcup \mathscr{A}$

Para confirmar que f é uma função:
p_1 & $q \in \mathscr{A}$
$n \in \text{domain}(p) \cap \text{domain}(q) \implies p(n) = q(n)$

$$
\begin{aligned}
BASE: \ 0 \in \ \text{domain}(p) \ \wedge \ \text{domain}(p) \ &\implies p(0) = q(0) \\
p(0) = a = q(0) \quad &\text{(def de } \mathscr{A}\text{)} \\
H.P: \ n \in \text{domain}(p) \wedge \ \text{domain}(q) \ &\implies p(n) = q(n) \\
P.I: \ S_n \in \text{domain}(p) \wedge \text{domain}(q) \ &\implies p(S_n) = q(S_n) \\
\text{Suponha } S_n \in \text{domain}(p) \wedge \text{domain}(q): & \\
p(S_n) = h(p(n)) \quad &(p \in \mathscr{A}) \\
= h(q(n)) \quad &\text{(H.I)(def de } \mathscr{A}\text{)} \\
= q(S_n) \quad &\text{(def de } \mathscr{A}\text{)}
\end{aligned}
$$

Agora que $\text{domain}(f) = \mathbb{N}$

$$
\begin{aligned}
0 &\in \text{domain}(f) \\
n \in \text{domain}(f) \ &\implies \ S_n \in \ \text{domain}(f)
\end{aligned}
$$

□

... Certamente para ser melhorado depois!

1.7 Os Naturais na ZFC

Para estabelecermos os Naturais na ZFC, temos que garantir duas coisas:

- Existência de $\mathbb{N}$
- Singularidade de $\mathbb{N}$

Para tal, iremos precisar do Teorema da Recursão (1.6).

1.7.1 Existência de $\mathbb{N}$

O Axioma da Infinidade (1.1.7) garante a exitência de um conjunto I tal que:

$$
\begin{aligned}
\emptyset &\in I \\
(\forall n)[n \in I \ &\implies \ \{n\} \in I]
\end{aligned}
$$

Usando esse I, primeiro vamos definir uma família de conjuntos:
Seja $\jmath$ = todos os conjuntos $C \subseteq I$ tal que satisfaz o ZF7 (1.1.7)

$\jmath = \{C \in \wp I \mid \emptyset \in C \wedge (\forall x \in C)[\lambda x.\{x\} \in C]\}$

Seja $\mathbb{N} \triangleq \bigcap \jmath$
Seja $0_1 = \emptyset$
Seja $S_1 = \lambda x.\{x\}$

Primeiro, criamos o $\jmath$ porque o Axioma da Infinidade (1.1.7) não nos garante que existe apenas um I, mas que, certamente, existe esse I definido (então, vamos considerar todos os I que existem como $\jmath$). Então, definimos $\mathbb{N} \triangleq \cap\jmath$ porquê assim iremos conseguir ignorar todos os outros conjuntos I dois quais nos não precisamos e iremos manter apenas o que queremos, tendo em vista que todos os C terão o $\emptyset$ e mais alguma coisa (lixo[3]), e o nosso I procurado terá apenas $\emptyset$. Dessa forma, o lixo foi ignorado.

Para terminar a prova, é suficiente provar que $(\mathbb{N}, 0, S)$ é um sistema de Peano. Para começar, $\mathbb{N} \in \jmath$, por que $X \in \jmath \implies \emptyset \in X$ e consequentemente $\emptyset \in \bigcap = \mathbb{N}$. Encaixando com os Axiomas de Peano:

1. $(\forall x \in J)[\emptyset \in X]$, então $\emptyset \in \cap J$ e $\emptyset \in \mathbb{N}$

2. Também, pela própria definição de J

3. $a \neq b \iff S_a \neq S_b$ ou $a \neq b \iff \{a\} \neq \{b\}$

4. $\forall x\{x\} \neq \emptyset$

5. Seja $X \subseteq \mathbb{N}$, tal que
   $0 \in$ X
   $(\forall x \in X)[S_X \in X]$
   Seja $n \in \mathbb{N}$
   $\exists p : x = \{p\}$
   –> Mesmo que $\{p\} \in \cap J = \mathbb{N}$
   –> Mesmo que $n \in X$ e $x \geq \mathbb{N}$

Basicamente, podemos descrever $\mathbb{N}$ de duas maneiras agora (dependendo de qual das duas versões do Axioma da Infinidade (1.1.7) você resolveu usar nessa etapa):
0 $\emptyset = \emptyset$
1 $\{\emptyset\} = \{0\}$
2 $\{\{\emptyset\}\} = \{1\}$
3 $\{\{\{\emptyset\}\}\} = \{2\}$

[3]Termo não tão apropriado, mas com uma noção interessante se traçarmos um parelelo com linguagem de programação.

$\vdots$
S $= \lambda$x.$\{x\}$
ou
0 $\quad \emptyset = \emptyset$
1 $\quad \{\emptyset\} = \{0\}$
2 $\quad \{\emptyset, \{\emptyset\}\} = \{0, 1\}$
3 $\quad \{\emptyset, \{\emptyset\}\, \{\{\emptyset\}\}\}$
$\vdots$
S $= \lambda x.x \cup \{x\}$

1.7.2 Singularidade de $\mathbb{N}$

"$\mathbb{N}$ is unique up to isomorphism:"

$$\pi : (\mathbb{N}_1; 0_1; S_1) \rightarrowtail\!\!\!\!\!\rightarrow (\mathbb{N}_1; 0_2; S_2).$$
$$\pi(0_1) = 0_2$$
$$\pi(S_1 n_1) = S_2 \pi(n_1)$$

Se traçarmos um paralelo com o Teorema da Recursão (1.6), para tentarmos provar a singularidade de $\mathbb{N}$, podemos realizar as seguintes associações:

$\mathbb{N} : \mathbb{N}_1$
$E : \ \mathbb{N}_2$
$a : \ 0_2$
$h : \ S_2$

Demonstração. $\pi : \mathbb{N}_1 \implies \mathbb{N}_2$
$\pi[\mathbb{N}_1] = \mathbb{N}_2$

- $0_2 \in \pi[\mathbb{N}_1]$
  –> Como $\pi(0_1) \implies S_2 n_2 \in \pi[\mathbb{N}_1]$

- $n_2 \in \pi[\mathbb{N}_1]$
  –> Suponha que $n_2 \in \pi[\mathbb{N}_1]$ –> H.I
  –> $(\exists n_1 \in \mathbb{N}_1)[\pi(n_1) = n_2]$
  –> $\pi(S_1 n_1) = S_2(\pi(n_2))$ que $= S_2 n_2$

... [4]

□

[4] Essa prova ainda não está terminada.

1.8 String Recursion

- Dado [ ] $\in [\mathbb{N}]$
- Se $n \in \mathbb{N}$, e $L \in [\mathbb{N}]$, então $(n : L) \in [\mathbb{N}]$

Exemplo: 2:3:4:[ ] = [2,3,4]

Alguns exemplos de funções recursivas que podemos definir utilizando String Recursion:

$$
\begin{aligned}
iszero \ : \ & [\mathbb{N}] \to \mathbb{B} \\
& iszero \ 0 = true \\
& iszero \ S_n = false
\end{aligned}
$$

$$
\begin{aligned}
empty \ : \ & [\mathbb{N}] \to \mathbb{B} \\
& empty \ [\ ] = true \\
& empty \ (x : x_s) = false
\end{aligned}
$$

$$
\begin{aligned}
++ : [\mathbb{N}] \to [\mathbb{N}] \to [\mathbb{N}] \\
[\ ] \ ++ \ y_s = y_s \\
(x : x_s) \ ++ \ y_s = x : (x_s ++ y_s)
\end{aligned}
$$

$$
\begin{aligned}
Ex : [1,2] ++ [6,7,8,9] &= 1 : 2 : [\ ] ++ [6,7,8,9] \\
&= 1 : (2 : [\ ] ++ [6,7,8,9]) \\
&= 1 : (2 : ([\ ] ++ [6,7,8,9])) \\
&= 1 : 2 : [6,7,8,9] \\
&= [1,2,6,7,8,9]
\end{aligned}
$$

$$
\begin{aligned}
reverse \ : \ & [\mathbb{N}] \to [\mathbb{N}] \\
& reverse \ [\ ] = [\ ] \\
& reverse \ [x] = [x] \\
& revese \ (x : x_s) = reversexs ++ [x]
\end{aligned}
$$

$$
\begin{aligned}
\sqsubseteq :\ & [\mathbb{N}] \to [\mathbb{N}] \to \mathbb{B} \\
& (x : x_s) \sqsubseteq [\ ] = false \\
& '[\ ] \sqsubseteq y_s = true \\
& (x : x_s) \sqsubseteq (y : y_s) = (x = y) \wedge xs \sqsubseteq ys
\end{aligned}
$$

$$
\begin{aligned}
Ex : [2,3,4,5] \sqsubseteq [2,3,5,7] &= (2 = 2) \wedge ([3,4,5] \sqsubseteq [3,5,7]) \\
&= (3 = 3) \wedge ([4,5] \sqsubseteq [5,7]) \\
&= (4 = 5) \wedge ([5] \sqsubseteq [7]) \\
&= \text{FALSE}
\end{aligned}
$$

$$
\begin{aligned}
\in :\ & \mathbb{N} \to [\mathbb{N}] \to \mathbb{B} \\
& n \in x\ [\ ] = false \\
& x \in (x : x_s) = (n = x) \vee (n \in x_s)
\end{aligned}
$$

$$
\begin{aligned}
& find : \mathbb{N} \to [\mathbb{N}] \to \mathbb{B} \\
& find\ n\ [\ ] = 0 \\
& find\ n\ (n : nx) = 0 \\
& find\ n\ (x : x_s) = 1 + find\ n\ x_s
\end{aligned}
$$

$$
\begin{aligned}
sum\ :\ & [\mathbb{N}] \to \mathbb{N} \\
& sum[\ ] = 0 \\
& sum(x : xs) = x + sumxs
\end{aligned}
$$

$$
\begin{aligned}
\oplus\ :\ & [\mathbb{N}] \to [\mathbb{N}] \to [\mathbb{N}] \\
& '[\ ] \oplus y_s = y_s \\
& x_s \oplus [\ ] = x_s \\
& (x : x_s) \oplus (y : y_s) = (x + y) : (x_s \oplus y_s)
\end{aligned}
$$

$$
\begin{aligned}
circle\ :\ (\mathbb{N} \to \mathbb{N} \to \mathbb{N}) \to [\mathbb{N}] \to [\mathbb{N}] \to [\mathbb{N}] \\
circle\ f\ [\ ]\ y_s &= [\ ] \\
f x s [\ ] &= [\ ] \\
f\ (x : x_s)\ (y : y_s) &= [f\ x\ y] : (circle\ f\ x_s\ y_s)
\end{aligned}
$$

2 λ-Calculus [6]

Tomando como base o conteúdo visto em **Lambda-calculus and combinators** [2] e **Denotational semantics** [5], passamos para a notação de λ dando uma sintaxe formal e regras de tranformação sintatíca (conversão).

2.1 O conjunto de λ-termos

Sendo $\Lambda = \lambda$-termos;

$$
\begin{aligned}
X &\in \Lambda \\
s, t \in \Lambda &\implies (s\ t) \in \Lambda \\
x \in var, t \in \Lambda &\implies \lambda X.t \in \Lambda
\end{aligned}
$$

2.2 Conversões $\alpha,\ \beta\ e\ \eta$

2.2.1 Conversão α

Determina que a escolha da variável ligada, na abstração lambda, não importa (normalmente):

$\lambda x.x =_\alpha \lambda y.y$

$\lambda x.\lambda x.x =_\alpha \lambda y.\lambda x.x$ Note que isso não poderá ser transformado em $\lambda y.\lambda x.y$

Primeiro, quando alfa-conversão atua em uma abstração, as únicas ocorrências de variáveis que podem ser renomeados são aqueles que são vinculados a esta mesma abstração. No segundo exemplo, portanto:

$\lambda x.\lambda x.x \neq_\alpha \lambda y.\lambda x.y$ Este último tem um significado diferente do original.

Em segundo lugar, uma conversão α não é possível se isto irá resultar em uma variável sendo capturada por uma abstração diferente. Por exemplo, se substituirmos x com y em $\lambda x.\lambda y.x$, nós obteríamos $\lambda y.\lambda y.y$, que tem um significado diferente da expressão anterior.

2.2.2 Redução β

Redução β é a ideia de aplicar uma função. Por exemplo, se temos $f(x) = x * 2$, para $x = 2$ aplicamos o valor a função que irá ficar como $f(2) = 2 * 2$. Essa é basicamente a ideia da redução β.

$(\lambda x.x * 2)\ 2 =_\beta 2 * 2$

2.2.3 Redução η

Eta-conversão expressa a ideia de extensionalidade, que neste contexto é que duas funções são as mesmas se e somente se eles dão o mesmo resultado para todos os argumentos.[5]

2.3 Aritmética em λ-Calculus

Utilizando os numerais de **Church** [6] [7], os números naturais podem ser definidos da seguinte maneira:

$$
\begin{aligned}
0 &:= \lambda f.\lambda x.\ x \\
1 &:= \lambda f.\lambda x.\ f\ x \\
2 &:= \lambda f.\lambda x.\ f\ (f\ x) \\
3 &:= \lambda f.\lambda x.\ f\ (f\ (f\ x)) \\
&\vdots \\
n &:= \lambda f.\lambda x.\ f^n\ x
\end{aligned}
$$

Isso quer dizer que um número natural n é representado pelo numeral de Church n, que tem a propriedade que, para quaisquer termos $\lambda\ F$ e X,

$$n\ F\ X =_\beta\ F^n\ x$$

2.3.1 Cálculo com os numerais de Church

No cálculo lambda, as funções numéricas são representáveis por funções correspondentes nos numerais de Church.

Para começar, vamos definir a função succ, que recebe um número n e retorna $n + 1$ pela adição de outra aplicação de f.

$SUCC\ :=\ \lambda n.\lambda f.\lambda x.f(n\ f\ x)$

[5] Sujeito a severas mudanças no futuro. Visite 3 para saber mais sobre.

$(n\ f\ x)$ quer dizer que f será aplicada n vezes em x. Por exemplo:

3 sqrt 7 = sqrt(sqrt(sqrt(7)))

Tendo em vista isso, queremos continuar aumentando nosso leque de operações aritméticas. Vamos com a soma. Sabemos que, por exemplo, fizermos $2+3$, deveremos somar 3 vezes o valor 2 à 1, de modo que $2+1+1+1=5$. Podemos então utilizar nosso função de SUCC, já definida, para fazer isso, e realizarmos m vezes SUCC de um valor n:

$PLUS\ :=\ \lambda m.\lambda n.m$ SUCC $n0$

Isso quer dizer que estamos aplicando SUCC m vezes em n. Agora ficou fácil continuar ampliando nossa aritmética:
Exercicio:

$-MULT\ :=$
$-POW\ :=$

2.4 Booleanos naturais no Λ

O λ-Calculus pode ser usado para modelar valores booleanos, aritmética, estruturas de dados e recursividade. Vamos focar agora em como podemos determinar operadores booleanos em λ-Calculus:

$$\lambda x.(\lambda y.x) \coloneqq true \coloneqq fst$$
$$\lambda x.(\lambda y.y) \coloneqq false \coloneqq snd$$

Um importante conceito no booleanos no λ-Calculus é do $iftheelse$. Ele pode ser explicado traçando um paralelo com operador ternário em linguagem de programação, no caso de $b?x:y$ para linguagem C. Em suma, ele irá testar o booleano b e irá para x se b TRUE, ou para y, caso contrário. Assim, "if b then x else y".

Com ele, fica mais fácil definirmos outros operadores booleanos no λ-Calculus. Antes, é importante nos lembrarmos de algumas propriedas interessantes:

$$a \wedge b = \begin{cases} b, \text{ se } a = \text{ TRUE} \\ a, \text{ se } a = \text{ FALSE} \end{cases} \qquad a \vee b = \begin{cases} \text{TRUE}, \text{ se } a = \text{ TRUE} \\ b, \text{ se } a = \text{ FALSE} \end{cases}$$

Sabendo disso...

Exercicio: Ache λ-Termos que se comportem como o:

- “not”

- “or”

- “and”

2.5 Combinadores I, K, B, S [2] e [4]

O sistema de combinadores foi criado para fazer o mesmo trabalho que os sistemas de λ-Calculus, mas sem o uso de variáveis ligadas. Na verdade, as irritantes complicações técnicas envolvida na substituição e na α-conversão vão ser evitadas completamentes a partir daqui. Entretanto, para essa técnica avançada no teremos que sacrificar a clareza intuitiva da notação λ.

Para motivar combinadores, considere a lei da comutatividade da adição em aritmética, que diz:

$$(\forall x, y)\ x + y = y + x$$

A expressão acima contém variáveis ligadas “x” e “y”. Mas elas podem ser removidas, como a seguir. Nos vamos primeiro definir um operador de adição A por:

$$A(x, y) = x + y \qquad \text{(para todo } x, y)$$

e vamos introduzir um operador C definido po:

$$(C(f))(x, y) = f(y, x) \qquad \text{(para todo } f, x, y)$$

Então a lei da comutatividade se torna simples

$$A = C(A)$$

O operador C pode ser chamado de **combinador**.

Assuma que é dado uma sequência infinita de expressões $v_0, v_00, v_000, \cdots$ chamadas “variáveis”, e um finita ou infinita sequencia de expressões chamada “constantes atômicas”, incluindo três chamadas “combinadores básicos”: **I**, **K** e **S** (Se **I**, **K** e **S** são as únicas constantes atômicas o sistema será chamado de puro, caso contrário, será aplicado). O conjunto de expressões chamado Termos Lambda é definido por indução como:

(a) todas as variáveis e constantes atômicas, incluindo **I**, **K** e **S** são Termos Lambda

(b) se X e Y são Termos Lambda, então é assim (X, Y)

$K = \lambda x \lambda y.x$	Retorna uma função contante
$B = \lambda x \lambda y \lambda z.x(yz)$	"Composition"
$I = \lambda x.x$	Retorna seu argumento
$S = \lambda x \lambda y \lambda z.xz(yz)$	Operador de substituição.

Exercicio: Mostre que
- $S(KS)K \to B$:

$(SKK)x = KxKx$	Pela definição de S
$= x$	Pela definição de K
$= I$	Pela definião de I

- $S(KS)K \to B$:

$(S(KS)K)xyz = ((KSx)Kx)yz$	Pela definição de S
$= (SKx)yz$	Pela definição de K
$= (Kxy)(yz)$	Pela definição de S
$= x(yz)$	Pela definição de K
$= B$	Pela definição de$B$.

3 Política de Colaboração

Você é capaz de alterar o conteúdo desse documento, para corrigir erros, melhorar suas explicações ou dar dicas/exemplos adicionais. Esse foi o objetivo desde começo.

Visita a página remota do documento para obter sua versão mais atualizada e/ou colaborar também.

Como base foram utilizados os livros "Notes on Set Theory" [3] e "Classic Set Theory" [1]. Caso você queira continuar usando-os como base para esse documento, sinta-se a vontade. Caso você queira, também, utilizar outras fontes, não deixe de citá-las para que sejam adicionadas as refêrencias. Tenha em mente que outros estudantes podem estar querendo estudar baseado nesse documento, então busque sempre informações confiáveis que podem ser atestadas (por meio da biblografia, por exemplo).

3.1 Colaboradores

- Yuri Alessandro Martins

- Thanos Tsouanas
- Elton Viana
- Gilney Junior
- João Victor

Referências

[1] DC Goldrei. *Classic Set Theory: For Guided Independent Study*. CRC Press, 1996.

[2] Jonathan P. Seldin J. Roger Hindley. *Lambda-calculus and combinators, an introduction*. Cambridge University Press, 2008.

[3] Yiannis Moschovakis. *Notes on set theory*. Springer Science & Business Media, 2006.

[4] Sören Stenlund. *Combinators, λ-terms and Proof Theory*, volume 42. Springer Science & Business Media, 2012.

[5] Joseph E Stoy. *Denotational semantics: the Scott-Strachey approach to programming language theory*. MIT press, 1977.

[6] Wikipedia. Lambda calculus — wikipedia, the free encyclopedia, 2007. [Online; accessed 12-May-2016].

[7] Wikipedia. Church encoding — wikipedia, the free encyclopedia, 2016. [Online; accessed 13-May-2016].