

INDUÇÃO MATEMÁTICA

RAUL H.C. LOPES

1. INDUÇÃO MATEMÁTICA

1.1. **Conjunto indutivamente definido.** Considere a seguinte definição, encontrada em [5].

Definição de um conjunto indutivo. Um conjunto de números reais é chamado de conjunto indutivo se ele tem as seguintes propriedades:

- (1) O número 1 pertence ao conjunto.
- (2) Para todo x pertencente ao conjunto, o número $x + 1$ também pertence ao conjunto.

Essa definição apresenta um *método para computar* números reais a partir de um elemento inicial: o número 1. Note que essa definição limita-se a estabelecer como encontrar elementos de **um** conjunto indutivo de reais (os inteiros positivos): ela nem tenta cobrir todo o conjunto de reais e nem mesmo determinar exatamente todos os elementos do conjunto indutivo em questão.

Veja agora a seguinte definição retirada da mesma fonte.

Definição de inteiros positivos. Um número real é chamado inteiro positivo se ele pertence a todo conjunto indutivo.

A definição acima, claramente referindo-se e complementando a anterior, tentar definir exatamente o conjunto dos inteiros positivos estabelecendo que ele é a interseção de todos os conjuntos possíveis de se obter, usando a primeira definição.

Além de estabelecer um “método para enumerar” todos os elementos do conjunto de inteiros positivos, essas duas definições sugerem:

- i um método para testar se qualquer z é um inteiro positivo: tal teste consistiria em verificar se z é o elemento inicial do conjunto definido ou se ele pode ser reescrito como $x + 1$, para algum x pertencente ao conjunto definido.
- ii um método de prova de propriedades sobre conjuntos de inteiros positivos. Para provar a propriedade P_n , para todo n inteiro positivo basta:
 - 1 Provar que a propriedade P é válida para o elemento inicial do conjunto de interesse;

2 Provar que Pk implica $P(k + 1)$, ou seja, que, para qualquer elemento do conjunto em questão, se P vale para esse elemento, então P vale para o seu sucessor.

Exemplo 1. *Provar que para todo n*

$$+i : 1 \leq i < n : i = \frac{n(n-1)}{2}$$

Prova.

Aplicar o método de prova acima delineado demanda:

- *Identificar a propriedade a provar. Seja P uma função que mapeia um inteiro para booleanos (conjunto $\{True, False\}$), definido por*

$$P = \lambda n. +i : 1 \leq i < n : i = \frac{n(n-1)}{2}$$

Dada esta definição, a propriedade que se deseja provar seria

$$\wedge n : : P n$$

- *Garantir que a propriedade trata de conjuntos indutivos, no caso inteiros positivos. Isto demanda ser mais preciso na definição do n : assumir que o objetivo é provar P para todo inteiro positivo. Logo, P assume a seguinte definição:*

$$P = \lambda n. +i : i \in \mathbb{Z}^+ \wedge 1 \leq i < n : i = \frac{n(n-1)}{2}$$

O objetivo agora é provar:

$$\wedge n : n \in \mathbb{Z}^+ : P n$$

- *Provar que a nova propriedade vale para 1.*

$$\begin{aligned} P1 &= (+i : i \in \mathbb{Z}^+ \wedge 1 \leq i < 1 : i = \frac{1(1-1)}{2}) \\ &= \langle \text{Axioma Empty Range} \rangle \\ 0 &= \frac{1(1-1)}{2} \\ &= \langle \text{Aritmética} \rangle \\ 0 &= 0 \\ &= \langle (\wedge x : : x = x) \rangle \\ &True \end{aligned}$$

Observações importantes sobre o trecho de prova acima:

- Por transitividade da igualdade, o trecho mostra que $P1 = \text{True}$, o que diz em essência que $P1$ é verdadeiro.
- Cada passo de prova é um processo de reescrita baseado em algum axioma.
- A igualdade é usada como equivalência lógica: duas expressões com valores booleanos iguais são logicamente equivalentes.
- Provar que, quando Pk é verdadeiro, onde $k \in \mathbb{Z}^+$, então $P(k+1)$ também é verdadeiro. Reescrevendo $P(k+1)$:

$$P(k+1) = ((+i : i \in \mathbb{Z}^+ \wedge 1 \leq i < k+1 : i) = \frac{(k+1)(k+1-1)}{2})$$

Note que assumir Pk como verdadeiro, consiste em assumir

$$(+i : i \in \mathbb{Z}^+ \wedge 1 \leq i < k : i) = \frac{k(k-1)}{2}$$

$$\begin{aligned} & +i : i \in \mathbb{Z}^+ \wedge 1 \leq i < k+1 : i \\ = & \langle \text{Definição Indutiva de Somatória} \rangle \\ & +i : i \in \mathbb{Z}^+ \wedge 1 \leq i < k : i + k \\ = & \langle \text{Assumindo } Pk \rangle \\ & \frac{k(k-1)}{2} + k \\ = & \langle \text{Aritmética} \rangle \\ & \frac{k(k+1)}{2} \end{aligned}$$

Observações importantes sobre o trecho de prova acima:

- Neste trecho de prova, foi feita a opção de se reescrever o lado esquerdo da igualdade que $P(k+1)$ representa até encontrar o lado direito dessa mesma igualdade.
- Cada passo da prova foi novamente um passo de reescrever alguma fórmula usando um axioma ou uma hipótese previamente apresentados.
- A transitividade da igualdade novamente garante a consistência da prova.
- Você é livre para escolher seu estilo de prova, mas cada passo deve ser logicamente justificado.

□

O princípio de prova delineado acima e usado no Exemplo 1 é formalizado como *Princípio da Indução Matemática*.

Definição 1. *Princípio da Indução Matemática sobre \mathbb{N}*

$$(\wedge n : n \in \mathbb{N} : P n) = P 0 \wedge (\wedge k : k \in \mathbb{N} : P k \Rightarrow P(k+1))$$

Exercício 1. *Prove que*

$$(\wedge n 0 : n 0 \mathbb{N} : (\wedge n : n \in \mathbb{N} n 0 \leq n : P n) = P n_0 \wedge (\wedge k : k \in \mathbb{N} \wedge k \leq n_0 : P k \Rightarrow P(k+1)))$$

Exemplo 2. *Prove que*

$$(\wedge n : n \in \mathbb{N} \wedge n \geq 4 : n^2 \leq 2^n)$$

Prova.

$$\begin{aligned} & (\wedge n : n \in \mathbb{N} \wedge n \geq 4 : n^2 \leq 2^n) \\ & = \langle \text{Exercício 1} \rangle \\ & (4^2 \leq 2^4) \wedge (\wedge k : k \in \mathbb{N} \wedge k \geq 4 : k^2 \leq 2^k \Rightarrow (k+1)^2 \leq 2^{k+1}) \end{aligned}$$

Prova de: $4^2 \leq 2^4$

$$4^2 \leq 2^4$$

= $\langle \text{Aritmética} \rangle$

True

Prova de: $(\wedge k : k \in \mathbb{N} \wedge k \geq 4 : k^2 \leq 2^k \Rightarrow (k+1)^2 \leq 2^{k+1})$

$$\mathbf{HI:} \quad k_0 \in \mathbb{N}, k_0 \geq 4, P k = k_0^2 \leq 2^{k_0}$$

Prova de: $(k_0 + 1)^2 \leq 2^{k_0+1}$

$$(k_0 + 1)^2$$

= $\langle \text{Aritmética} \rangle$

$$k_0^2 + (2k_0 + 1)$$

$\leq \langle P k \rangle$

$$2^{k_0} + (2k_0 + 1)$$

$\leq \langle \text{when } k_0 \geq 3, 2k_0 + 1 \leq k_0^2 \rangle$

$$2^{k_0} + k_0^2$$

$\leq \langle P k \rangle$

$$2^{k_0} + 2^{k_0}$$

= $\langle \text{Aritmética} \rangle$

$$2^{k_0+1}$$

□

Exercício 2. *Prove por indução:*

- (1) $(+i : 1 \leq i \leq n : i) = n(n+1)/2$
- (2) $(+i : 0 \leq i < n : 2^i) = 2^n - 1$
- (3) $(+i : 0 \leq i < n : 3^i) = (3^n - 1)/2$
- (4) $(+i : 1 \leq i \leq n : i^2) = n(n+1)(2n+1)/6$
- (5) $(+i : 0 \leq i \leq n : i \cdot 2^i) = (n-1) \cdot 2^{n+1} + 2$
- (6) $(+i : 0 \leq i \leq n : (2i+1)^2) = (n+1)(2n+1)(2n+3)/3$
- (7) $(+i : 0 \leq i \leq n : i(i+1)(i+2)) = n(n+1)(n+2)(n+3)/4$
- (8) $(\wedge n, a : n \geq 1 \wedge a \neq 1 : (+i : 0 \leq i < n : a^i) = (1 - a^n)/(1 - a))$
- (9) $(\wedge n : n \geq 1 : (+i : 1 \leq i \leq n : 1/(i \cdot (i+1)))) = n/(n+1)$
- (10) $(\wedge n : n \geq 3 : n+1 < 2^n)$
- (11) $(\wedge n : n \geq 4 : n^2 \leq 2^n)$
- (12) $(\wedge n : n \geq 7 : 3^n < n!)$

1.2. Ouro de tolo? Já dá para conjecturar: o método de prova acima, o chamado *Princípio da Indução*, deve ser aplicável à prova de propriedades de qualquer conjunto contável. Então... Dado o conjunto de humanos, que nem mesmo é infinito, ficamos tentados a realizar aplicações mais interessantes.

Exercício 3. *Prove que todas as garotas loiras têm olhos claros.*

Para obter a prova acima, você pode:

- (1) Esquecer o Princípio da Indução e partir para o experimento. Que tal na sua turma de *Estruturas de Dados*? É claro que isso demanda definir:
 - i O que são garotas? *Ok! Isso pode não ser politicamente correto!*
 - ii O que são olhos claros?
- (2) Tentar Princípio da Indução, mas quem são Pk e $P(k+1)$? Aliás, esse é o ponto fundamental de toda a prova por indução: indentificar Pk , a *hipótese de indução*, e realizar o *passo indutivo* para obter $P(k+1)$.

Ok! De volta ao reino de \mathbb{N} ...

Veja o seguinte “teorema” e sua “prova”, retirados de [3].

Teorema Suspeito 1. *Dados inteiros positivos a e n , $a^{n-1} = 1$.*

Prova Suspeita.

1 Para $n = 1$, $a^{1-1} = a^0 = 1$.

2 Para $n = k + 1$,

$$a^{(k+1)-1} = a^k = a^{k-1} \cdot a = \frac{a^{k-1} \cdot a^{k-1}}{a^{(k-2)}} = \frac{1 \cdot 1}{1} = 1$$

O que está errado? Talvez o fato de que para se obter a prova para $n = k + 1$, foram usados como hipóteses o fato de que a assertiva seria válida para $n = k$ e $n = k - 1$? Não! Na verdade, como mostrado na próxima seção, esse é um raciocínio válido.

1.3. Indução para *Well-founded sets*. Considere a seguinte série, conhecida como série de Fibonacci¹.

Definição 2. *A série de Fibonacci é o conjunto indutivo dos F_i , definidos por:*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2}, \text{ para } n > 1 \end{aligned}$$

Exemplo 3. *Prove que*

$$(\wedge n : n \in \mathbb{N} : F_n < 2^n)$$

Prova.

$$\begin{aligned} &(\wedge n : n \in \mathbb{N} : F_n < 2^n) \\ &= \langle \text{Definição 1} \rangle \\ &F_0 < 2^0 \wedge (\wedge k : k \in \mathbb{N} : F_k < 2^k \Rightarrow F_{k+1} < 2^{k+1}) \end{aligned}$$

Prova de: $F_0 < 2^0$

$$\begin{aligned} &F_0 < 2_0 \\ &= \langle F \text{ (axioma 0) e Aritmética} \rangle \\ &0 < 1 \\ &= \\ &\text{True} \end{aligned}$$

Prova de: $(\wedge k : k \in \mathbb{N} : F_k < 2^k \Rightarrow F_{k+1} < 2^{k+1})$

HI: $k \in \mathbb{N}, P k = F_k < 2^k$

$$\begin{aligned} &F_{k+1} < 2^{k+1} \\ &= \langle F \text{ (axioma 2)} \rangle \\ &F_k + F_{k-1} \\ &< \langle P k \rangle \\ &2^k + F_{k-1} \\ &\langle \text{Nenhuma hipótese sobre } F_{k-1} \rangle \end{aligned}$$

¹Veja em [2] e [3] história, fatos e folclore em torno do números de Fibonacci.

O Exemplo 3 não pode ser completado porque não há hipóteses sobre o valor de F_{k-1} . A seguinte variante do princípio da indução poderia ajudar, se usada com cuidado.

Definição 3. Princípio da Indução Forte.

$$(\wedge n : n \in \mathbb{N} : P n) = (\wedge n : n \in \mathbb{N} : (\wedge k : k \in \mathbb{N} \wedge k < n : P k \Rightarrow P n))$$

O Exemplo 3 poderia agora ser completado desde que se Definição 3 para assumir que $F_{k-1} < 2^{k-1}$. Mas, seguindo o mesmo raciocínio, a seguinte prova também seria trivialmente obtida.

Exemplo 4. Use a Definição 3 e o raciocínio do Exemplo 3 para provar que todos os números da série de Fibonacci são pares.

Humm... Mas, 1, 3, 5 todos estão na série de Fibonacci e são ímpares. Bem, talvez sejam apenas uns poucos, uma minoria. Basta disfarçar e fingir que não estão lá. O teorema seria “aproximadamente válido.” Ou... “esquece tudo: essa tal de Ciência da Computação não funciona porque o fundamento matemático não é confiável.” Ou... O que está errado? A série de Fibonacci é um conjunto indutivo com dois *elementos iniciais*: $F_0 = 0$ e $F_1 = 1$. O chamado *caso base* da prova por indução deve demandar a prova de que a propriedade em questão vale para esses dois elementos. No Exemplo 4, no *caso base*, demanda-se provar que F_1 é par. A prova falha nesse ponto.

Considere o seguinte problema.²

Exercício 4. Para qualquer número $n \geq 8$, é possível obter a partir de uma soma de fatores todos iguais a 3 ou 5.

Exercício 5. Prove que os princípios da Definição 1 e Definição 3 são equivalentes

1.4. Classe indutiva.

Definição 4. \mathfrak{X} é uma classe indutiva se:

- (i) seus elementos são gerados a partir de uma das seguintes operações:
 1. existe uma especificação inicial \mathcal{B} de um conjunto de elementos que pertencem a \mathfrak{X} (chamados *elementos iniciais*);
 2. existe um conjunto \mathcal{M} de operações que aplicadas a elementos de \mathfrak{X} produzem novos elementos de \mathfrak{X} .
- (ii) Todos elemento de \mathfrak{X} pode ser obtido começando com um ou mais elementos de \mathcal{B} e aplicando a cada passo uma operação de \mathcal{M} a elementos obtidos em algum passo anterior ou obtidos de \mathcal{B} .

²Retirado das notas de aula de Tom Leighton, do curso de Mathematics for Computer Science, MIT, Setembro/97.

comum dizer que \mathcal{X} é a menor classe (ou o fecho universal da classe), cujos elementos iniciais são dados por \mathcal{B} e que tem novos elementos construídos por aplicações repetidas de \mathcal{M} .³

Exemplo 5. A série de Fibonacci é uma classe indutiva, cujos elementos iniciais são $F_0 = 0$ e $F_1 = 1$. Qualquer F_n , para $n > 1$, é obtido da soma de F_{n-1} e F_{n-2} .

O Princípio da Indução fica mais claro quando aplicado ao conceito de classe indutiva.

Definição 5. Uma propriedade P é válida para todos os elementos de uma classe indutiva \mathfrak{X} se:

- (i) P é válida para todos os elementos iniciais de \mathfrak{X} (P é válida para os elementos gerados por \mathcal{B}).
- (ii) Qualquer operação \mathcal{M} (que permite gerar elementos de \mathfrak{X} a partir de elementos já conhecidos) preserva a validade de P .

Exemplo 6. Uma propriedade é válida para classe da série de Fibonacci, introduzida na Definição 2, se ela for válida para seus elementos iniciais: 0 e 1. Logo, a propriedade de que todos os elementos da classe são pares, do Exemplo 4, não é válida.

Exercício 6. Reveja sua prova do Exemplo 3 à luz do conceito de classe indutiva.

Exercício 7. Seja $\phi = (1 + \sqrt{5})/2$. Prove que para $n \geq 1$

$$\phi^{n-2} \leq F_n \leq \phi^{n-1}$$

2. WELL-FOUNDED SETS

O princípio da Indução apresentado na Definição 3 pode ser generalizado para qualquer conjunto que possa ser caracterizado como *well founded*. Classes indutivas podem ser caracterizadas a partir do conceito de *well-foundedness*.

Definição 6. Elemento Minimal. Dado um par $\langle U, \prec \rangle$, y é um elemento minimal de S , denotado por $y \in \downarrow(U, \prec)$, se

$$y \in S \wedge (\wedge x : x \prec y : x \notin S : .)$$

Definição 7. Um conjunto U é dito well founded por uma relação \prec , denotado por $\mathbf{wff}(U, \prec)$, quando todo subconjunto não vazio de U tem um elemento minimal.

³Veja capítulos iniciais [1] sobre conceitos de classe indutiva, algoritmo, e questões definidas e semi-definidas.

Exercício 8. Defina uma relação de precedência para a série de Fibonacci e liste seus elementos minimais.

Definição 8. Indução Matemática sobre Well-Founded Sets. Dado um par $\langle U, \prec \rangle$, tal que $\mathbf{wff}(U, \prec)$, a seguinte equivalência define o princípio de prova por indução para propriedades sobre U :

$$\begin{aligned} & (\wedge x : x \in U : P x) \\ & = \\ & (\wedge x : x \in U : (\wedge y : y \in U \wedge y \prec x : P y) \Rightarrow P x) \end{aligned}$$

Note que a Definição 8 demanda provar $P x$ para todo $x \in U$, mas permite assumir como hipótese que P vale para todos os elementos que precedem x . Isso equivale dizer que essa definição demanda provar que $P k$ sempre que k é um elemento minimal em $\mathbf{wff}(U, \prec)$. Isso é explicitamente colocado na Definição 9.

Definição 9. Indução Matemática sobre Well-Founded Sets (alternativa). Dado um par $\langle U, \prec \rangle$, tal que $\mathbf{wff}(U, \prec)$, a seguinte equivalência define o princípio de prova por indução para propriedades sobre U :

$$\begin{aligned} & (\wedge x : x \in U : P x) \\ & = \\ & (\wedge y : y \in \downarrow(U, \prec) : P y) \\ & \wedge \\ & (\wedge x : x \in U \wedge x \notin \downarrow(U, \prec) : (\wedge y : y \in U \wedge y \prec x : P y) \Rightarrow P x) \end{aligned}$$

Exercício 9. Use a Definição 9 para provar que $F_n < 2^n$.

Teorema 1. Qualquer $M \subseteq \mathbb{N}$, tal que M não é vazio, tem um elemento mínimo.

Prova.

Assumindo que $M \subseteq \mathbb{N}$ e que M não tem elemento mínimo, chega-se à contradição de que M é vazio. Por Princípio da Indução Forte, prova-se que $\forall n : n \in \mathbb{N} : n \notin M$.

(1) Para $n = 0$, $n \notin M$ ou n seria mínimo, pois $M \subseteq \mathbb{N}$ e 0 é mínimo em \mathbb{N} .

(2) Assumindo-se que $\forall k \in \mathbb{N} : \leq 0kn : k \notin M$, conclui-se que $k+1 \notin M$, ou $k+1$ seria mínimo de M .

Logo M é vazio, ou M tem mínimo. □

Exercício 10. Construa a prova do Exercício 7 usando a Definição 9.

3. INDUÇÃO E COMPUTAÇÃO

Tipos de dados são representações em alguma linguagem de programação de classes indutivas. Na verdade, qualquer computação pode ser vista como a atividade de enumerar os elementos de uma class indutiva.

Exercício 11. *Construa um argumento a favor de ou contra a seguinte afirmação:*

Só é computavel o que pode ser definido indutivamente.

Exercício 12. *Construa um argumento a favor ou contra de: Nem todas as funções são computáveis*

Exemplo 7. *Considere a classe **Natnum** como a menor classe que contém:*

- (i) *Z, como único elemento inicial;*
- (ii) *(Sx), quando x já pertence a **Natnum**.*

3.1. Admissibilidade de funções. Uma representação em Haskell para a classe **Natnum** seria:

```
data Natnum = Z | (S Natnum)
  deriving (Eq)
```

Essa definição contém três elementos importantes:

- (1) Ela introduz *Z* como construtor do único objeto inicial (elemento minimal) da classe.
- (2) Ela introduz um operador *S* para construir novos elementos da classe a partir de elementos já existentes.
- (3) Através da cláusula **deriving (Eq)**, ela introduz extensionalidade que estabelece:

$$Z = Z$$

$$(\wedge n, m : n, m \in \mathbf{Natnum} : (S(n) = S(m)) = (m = n))$$

Um exemplo de função para adição de elementos dessa classe seria:

```
itsum m Z = m
itsum m (S n) = itsum (S m) n
```

É importante entender os parâmetros dessa definição como variáveis universalmente quantificadas. Dessa forma, a definição de **itsum** acima

estaria estabelecendo que:

$$(\wedge m : m \in \mathbf{Natnum} : itsum\ m\ Z = m)$$

$$(\wedge m, n : m, n \in \mathbf{Natnum} : itsum\ m\ (Sn) = itsum\ (Sm)\ n)$$

A função **itsum** acima definida permite introduzir dois conceitos importantes:

1. *Admissibilidade da função:* a função acima será dita admissível porque existe uma medida de complexidade dos seus argumentos que é sempre decrescente em qualquer chamada recursiva da mesma. Nessa função, o segundo argumento da chamada recursiva é sempre mais simples que o segundo argumento da chamada que a precede.

Por exemplo, assumamos um mapeamento $f : \mathbf{Natnum} \rightarrow \mathbb{N}$ que associa:

$$f(m, Z) = 0$$

$$f(m, (Sn)) = f\ n + 1$$

Claramente f é uma medida de complexidade para os argumentos de chamada de **itsum** que decresce a cada chamada recursiva, formando uma cadeia finita decrescente, com mínimo em (m, Z) .

2. Diferentemente da função **recsum** abaixo, **itsum** pode ser avaliada por um processo de repetição substituições simples da chamada original pela chamada recursiva.

$$\text{recsum } m\ Z = m$$

$$\text{recsum } m\ (S\ n) = S(\text{recsum } m\ n)$$

A computação de **recsum** demanda que uma memória seja mantida das sucessivas chamadas recursivas para os **S** sejam adicionados ao resultado final da chamada não recursiva.

A função **itsub** introduzida abaixo estabelece a diferença absoluta de dois elementos da class **Natnum**.

$$\text{itsub } m\ Z = m$$

$$\text{itsub } Z\ n = n$$

$$\text{itsub } (S\ m)\ (S\ n) = \text{itsub } m\ n$$

Exercício 13. Prove que **itsub** sempre termina.

Exercício 14. Prove:

$$(\wedge m, n : m, n \in \mathbf{Natnum} : itsub\ m\ n = itsub\ n\ m)$$

Exercício 15. Defina uma função **pred**, que obtém o predecessor de um **Natnum**, e prove que

$$\text{itsub } m (S n) = \text{pred}(\text{itsub } m n)$$

Use indução sobre n (segundo argumento do (m, n) da chamada recursiva.) No passo indutiva, considere dois casos: quando m é **Z** e quando m é $(S u)$ para algum u .

3.2. Generalizando indução. É importante observar que a medida de complexidade de argumentos de uma função não precisa se restringir a um único argumento. Através do conceito de *well-foundedness* (ver em [2] e também [4], página 20, exercício 15), podemos generalizar todo o conceito de indução matemática e de admissibilidade de funções se, dado um conjunto S (de inteiros, **Natnum**, tuplas, etc), pudermos estabelecer um relação \prec tal que:

- toda a cadeia em S tem um mínimo x : para todo $y \in S$ existe um mínimo x tal que $x \prec x_1 \prec x_2 \prec \dots \prec y$.
- Se, para $x, y, z \in S$, $x \prec y$ e $y \prec z$, então $x \prec z$.

Uma função recursiva é admissível se os argumentos das suas chamadas recursivas admitem algum relação de precedência que determina um *well-founded set*

Exemplo 8.

$$\begin{aligned} \text{gcd } m n &= \text{if } m < n \\ &\quad \text{then gcd } m (n - m) \\ &\quad \text{else if } m > n \\ &\quad \text{then gcd } (m - n) n \\ &\quad \text{else } m \end{aligned}$$

A relação abaixo

$$\begin{aligned} (m_0, n_0) &\prec (m_1, n_1) \text{ quando } m_0 < m_1 \\ (m_0, n_0) &\prec (m_1, n_1) \text{ quando } (m_0 = m_1) \wedge (n_0 < n_1) \end{aligned}$$

e o fato de que nas chamadas recursivas de **gcd** os argumentos de chamada são sempre maiores ou iguais do zero pode ser usada para provar que a definição de **gcd** é admissível (sempre termina.)

Por outro lado, usando indução forte, a relação \prec acima e o fato de que o maior divisor de m e n é também o maior divisor de m e $m - n$ (quando m é maior do que n), permite provar que a função acima calcula corretamente o maior divisor comum de dois inteiros positivos.

Exercício 16. *Prove que $\text{gcd}(m, n)$ sempre termina e que calcula corretamente o maior divisor comum de m e n .*

3.3. Tipos abstratos de dados.

Exercício 17. *Crie em Haskell, ao menos, as seguintes funções sobre o tipo **Natnum**. Defina e prove condições de correção para as mesmas:*

- (1) *Funções para cálculo de:*
 - soma de dois objetos;
 - diferença de dois objetos;
 - produto de dois objetos;
 - divisão de dois objetos;
 - resto da divisão inteira de dois objetos.
- (2) *Funções implementar as relações lógicas, apresentando condições de correção:*
 - (a) *igualdade;*
 - (b) *desigualdade;*
 - (c) *menor que;*
 - (d) *maior que;*

Exercício 18. *Dadas as definições de multiplicação, divisão e resto da divisão, prove o teorema correspondente a $m = q * n + r$, quando q é quociente da divisão de m por n e r é o resto dessa divisão.*

REFERÊNCIAS

- [1] Haskell B. Curry, *Foundations of mathematical logic*, Dover Publications, Inc., 1977.
- [2] David Gries and Fred B. Schneider, *A logical approach to discrete mathematics*, Springer-Verlag, 1993.
- [3] Donald E. Knuth, *Fundamental algorithms*, third ed., The Art of Computer Programming, vol. I, Addison-Wesley, 1998.
- [4] ———, *Sorting and searching*, The Art of Computer Programming, vol. III, Addison-Wesley, 1998.
- [5] Tom M. Apostol, *Calculus: One-variable calculus, with an introduction to linear algebra*, second edition ed., vol. I, John Wiley & Sons, Inc., 1967.