Introdução à Informática

Modelos abstratos: Computabilidade

Definição:

O que pode ser computado por meio de um processo automático.

Argumento:

Se um problema pode ser resolvido, por meio de uma série de operações em um algoritmo, ele pode ser computado por um processo automático.

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)

    Modelo determinístico computacional abstrato formado por um número inteiro positivo de regras de deleção (m), um alfabeto finito de símbolos (A) e uma regra de produção (P).
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

Post Tag-machine:

```
2 P(H)                          // tag system
   A : {a,b,c,H}                // alfabeto
         a -> ccbaH
         b -> cca
         c -> cc

     baa                        // inicio da computação
        acca
           caccbaH
              ccbaHcc
                 baHcccc
                    Hccccca     // Pára
```

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos

    Regras de substituição:

    A -> cerveja; B -> amigo; C -> no bar; no bar -> na Nina

    Fui tomar A, com os Bs, C.
    Fui tomar cerveja, com os Bs, C.
    Fui tomar cerveja, com os amigos, C.
    Fui tomar cerveja, com os amigos, no bar.
    Fui tomar cerveja, com os amigos, na Nina.
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade

    Aplicação repetida e **finita** de uma função em si mesma.

    `f(n) = n (OP) f(n-1)`

  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Entre 1930-1940: vários estudos sobre o problema da computabilidade.
  - Emil Post: Sistema de produção (Tag System)
  - Markov: Algoritmos
  - Kleene: Recursividade
  - Schönfinkel - Curry: Lógica Combinatória

    Elementos combinatórios suficientes para a declaração de predicados arbitrários de primeira ordem que podem ser expressos sem o uso de variáveis dependentes.
  - Church: Cálculo lambda
  - Turing: Máquinas Computacionais Lógicas (LCM)

Introdução à Informática

Modelos abstratos: Computabilidade

- Cálculo lambda:
  - Teoria baseada em funções computáveis.
  - Modelo matemático de tradução de algoritmos específicos em máquinas que possam calculá-los.
  - Conjunto de operadores, operandos e aplicações.
  - argumentos, abstrações (dos argumentos em funções) e aplicações (de operandores sobre operandos).
  - Regras de substituição (redução).

Introdução à Informática

Modelos abstratos: Computabilidade

Cálculo Lambda:

Na matemática tradicional podemos formular a abstração como:

$$f(x) = x^2 + xy + z$$

para qualquer aplicação em f(x) resulta na substituição de x pela aplicação.

No cálculo lambda escrevemos:

$$\lambda x.x^2 + xy + z$$

Introdução à Informática

Modelos abstratos: Computabilidade

Cálculo Lambda:

Exemplo: suponha que, na abstração:

$$\lambda x.x^2 + xy + z$$

apliquemos y, ou seja, cada ocorrência livre de x deve ser substituída por y:

$$\lambda x.(x^2 + xy + z)\, y$$

então a expressão se reduz a:

$$2y^2 + z$$

Introdução à Informática

Modelos abstratos: Computabilidade

Cálculo Lambda:

Se

$$f(x)=(x^2+4)\equiv\lambda x.x^2+4$$

a aplicação de f(2) pode ser escrita como:

$$\lambda x.(x^2+4)(2)$$

que se reduz pela regra de substituição de toda ocorrência livre de x por 2:

$$(2^2+4)\Rightarrow 8$$

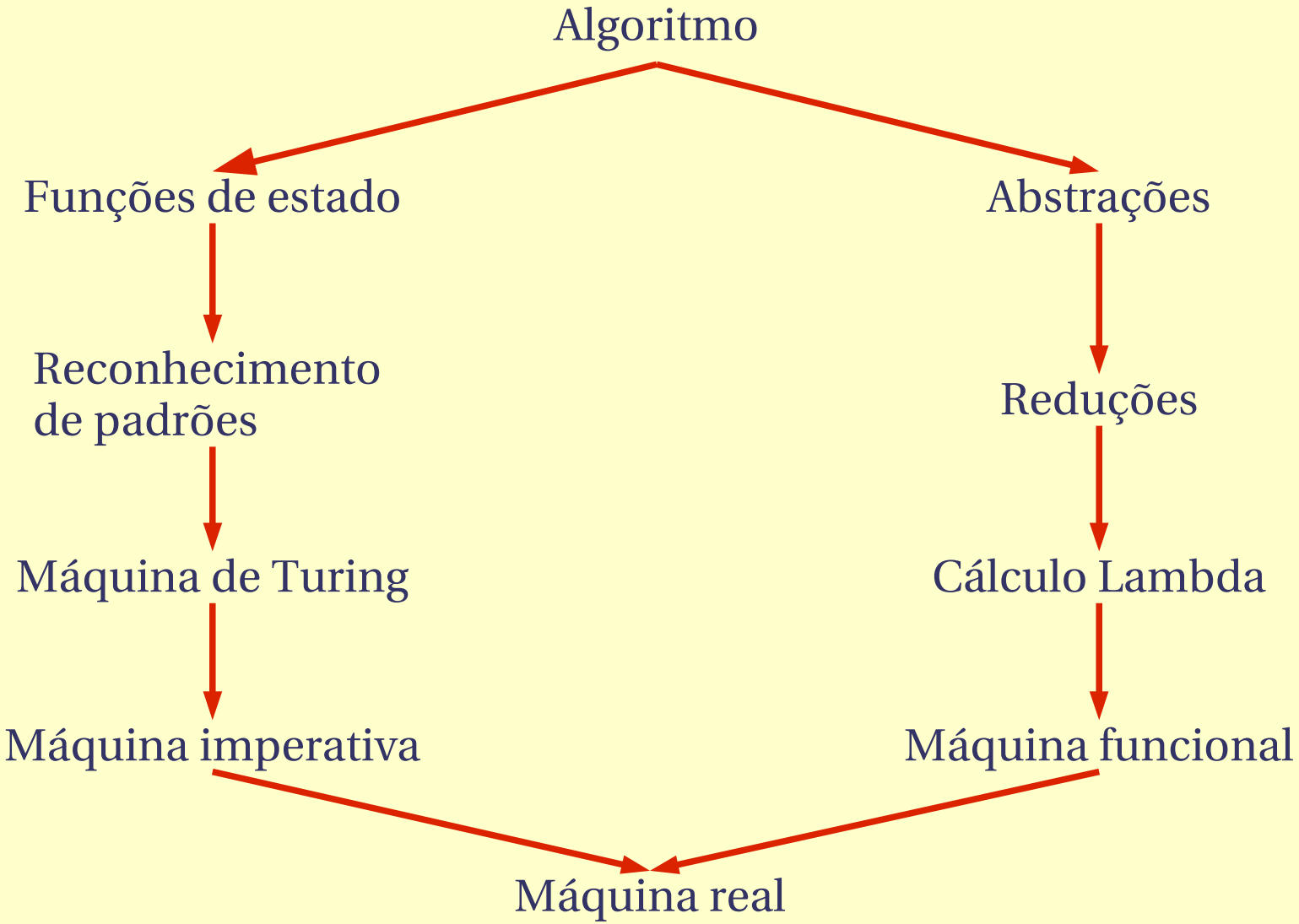
Introdução à Informática

Modelos abstratos: Computabilidade

- Máquina de Turing:
  - Modelo de nível elementar e ideal.
  - Imita o comportamento do computador.
  - Fita com alfabeto finito
  - Processador primitivo com movimento bidirecional e capacidade de leitura e escrita.
  - Um número de estados finitos, previamente descritos.
  - Capaz de computar qualquer problema que pode ser descrito por um algoritmo.

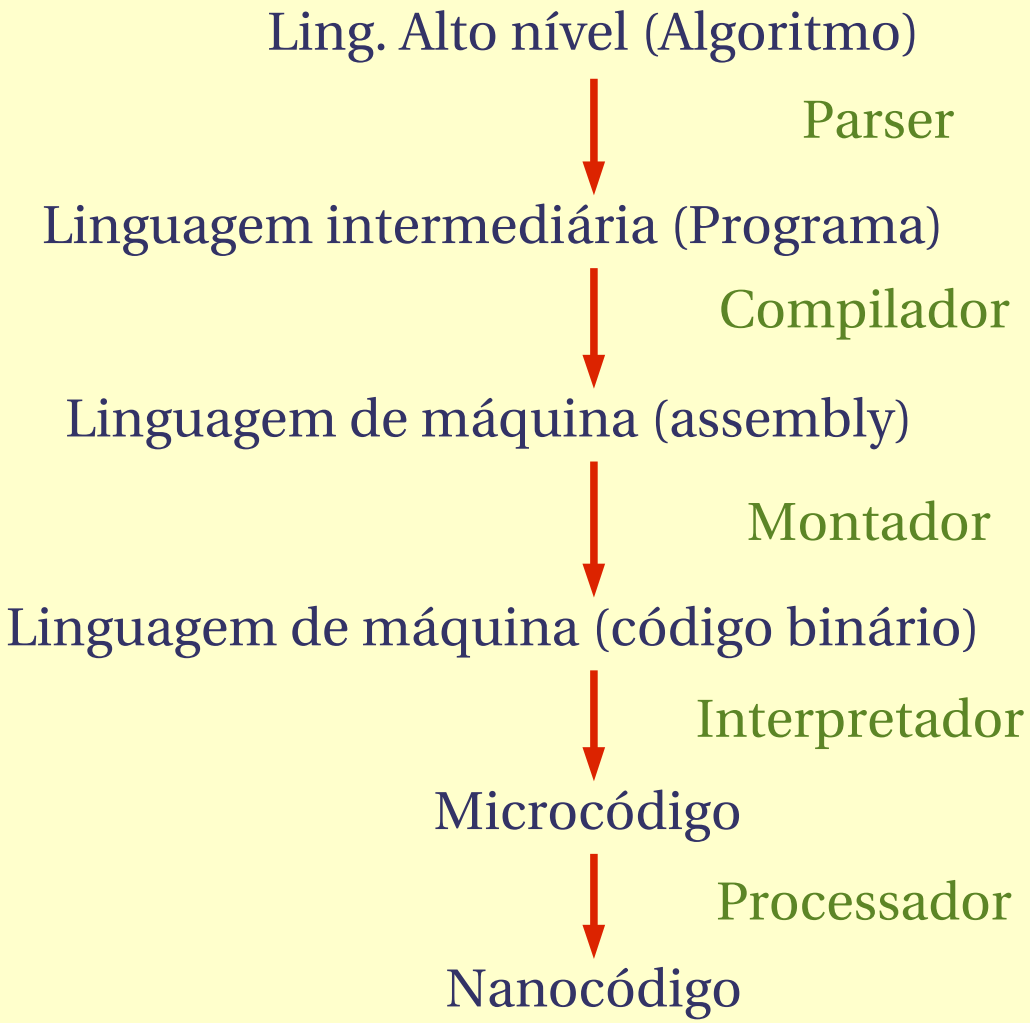
Introdução à Informática

Modelos abstratos: Computabilidade (nível lógico)



Introdução à Informática

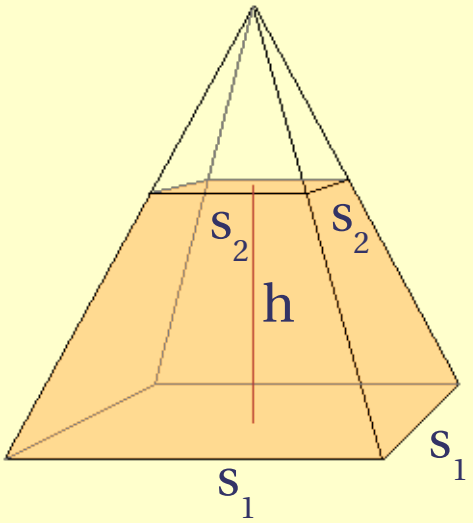
Modelos abstratos: Computabilidade (nível físico)



Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)

Considere um tronco de pirâmide (frusto):



Seu volume pode ser calculado por:

$$V_t = \frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2))$$

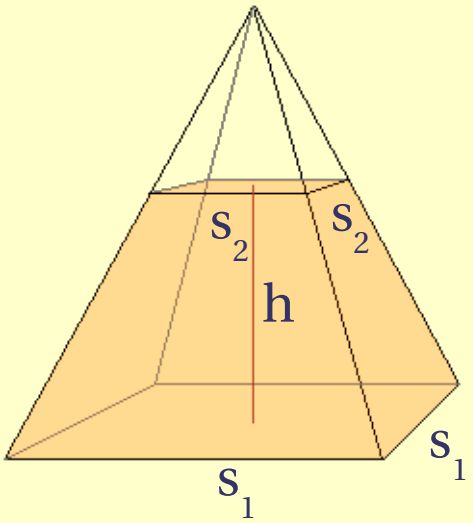
Desde que satisfaça as seguintes condições:

$$(h, s_1, s_2 \in \Re^+) \wedge (s_1 > s_2)$$

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)

Considere um tronco de pirâmide (frusto):



Seu volume pode ser calculado por:

$$V_t = \frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2))$$

Desde que satisfaça as seguintes condições:

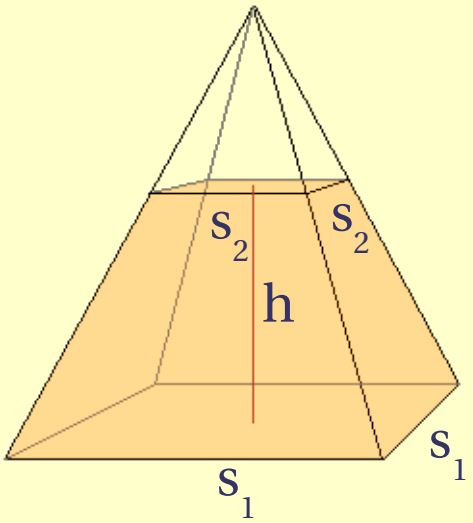
$$(h, s_1, s_2 \in \Re^+) \wedge (s_1 > s_2)$$

V_t descreve um algoritmo para o cálculo do volume do frusto. Ele deve receber, como argumentos, os valores de h, s_1 e s_2, desde que respeitadas as condições.

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)

Considere um tronco de pirâmide (frusto):



Seu volume pode ser calculado por:

$$V_t = \frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2))$$

Desde que satisfaça as seguintes condições:

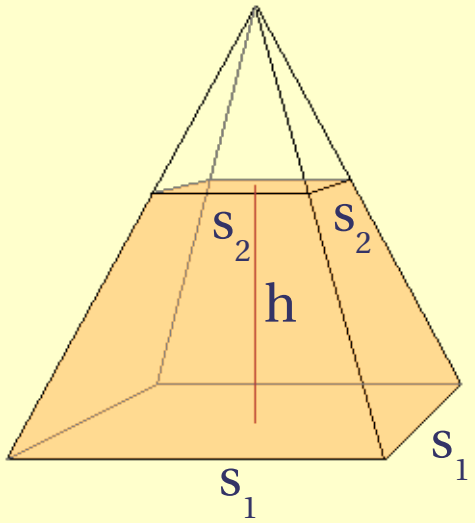
$$(h, s_1, s_2 \in \Re^+) \wedge (s_1 > s_2)$$

V_t é, então, uma função de 3 argumentos:

$$V_t = f(h, s_1, s_2) = (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)



$$V_t = f(h, s_1, s_2) = (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

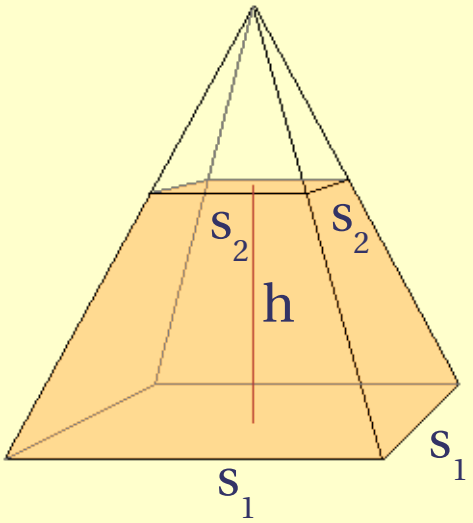
Podemos, então, criar um construtor (uma abstração) que vai associar os argumentos à expressão.

$$f(h, s_1, s_2) \equiv \lambda\, h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)

Considere um tronco de pirâmide (frusto):



$$V_t = f(h, s_1, s_2) = (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Podemos, então, criar um construtor (uma abstração) que vai associar os argumentos à expressão.

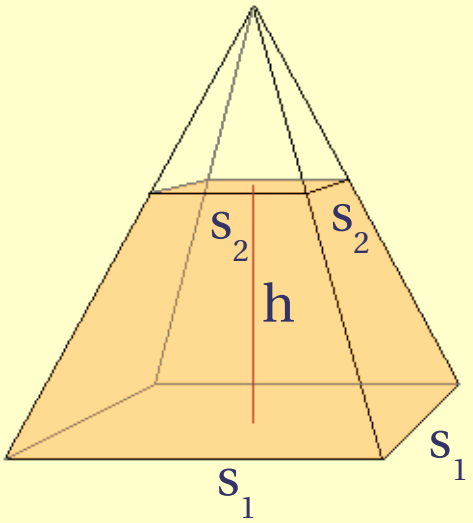
$$f(h, s_1, s_2) \equiv (\lambda\ h\ s_1\ s_2\ (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

abstração lambda

expressão

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)



$$V_t = f(h, s_1, s_2) = (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Podemos, então, criar um construtor (uma abstração) que vai associar os argumentos à expressão.

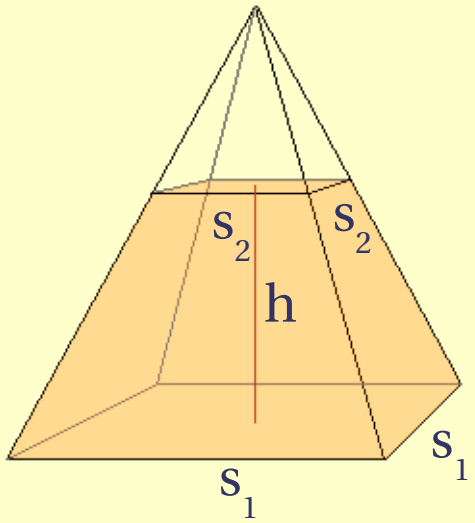
$$f(h, s_1, s_2) \equiv \lambda\ h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Uma aplicação atribuirá valores aos argumentos, ou seja, para cada ocorrência livre de h, s1 e s2 na expressão, ocorrerá uma substituição pelos valores da aplicação, por exemplo:

$$\lambda\ h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))(a, b > c \in \Re^+)$$

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)



$$V_t = f(h, s_1, s_2) = (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

Podemos, então, criar um construtor (uma abstração) que vai associar os argumentos à expressão.

$$f(h, s_1, s_2) \equiv \lambda\, h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))$$

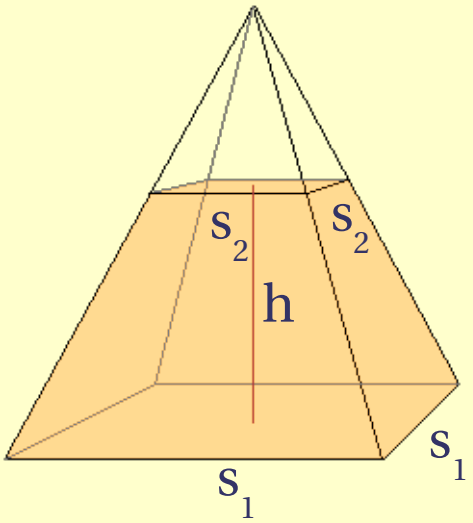
Uma aplicação atribuirá valores aos argumentos, ou seja, para cada ocorrência livre de h, s1 e s2 na expressão, ocorrerá uma substituição pelos valores da aplicação, por exemplo:

$$\lambda\, h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))(a, b > c \in \Re^+)$$

aplicação

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)



$$\lambda\, h\ s_1\ s_2 . (\frac{3}{4} h((s_1 + s_2)^2 - (s_1 \cdot s_2)))(a, b > c \in \Re^+)$$

O que significa que, sendo a,b e c números reais positivos, sendo b > c, então todo h deve ser substituído por a, todo s_1 deve ser substituído por b e todo s_2 deve ser substituído por c.

É evidente que, a,b e c devem ser valores numéricos para que o resultado possa ser calculado.

Introdução à Informática

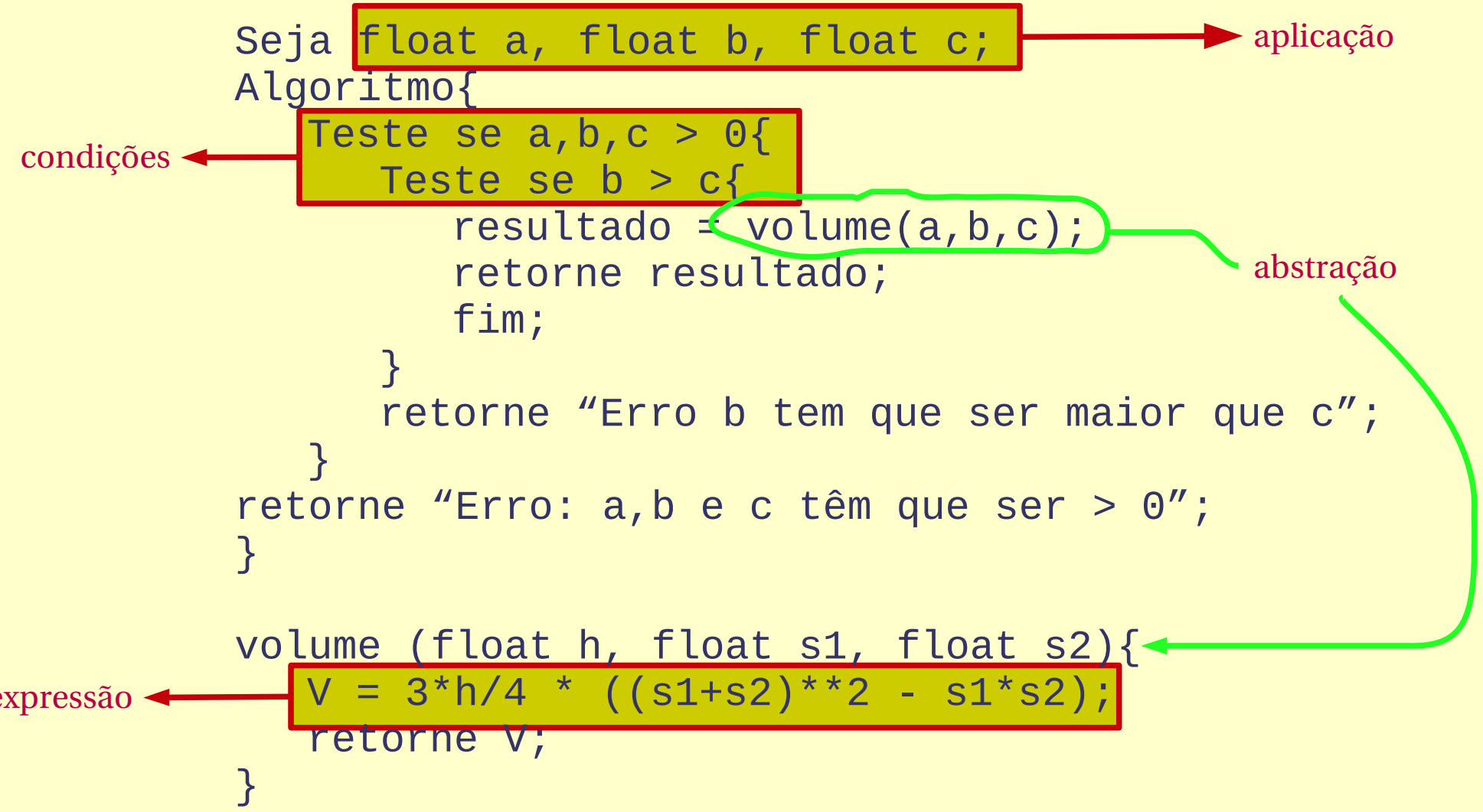
Modelos abstratos: Computabilidade (Exemplo)

```
Seja float a, float b, float c;
Algoritmo{
   Teste se a,b,c > 0{
      Teste se b > c{
         resultado = volume(a,b,c);
         retorne resultado;
         fim;
      }
      retorne “Erro b tem que ser maior que c”;
   }
retorne “Erro: a,b e c têm que ser > 0”;
}

volume (float h, float s1, float s2){
   V = 3*h/4 * ((s1+s2)**2 - s1*s2);
   retorne V;
}
```

Introdução à Informática

Modelos abstratos: Computabilidade (Exemplo)



Introdução à Informática

Modelos abstratos: Computabilidade (Ex: C)

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

float v_frusto (float,float, float);
float a=3.0, b=2.0, c=1.0;

int main(){
    float resultado;
    if ((a>0 && b>0 && c>0) && (b>c)){
        resultado = v_frusto(a,b,c);
        printf("Volume = %f\n", resultado);
        exit(0);
    }
        printf("Erro: a,b,c têm que ser >0 e b maior que c\n");
        exit(1);
}

float v_frusto (float h, float s1, float s2){
    float V;
    V = 3.0*h/4.0 * (pow((s1+s2),2) - s1*s2);
    return V;
}
```

Introdução à Informática

Modelos abstratos: Computabilidade (Ex: Haskell)

```
module Frusto (volFrusto) where

volFrusto :: (Ord a, Fractional a) => a -> a -> a -> a
volFrusto h s1 s2
       | condic h s1 s2 == True  = 3/4 * h * ((s1+s2)^2 - s1*s2)
       | otherwise               = -1.0

condic :: (Num a, Ord a, Num a1, Ord a1) => a -> a1 -> a1 -> Bool
condic x y z = if(x<=0 || y<=0 || z<=0 || y<z)then False else True
```

Introdução à Informática

Modelos abstratos: Recursividade

Uma função definida pela aplicação repetida da própria função.

Exemplo:

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

que pode ser definido como:

$$f(n) = n! = n \times f(n-1), \quad n > 0$$

Introdução à Informática

Modelos abstratos: Recursividade

A forma:

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

representa um algoritmo não recursivo e a forma:

$$f(n) = n! = n \times f(n-1), \quad n > 0$$

representa um algoritmo recursivo.

Introdução à Informática

Modelos abstratos: Recursividade

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

```
f := 1;
Para i=1 até i= n;
   f := f * I;
   i := i + 1;
   retorne f;
   fim;
```

Introdução à Informática

Modelos abstratos: Recursividade

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

```
/* fatorial.c (forma não recursiva) */
#include <stdlib.h>
#include <stdio.h>

int main(){
   int i, f=1, n=10;
   for (i=1;i<=n;i++){
      f = f * i;
   }
   printf("Fatorial de %d = %d\n", n, f);
   exit(0);
}
```

Introdução à Informática

Modelos abstratos: Recursividade

$$f(n) = n! = n \times f(n-1),\ n>0$$

```
/* fat_recurs.c (forma recursiva) */
#include <stdlib.h>
#include <stdio.h>

int fat (int);

int main(){
   int f, x=10;
   f = fat(x);
   printf("Fatorial de %d = %d\n", x, f);
   exit(0);
}

int fat(int n){
   if(n==0) return 1;
   else return n* fat(n-1);
}
```

Introdução à Informática

Modelos abstratos: Recursividade

$$f(n) = n! = n \times f(n-1),\ n>0$$

```
-- fatorial.hs (Forma recursiva funcional)

module Main (print fatorial 10) where

fatorial :: Int -> Int
fatorial 0 = 1
fatorial n = n * fatorial (n-1)
```