Expressões Condicionais

Cap. 4

Expressões condicionais

- Em diversas situações, a ação a ser realizada depende de algo:
  - se estiver chovendo, leve o guarda-chuva
  - se o sinal estiver vermelho, pare
  - se o bolo estiver pronto, retire do forno
  - …

Expressões condicionais

- Em diversas situações, a ação a ser realizada depende de algo:
  - se estiver chovendo, leve o guarda-chuva
  - se o sinal estiver vermelho, pare
  - se o bolo estiver pronto, retire do forno
  - …

condições

O que é uma condição?

É uma expressão cujo valor é

- Verdadeiro, ou
- Falso

Booleano = {Verdadeiro, Falso}

Em Racket: **Booleano = { #true, #false }**
{ true, false }
{ #t, #f }

Expressões Booleanas

1 < 2 — verdadeiro

3 = 4 — falso

3 > 4 — falso

x > 1 — ???

Expressões Booleanas

(< 1 2) → true

< : Número Número → Booleano

(= 3 4) → false

= : Número Número → Booleano

(> 3 4) → false

> : Número Número → Booleano

(> x 1) → ???

Booleano = {true, false}

Operações Booleanas

Booleano = {true, false}

not : Booleano → Booleano

(not true) = false
(not false) = true

and : Booleano Booleano → Booleano

(and true true) = true
(and true false) = false
(and false true) = false
(and false false) = false

or : Booleano Booleano → Booleano

(or true true) = true
(or true false) = true
(or false true) = true
(or false false) = false

Exemplos

Se tem sol eu vou na praça.

C1

Se não tem sol eu vou na praça.

(not C1)

Se não tem sol e tem pipoca eu vou assistir TV.

(and (not C1) C2)

Se tem chocolate ou tem pipoca eu vou assistir TV.

(or C3 C2)

Exemplos

(and (= 1 2) (< 2 3))

false

(or (= 1 2) (< 2 3))

true

(not (< 2 3))

false

(not (< x 3))

???

(and (= x y) (< x y))

false

(or (= x y) (< x y) (> x y))

true

Em Racket...

`true`	*verdadeiro*
`false`	*falso*
`(= x y)`	*x é igual a y*
`(< x y)`	*x é menor do que y*
`(> x y)`	*x é maior do que y*
`(<= x y)`	*x é menor ou igual a y*
`(>= x y)`	*x é maior ou igual a y*
`(not expressão)`	*negação de expressão*
`(and exp_1 ... exp_n)`	*todas expressões (1 a n) verdadeiras*
`(or exp_1 ... exp_n)`	*uma expressão (1 a n) verdadeira*

Exercícios

- **Exercício 4.1.1.** Qual o resultado das seguintes condições em Racket?

  1. `(and (> 4 3) (<= 10 100))`
  2. `(or (> 4 3) (<= 10 100))`
  3. `(not (= 2 3))`

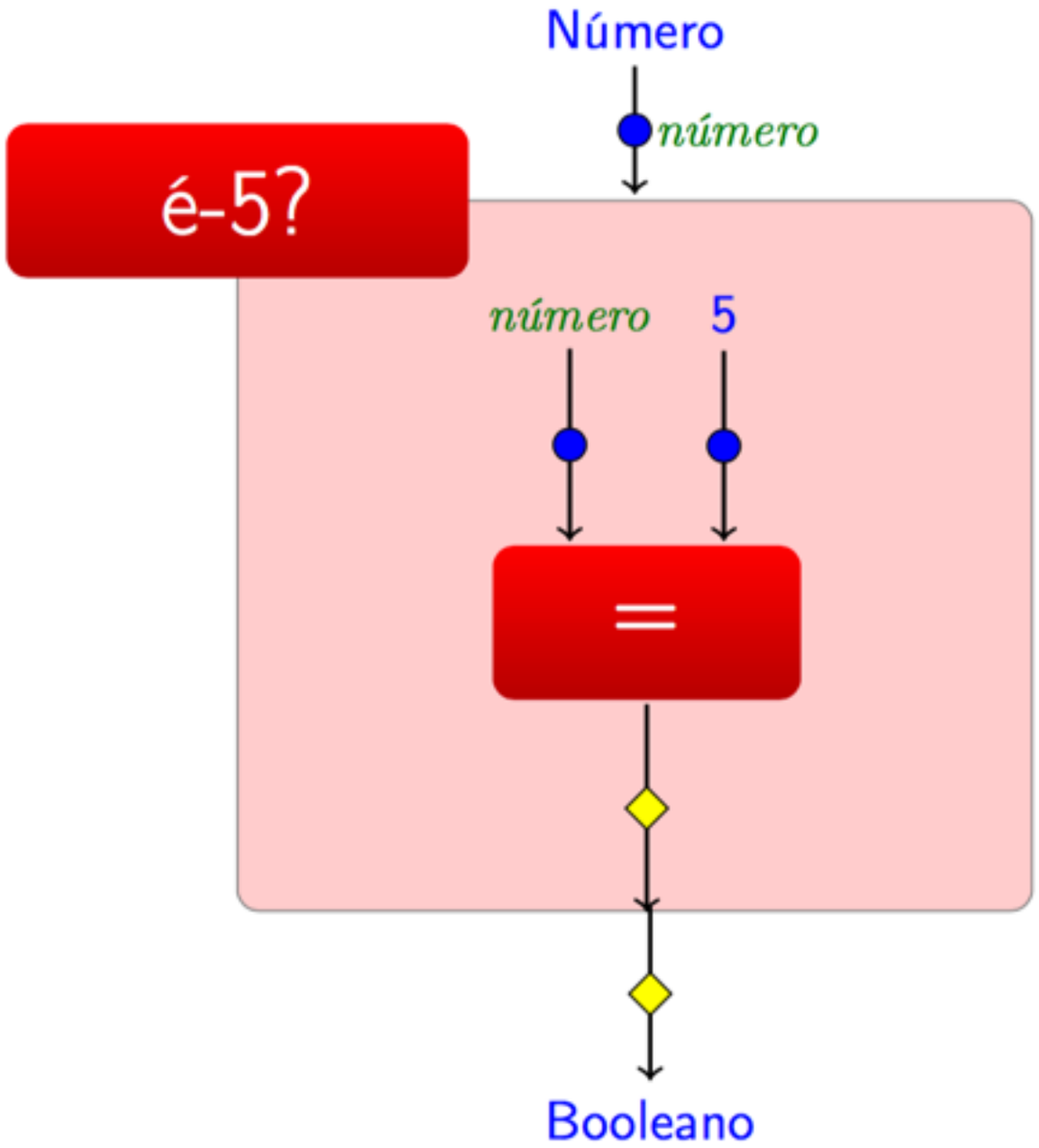
- **Exercício 4.1.2.** Qual o resultado de

  1. `(> 4 3)`
  2. `(and (> 4 x) (> x 3))`
  3. `(= (* x x) x)`

  para: (a) $x = 4$, (b) $x = 2$, (c) $x = \frac{7}{2}$.

Exemplo

- Função que testa se a entrada é 5

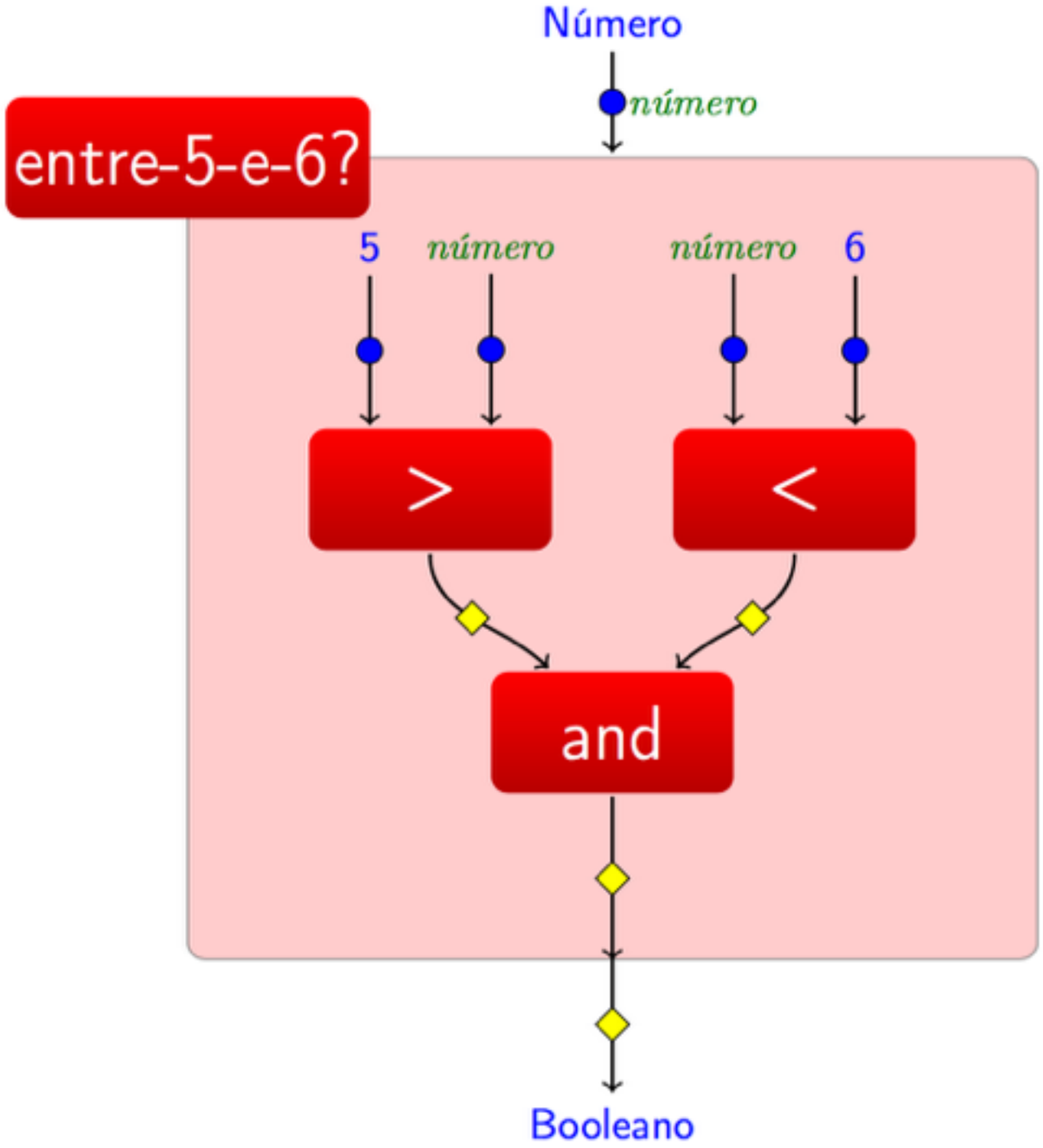


é-5 : Número → Booleano

```
;; Obj: Dado um número, verifica se ele é 5
;; Exemplos: …

(define (é-5? número)
        (=
           número
           5
        )
 )
;;Testes: …
```

Exemplo

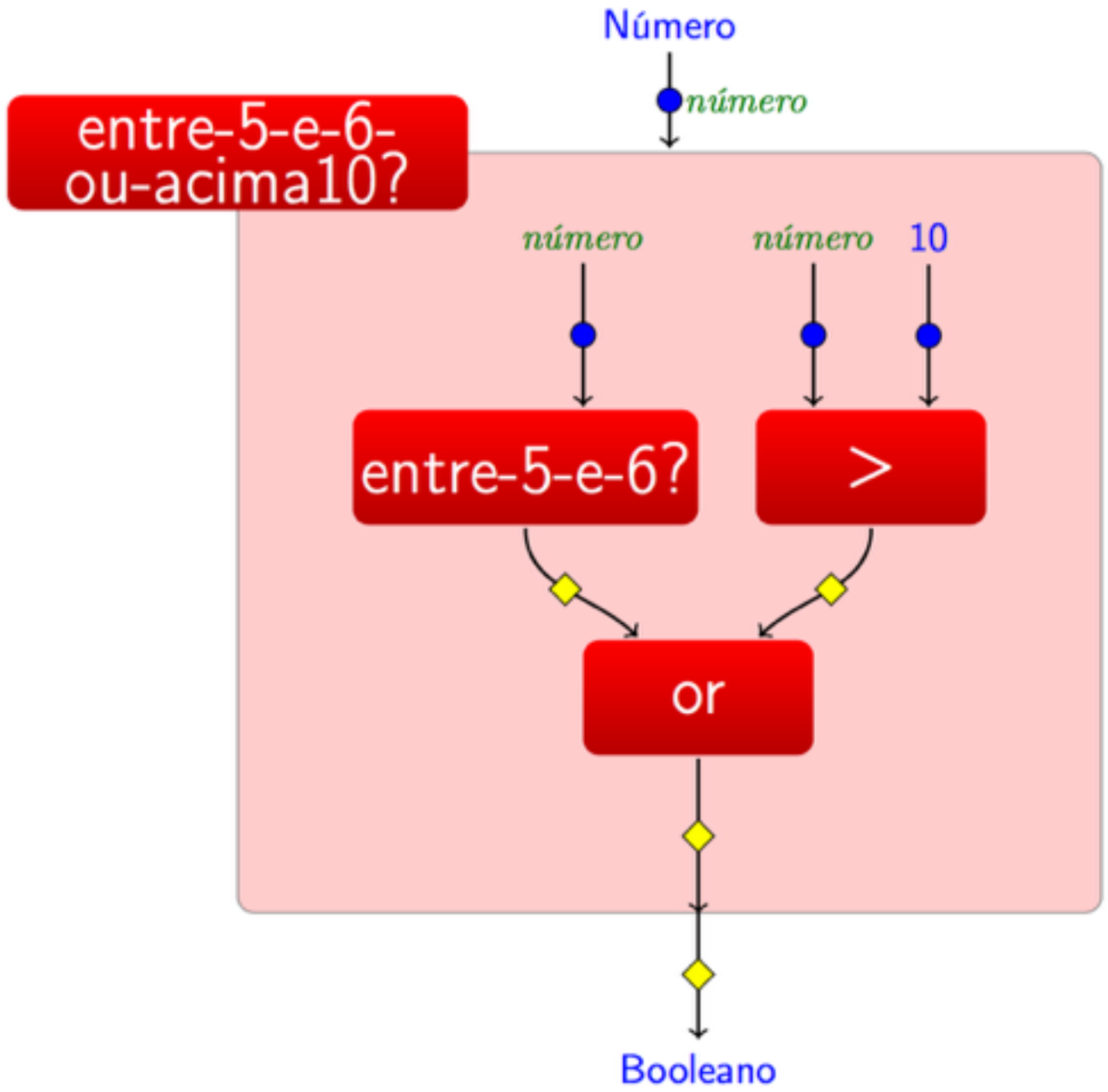
- Função que testa se a entrada está entre 5 e 6 (exclusive)




```
entre-5-e-6? : Número → Booleano
;; Obj: Dado um número, verifica se ele
        está entre 5 e 6 (exclusive)
;; Exemplos: …

(define (entre-5-e-6? número)
        (and
            (< 5 número)
            (< número 6)
        )
)
;; Testes: …
```

Exemplo

- Função que testa se a entrada está entre 5 e 6 (exclusive) ou é maior que 10



entre-5-6-ou-acima10? : Número → Booleano

```
;; Obj: Dado um número, verifica se ele está
  entre 5 e 6 (exclusive) ou é maior que 10
;; Exemplos: …

(define (entre-5-6-ou-acima10? número)
        (or
           (entre-5-6? número)
           (> número 10)
        )
)

;; Testes: …
```

Exercícios

- **Exercício 4.2.1.** Traduza os intervalos abaixo para funções em Racket que consomem (aceitam) um número e produzem `true` se o número está no intervalo e `false`, caso contrário.

  1. $(3, 7]$
  2. $[3, 7]$
  3. $[3, 9)$
  4. União de $(1, 6)$ e $(9, 14)$
  5. Na parte de fora de $[1, 3]$

Exercícios

- **Exercício 4.2.2.**

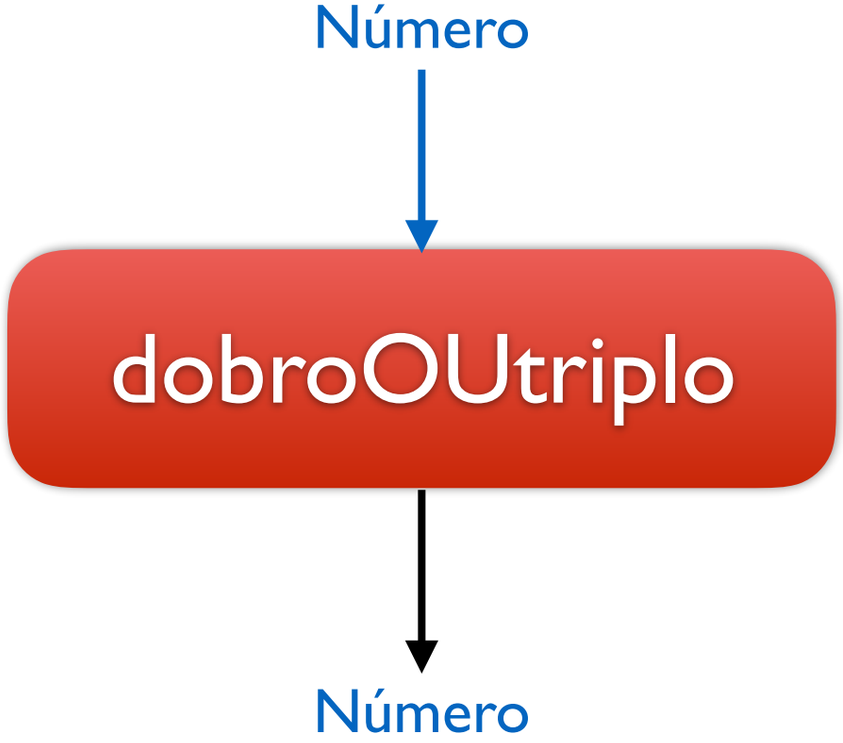
(a) Traduza as funções abaixo para intervalos:

```
(define (no-intervalo-1? x)
    (and (< -3 x) (< x 0)))

(define (no-intervalo-2? x)
    (or (< x 1) (> x 2)))

(define (no-intervalo-3? x)
    (not (and (<= 1 x) (<= x 5))))
```

(b) Escreva os contratos e objetivos para cada uma das funções acima.

Exemplo

- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.

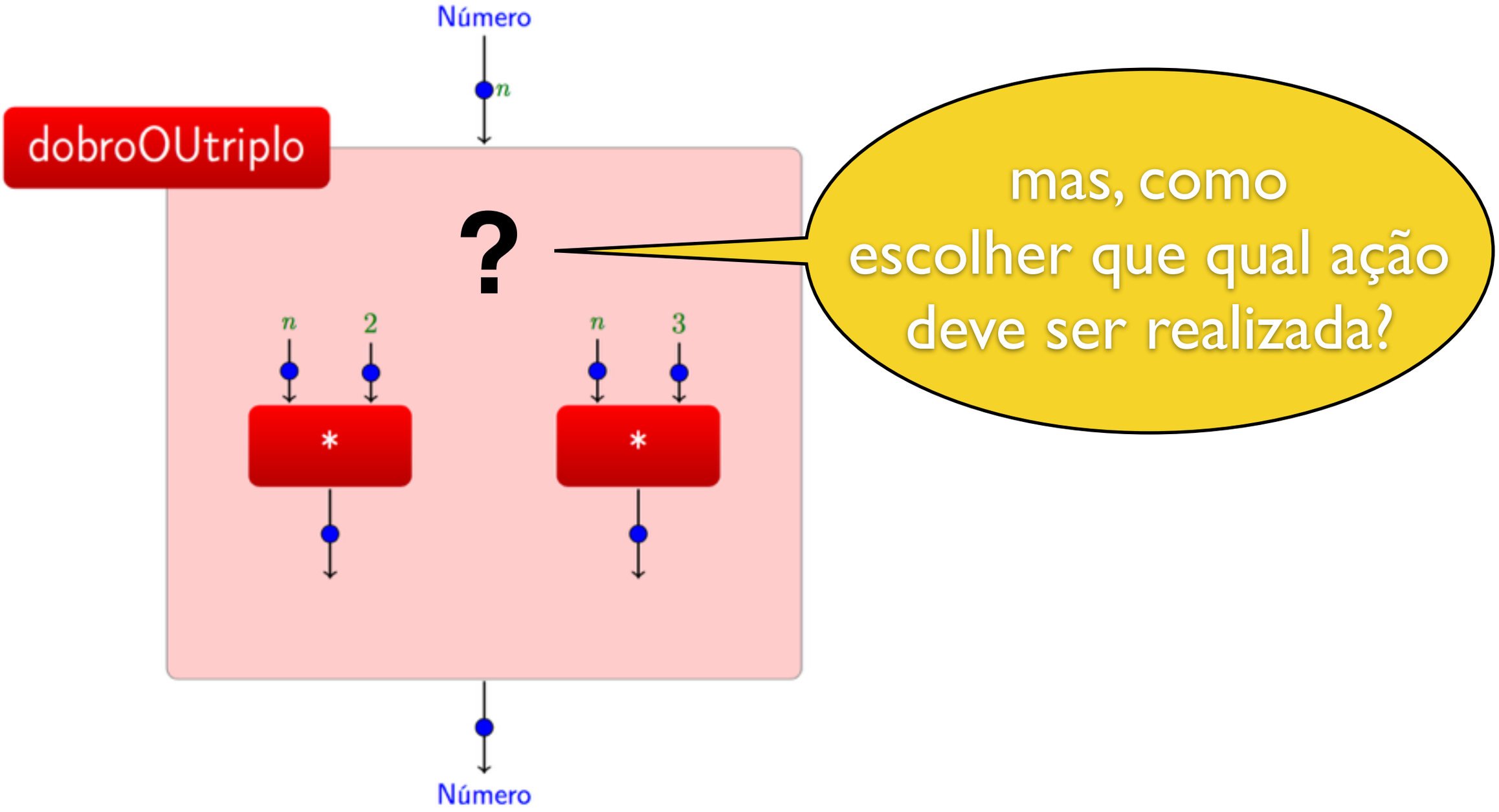


contrato

dobroOUtriplo : Número → Número

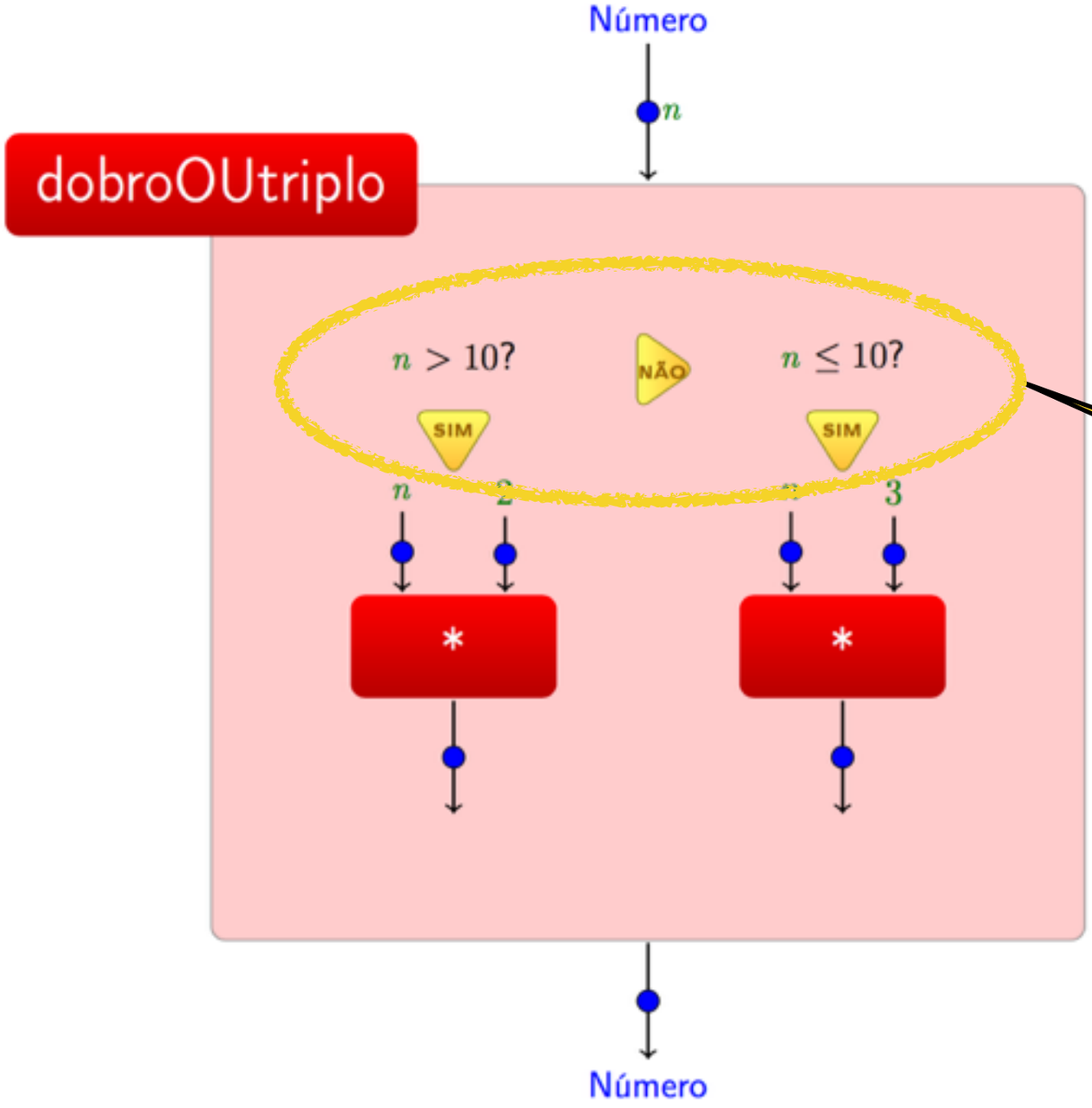
Exemplo

- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.



Exemplo

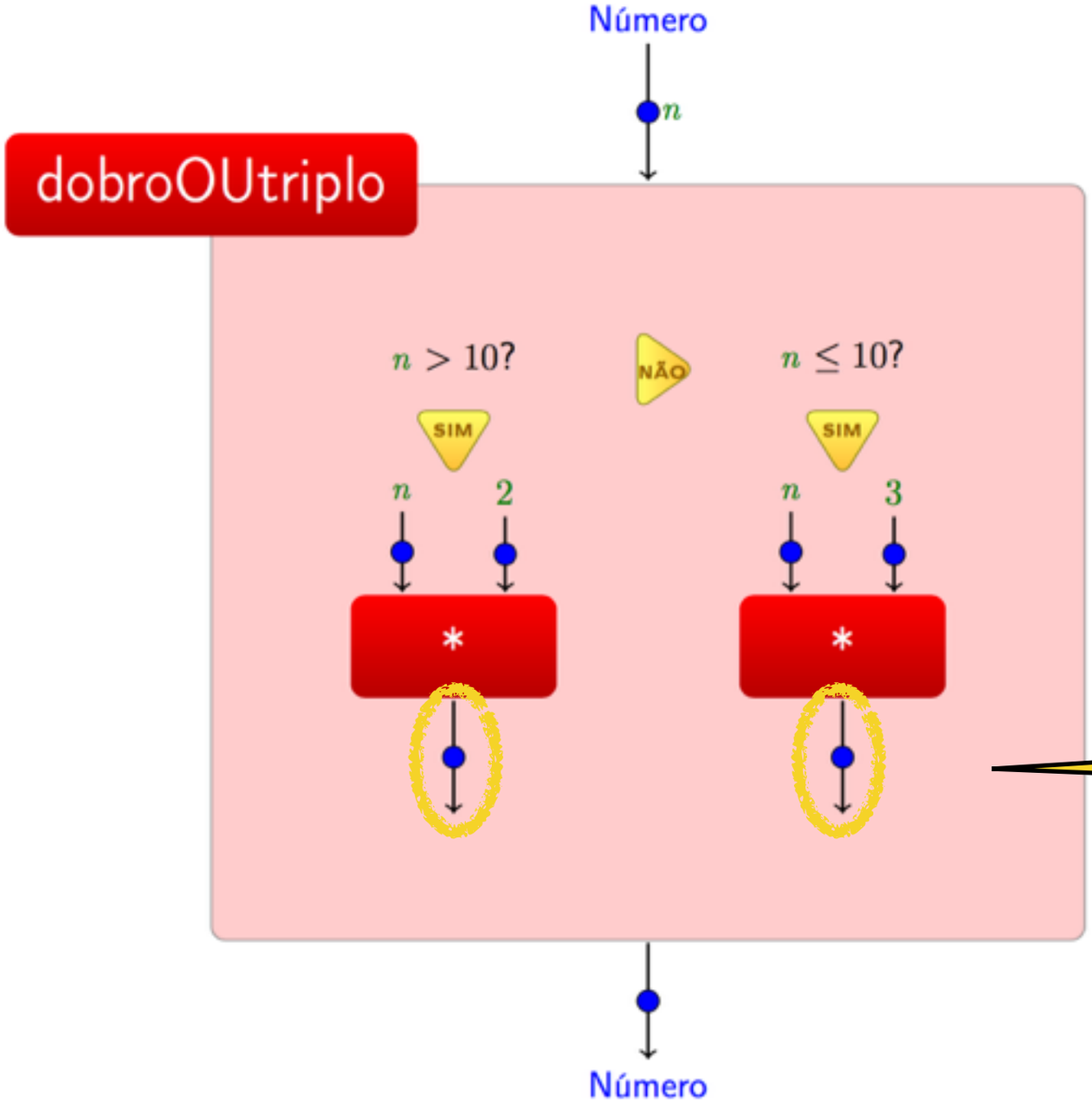
- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.



fazendo um teste!

Exemplo

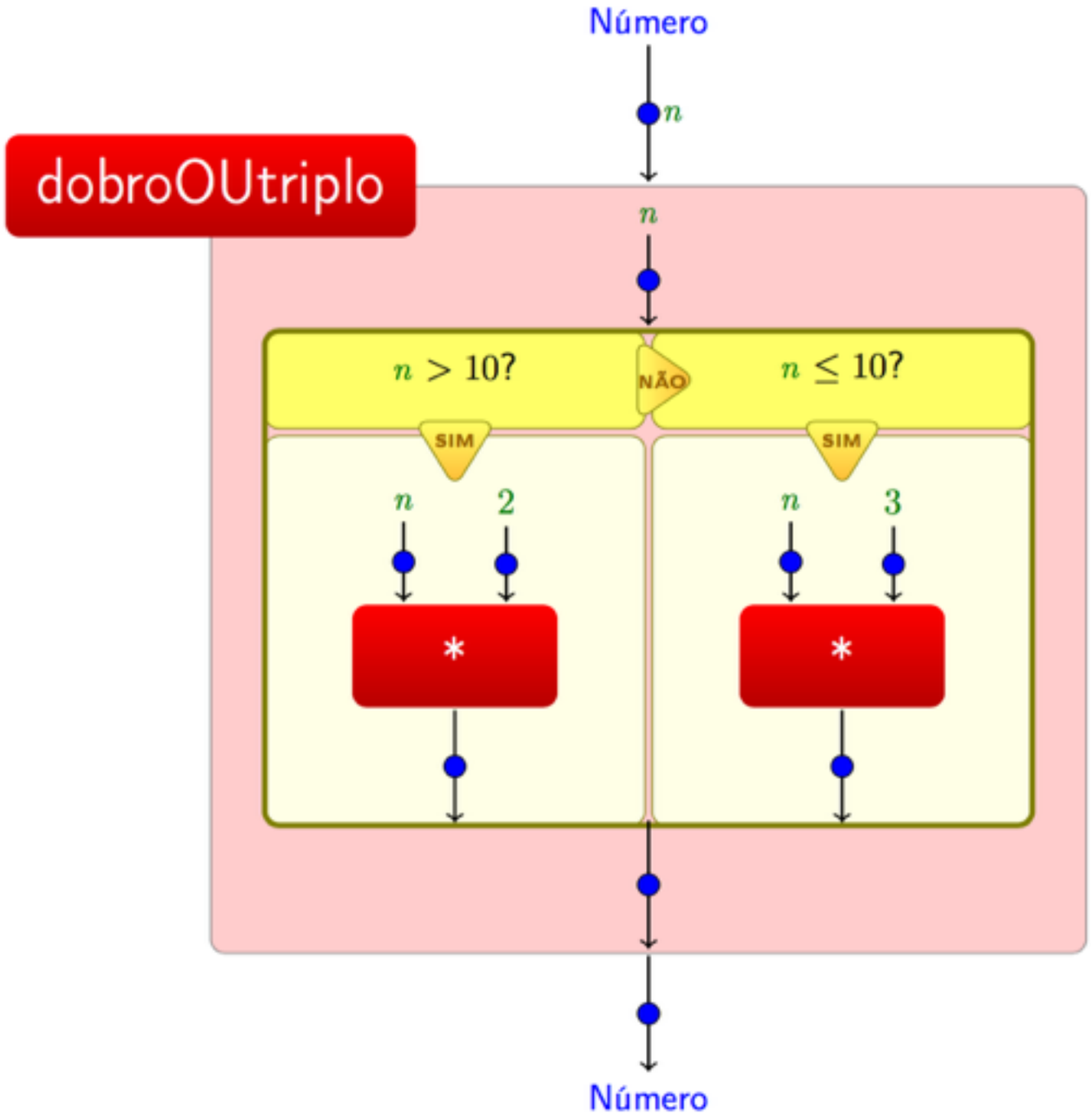
- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.



e como fazer para não termos 2 resultados?

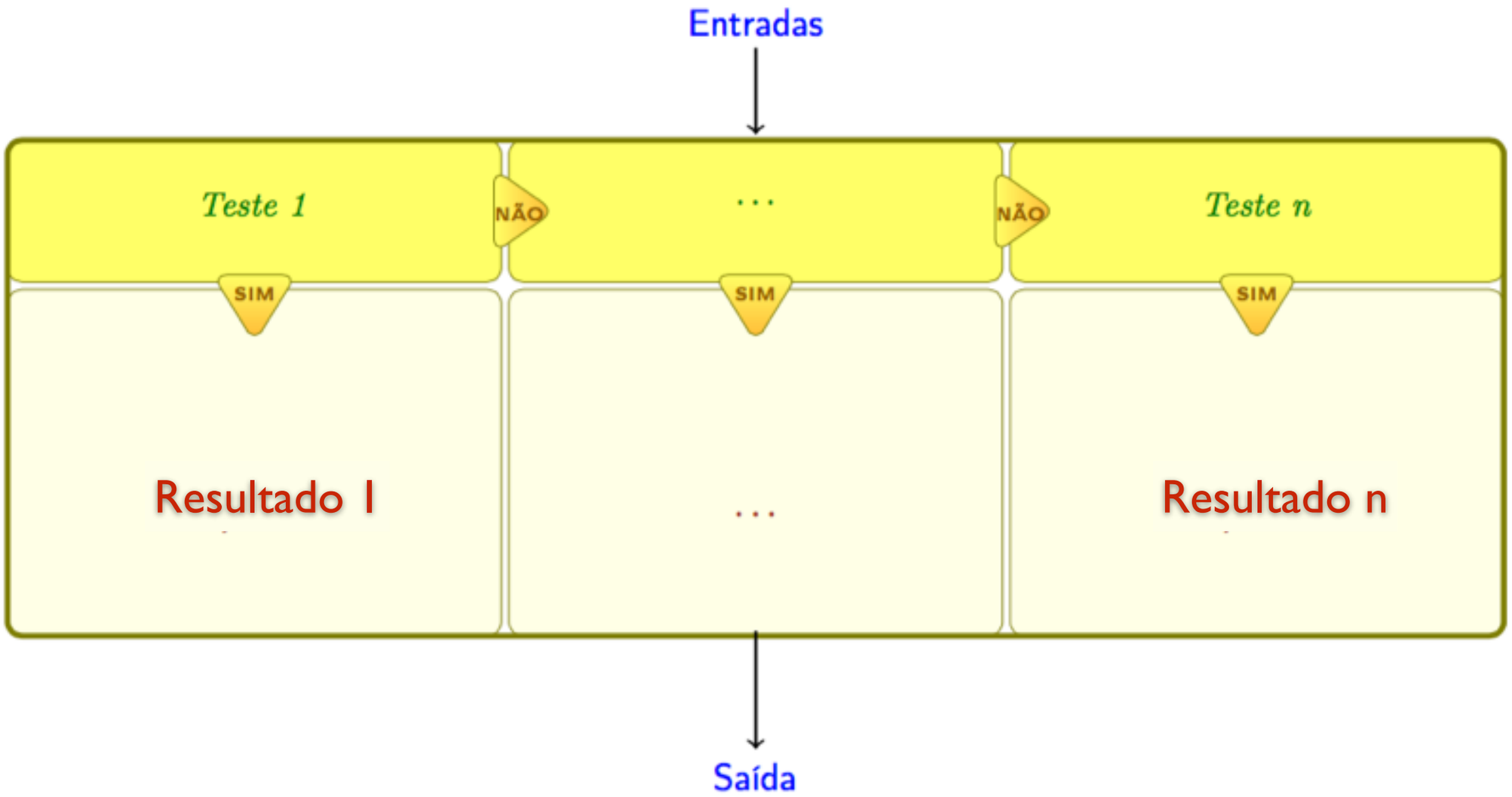
Exemplo

- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.



encapsulando os resultados diferentes em um ***bloco condicional***!

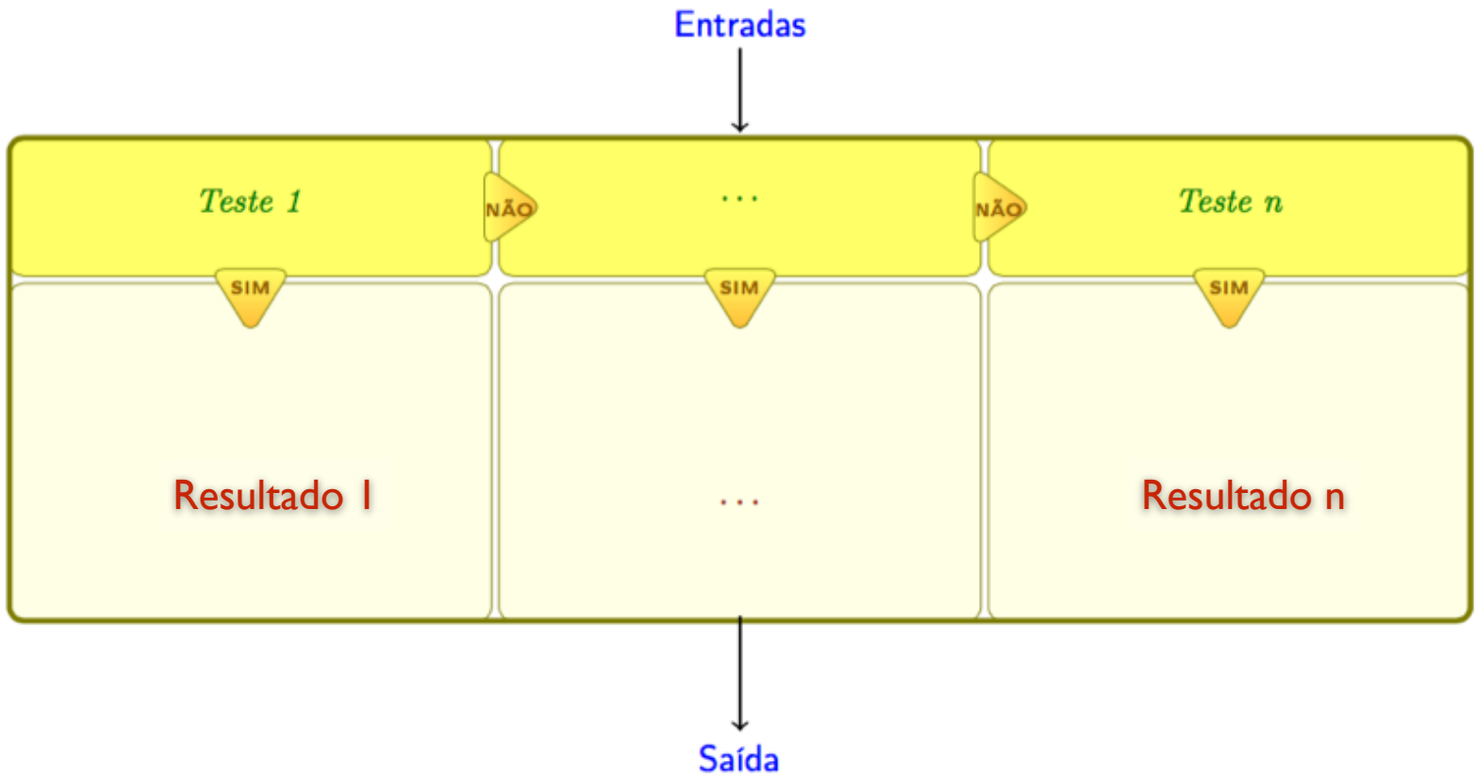
Expressão Condicional



Uma expressão condicional é um grupo de pares:

(teste, resultado)

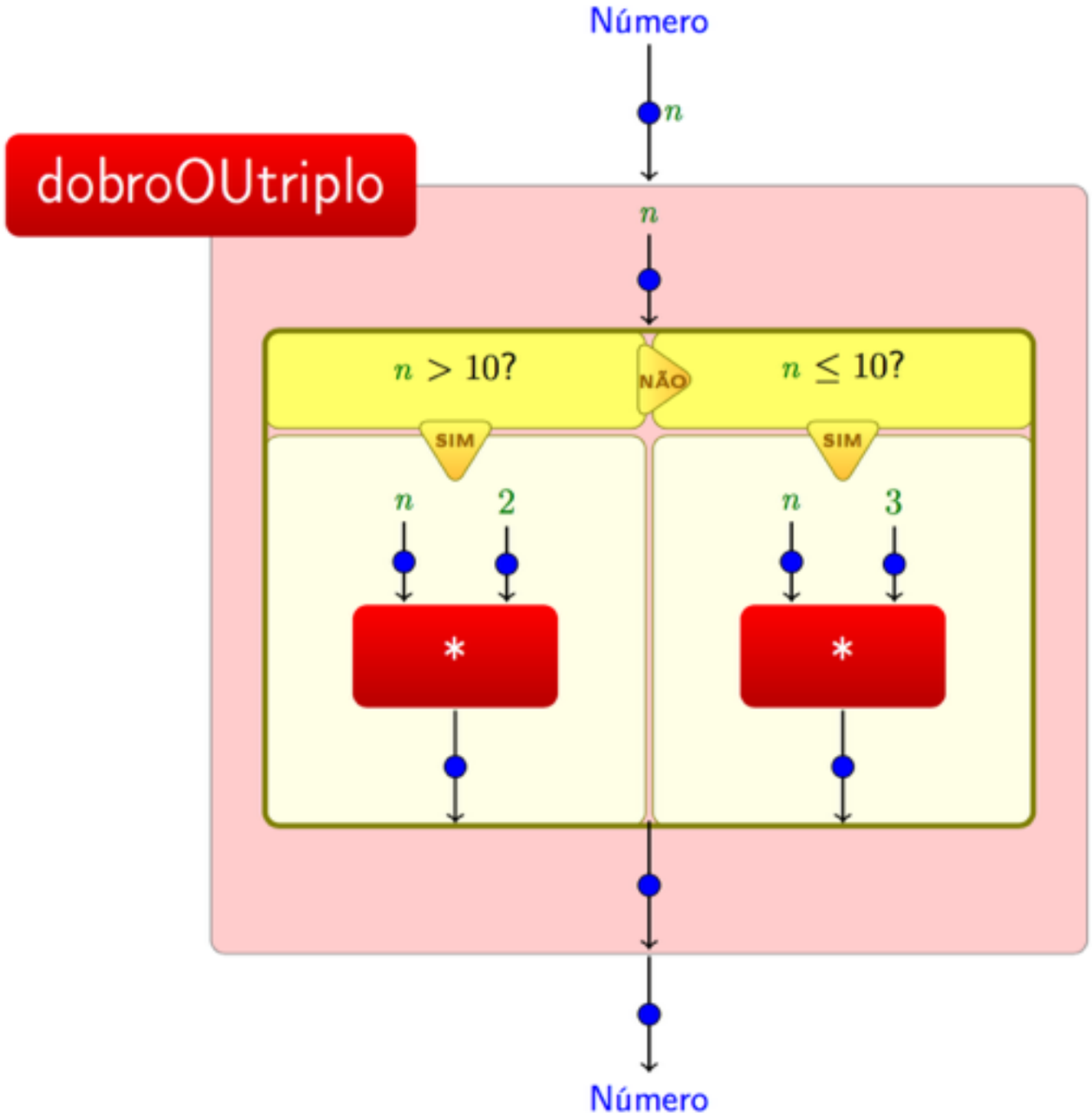
Textualmente…




```
( cond
      [ Teste 1   Resultado 1]
      [   ...       ...     ]
      [ Teste n   Resultado n]
)
```

Exemplo

- Construir um algoritmo que, dado um número, se ele for maior que 10 devolve o seu dobro, senão, devolve o seu triplo.



dobroOUtriplo : Número → Número

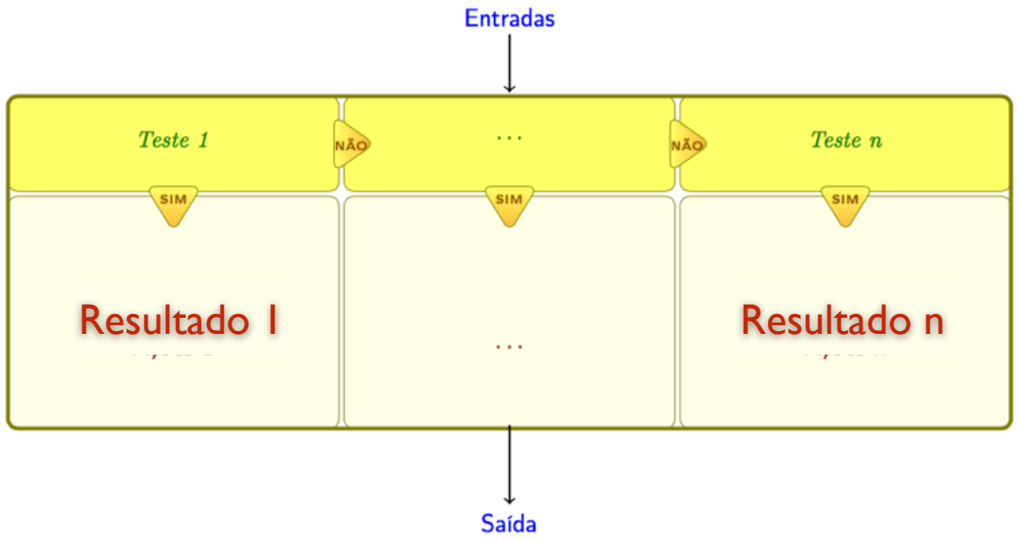
```
(define (dobroOUtriplo n)
  (cond
        [(> n 10)   (* 2 n)]
        [(<= n 10) (* n 3)]
  )
)
```

teste

resultado

Expressão Condicional

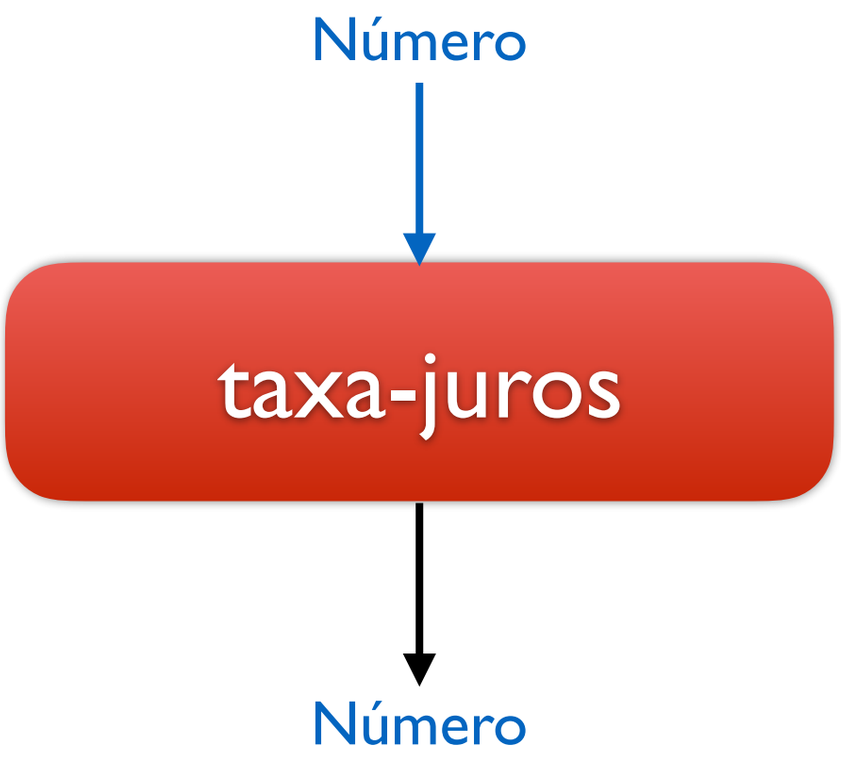
```
(cond
        [teste1  resultado1]
        [teste2 resultado2]
          ...
        [testen resultado n]
)
```



- O **teste** deve ser uma **expressão booleana**
- O **resultado** deve ser uma **expressão válida (qualquer tipo)**
- Se nenhum teste der resultado verdadeiro, o condicional resultará em erro ao executar
- Pode-se usar **else** ao invés da última expressão de teste
- Se mais de uma expressão de teste for verdadeira, a primeira será executada
- NUNCA mais de uma instrução de um condicional é executada

Exemplo

"Suponha que um banco pague juros de 4% para depósitos de até R$ 1000 (inclusive), 4.5% para depósitos de até R$ 5000 (inclusive) e de 5% para depósitos de mais de R$ 5000. Escreva um programa que, dado o valor a ser depositado, produza a taxa de juros correspondente a esse valor."



contrato

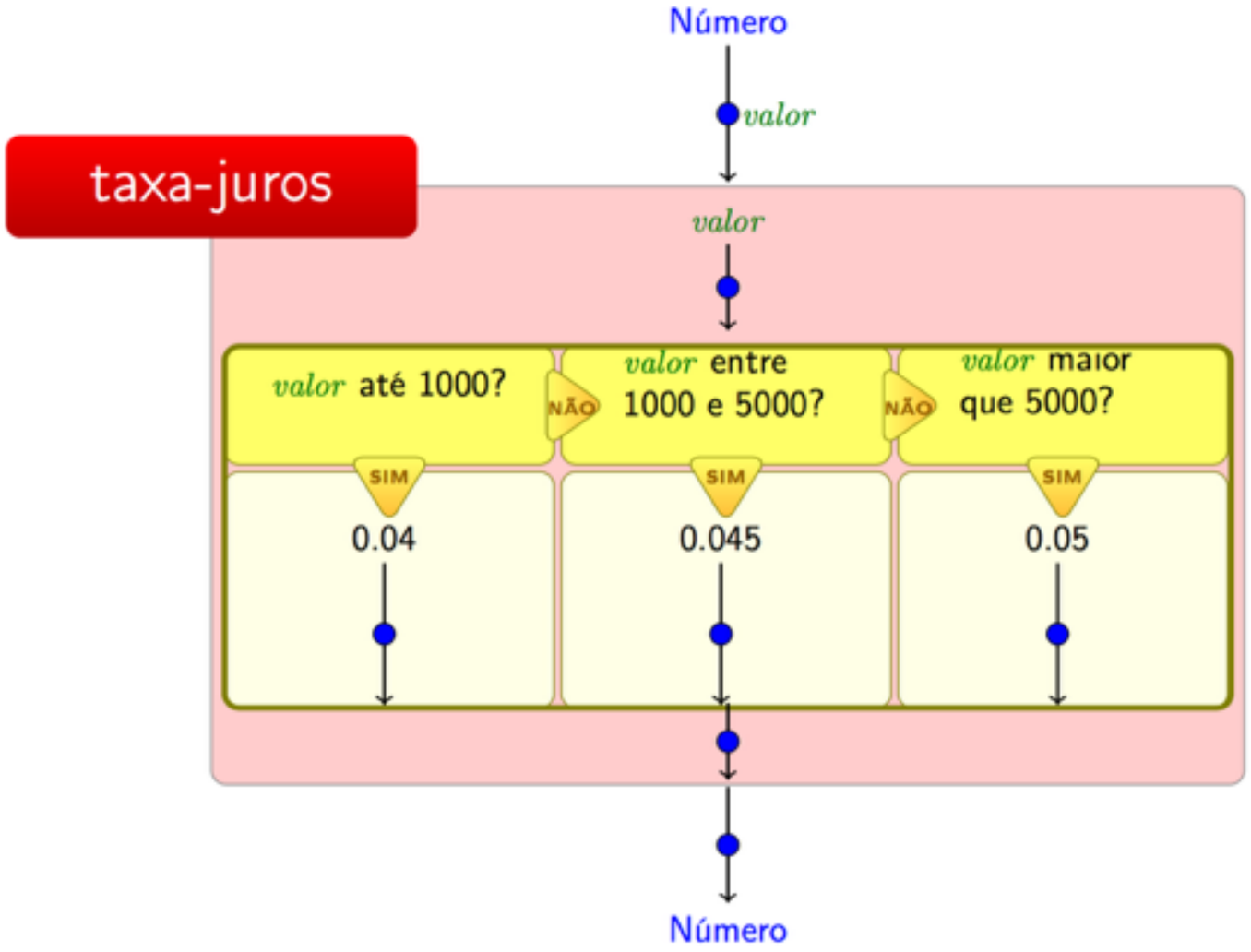
taxa-juros : Número → Número

Exemplo

"Suponha que um banco pague juros de 4% para depósitos de até R$ 1000 (inclusive), 4.5% para depósitos de até R$ 5000 (inclusive) e de 5% para depósitos de mais de R$ 5000. Escreva um programa que, dado o valor a ser depositado, produza a taxa de juros correspondente a esse valor."

```
;; Exemplos:
;;  (taxa-juros 1000) = ...
;;  (taxa-juros 5000) = ...
;;  (taxa-juros 10000) =  ...
;;  (taxa-juros 245) =  ...
```

Exemplo



taxa-juros : Número → Número

```
(define (taxa-juros valor)
  (cond
        [(<= valor 1000)   0.04]
        [(and (> valor 1000)
              (<= valor 5000)) 0.045]
        [(> valor 5000) 0.05]
  )
)
```

Qual a diferença?

```
taxa-juros : Número → Número

(define (taxa-juros valor)
   (cond
         [(<= valor 1000)   0.04]
         [(< valor 5000)) 0.045]
         [(>= valor 5000) 0.05]
    )
)
```

```
taxa-juros : Número → Número

(define (taxa-juros valor)
   (cond
         [(<= valor 1000)   0.04]
         [(and (> valor 1000)
               (< valor 5000)) 0.045]
         [(>= valor 5000) 0.05]
    )
)
```

Exemplo

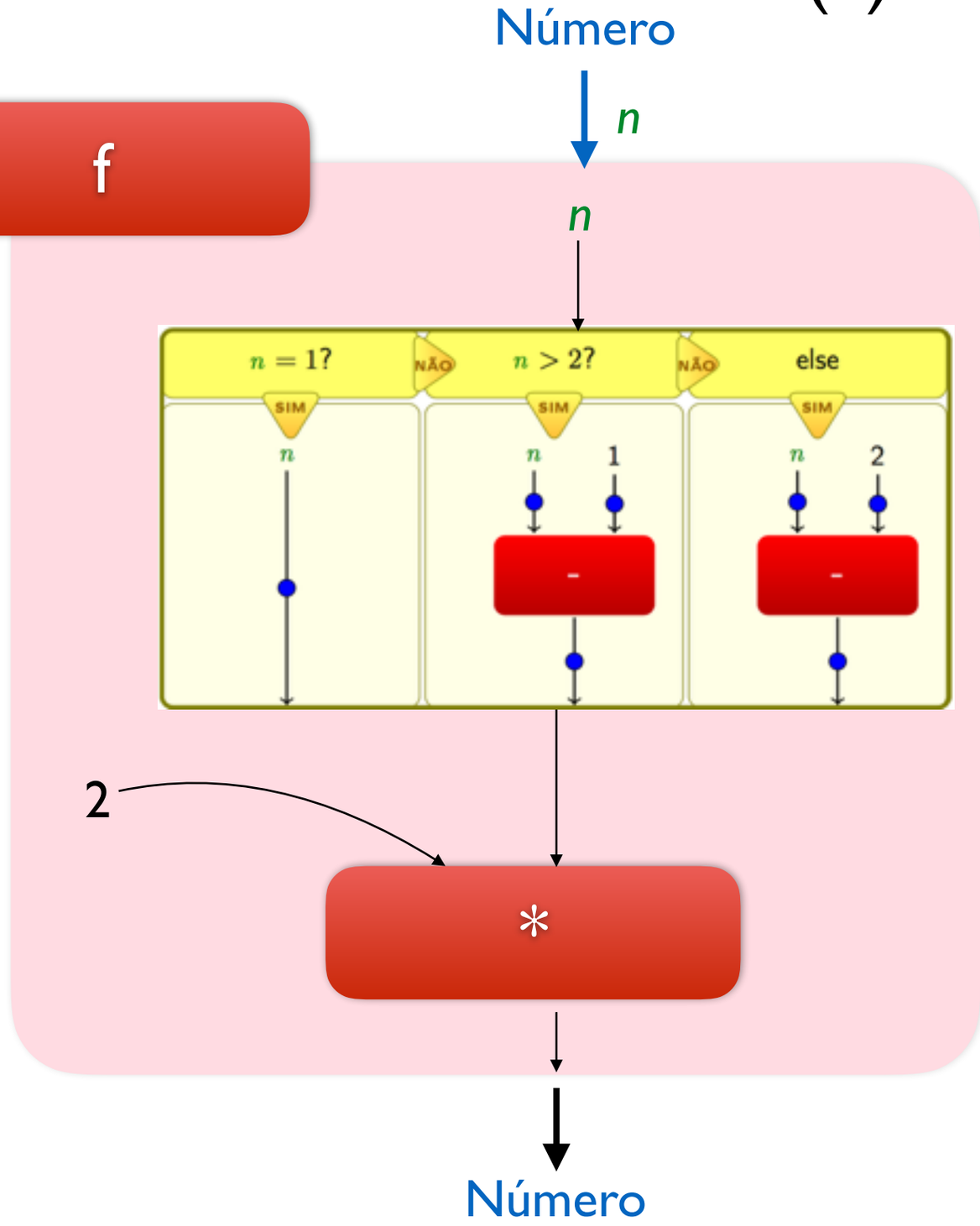
$$f : \text{Número} \rightarrow \text{Número}$$

$$f(n) = \begin{cases} 2 \times n & \text{, se } n=1 \\ 2 \times (n - 1) & \text{, se } n>2 \\ 2 \times (n - 2) & \text{, caso contrario} \end{cases}$$

$$f(n) = 2 \times \begin{cases} n & \text{, se } n=1 \\ (n - 1) & \text{, se } n>2 \\ (n - 2) & \text{, caso contrario} \end{cases}$$

Exemplo

$$f(n) = 2 \times \begin{cases} n & \text{, se } n=1 \\ (n-1) & \text{, se } n>2 \\ (n-2) & \text{, caso contrario} \end{cases}$$



f : Número → Número

```
(define (f n)
  (* 2
     (cond
           [(= n 1)   n]
           [(> n 2)   (- n 1)]
           [else      (- n 2)]
     )
  )
)
```

Exemplo

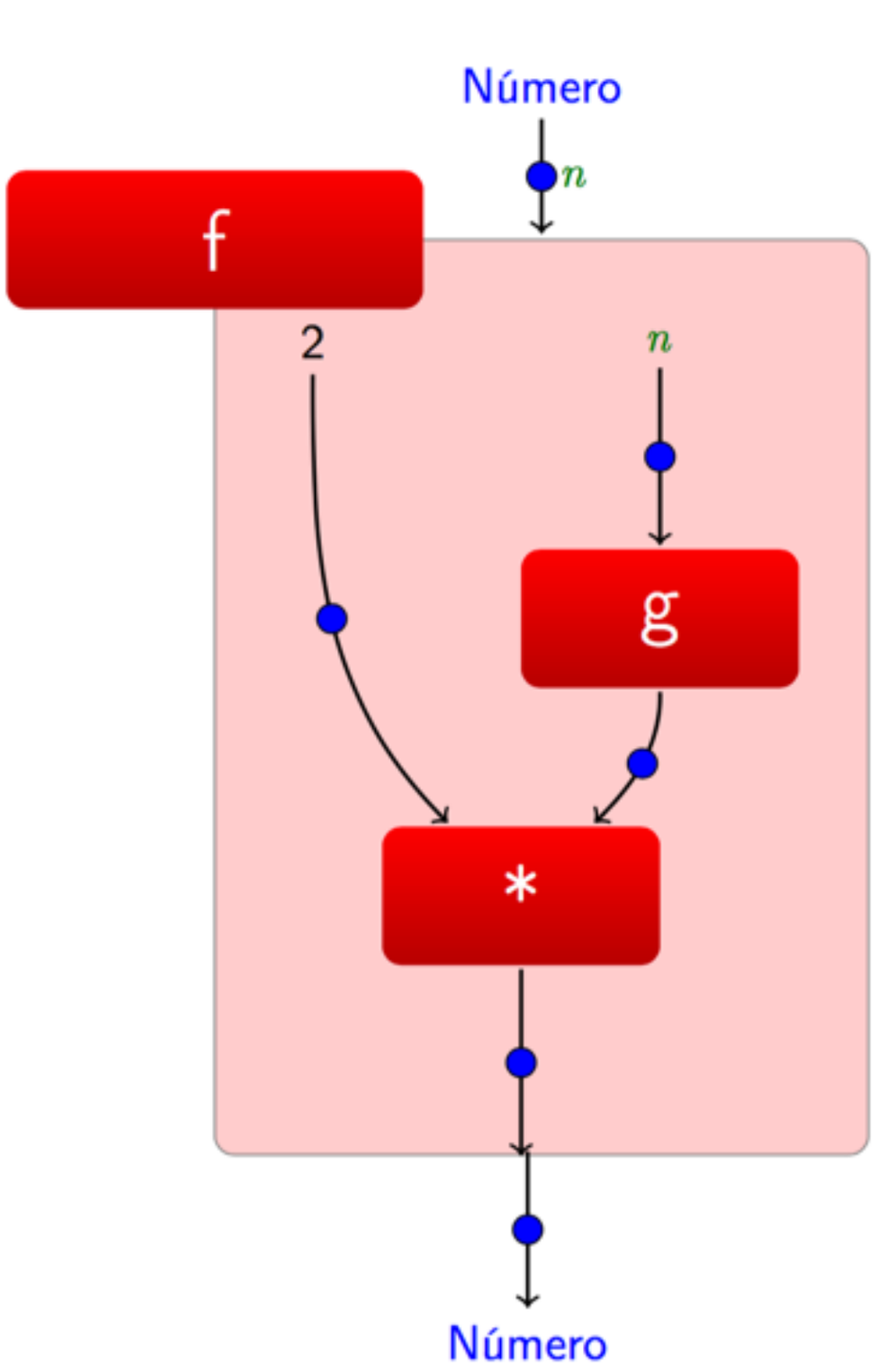
$$f : \text{Número} \rightarrow \text{Número}$$

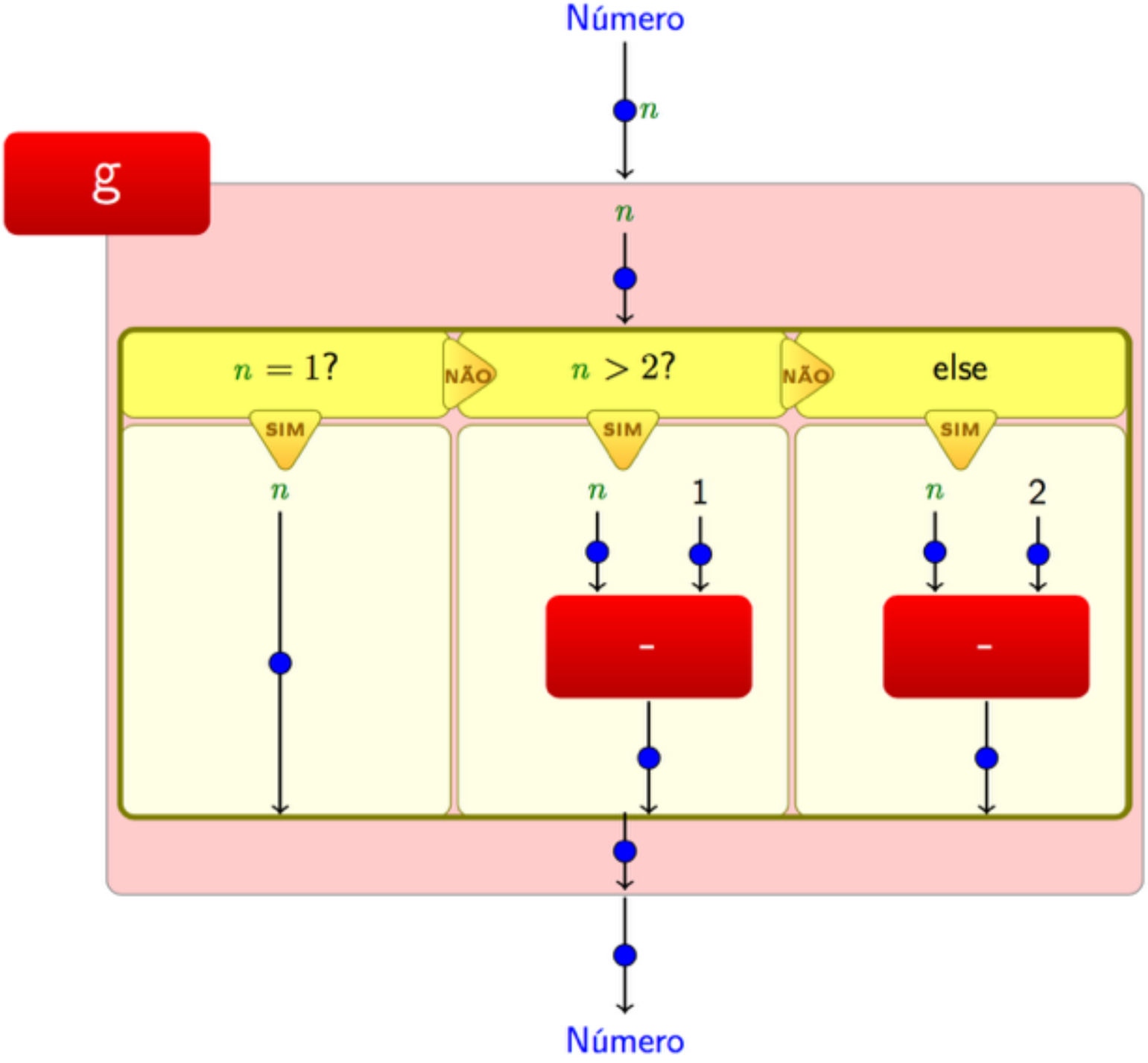
$$f(n) = 2 \times g(n)$$

$$g(n) = \begin{cases} n & , \text{se } n = 1 \\ (n - 1) & , \text{se } n > 2 \\ (n - 2) & , \text{caso contrario} \end{cases}$$

$$f(n) = 2 \times g(n)$$

$$g(n) = \begin{cases} n & \text{, se } n=1 \\ (n-1) & \text{, se } n>2 \\ (n-2) & \text{, caso contrario} \end{cases}$$





Lembrete

1. Leia os capítulos 1 a 4 do livro `www.htdp.org`
2. Teste os exemplos do livro no DrRacket
3. Faça os exercícios da lista !