

Structure and Interpretation of Computer Programs

Second Edition

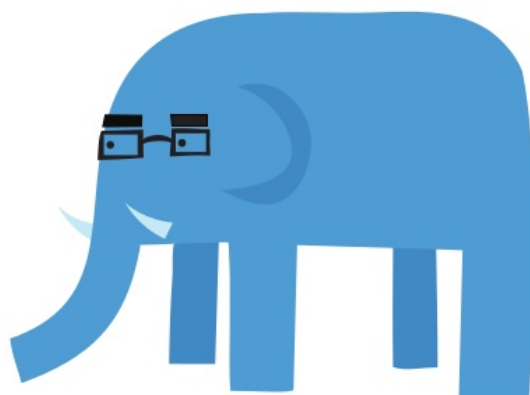


Harold Abelson and
Gerald Jay Sussman
with Julie Sussman



Learn You a Haskell for Great Good!

A Beginner's Guide



Miran Lipovača

<https://repl.it/languages/scheme>

Github хранилище с материалите от семинара

1. <https://github.com/triffon/fp-2021-22>
2. Отивате в `exercises/8`

Всеки мощен език за програмиране ни дава 3 основни средства за организация на нашите идеи при решаването на даден проблем:

- **примитивни изрази** – най-простите елементи
- **средства за комбинация** – за създаване на съставни елементи от по-прости
- **средства за абстракция** – за именуване на съставни елементи, които да използваме като примитивните елементи

Израз

Infix notation

1 + 2

Prefix notation

(+ 1 2)
(+ 1 2 3 4 5) ; повече аргументи

Атоми

Примитивните изрази в езика

```
42  
3.14  
#t  
#f
```



LISTS

LISTS EVERYWHERE

Списъци

Средствата за комбинация в езика

```
(operation operand1 operand2 ...)
```

```
(sum-of-squares 2 3)
```

```
(+ (* 2 2) (* 3 3))
```

Вградени процедури

Те също са изрази

`+` , `-` , `*` , `/` , `remainder` , `quotient` , `modulo`

Вградени предикати

Те също са процедури

= , < , > , <= , >= , not , and , or

**“Lisp programmers know the value
of everything but the cost of nothing.”**

*Alan Perlis
(paraphrasing Oscar Wilde)*

Оценка на комбинация

(подизраз1 подизраз2 подизраз3 ...)

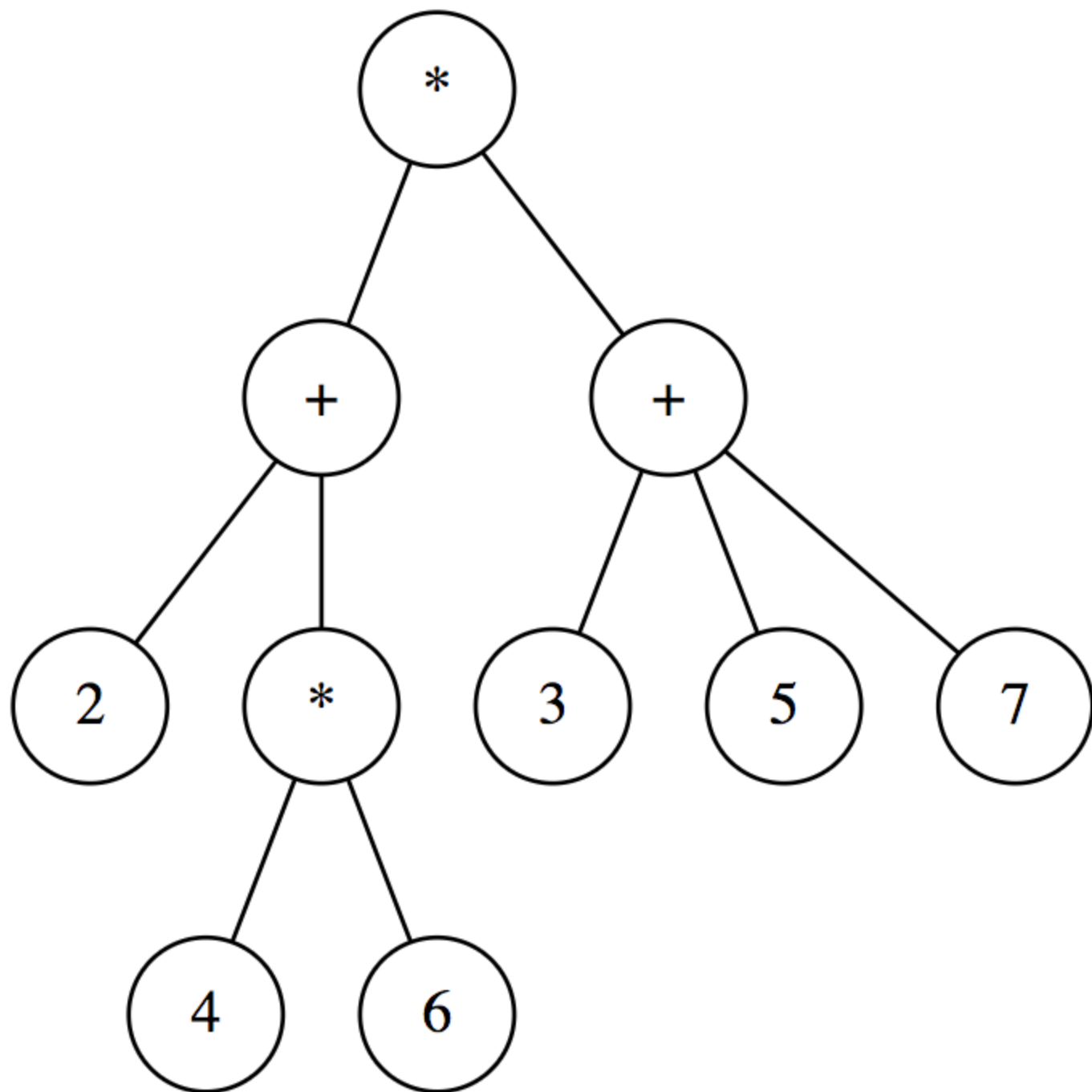
1. Оценяваме всички под-изрази в комбинацията (списъка).

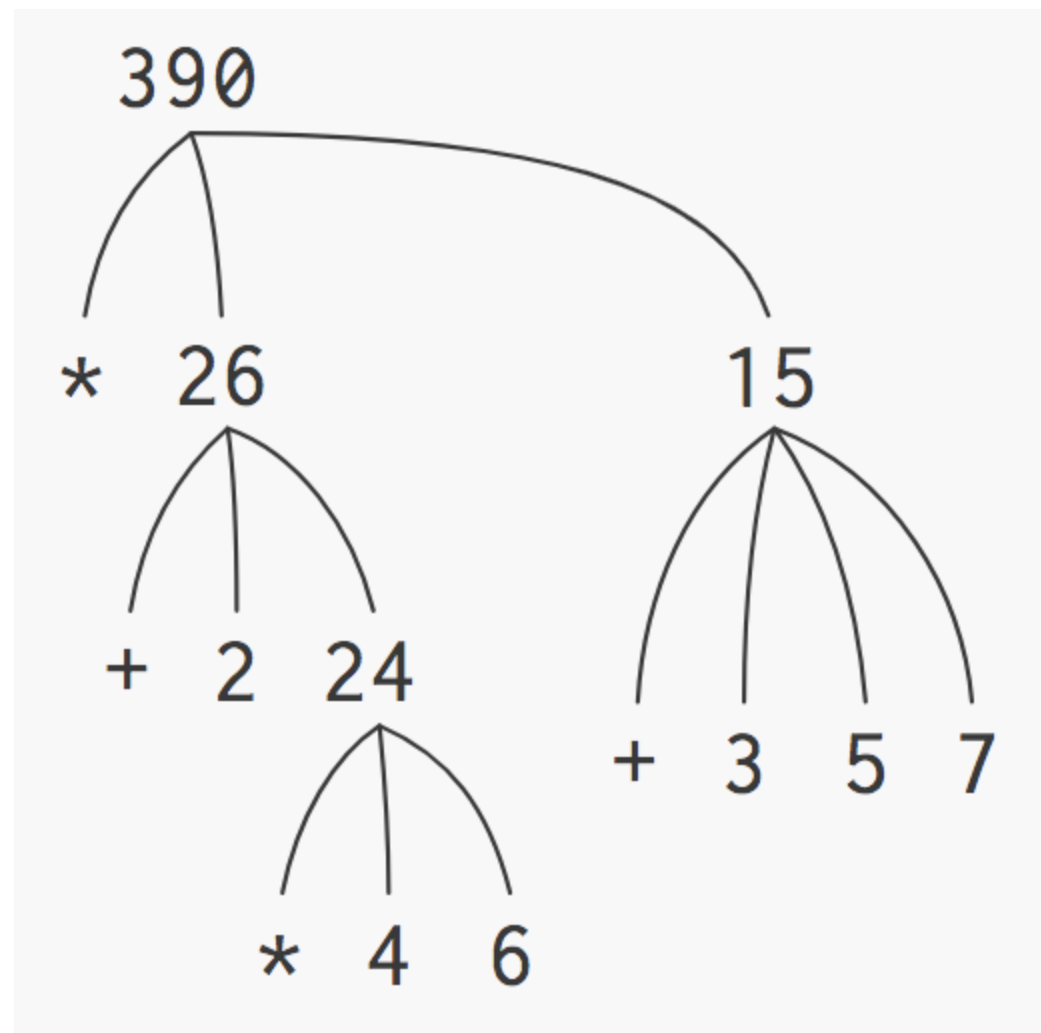
(процедура аргумент1 аргумент2 ...)

2. Прилагаме стойността на най-левия подизраз (процедура)
върху аргументите, които са стойностите на останалите
подизрази.

Оценете комбинацията

```
( * ( + 2 ( * 4 6 ) )  
    ( + 3 5 7 ) )
```





Именуване на израз

Средство за абстракция

```
(define grade 6)
```

```
(define answer  
  (+ 3  
     (* 3 13)))
```

```
(define + -)
```

Дефиниране на процедура

По-мощно средство за абстракция

```
(define (square x) (* x x))
```

| | | | |

To square something, multiply it by itself.

```
(define (sum-of-squares x y)
  (+ (square x) (square y)))
```

Прилагане на процедура

```
(square 7) ; 49
```

```
(sum-of-squares 3 4) ; 25
```

```
(sum-of-squares (+ 1 2) (* 2 2)) ; 25
```

if

```
(if (= 1 1) #t #f) ; #t
```

```
(if (not (= 1 1))  
    (+ 3 3)  
    (+ 1 1)) ; 2
```

```
(if <условие>  
    <истина>  
    <лъжа>)
```

cond

```
(define (abs x)
  (cond ((> x 0) x)
        ((< x 0) (- x))
        (else 0)))
```

```
(cond (<условие-1> <клауза-1>)
      (<условие-2> <клауза-2>)
      ...
      (<условие-N> <клауза-N>)
      (else <клауза-по-подразбиране>))
```

Логически оператори

```
(and <p1> <p2> ... <pN>)
```

```
(or <p1> <p2> ... <pN>)
```

```
(not <p>)
```

Примери

```
(define (>= x y) (or (> x y) (= x y)))
```

```
(define (>= x y) (not (< x y)))
```