Модел на средите и изчислителни процеси

Трифон Трифонов

Функционално програмиране, спец. Информатика, 2016/17 г.

13–20 октомври 2016 г.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.
- В даден момент могат да съществуват много среди.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.
- В даден момент могат да съществуват много среди.
- Символите винаги се оценяват в една конкретна среда.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.
- В даден момент могат да съществуват много среди.
- Символите винаги се оценяват в една конкретна среда.
- Символите могат да има различни оценки в различни среди.

Среди в Scheme

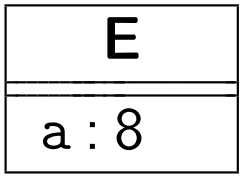
- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.
- В даден момент могат да съществуват много среди.
- Символите винаги се оценяват в една конкретна среда.
- Символите могат да има различни оценки в различни среди.
- При стартиране Scheme по подразбиране работи в **глобалната среда**.

Среди в Scheme

- Връзката между символите и техните оценки се записват в речник, който се нарича **среда**.
- Всеки символ има най-много една оценка в дадена среда.
- В даден момент могат да съществуват много среди.
- Символите винаги се оценяват в една конкретна среда.
- Символите могат да има различни оценки в различни среди.
- При стартиране Scheme по подразбиране работи в **глобалната среда**.
- В глобалната среда са дефинирани символи за стандартни операции и функции.

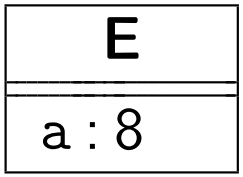
Пример за среда

- `(define a 8)`



Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!



Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!
- `(define r 5)`

E
a : 8
r : 5

Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!
- `(define r 5)`
- `(+ r 3)` $\longrightarrow$ `8`

E
a : 8
r : 5

Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!
- `(define r 5)`
- `(+ r 3)` $\longrightarrow$ `8`
- `(define (f x) (* x r))`

Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!
- `(define r 5)`
- `(+ r 3)` $\longrightarrow$ `8`
- `(define (f x) (* x r))`
- `(f 3)` $\longrightarrow$ `15`

Пример за среда

- `(define a 8)`
- `r` $\longrightarrow$ Грешка!
- `(define r 5)`
- `(+ r 3)` $\longrightarrow$ `8`
- `(define (f x) (* x r))`
- `(f 3)` $\longrightarrow$ `15`
- `(f r)` $\longrightarrow$ `25`

Функции и среди

- Всяка функция `f` пази указател към средата **E**, в която е дефинирана.

Функции и среди

- Всяка функция `f` пази указател към средата **E**, в която е дефинирана.
- При извикване на `f`:

Функции и среди

- Всяка функция `f` пази указател към средата $\mathbf{E}$, в която е дефинирана.
- При извикване на `f`:
  - създава се нова среда $\mathbf{E_1}$, която разширява $\mathbf{E}$

Функции и среди

- Всяка функция `f` пази указател към средата $\mathbf{E}$, в която е дефинирана.
- При извикване на `f`:
  - създава се нова среда $\mathbf{E_1}$, която разширява $\mathbf{E}$
  - в $\mathbf{E_1}$ всеки символ означаващ формален параметър се свързва с оценката на фактическия параметър

Функции и среди

- Всяка функция `f` пази указател към средата $\mathbf{E}$, в която е дефинирана.
- При извикване на `f`:
  - създава се нова среда $\mathbf{E_1}$, която разширява $\mathbf{E}$
  - в $\mathbf{E_1}$ всеки символ означаващ формален параметър се свързва с оценката на фактическия параметър
  - тялото на f се оценява в $\mathbf{E_1}$

Дърво от среди

- Всяка среда пази указател към своя “родителска среда”, която разширява

Дърво от среди

- Всяка среда пази указател към своя “родителска среда”, която разширява
- така се получава дърво от среди

Дърво от среди

- Всяка среда пази указател към своя “родителска среда”, която разширява
- така се получава дърво от среди
- при оценка на символ в дадена среда **E**

Дърво от среди

- Всяка среда пази указател към своя “родителска среда”, която разширява
- така се получава дърво от среди
- при оценка на символ в дадена среда **E**
  - първо се търси оценката му в **E**

Дърво от среди

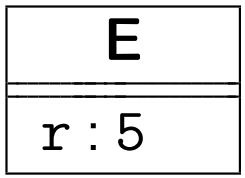
- Всяка среда пази указател към своя “родителска среда”, която разширява
- така се получава дърво от среди
- при оценка на символ в дадена среда **E**
  - първо се търси оценката му в **E**
  - ако символът не е дефиниран в **E**, се преминава към родителската среда

Дърво от среди

- Всяка среда пази указател към своя “родителска среда”, която разширява
- така се получава дърво от среди
- при оценка на символ в дадена среда **E**
  - първо се търси оценката му в **E**
  - ако символът не е дефиниран в **E**, се преминава към родителската среда
  - при достигане на най-горната среда, ако символът не е дефиниран и в нея се извежда съобщение за грешка

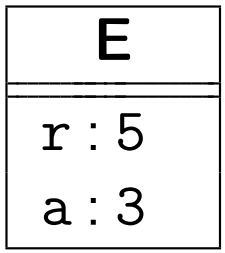
Извикване на дефинирана функция

- `(define r 5)`



Извикване на дефинирана функция

- `(define r 5)`
- `(define a 3)`



Извикване на дефинирана функция

- `(define r 5)`
- `(define a 3)`
- `(define (f x) (* x r))`

Извикване на дефинирана функция

- `(define r 5)`
- `(define a 3)`
- `(define (f x) (* x r))`
- {**E**} `(f a)`

Извикване на дефинирана функция

- `(define r 5)`
- `(define a 3)`
- `(define (f x) (* x r))`
- {**E**} `(f a)`

  ↓

  {**E**} `(f 3)`

Извикване на дефинирана функция

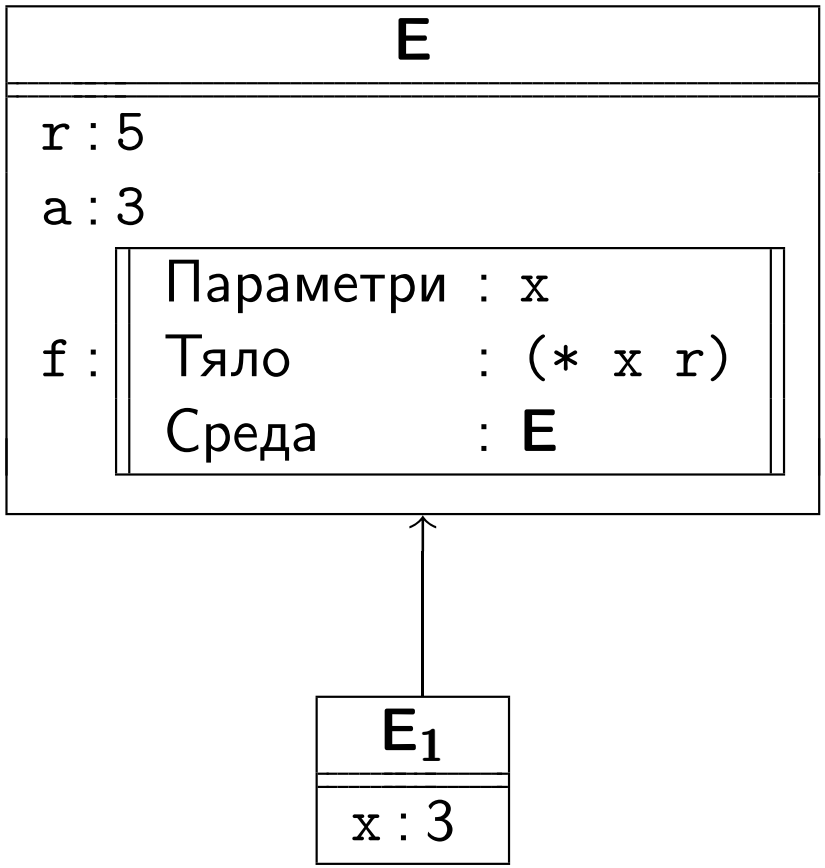
- `(define r 5)`
- `(define a 3)`
- `(define (f x) (* x r))`
- {E} `(f a)`

  ↓

  {E} `(f 3)`

  ↓

  $\{E_1\}$ `(* x r)`



Извикване на дефинирана функция

- `(define r 5)`
- `(define a 3)`
- `(define (f x) (* x r))`
- {E} `(f a)`

  ↓

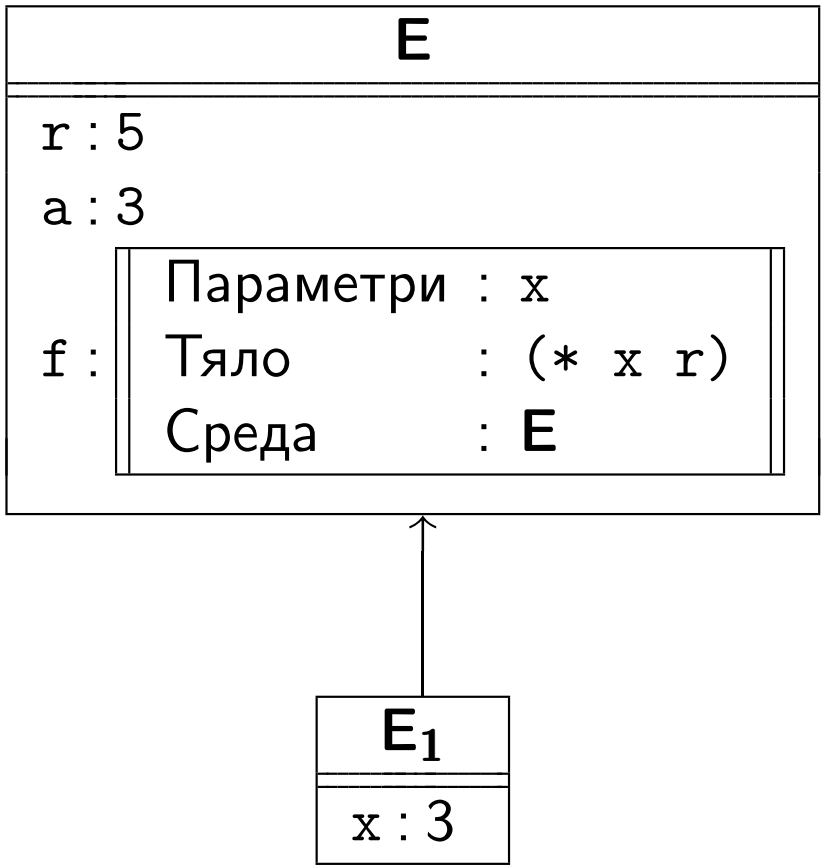
  {E} `(f 3)`

  ↓

  {E_1} `(* x r)`

  ↓

  15



Какво е рекурсия?

Какво е рекурсия?

Какво е рекурсия?

Какво е рекурсия?

Какво е рекурсия?

Повторение чрез позоваване на себе си

Какво е рекурсия?

Повторение чрез позоваване на себе си

Рекурсивна функция: дефинира се чрез себе си

$$n! = \begin{cases} 1, & \text{при } n = 0, \quad \textbf{(база)} \\ n \cdot (n-1)!, & \text{при } n > 0. \quad \textbf{(стъпка)} \end{cases}$$

Какво е рекурсия?

Повторение чрез позоваване на себе си

Рекурсивна функция: дефинира се чрез себе си

$$n! = \begin{cases} 1, & \text{при } n = 0, \quad \textbf{(база)} \\ n \cdot (n-1)!, & \text{при } n > 0. \quad \textbf{(стъпка)} \end{cases}$$

- Дава се отговор на най-простата задача (база, дъно)
- Показва се как сложна задача се свежда към една или няколко по-прости задачи от същия вид (стъпка)