Volume II: Systems

Introduction to System Programming

Namdak Tonpa

2022 · Groupoid Infinity

II

Зміст

Issue VII: Cartesian Interpreter

Maksym Sokhatskyi [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnical Institute
29 квітня 2025 р.

Анотація

Minimal language for sequential computations in cartesian closed categories.

Keywords: Lambda Calculus, Cartesian Closed Categories

0.1 The Joe Language

Мова програмування **Joe** — це чиста нетипізована мова, що є внутрішньою мовою декартово-замкнених категорій. Вона базується на лямбда-численні, розширеному парами, проєкціями та термінальним об'єктом, забезпечуючи мінімальну модель для обчислень у категорійному контексті.

Синтаксис Терми **Joe** складаються зі змінних, лямбда-абстракцій, застосувань, пар, проєкцій (першої та другої) та термінального об'єкта. Це мінімальна мова, що підтримує обчислення через бета-редукцію та проєкції.

```
I = #identifier
O = I | ( O ) | O O | λ I -> O | O , O | O.1 | O.2 | 1
```

```
type term =
  | Var of string
  | Lam of string * term
  | App of term * term
  | Pair of term * term
  | Fst of term
  | Snd of term
  | Unit
```

Правила обчислень Основними правилами обчислень у **Joe** є бета-редукція для лямбда-абстракцій та правила проєкцій для пар. Термінальний об'єкт є незвідним.

```
App (Lam (x, b), a) → subst x a b
Fst (Pair (t1, t2)) → t1
Snd (Pair (t1, t2)) → t2
```

$$\frac{(\lambda x.b)\ a}{b[a/x]} \quad \frac{\text{fst}\ \langle t_1, t_2 \rangle}{t_1} \quad \frac{\text{snd}\ \langle t_1, t_2 \rangle}{t_2}$$

```
let rec subst x s = function
  | Var y -> if x = y then s else Var y
  | Lam (y, t) when x <> y -> Lam (y, subst x s t)
  | App (f, a) -> App (subst x s f, subst x s a)
  | Pair (t1, t2) -> Pair (subst x s t1, subst x s t2)
  | Fst t -> Fst (subst x s t)
  | Snd t -> Snd (subst x s t)
  | Unit -> Unit
  | t -> t
```

```
let rec equal t1 t2 =
  match t1, t2 with
  | Var x, Var y -> x = y
  | Lam (x, b), Lam (y, b') -> equal b (subst y (Var x) b')
  | Lam (x, b), t -> equal b (App (t, Var x))
  | t, Lam (x, b) -> equal (App (t, Var x)) b
  | App (f1, a1), App (f2, a2) -> equal f1 f2 && equal a1 a2
  | Pair (t1, t2), Pair (t1', t2') -> equal t1 t1' && equal t2 t2'
  | Fst t, Fst t' -> equal t t'
  | Snd t, Snd t' -> equal t t'
  | Unit, Unit -> true
  | _ -> false
```

```
let rec reduce = function
  | App (Lam (x, b), a) -> subst x a b
  | App (f, a) -> App (reduce f, reduce a)
  | Pair (t1, t2) -> Pair (reduce t1, reduce t2)
  | Fst (Pair (t1, t2)) -> t1
  | Fst t -> Fst (reduce t)
  | Snd (Pair (t1, t2)) -> t2
  | Snd t -> Snd (reduce t)
  | Unit -> Unit
  | t -> t
```

```
let rec normalize t =
  let t' = reduce t in
  if equal t t' then t else normalize t'
```

Внутрішня мова ДЗК Мова **Joe** є внутрішньою мовою декартово-замкненої категорії (ДЗК). Вона включає лямбда-абстракції та застосування для замкнутої структури, пари та проєкції для декартового добутку, а також термінальний об'єкт для відновлення повної структури ДЗК.

Бібліографія

1. Alonzo Church. *A Set of Postulates for the Foundation of Logic*. 1933.
2. Alonzo Church. *An Unsolvable Problem of Elementary Number Theory*. 1941.
3. Haskell Curry, Robert Fey. *Combinatory Logic, Volume I*. 1951.
4. Dana Scott. *A Type-Free Theory of Lambda Calculus*. 1970.
5. John Reynolds. *Towards a Theory of Type Structure*. 1974.
6. Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. 1984.

7. G. Cousineau, P.-L. Curien, M. Mauny. *The Categorical Abstract Machine*. 1985.

Issue VIII: Linear Interpreter

Maksym Sokhatskyi [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnical Institute
29 квітня 2025 р.

Анотація

Minimal language for parallel computations in symmetric monoidal categories.
Keywords: Interaction Networks, Symmetric Monoidal Categories

0.2 The Bob Language

Мова програмування **Bob** — це внутрішня мова симетричних моноїдальних категорій, що реалізує паралельні обчислення через взаємодію комбінаторів (ζ, δ, ϵ) з правилами анігіляції та комутації, придатна для моделювання лінійних і паралельних систем.

Definition 1. Терми **Bob** складаються зі змінних, комбінаторів (Con, Dup, Era), пар, обміну (Swap), зв'язування (Let) та одиниці (Unit). Мова підтримує афінну логіку, забороняючи повторне використання змінних.

```
I = #identifier
BOB = I | Con BOB  | Dup BOB | Era BOB
        | Pair (BOB, BOB)    | Unit
        | Let (I, BOB, BOB)  | Swap BOB
```

Definition 2. Кодування термів у мові OCaml:

```
type term =
    | Var of string
    | Con of term
    | Dup of term
    | Era of term
    | Pair of term * term
    | Swap of term
    | Let of string * term * term
    | Unit
```

Theorem 1. Правила обчислень у **Bob** базуються на анігіляції та комутації комбінаторів:

```
Con (Con x) → Pair (x, x)
Dup (Dup x) → Pair (x, x)
Era (Era x) → Unit
Con (Dup x) → Dup (Con x)
Con (Era x) → Pair (Era x, Era x)
Dup (Era x) → Pair (Era x, Era x)
Swap (Pair (t, u)) → Pair (u, t)
Let (x, t, u) → subst x t u
```

$$\frac{\zeta(\zeta(x))}{(x,x)} \qquad (\zeta\text{-annihilation})$$

$$\frac{\delta(\delta(x))}{(x,x)} \qquad (\delta\text{-annihilation})$$

$$\frac{\varepsilon(\varepsilon(x))}{\mathbf{1}} \qquad (\epsilon\text{-annihilation})$$

Definition 3. Підстановка в **Bob**:

```
let rec subst env var term = function
    | Var v ->
      if v = var then
          if is_bound var env then failwith "Affine violation: variable used twice"
          else term
      else Var v
    | Con t -> Con (subst env var term t)
    | Dup t -> Dup (subst env var term t)
    | Era t -> Era (subst env var term t)
    | Pair (t, u) -> Pair (subst env var term t, subst env var term u)
    | Swap t -> Swap (subst env var term t)
    | Let (x, t1, t2) ->
      let t1' = subst env var term t1 in
      if x = var then Let (x, t1', t2)
      else Let (x, t1', subst env var term t2)
    | Unit -> Unit
```

Definition 4. Редукція термів у **Bob**:

```
let reduce env term =
    match term with
    | Con (Con x) -> Pair (x, x)
    | Dup (Dup x) -> Pair (x, x)
    | Era (Era x) -> Unit
    | Con (Dup x) -> Dup (Con x)
    | Con (Era x) -> Pair (Era x, Era x)
    | Dup (Era x) -> Pair (Era x, Era x)
    | Swap (Pair (t, u)) -> Pair (u, t)
    | Let (x, t, u) -> subst env x t u
    | _ -> term
```

Definition 5. Пошук активних пар для редукції:

```
let rec find_redexes env term acc =
    match term with
    | Con (Con x) -> (term, Pair (x, x)) :: acc
    | Dup (Dup x) -> (term, Pair (x, x)) :: acc
    | Era (Era x) -> (term, Unit) :: acc
    | Con (Dup x) -> (term, Dup (Con x)) :: acc
    | Con (Era x) -> (term, Pair (Era x, Era x)) :: acc
    | Dup (Era x) -> (term, Pair (Era x, Era x)) :: acc
    | Swap (Pair (t, u)) -> (term, Pair (u, t)) :: acc
    | Let (x, t, u) -> (term, subst env x t u) :: find_redexes env t (find_redexes env u acc)
    | Con t ->
      (match t with
       | Dup _ | Era _ -> acc
       | Con x -> find_redexes env t ((term, reduce env term) :: acc)
       | _ -> find_redexes env t acc)
    | Dup t -> find_redexes env t acc
    | Era t -> find_redexes env t acc
    | Pair (t, u) ->
      let acc' = find_redexes env t acc in
      find_redexes env u acc'
    | Swap t -> find_redexes env t acc
    | Var _ | Unit -> acc
```

Definition 6. Паралельна редукція:

```
let eval_parallel pool env term =
    let rec loop term =
        let redexes = find_redexes env term [] in
        if redexes = [] then term
        else
            let new_term = Task.run pool (fun () ->
                List.fold_left
                    (fun acc (old_t, new_t) -> replace_subterm old_t new_t acc)
                    term redexes
            ) in
            loop new_term
    in
    loop term
```

Theorem 2. Доведення, що мова **Bob** є внутрішньою мовою симетричних моноїдальних категорій:

$$\begin{cases} \text{Let} : A \to C(u \cdot t, t : A \to B, u : B \to C), \\ \text{Pair} : A \to B \to A \otimes B, \\ \text{Swap} : A \otimes B \to B \otimes A, \\ \text{Con} : A \otimes A \to A, \\ \text{Dup} : A \to A \otimes A, \\ \text{Era} : A \to \mathbf{1}, \\ \text{Var} : A, \\ \text{Unit} : \mathbf{1}. \end{cases}$$

Конструктори відповідають аксіомам СМК:

- Pair моделює тензорний добуток $\otimes$. - Swap реалізує симетрію $\sigma_{A,B}$ з умовою $\sigma_{B,A} \circ \sigma_{A,B} = \text{id}_{A \otimes B}$. - Unit є одиничним об'єктом I для якого $A \otimes I \cong A$. - Let моделює композицію морфізмів (асоціативність). - Dup та Era утворюють структуру комоноїда. - Con діє як контракція.

Лямбда-функція і аплікація:

$$\begin{cases} \lambda x.t \vdash \text{Con}(\text{Let}(x, \text{Var}(x), t)), \\ tu \mapsto \text{Con}(\text{Pair}(t, u)). \end{cases}$$