Bab VIII
Pemrograman Fungsional

Bahasa Pemrograman Fungsional :

- Disebut aplikatif karena fungsi yang diaplikasikan ke dalam argumentasi menjadi deklaratif dan non prosedural

- Didasarkan pada konsep matematika dari sebuah fungsi dan bahasa pemrograman fungsional, meliputi :
  - Suatu set fungsi primitif
  - Suatu set format fungsional
  - Aplikasi operasi
  - Suatu set objek data dan fungsi asosiasi
  - Suatu mekanisme untuk memberikan rujukan sebuah nama terhadap suatu fungsi

- Merupakan hasil dari fungsi meringkas dan men-generalisir type data dari peta

3 Komponen Primer Bahasa Fungsional :

- **Kumpulan objek data**
  Menggunakan mekanisme struktur data tingkat tinggi.
  Contoh : Array atau List

- **Kumpulan fungsi built-in**
  Untuk memanipulasi objek data dasar yang menyediakan sejumlah fungsi untuk membuat dan mengakses list.
  Contoh :
  LISP →
  - bahasa unuk komputasi simbolik, nilai direpresentasikan dengan ekspresi simbolik.
  - banyak digunakan di wilayah kecerdasan buatan (robotika, sistem cerdas).
  - biasa dieksekusi di bawah kendali interpreter

  Ekspresi terdiri dari atom atau list.
  Atom → string dan karakter (huruf, angka)
  Contoh : A68000

  List → urutan dari atom atau list, dipisahkan dengan spasi, ditutup dengan tanda kurung.
  Contoh : (PLUS AB)
  ((daging ayam) (sawi kangkung bayam) air))

  ML (Meta Language) →
  - Merupakan bahasa aplikatif dengan program-program yang ditulis menggunakan gaya C atau Pascal dan dengan konsep yang lebih advance tentang tipe data

- Mendukung polimorfisme dan abstraksi data
- Berjalan dengan interpreter

- **Kumpulan fungsional forms**
  Untuk membuat fungsi baru, yang mengizinkan programmer mendefinisikan operasi baru dari kombinasi fungsi yang ada.

Lambda Calculus

Adalah :
- Bahasa sederhana dengan ilmu semantik sederhana, ekspresif yang menyatakan semua fungsi dapat diperhitungkan
- Merupakan suatu bentuk formal dengan fungsi sebagai aturan

Contoh :
- Dengan ekspresi polynomial $X^2 + 3X - 5$
- Dengan fungsi lebih dari 1 variabel
  (+ x y) ditulis ((+ x)y) dimana fungsi (+ x) adalah fungsi yang menambahkan sesuatu ke x

Lambda Calculus murni mempunyai 3 buah elemen :

- Lambang primitif
- Aplikasi fungsi
- Fungsi ciptaan

Lambda calculus murni tidak mempunyai fungsi tetap atau konstanta

Kalkulasi dalam lambda calculus adalah :
Menulis ulang(mengurangi) suatu lambda-expression menjadi suatu format formal.

Ilmu Semantik Operasional

Inti denotasional Ilmu Semantik adalah : terjemahan dari program konvensional ke dalam persamaan fungsional.

Tujuan denotasional semantik dari suatu bahasa adalah : menugaskan suatu nilai kepada setiap ekspresi dalam bahasa.

Ilmu semantik dapat dinyatakan dalam lambda calculus sebagai fungsi mathematical, Eval, dari ekspresi ke nilai.
Contoh : Eval[+ 3 4] = 7 menggambarkan bahwa nilai ekspresi (+ 3 4) untuk menjadi 7

Fungsi Rekursif

Perluasan syntax Lambda-calculus yang mencakup ekspresi yang telah dinamai (named expressions).

L :: ...|x : L|...

X = nama dari Ekspresi Lambda L

```
      FAC : \n.(if (= n 0) 1 (* n (FAC (- n 1))))
            dengan syntactic sugaring :
    FAC : \n.if (n = 0) then 1 else (n * FAC (n – 1))

FAC : (\fac.(\n. (if (= n 0) (* n (fac (- n 1))))) FAC)

      H : \fac.(\n. (if (= n 0) 1 (* n (fac (- n 1)))))

            FAC : (H FAC)
```

Aturan Lingkup Leksikal

```
      let n : E in B
adalah penyingkatan untuk (\n.B) E

      let x : 3 in (* x x)

\y. let x : 3 in (* y x) Ekuivalen \y. (* y 3)

      letrec n : E in B
adalah peyingkatan untuk let n : Y (\n.E) in B

let n : E in B          = (\n.B) E
letrec n : E in B       = let n : Y (\n.E ) in B
```

Semantic Translasi dan Kombinator

```
Kombinator :
S = \f.(\g.(\x. f x (g x)))
K = \x. \y. x
I = \x.x
Y = \f. \x. (f (x x)) \x. (f (x x))

Aturan reduksi untuk kalkulus SKI adalah :
S f g x → f x (g x)
K c x   → c
I x     → x
Y e     → e (Y e)
(A B)   → A B
(A B C) → A B C
```

- Aturan Reduksi dijalankan dari kanan ke kiri
- Jika tidak ada reduksi S,K,I,Y maka tanda kurung akan dibuang, dan proses reduksi diteruskan

```
Semantik Translasi untuk Lambda Calculus :
Compile [s]             → s
Compile [(E1 E2)]       → (Compile [E1] Compile [E2])
Compile [\x.E]          → Abstract [(x, Compile [E])]
Abstract [(x, s)]       → if (s = x) then I else (K s)
Abstract [(x, (E1 E2))] → ((S Abstract [(x, E1)]) Abstract [(x, E2)])
dimana s adalah symbol
```

Scheme

- Turunan dari LISP, didasarkan pada Lambda Calculus. Dikonsentrasikan ke fitur lambda-calculus

- Scheme mempunyai dua object :
  - Atoms : Untaian Karakter yang bukan blank
  - List : Rangkaian Atom atau List dipisahkan oleh blank dan berada dalam tanda

- Sebuah fungsi dapat terbuat atas fungsi yang lain dan dapat diaplikasikan pada list atau argumen

Dari Sisi Sejarah

Alternatif teori dasar matematika

- Alonso Church, lambda-calculus, 1930-an
- Haskell B. Curry, logika kombinatorial

1958, LISP (LISt Processing), pemrosesan list berdasarkan fungsi rekursif.
- Recursion
- First class function
- Garbage collection

1960an, penggunaan lambda-calculus di dalam ilmu computer → Semantik Denotasional
***Teori Semantik Formal untuk bahasa pemrograman (Peter Landin, Christopher* Strachy, dll)**

1969, Model Matematika pertama untuk lambda-calculus bertipe bebas (Dana Scott)

Haskell
- Bahasa modern yang dinamai sama dengan Haskell B. Curry
- Didesain oleh 15 orang anggota komite internasional
- Pembentukan bahasa fungsional yang memasukkan :
  - ide-ide baik yang sebelumnya ada dalam riset bahasa fungsional
  - yang sesuai untuk pengajaran, riset dan aplikasi
  - Fasilitas Overloading, yang dipadukan dengan sistem bertipe polimorfis, i/o fungsional, abstraksi data dan penyembunyian informasi

Functional Programming urutan mesin virtual.
- Bahasa pemrograman fungsional → lambda calculus
- Lambda calculus → logika kombinatorial
- Logika kombinatorial → kode mesin reduksi graf

Kesemuanya adalah mesin virtual