ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ

Лекція 10

ТЕОРІЯ ФУНКЦІОНАЛЬНИХ ПРОГРАМ

2020

Повний текст лекції буде розміщений на сайті baklaniv.at.ua

Основні задачі теорії функціональних програм

Основні завдання, для вирішення яких призначена теорія функціональних програм можна розділити на наступні три класи.

1. Верифікація різних властивостей функціональних програм (до числа яких відносяться властивості коректний, безпеки, оптімальності і т.п.)
2. Оптимізує перетворення функціональних програм (тобто перетворення їх в такі програми, які обчислює ті ж самі функції, що і вихідні програми, але при тому працює швидше, ніж вихідні програми).

3. Автоматизація проектування (синтез) функціональних програм.

Метод функціонального програмування. Опис методу

Метод функціонального програмування полягає в побудові опису обчислюваних функцій у вигляді систем функціональних рівнянь, рішеннями яких повинні бути описані функції.
Цей метод альтернативний по відношенню до методу імперативного програмування, при якому обчислювана функція визначається шляхом вказівки дій, що перетворюють вхідні дані у вихідні.

Описи обчислюваних функцій, побудовані на основі методу функціонального програмування, називаються *функціональними програмами (ФП)*.

Метод функціонального програмування є одним з найбільш ефективних інструментів програмування, він дозволяє розробляти в короткі терміни легко розуміються і надійні програми.

Функціональне програмування можна також використовувати для

- опису специфікацій, тобто властивостей розроблюваних програм, до числа яких відносяться, наприклад, властивості коректності, оптимальності, безпеки, стійкості, і т.д., а також
- швидкого створення прототипів необхідних функцій, які можна потім еалізовивать більш ефективно на мовах програмування низького рівня.

У другому випадку опис функції у вигляді ФП розглядається

- не як опис реального процесу обчислення її значень,
- а як еталон, призначений для контролю правильності реалізації тієї функції на мові програмування низького рівня.

Приклади функціональних програм

Наведена нижче ФП описує функцію конкатенації рядків

```
append : S × S → S
```

яка має два аргументи.

Ця функція перетворює пару рядків (u, v) в їх конкатенацію, тобто

- у рядок a1. . . an b1. . . bm, якщо u і v мають вигляд відповідно

a1. . . an і b1. . . bm (n, m> 0)

- рядок v, якщо рядок u порожній,
- рядок u, якщо рядок v порожній.

ФП, що описує функцію append, має вигляд

$$\varphi(u, v) = \begin{array}{l} \textbf{if } eq(u, \varepsilon) \textbf{ then } v \\ \textbf{else } conc(head(u), \varphi(tail(u), v)) \end{array} \quad (1)$$

де символ ε позначає порожній рядок.

ФП (1) складається з одного функціонального рівняння, невідома величина в якому функціональна змінна φ.

Ліва частина рівняння (1) складається з

- функціональної змінної φ, що відповідає описуваної функції, і
- списку формальних параметрів цієї функції (u і v).

Права частина рівняння (1) являє собою вираз, що описує зв'язок значення описуваної функції на парі аргументів
(u, v)
з її значеннями на інших аргументах.

Словесно цей зв'язок можна виразити таким чином: значенням описуваної функції на парі аргументів (u, v) є

- рядок v, якщо u порожня, і
- рядок, що складається з першого елемента рядка u, за яким слідує рядок, що є значенням описуваної функції на парі (Tail (u), v), якщо u непорожній.

Неважко довести, що функція append є єдиним рішенням рівняння (1). Рівняння (1) можна розглядати як рекурсивний алгоритм обчислення функції append.

Іншим прикладом є ФП, що описує функцію інвертування рядки

reverse: S → S

яка перетворює кожну рядок u в рядок, що отримується з u його записом в зворотному порядку, тобто

reverse (a1... an) = an. . . . a1

ФП, що описує функцію reverse, має вигляд

φ (u) = if eq (u, ε) then ε
else append (φ (tail (u)), head (u))

У цій ФП використовується функція append, яку ми описали у вигляді ФП в попередньому параграфі.

Третім прикладом є ФП, що описує функцію пошуку підрядка

find : S x S → C

Значенням даної функції на двох рядках u, v є символ

- 1, якщо u є підрядком рядка v, тобто якщо v є значенням виразу append (x, append (u, y)) для деяких рядків x, y, і
- 0, інакше.

ФП, що описує функцію find, має вигляд

$$
\begin{cases}
\varphi_1(u,v) = & \textbf{if } \varphi_2(u,v) \textbf{ then } 1 \\
 & \textbf{else} \begin{pmatrix} \textbf{if } eq(v,\varepsilon) \textbf{ then } 0 \\ \textbf{else } \varphi_1(u, tail(v)) \end{pmatrix} \\
\\
\varphi_2(u,v) = \\
\quad = \quad \textbf{if } eq(u,\varepsilon) \textbf{ then } 1 \\
\qquad \textbf{else} \begin{pmatrix} \textbf{if } eq(v,\varepsilon) \textbf{ then } 0 \\ \textbf{else} \begin{pmatrix} \textbf{if } eq\left(head(u), head(v)\right) \\ \textbf{then } \varphi_2(tail(u), tail(v)) \\ \textbf{else } 0 \end{pmatrix} \end{pmatrix}
\end{cases}
$$

Дана ФП являє собою систему з двох функціональних рівнянь. Ця система має єдине рішення, яке представляє собою пару функцій

(Find, prefix)

(Find відповідає φ1, а prefix φ2), де
• функція find є тією функцією, для опису якої призначена дана ФП, і
• функція prefix є допоміжною функцією, вона має вигляд

prefix: S x S → C

її значенням на парі рядків u, v є символ

1, якщо u є префіксом рядки v, тобто якщо v є значенням виразу append (u, x) для деякого рядка x, і

0, інакше.

Функціональні програми

У цьому параграфі ми формально визначаємо поняття функціональної програми. Для цього ми спочатку вводимо необхідні синтаксичні конструкції.

Домени і функції на доменах

Типи і домени

Ми припускаємо, що задано множина Types типів даних (або просто типів). З кожним типом t пов'язано непорожня множина Dt, що називається доменом типу t. Елементи домену Dt називаються значеннями типу t.

Нижче ми перераховуємо деякі типи і вказуємо відповідні їм домени.

- `char`, домен типу `char` складається з символів
- `string`, домен типу `string` складається з символьних рядків
- `int`, домен типу `int` складається з цілих чисел
- `nat`, домен типу `nat` складається з натуральних чисел (`0, 1, 2,` і т.д.)

- bool, домен типу bool складається з двох елементів, які називаються істина і брехня, і позначаються символами **>** і $\perp$ відповідно.

Ми будемо позначати символами C, S, Z, N і B домени типів char, string, int, nat і bool відповідно.

Поповнені домени

Для кожного домена D запис D позначає множину, що складається з

- всіх елементів домену D, і
- ще одного елемента, який не входить в D, позначається символом $\boldsymbol{\omega}$ (для позначення цього елемента використовується один і той же символ ω для всіх доменів), і називається *невизначеним значенням*.

Для кожного типу t множина D_t називається поповненим доменом типу t. Ми будемо розглядати D_t як *частково впорядкована множина (ЧВМ)*, відношення порядку на якому визначається наступним чином:

$\forall\ d_1, d_2 \in D_t$

$d_1 \leq d_2 \Leftrightarrow d_1 = \omega$ або $d_1 = d_2$ (2)

Для кожного списку D1,. . . , Dn
поповнених доменів ми будемо
розглядати їх декартовий добуток
D1 x. . . x Dn (3)
теж як ЧВМ, відношення порядку на
якому визначається наступним чином:
для будь-яких списків
d ¯i d ¯0 з множини (3), де
d ¯= (d1,..., dn),
d ¯0 = (d0 1,..., d0 n)
ми будемо вважати d ¯≤ d ¯0, якщо
d1 ≤ d0 1,. . . , Dn ≤ d0 n

Монотонні функції

Функція f виду

$f: D1 \times . . . \times Dn \to D$ (4)

називається монотонною, якщо для кожної пари списків

$d\bar{}, d\bar{}0 \in D1 \times . . . \times Dn$

вірна імплікація

$d\bar{} \le d\bar{}0 \Rightarrow f(d\bar{}) \le f(d\bar{}0)$

Прикладом немонотонної функції є функція несуворої рівності, яка має вигляд

$D \times D \to B$

і зіставляє кожній парі

(d1, d2) ∈ D x D значення

$$(d_1 \equiv d_2) \stackrel{\text{def}}{=} \begin{cases} \top & \text{если } d_1 = d_2 \\ \bot & \text{иначе} \end{cases}$$

Немонотонність цієї функції випливає з того, що

- $(\omega, \omega) \leq (\omega, d)$, где $d \in D$, но
- $(\omega \equiv \omega) = \top, \quad (\omega \equiv d) = \bot, \quad \top \not\leq \bot.$

Природні продовження

Нехай задана часткова функція f виду

$f: D1 \times . . . \times Dn \rightarrow D$ (5)

де $D1, . . . , Dn$, D деякі домени.

Ми будемо позначати тим же символом f тотальну (тобто всюди визначену) функцію вигляду

$f: D1\ x. \ . \ . \ x\ Dn \rightarrow D$ (6)

Яка визначається наступним чином: для будь-якого списку

$\bar{d} = (d1, ..., dn) \in D1 \times . . . \times Dn$

• якщо $\forall i = 1,...,N\ di \neq \omega$, тобто

$\bar{d} \in D1 \times . . . \times Dn$

то значення функції (6) на $\bar{d}$ буде значення часткової функції (5) на $\bar{d}$, якщо це значення визначено елементу ω, інакше

• якщо $\exists\ i \in \{1, . . . , N\}: di = \omega$, то значення функції (6) на $\bar{d}$ дорівнює ω.

Функція (6) називається *природним продовженням часткової функції* (5). Неважко довести, що природне продовження будь-якої часткової функції є монотонною функцією.

Частково впорядковані множини монотонних функцій

Функціональний тип (ФТ) це запис виду

$(T1, ..., Tn) \to t$ (7)

де $n \geq 0$, і $t1, ..., tn, t \in Types$.

У разі $n = 0$ та частина ФТ (7), яка знаходиться перед стрілкою, є марною послідовністю.

Для кожного ФТ $f\ t = (t1,..., tn) \to t$ знакосочетаній $Df\ t$ позначає множину всіх монотонних функцій f виду

$f: Dt1\ x.\ .\ .\ x\ Dtn \to Dt$ (8)

У разі $n = 0$ область визначення функції (8) є одноелементна множина.

Ми будемо розглядати $D_{f\ t}$ як ЧВМ, відношення порядку на якому визначається наступним чином: для будь-яких $f_1, f_2 \in D_{f\ t}$ ми будемо вважати $f_1 \leq f_2$, якщо

$$\forall\, \bar{d} \in D_{t1} \times \ldots \times D_{tn} \quad f_1(\bar{d}) \leq f_2(\bar{d})$$

де відношення нерівності на D_t розуміється в сенсі визначення (2). Позначимо символом ω_m функцію з $D_{f\ t}$, приймаючу на кожному своєму аргументі значення ω.

Неважко бачити, що

$\forall f \in Df\ t \quad \omega\ m \leq f$ (9)

Ланцюг в $Df\ t$ це послідовність $\{fi \mid i \geq 0\}$ елементів $Df\ t$, що задовольняють умові:

$f0 \leq f1 \leq f2 \leq \ldots$

Ця умова еквівалентно тому, що для кожного d ¯з множини

$Dt1\ x.\ .\ .\ x\ Dtn$ (10)

правильне співвідношення

$f0\ (d)\ \bar{}\leq f1\ (d)\ \bar{}\leq f2\ (d)\ \bar{}\leq \ldots$(11)

Згідно з визначенням відносини порядку на поповнених доменах (див. (2)), співвідношення (11) еквівалентно такій умові:

- або $\forall\ i \geq 0\ f_i(d)\ \bar{=}\ \omega$,
- або $\exists\ i \geq 0,\ \exists\ d_0 \in D_t$:

$\forall j < i\ f_j(d)\ \bar{=}\ \omega$

$\forall j \geq i\ f_j(d)\ \bar{=}\ d_0$ (12)

Позначимо записом

$\sup\{f_i \mid i \geq 0\}$

функцію виду (8), яка визначається наступним чином: для кожного списку

d ¯ з множини (10)

Неважко довести, що функція $\sup \{f_i \mid i \geq 0\}$

- монотонна (тобто належить $Df\ t$), і
- має такі властивості:

$\forall\ i \geq 0\ f_i \leq \sup \{f_i \mid i \geq 0\}$

якщо для деякої функції $f \in Df\ t$ правильне співвідношення $\forall\ i \geq 0\ f_i \leq f$ то $\sup \{f_i \mid i \geq 0\} \leq f$.

Повні частково впорядковані множини

ЧВМ P називається *повним*, якщо

• P містить найменший елемент, тобто такий елемент 0, що

$\forall p \in P \; 0 \leq p$

• для кожного ланцюга $p0 \leq p1 \leq p2 \leq . . .$ елементів P існує точна верхня грань цьому ланцюзі, тобто такий елемент $\sup \{pi \mid i \geq 0\} \in P$ (14)

який має такі властивості

$\forall i \geq 0 \; pi \leq \sup \{pi \mid i \geq 0\}$

якщо елемент $p0 \in P$ такий, що

$\forall\ i \geq 0\ p_i \leq p_0$
то $\sup \{p_i \mid i \geq 0\} \leq p_0$.
Нижче ми будемо позначати елемент виду (14) більш коротким записом $\sup p_i$.
З міркувань в кінці попереднього пункту випливає, що $Df\ t$ повне ЧВМ. З (9) випливає, що найменшим елементом $Df\ t$ є функція $\omega\ m$.

На наступній лекції ми розглянемо поняття термів ФП і перейдемо до формального визначення функціональної програми.