Верификация функциональных программ методом построения диаграмм состояний

Миронов А.М.[1]

В статье вводятся графовые объекты, соответствующие функциональным программам, называемые диаграммами состояний. Показано, как можно использовать диаграммы состояний для решения задач верификации функциональных программ. Предлагаемый подход иллюстрируется примером верификации функциональной программы сортировки.

Ключевые слова: верификация программ, функциональные программы, диаграммы состояний.

1. Введение

Проблема **верификации программ**, которая заключается в доказательстве утверждений о том что анализируемые программы обладают заданными свойствами, является одной из основных проблем теоретической информатики. Для различных классов программ используются различные методы верификации. Например, для верификации последовательных программ используются метод индуктивных утверждений Флойда [2], логика Хоара [3], и т.д. Для верификации параллельных и распределенных программ используются методы, основанные на исчислении взаимодействующих систем (CCS) и π-исчислении Милнера [4], [5], теория взаимодействующих последовательных процессов Хоара (CSP) и ее обобщения [6], [7], темпоральная логика и model checking [8], процессная алгебра [9], сети Петри [10], и т.д.

[1] *Миронов Андрей Михайлович* — к.ф.-м.н., доцент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ имени М.В.Ломоносова, e-mail: amironov66@gmail.com.

Mironov Andrey Mihailovich — Candidate of Physical and Mathematical Sciences, Associate Professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intelligent Systems.

Основными методами верификации функциональных программ (ФП) являются вычислительная индукция и структурная индукция [1]. Главные проблемы при использовании этих методов связаны с высокой сложностью построения формальных доказательств корректности анализируемых ФП. Существуют также другие методы верификации ФП, отметим среди них следующие: методы, основанные на анализе потоков управления [11], методы, основанные на рассуждениях с типами данных, абстрактной интерпретации и выводе типов [12], методы, основанные на использовании подхода model checking [13], [14], методы, основанные на понятии мультипараметрического преобразователя деревьев [15].

В настоящей работе мы рассматриваем ФП как системы алгебраических уравнений над строками. Мы вводим понятие диаграммы состояний для таких ФП, и излагаем метод верификации, основанный на диаграммах состояний. Основные преимущества излагаемого подхода по сравнению со всеми вышеперечисленными подходами к верификации ФП связаны с тем, что данный подход позволяет представить доказательства корректности ФП в виде простых свойств их диаграмм состояний.

Основная идея предлагаемого подхода заключается в следующем:

- предполагается, что спецификация свойств анализируемой ФП Σ определяется другой ФП Σ', входными значениями для которой являются выходные значения ФП Σ,

- ФП Σ считается корректной относительно спецификации, выражаемой ФП Σ', если и только если суперпозиция $f_{\Sigma'}(f_{\Sigma})$ функций, соответствующих ФП Σ и Σ', принимает значение 1 на всех своих аргументах, данное утверждение обозначается записью

$$f_{\Sigma'}(f_{\Sigma}) = 1 \tag{1}$$

- обоснование утверждения (1) сводится к анализу диаграммы состояний для ФП $\Sigma'(\Sigma)$, которой соответствует суперпозиция $f_{\Sigma'}(f_{\Sigma})$.

Предложенный метод верификации ФП иллюстрируется примером верификации ФП сортировки вставкой. Сначала мы представляем полное доказательство корректности этой ФП на основе структурной индукции. Это делается для сравнения сложности верификации ФП на основе метода структурной индукции, и сложности предлагаемого метода верификации ФП на основе диаграмм состояний. Далее, мы представляем доказательство корректности ФП методом, основанным на построении

диаграмм состояний. Доказательство вторым методом может быть порождено автоматически. Это демонстрирует преимущества предлагаемого метода верификации ФП по сравнению с верификацией на основе метода структурной индукции.

2. Основные понятия

2.1. Термы

Мы предполагаем, что задано множество $\mathcal{D}$ **значений**, и каждое значение из $\mathcal{D}$ имеет один из трех типов: **C**, **S** или **B**. Значения типа **C** называются **символами**, значения типа **S** называются **символьными строками** (или просто **строками**), значения типа **B** называются **булевыми значениями**. Существует два булевых значения: $\top$ (истина) и $\bot$ (ложь). Каждое значение типа **S** представляет собой конечную (возможно пустую) последовательность значений типа **C**. Совокупности значений типа **C**, **S** и **B** обозначаются записями $\mathcal{D}_{\mathbf{C}}$, $\mathcal{D}_{\mathbf{S}}$ и $\mathcal{D}_{\mathbf{B}}$ соответственно.

Также предполагаем, что заданы множества

- $\mathcal{X}$ **переменных по данным** (или просто **переменных**),
- $\mathcal{C}$ **констант**,
- $\mathcal{F}$ **функциональных символов (ФС)**, и
- Φ **функциональных переменных**.

Каждый элемент x какого-либо из вышеперечисленных множеств связан с **типом**, обозначаемым знакосочетанием $\tau(x)$, и

- если $x \in \mathcal{X}$ или $x \in \mathcal{C}$, то $\tau(x) \in \{\mathbf{C}, \mathbf{S}, \mathbf{B}\}$,
- если $x \in \mathcal{F}$ или $x \in \Phi$, то $\tau(x)$ – это запись вида $(t_1, \ldots, t_n) \to t$, где $t_1, \ldots, t_n, t \in \{\mathbf{C}, \mathbf{S}, \mathbf{B}\}$.

Каждой константе $c \in \mathcal{C}$ соответствует элемент множества $\mathcal{D}_{\tau(c)}$, называемый **значением** этой константы. Символ ε обозначает константу типа **S**, значением которой является пустая строка. Существуют константы типа **B**, которым соответствуют значения $\top$ и $\bot$, данные константы обозначаются теми же символами $\top$ и $\bot$ соответственно.

Каждому ФС $f \in \mathcal{F}$ соответствует частичная функция, обозначаемая тем же символом f, и имеющая вид

$$f : \mathcal{D}_{t_1} \times \ldots \times \mathcal{D}_{t_n} \to \mathcal{D}_t, \quad \text{где } \tau(f) = (t_1, \ldots, t_n) \to t.$$

Ниже мы перечисляем некоторые ФС из $\mathcal{F}$, рядом с каждым ФС мы указываем (через двоеточие) его тип.

1) $head : \mathbf{S} \to \mathbf{C}$. Функция $head$ определена на непустых строках, она отображает каждую непустую строку в ее первый элемент (т.е. если строка u имеет вид $a_1 \dots a_n$, то $head(u) = a_1$).

2) $tail : \mathbf{S} \to \mathbf{S}$. Функция $tail$ определена на непустых строках, она отображает каждую непустую строку u в строку (называемую **хвостом** строки u), получаемую из u удалением ее первого элемента.

3) $conc : (\mathbf{C}, \mathbf{S}) \to \mathbf{S}$. Для каждой пары $(a, u) \in \mathcal{D}_{\mathbf{C}} \times \mathcal{D}_{\mathbf{S}}$ строка $conc(a, u)$ получается путем приписывания символа a в начало строки u.

4) $=: (t, t) \to \mathbf{B}$, где $t \in \{\mathbf{C}, \mathbf{S}, \mathbf{B}\}$, т.е. символ $=$ обозначает три ФС. Значение функции $=$ на паре (x, y) равно $\top$, если x и y совпадают, и $\bot$, иначе.

5) $\leq: (\mathbf{C}, \mathbf{C}) \to \mathbf{B}$. Мы предполагаем, что $\mathcal{D}_{\mathbf{C}}$ – линейно упорядоченное множество, и значение функции $\leq$ на паре (a, b) равно $\top$, если $a \leq b$, и $\bot$, иначе.

6) Булевы ФС: $\neg : \mathbf{B} \to \mathbf{B}$, $\wedge : (\mathbf{B}, \mathbf{B}) \to \mathbf{B}$, и т.д., соответствующие функции являются стандартными булевыми функциями на аргументах $\top$ и $\bot$ (т.е. $\neg(\top) = \bot$, и т.д.).

7) $if\_then\_else : (\mathbf{B}, t, t) \to t$, где $t \in \{\mathbf{C}, \mathbf{S}, \mathbf{B}\}$, т.е. запись if_then_else обозначает три ФС. Функции, соответствующие этим ФС, определяются одинаково:

$$if\_then\_else\ (a, x, y) \stackrel{\text{def}}{=} \begin{cases} x, & \text{если } a = \top, \\ y, & \text{если } a = \bot. \end{cases}$$

Понятие **терма** определяется индуктивно. Каждый терм e связан с некоторым типом $\tau(e) \in \{\mathbf{C}, \mathbf{S}, \mathbf{B}\}$. Определение терма имеет следующий вид:

- каждая переменная по данным и каждая константа является термом, тип которого равен типу этой переменной или константы,

- если f – ФС или функциональная переменная, $e_1, \ldots, e_n$ термы, и

$$\tau(f) = (\tau(e_1), \ldots, \tau(e_n)) \to t,$$

  то запись $f(e_1, \ldots, e_n)$ – терм типа t.

Будем использовать следующие понятия и обозначения.

- Множество всех термов будем обозначать символом $\mathcal{E}$.
- $\forall\, e, e' \in \mathcal{E}$ e' называется **подтермом** терма e, если либо $e' = e$, либо e имеет вид $f(e_1, \ldots, e_n)$, и $\exists\, i \in \{1, \ldots, n\} : e'$ – подтерм терма e_i.
- $\forall\, e \in \mathcal{E}$ записи X_e, Φ_e обозначают множества всех переменных по данным и функциональных переменных соответственно, входящих в e.
- $\forall\, X \subseteq \mathcal{X}$ запись $\mathcal{E}_X$ обозначает множество $\{e \in \mathcal{E} \mid X_e \subseteq X\}$.
- Будем обозначать термы

$$\mathit{head}(e),\ \mathit{tail}(e),\ \mathit{conc}(e, e'), = (e, e'),\ \leq (e, e'),\ \mathit{if\_then\_else}\ (e, e', e'')$$

  записями e_h, e_t, ee', $e = e'$, $e \leq e'$, $[\![e]\!]\, e' : e''$, соответственно.
- Терм называется **простым**, если он имеет вид $e_1 \ldots e_n$, где $\forall\, i = 1, \ldots, n$ e_i – переменная или константа.
- Термы, содержащие булевы ФС, будут обозначаться так же как в математических текстах (т.е. в виде записей $e \wedge e'$, и т.д.), термы вида $e_1 \wedge \ldots \wedge e_n$ могут также обозначаться записями вида $\left\{ \begin{array}{c} e_1 \\ \ldots \\ e_n \end{array} \right\}$,
- $\forall\, e \in \mathcal{E}$ запись $\mathcal{D}_e$ обозначает множество $\mathcal{D}_{\tau(e)}$.
- Списки термов обозначаются записями вида $\bar{e}$.
- Если $\bar{e}$ – список термов вида $(e_1, \ldots, e_n)$, то
  - $\tau(\bar{e})$ обозначает список ($\tau(e_1)$, ..., $\tau(e_n)$),
  - $X_{\bar{e}}$, $\Phi_{\bar{e}}$ обозначают множества $\bigcup_{i=1}^{n} X_{e_i}$, $\bigcup_{i=1}^{n} \Phi_{e_i}$ соответственно,
  - $\mathcal{D}_{\bar{e}}$ обозначает множество $\mathcal{D}_{e_1} \times \ldots \times \mathcal{D}_{e_n}$.

- Если $\bar{e}' = (e'_1, \ldots, e'_n)$, $\bar{e}'' = (e''_1, \ldots, e''_n)$ – списки термов, $\tau(\bar{e}') = \tau(\bar{e}'')$, то
  - запись $\bar{e}' = \bar{e}''$ обозначает терм $(e'_1 = e''_1) \wedge \ldots \wedge (e'_n = e''_n)$,
  - если дополнительно предполагается, что для каждой пары i, j различных индексов из $\{1, \ldots, n\}$ терм e'_i не является подтермом терма e''_j, то
    * $\forall\, e \in \mathcal{E}$ запись
    $$e[e''_1/e'_1, \ldots, e''_n/e'_n] \tag{2}$$
    обозначает терм, получаемый из e заменой $\forall\, i = 1, \ldots, n$ каждого подтерма терма e, совпадающего с e'_i, на терм e''_i, терм (2) обозначается также записью $e[\bar{e}''/\bar{e}']$,
    * для каждого списка термов $\bar{e} = (e_1, \ldots, e_m)$ запись $\bar{e}[\bar{e}''/\bar{e}']$ обозначает терм
    $$(e_1[\bar{e}''/\bar{e}'], \ldots, e_m[\bar{e}''/\bar{e}']).$$

- **Уточнением** называется запись θ вида
$$e_1/x_1, \ldots, e_n/x_n, \tag{3}$$
где $x_1, \ldots, x_n$ – различные переменные, $e_1, \ldots, e_n$ – простые термы, причем $\forall\, i = 1, \ldots, n \;\; \tau(x_i) = \tau(e_i)$. $\forall\, e \in \mathcal{E}$ запись $e[\theta]$ обозначает терм $e[e_1/x_1, \ldots, e_n/x_n]$ (аналогичные записи используются, когда вместо терма e рассматривается список термов). (3) называется **переименованием**, если $e_1, \ldots, e_n$ – различные переменные.

2.2. Понятие функциональной программы

В настоящем тексте под **функциональной программой (ФП)** понимается конечная совокупность Σ равенств вида

$$\begin{cases} \varphi_1(x_{11}, \ldots, x_{1n_1}) = e_1 \\ \ldots \\ \varphi_m(x_{m1}, \ldots, x_{mn_m}) = e_m \end{cases} \tag{4}$$

где $\varphi_1, \ldots, \varphi_m$ – различные функциональных переменные, и для каждого $i = 1, \ldots, m \;\; \varphi_i(x_{i1}, \ldots, x_{in_i})$ и e_i – термы одинакового типа, причем

$$X_{e_i} = \{x_{i1}, \ldots, x_{in_i}\}, \quad \Phi_{e_i} \subseteq \{\varphi_1, \ldots, \varphi_m\}.$$

Главным термом ФП (4) называется левая часть первого равенства в (4) (т.е. терм $\varphi_1(x_{11}, \ldots, x_{1n_1})$).

Совокупность равенств, входящих в ФП (4), можно рассматривать как систему функциональных уравнений относительно функциональных переменных $\varphi_1, \ldots, \varphi_m$. Данная система определяет список

$$(f_{\varphi_1}, \ldots, f_{\varphi_m}) \tag{5}$$

частичных функций, соответствующих функциональным переменным $\varphi_1, \ldots, \varphi_m$, который является наименьшим (в смысле порядка на списках частичных функций, описанного в [1]) решением системы функциональных уравнений (4). Список (5) называется **наименьшей неподвижной точкой (ННТ)** ФП (4). Все детали, связанные с понятием ННТ ФП, м.б. найдены в книге [1]. Первая фунция в списке (5) (т.е. f_{φ_1}) обозначается записью f_Σ, и называется **функцией, определяемой ФП** Σ.

Пусть задана ФП Σ. Запись $\mathcal{E}_\Sigma$ обозначает множество всех термов, таких, что все входящие в них функциональные переменные входят в Σ.

Будем считать ФП Σ и Σ' одинаковыми, если Σ' получается из Σ переименованием переменных по данным и функциональных переменных, т.е. если $X\Phi_\Sigma$ и $X\Phi_{\Sigma'}$ – множества всех переменных по данным и функциональных переменных, входящих в Σ и Σ' соответственно, то существует взаимно однозначное соответствие $f : X\Phi_\Sigma \to X\Phi_{\Sigma'}$, такое, что Σ' получается из Σ заменой каждой переменной $v \in X\Phi_\Sigma$ на $f(v)$.

3. Пример спецификации и верификации функциональной программы

3.1. Пример функциональной программы

Рассмотрим следующую ФП:

$$\left\{\begin{array}{l} \mathbf{sort}(x) = [\![x = \varepsilon]\!]\ \varepsilon : \mathbf{insert}(x_h, \mathbf{sort}(x_t)) \\ \mathbf{insert}(a, y) = [\![y = \varepsilon]\!] \quad a\varepsilon \\ \qquad\qquad\qquad\qquad\ : \quad [\![a \le y_h]\!] \quad ay \\ \qquad\qquad\qquad\qquad\qquad\quad : y_h\ \mathbf{insert}(a, y_t) \end{array}\right. \tag{6}$$

Эта ФП определяет функцию сортировки на строках. ФП состоит из двух уравнений, которые определяют следующие функции:

- $\mathbf{sort} : \mathbf{S} \to \mathbf{S}$ – основная функция, и

- **insert** : $(\mathbf{C}, \mathbf{S}) \to \mathbf{S}$ – вспомогательная функция, отображающая пару $(a, y) \in \mathcal{D}_{\mathbf{C}} \times \mathcal{D}_{\mathbf{S}}$ в строку, получаемую вставкой символа a в строку y, причем выполнено следующее условие: если строка y упорядочена, то строка $\mathbf{insert}(a, y)$ тоже упорядочена (строка упорядочена, если ее компоненты образуют неубывающую последовательность).

3.2. Пример спецификации функциональной программы

Одно из свойств корректности ФП (6) имеет следующий вид: $\forall\, x \in \mathcal{D}_{\mathbf{S}}$ строка $\mathbf{sort}(x)$ упорядочена. Это свойство м.б. выражено формально. Рассмотрим ФП, определяющую функцию **ord** проверки упорядоченности строк:

$$\begin{array}{lll} \mathbf{ord}(x) = [\![x = \varepsilon]\!] & 1 & \\ & : [\![x_t = \varepsilon]\!] & 1 \\ & & : [\![x_h \le (x_t)_h]\!] \quad \mathbf{ord}(x_t) : 0 \end{array} \tag{7}$$

Функция **ord** позволяет выразить описанное выше свойство корректности в виде следующего математического утверждения:

$$\forall\, x \in \mathcal{D}_{\mathbf{S}} \quad \mathbf{ord}(\mathbf{sort}(x)) = 1. \tag{8}$$

3.3. Пример верификации функциональной программы

Проблема верификации свойства корректности (8) ФП (6) заключается в построении формального доказательства утверждения (8). Это утверждение может быть доказано как обычная математическая теорема, путем использования метода математической индукции. Например, доказательство этого утверждения может иметь следующий вид.

Если $x = \varepsilon$, то, согласно первому уравнению системы (6), равенство $\mathbf{sort}(x) = \varepsilon$ верно, и поэтому

$$\mathbf{ord}(\mathbf{sort}(x)) = \mathbf{ord}(\varepsilon) = 1.$$

Пусть $x \neq \varepsilon$. Докажем (8) для этого случая по индукции.

Предположим что для каждой строки y, длина которой меньше, чем длина строки x, равенство $\mathbf{ord}(\mathbf{sort}(y)) = 1$ верно. Докажем, что тогда

$$\mathbf{ord}(\mathbf{sort}(x)) = 1. \tag{9}$$

(9) равносильно равенству

$$\mathbf{ord}(\ \mathbf{insert}(x_h, \mathbf{sort}(x_t))) = 1. \tag{10}$$

По индуктивному предположению, $\mathbf{ord}(\mathbf{sort}(x_t)) = 1$, откуда следует (10) на основании следующей леммы.

Лемма.

Имеет место импликация:

$$\mathbf{ord}(y) = 1 \quad \Rightarrow \quad \mathbf{ord}(\mathbf{insert}(a, y)) = 1 \tag{11}$$

Доказательство.

Индукция по структуре y.

Если $y = \varepsilon$, то правая часть (11) имеет вид $\mathbf{ord}(a\varepsilon) = 1$, что верно по определению **ord**.

Пусть $y \neq \varepsilon$, и для каждой строки z, длина которой меньше чем длина y, верна импликация:

$$\mathbf{ord}(z) = 1 \quad \Rightarrow \quad \mathbf{ord}(\mathbf{insert}(a, z)) = 1 \tag{12}$$

Пусть $c \stackrel{\text{def}}{=} y_h$, $d \stackrel{\text{def}}{=} y_t$. Тогда (11) имеет вид

$$\mathbf{ord}(cd) = 1 \quad \Rightarrow \quad \mathbf{ord}(\mathbf{insert}(a, cd)) = 1 \tag{13}$$

Для доказательства импликации (13) необходимо доказать, что если $\mathbf{ord}(cd) = 1$, то верны следующие импликации:

(a) $a \leq c \quad \Rightarrow \quad \mathbf{ord}(a(cd)) = 1$,

(b) $c < a \quad \Rightarrow \quad \mathbf{ord}(c\ \mathbf{insert}(a, d)) = 1$.

(a) верно, т.к. из $a \leq c$ следует, что $\mathbf{ord}(a(cd)) = \mathbf{ord}(cd) = 1$.

Докажем (b).

- $d = \varepsilon$. В этом случае правая часть (b) имеет вид

$$\mathbf{ord}(c(a\varepsilon)) = 1 \tag{14}$$

  (14) следует из неравенства $c < a$.

- $d \neq \varepsilon$. Пусть $p \stackrel{\text{def}}{=} d_h$, $q \stackrel{\text{def}}{=} d_t$.

  В этом случае необходимо доказать, что если $c < a$, то

$$\mathbf{ord}(c\ \mathbf{insert}(a, pq)) = 1 \tag{15}$$

1) если $a \leq p$, то (15) имеет вид

$$\mathbf{ord}(c(a(pq))) = 1 \tag{16}$$

Т.к. $c < a \leq p$, то (16) следует из равенств

$$\mathbf{ord}(c(a(pq))) = \mathbf{ord}(a(pq)) = \mathbf{ord}(pq) =$$
$$= \mathbf{ord}(c(pq)) = \mathbf{ord}(cd) = 1$$

2) если $p < a$, то (15) имеет вид

$$\mathbf{ord}(c(p\ \mathbf{insert}(a, q))) = 1 \tag{17}$$

Т.к., по предположению

$$\mathbf{ord}(cd) = \mathbf{ord}(c(pq)) = 1$$

то $c \leq p$, и поэтому (17) можно переписать в виде

$$\mathbf{ord}(p\ \mathbf{insert}(a, q)) = 1 \tag{18}$$

Если $p < a$, то

$$\mathbf{insert}(a, d) = \mathbf{insert}(a, pq) = p\ \mathbf{insert}(a, q)$$

поэтому (18) можно переписать в виде

$$\mathbf{ord}(\mathbf{insert}(a, d)) = 1 \tag{19}$$

(19) следует из индуктивного предположения для леммы (т.е. из импликации (12), где $z \stackrel{\text{def}}{=} d$) из равенства

$$\mathbf{ord}(d) = 1$$

которое обосновывается цепочкой равенств

$$1 = \mathbf{ord}(cd) = \mathbf{ord}(c(pq)) = \quad (\text{т.к. } c \leq p)$$
$$= \mathbf{ord}(pq) = \mathbf{ord}(d). \quad \blacksquare$$

Из вышеприведенного примера мы видим, что даже для простейшей ФП, которая состоит из нескольких строк, доказательство ее корректности является нетривиальным математическим рассуждением, которое трудно проверить и гораздо более трудно построить.

Ниже мы представляем принципиально другой метод верификации ФП, основанный на построении диаграммы состояний ФП, и иллюстрируем его доказательством соотношения (8) на основе этого метода. Это доказательство может быть порождено автоматически, что является свидетельством преимуществ метода верификации ФП на основе диаграмм состояний.

4. Состояния функциональных программ

4.1. Понятие состояния функциональной программы

Состоянием ФП Σ называется запись s вида $\{\beta\}^{y}_{x_1,\ldots,x_n}$ где

- $\beta \in \mathcal{E}_\Sigma$ – терм типа **B**, называемый **условием** состояния s,
- y – простой терм, называемый **выходным термом** состояния s, и
- $x_1, \ldots, x_n$ – список простых термов, называемых **входными термами** состояния s.

Множество всех состояний ФП Σ обозначается записью $\mathcal{S}_\Sigma$. $\forall\, s \in \mathcal{S}_\Sigma$ запись X_s обозначает множество всех переменных по данным, входящих в состояние s. Если состояние s имеет вид $\{\beta\}^{y}_{x_1,\ldots,x_n}$, то будем обозначать записями β_s, y_s, и $\bar{x}_s$ термы β, y и список $x_1, \ldots, x_n$, соответственно. Если β_s имеет вид $e_1 \wedge \ldots \wedge e_n$, то s может обозначаться записью $\left\{ \begin{array}{c} e_1 \\ \ldots \\ e_n \end{array} \right\}^{y_s}_{\bar{x}_s}$. Если $\beta_s = \top$, то s может обозначаться записью $\{\}^{y_s}_{\bar{x}_s}$).

Состояние $s \in \mathcal{S}_\Sigma$ называется **начальным** состоянием ФП Σ (и обозначается записью s^0_Σ), если оно имеет вид $\{y = \varphi(\bar{x})\}^{y}_{\bar{x}}$ где

- φ – функциональная переменная, входящая в главный терм ФП Σ,
- $\bar{x}$ – список различных переменных,
- y – переменная, не входящая в $\bar{x}$, и
- $\tau(\varphi) = \tau(\bar{x}) \to \tau(y)$.

Состояние $s \in \mathcal{S}_\Sigma$ называется **терминальным**, если β_s не содержит функциональных переменных.

Если $s \in \mathcal{S}_\Sigma$ и θ – уточнение, то $s[\theta]$ обозначает состояние $\{\beta_s[\theta]\}^{y_s[\theta]}_{\bar{x}_s[\theta]}$. Если θ – переименование, то будем говорить, что $s[\theta]$ получается из s переименованием.

4.2. Равенство термов и состояний

Пусть задано множество переменных $X \subseteq \mathcal{X}$. **Означиванием** переменных из X называется функция $\xi : X \to \mathcal{D}$, сопоставляющая каждой переменной $x \in X$ значение $\xi(x)$ типа $\tau(x)$. Множество всех означиваний переменных из X обозначается записью $X^\bullet$. Для

- каждого означивания $\xi \in X^\bullet$, и
- каждого терма e, такого, что $X_e \subseteq X$ и $\Phi_e = \emptyset$,

запись e^ξ обозначает объект, который либо является значением из $\mathcal{D}$, либо не определен, и вычисляется рекурсивно:

- если $e = x \in X$, то $e^\xi = \xi(x)$,
- если $e = c \in \mathcal{C}$, то e^ξ – значение константы c,
- если $e = f(e_1, \ldots, e_n)$, где $f \in \mathcal{F}$, то
  - e^ξ равно значению $f(e_1^\xi, \ldots, e_n^\xi)$, если это значение определено,
  - e^ξ не определено, в противном случае.

Пусть задана ФП Σ. Для каждого $e \in \mathcal{E}_\Sigma$, и каждого $\xi \in X^\bullet$, где $X \supseteq X_e$, запись $e^{\xi,\Sigma}$ обозначает **значение** e **на** ξ **относительно** Σ, которое определяется так же как выше, со следующим отличием: каждая функциональная переменная из Φ_e рассматривается как ФС, с которым связана частичная функция, являющаяся соответствующей компонентой ННТ ФП Σ.

Термы e_1 и $e_2 \in \mathcal{E}_\Sigma$ считаются **равными** (относительно ФП Σ), если для каждого означивания $\xi \in (X_{e_1} \cup X_{e_2})^\bullet$ объекты $e_1^{\xi,\Sigma}$ и $e_2^{\xi,\Sigma}$ либо оба не определены, либо оба определены и совпадают.

Примеры пар равных термов:

$$e_1 e_1' = e_2 e_2' \text{ и } (e_1 = e_2) \wedge (e_1' = e_2'), \quad ee' = \varepsilon \text{ и } \bot,$$
$$[\![\top]\!] e : e' \text{ и } e, \quad [\![\bot]\!] e : e' \text{ и } e', \quad e = \top \text{ и } e, \quad e = \bot \text{ и } \neg e,$$
$$e \wedge (e' = e'') \text{ и } e[e''/e'] \wedge (e' = e'').$$

Пусть Σ – ФП, и s, s' – пара состояний из $\mathcal{S}_\Sigma$. Будем рассматривать s и s' как равные, если верно какое-либо из следующих условий:

- s' получается из s переименованием, и $\bar{x}_s = \bar{x}_{s'}$,
- $\beta_{s'} = \beta \wedge (x = e)$, где $x \in \mathcal{X}$, $x \notin X_s$, $e \in \mathcal{E}$, β получается из β_s заменой некоторых вхождений e на x, $y_{s'} = y_s$, $\bar{x}_{s'} = \bar{x}_s$,
- $\beta_s = \beta \wedge (x = e)$, где $x \in \mathcal{X}$, e – простой терм, $x \notin X_e$, $s' = s[e/x]$.

4.3. Переходы функциональных программ

В этом пункте определяется понятие перехода ФП. Каждый такой переход выражает связь между двумя состояниями ФП, и имеет метку, которая представляет собой

- либо одну из функциональных переменных,
- либо терм типа **B**, называемый **условием** этого перехода.

Переходом ФП Σ называется тройка $r = (s, s', l)$, где s, s' – состояния из $\mathcal{S}_\Sigma$, называемые **началом** и **концом** перехода r, соответственно, и l – запись, называемая **меткой** перехода r. Переход (s, s', l) называется переходом из s в s'. Он может обозначаться также записью $s \xrightarrow{l} s'$.

$\forall\, s \in \mathcal{S}_\Sigma$ есть следующие переходы с началом s: пусть $s = \{\beta\}^y_{\bar{x}}$, тогда

1) если β содержит подтерм вида $\varphi(\bar{e})$, и одно из уравнений в Σ имеет вид $\varphi(\bar{x}) = e$, то имеется переход (называемый **раскрытием**)

$$s \xrightarrow{\varphi} \{\beta[e[\bar{e}/\bar{x}]/\varphi(\bar{e})]\}^y_{\bar{x}},$$

2) если β содержит подтерм e типа **B**, то имеется пара переходов

$$s \xrightarrow{e} \{\beta \wedge e\}^y_{\bar{x}}, \quad s \xrightarrow{\neg e} \{\beta \wedge \neg e\}^y_{\bar{x}}, \tag{20}$$

3) если β содержит подтерм e типа **S**, то имеется пара переходов

$$s \xrightarrow{e=\varepsilon} \{\beta \wedge (e = \varepsilon)\}^y_{\bar{x}}, \quad s \xrightarrow{e=xx'} \{\beta \wedge (e = xx')\}^y_{\bar{x}}, \tag{21}$$

где x, x' **новые** переменные (т.е. не входящие в X_s).

Каждый из переходов, входящий в какую-либо из пар вида (20) или (21), называется **комплементарным** к другому переходу из этой пары.

Множество всех переходов ФП Σ обозначается записью $\mathcal{R}_\Sigma$.

5. Окрестности состояний

5.1. Развертки состояний

Пусть задана ФП Σ.

Разверткой состояния $s \in \mathcal{S}_\Sigma$ называется конечное дерево V,

- каждой вершине v которого сопоставлено состояние $s_v \in \mathcal{S}_\Sigma$, причем корню этого дерева сопоставлено состояние s, и
- каждому ребру r которого сопоставлен переход из $\mathcal{R}_\Sigma$ вида $s_v \xrightarrow{l} s_{v'}$, где v и v' – начало и конец ребра r, метка l этого перехода является также меткой ребра r.

Вершины и ребра V будут отождествляться с теми состояниями и переходами соответственно, которые им сопоставлены.

5.2. Понятие окрестности состояния

Пусть задана ФП Σ.

Каждому состоянию $s \in \mathcal{S}_\Sigma$ соответствует множество $\mathcal{U}_s$ всех **окрестностей** состояния s. Каждая окрестность $U \in \mathcal{U}_s$ является деревом,

- вершинам которого сопоставлены состояния из $\mathcal{S}_\Sigma$, и
- ребра имеют метки, представляющие собой списки меток, используемых в развёртках состояний.

Множество $\mathcal{U}_s$ определяется следующим образом.

1) Каждая развертка V состояния s, такая, что $\forall\, v \in V$ из вершины v
   - либо не выходит ни одного ребра (такая v называется **листом**),
   - либо выходит одно ребро, являющееся раскрытием,
   - либо выходит пара комплементарных ребер,

   принадлежит множеству $\mathcal{U}_s$.
2) Пусть $U \in \mathcal{U}_s$, и s' – вершина U, не являющаяся ни корнем, ни листом, тогда, если
   - ребро с концом в s', имеет вид $s_0 \xrightarrow{l} s'$, и

- ребра с началом s', имеют вид $s' \stackrel{l_1}{\to} s_1$, …, $s' \stackrel{l_n}{\to} s_n$,

то $\mathcal{U}_s$ содержит дерево U', получаемое из U

- удалением вершины s' и связанных с ней ребер, и
- добавлением ребер $s_0 \stackrel{ll_1}{\to} s_1, \ldots, s_0 \stackrel{ll_n}{\to} s_n$, где $\forall\, i = 1, \ldots, n$ ll_i – конкатенация списков l и l_i.

3) Пусть $U \in \mathcal{U}_s$, и s' – **противоречивая** вершина U (т.е. $\beta_{s'} = \bot$), не являющаяся корнем, тогда $\mathcal{U}_s$ содержит дерево U', получаемое из U удалением вершин, достижимых из s' (т.е. таких, в которые существует путь из s'), а также ребер, связанных с этими вершинами.

Окрестность U', получаемую из U согласно пунктам 2 и 3 данного определения, будем называть **редукцией** окрестности U.

Нетрудно доказать, что

- вершина s какой-либо окрестности противоречива тогда и только тогда, когда концы всех ребер, выходящих из s, противоречивы, и
- состояние противоречиво тогда и только тогда, когда все листья некоторой его окрестности противоречивы.

Если U – окрестность какого-либо состояния, то $\forall\, v \in U$ существует единственный путь из корня U в вершину v. Все вершины U, лежащие на этом пути и не совпадающие с v, будем называть **предками** v.

При графическом изображении окрестностей мы будем использовать следующее соглашение: если U – окрестность какого-либо состояния, то в графическом изображении окрестности U

- вершины, входящие в U, изображаются овалами, причем корневая вершина изображается двойным овалом, противоречивые вершины могут изображаться черными квадратами (■),
- $\forall\, s \in U$ овал O_s, изображающий s, выглядит следующим образом:
  - конъюнктивные члены, входящие в β_s, изображаются в столбик внутри O_s (а если $\beta_s = \top$, то внутри O_s ничего не рисуется),
  - список $\bar{x}_s$ входных термов и выходной терм y_s состояния s изображаются справа от O_s снизу и сверху соответственно, и

– идентификатор состояния s изображается слева сверху от O_s,

- ребра, входящие в U изображаются стрелками, соединяющими овалы: если U содержит ребро $s \xrightarrow{l} s'$, то
  - данное ребро изображается стрелкой из O_s в $O_{s'}$, и
  - рядом с этой стрелкой м.б. указаны компоненты метки l.

5.3. Примеры окрестностей

В этом пункте мы приводим примеры окрестностей состояний для ФП сортировки (6) и для ФП проверки упорядоченности строк (7).

5.3.1. Примеры окрестностей для программы сортировки

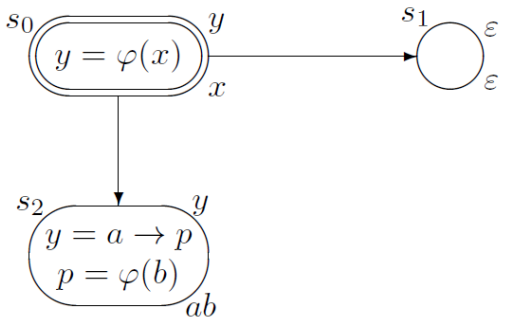
Перепишем ФП сортировки (6), используя более короткие обозначения для входящих в нее функциональных переменных (будем обозначать термы вида $\mathbf{sort}(x)$ и $\mathbf{insert}(a, y)$ записями $\varphi(x)$ и $a \to y$ соответственно):

$$\begin{cases} \varphi(x) = [\![x = \varepsilon]\!]\, \varepsilon : (x_h \to \varphi(x_t)) \\ a \to y = [\![y = \varepsilon]\!] a\varepsilon : \Big([\![a \le y_h]\!]\, ay : y_h(a \to y_t) \Big) \end{cases} \tag{22}$$

Одна из разверток состояния $s_0 \stackrel{\text{def}}{=} \{y = \varphi(x)\}_x^y$ состоит из следующих состояний и ребер:

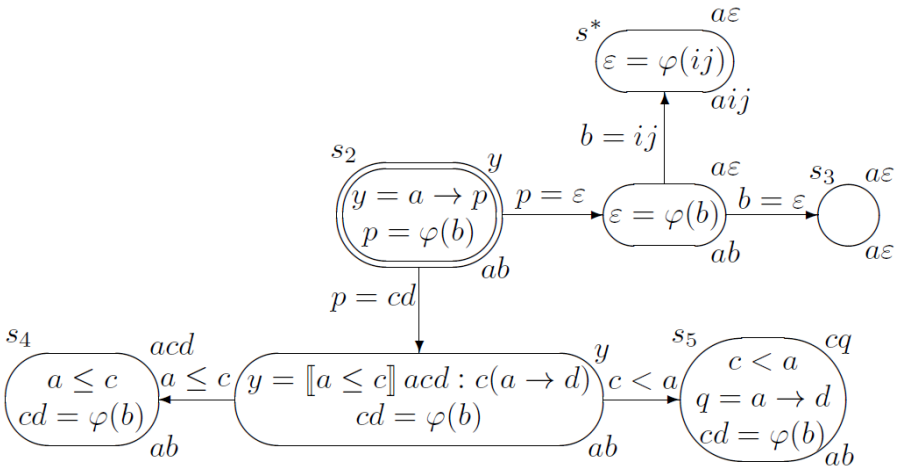
$$\begin{aligned} s_0 &\xrightarrow{\varphi} s \stackrel{\text{def}}{=} \{y = [\![x = \varepsilon]\!]\, \varepsilon : x_h \to \varphi(x_t)\}_x^y, \\ s &\xrightarrow{x=\varepsilon} \left\{ \begin{array}{l} y = [\![x = \varepsilon]\!]\, \varepsilon : x_h \to \varphi(x_t) \\ x = \varepsilon \end{array} \right\}_x^y = \{y = \varepsilon\}_\varepsilon^y = \{\}_\varepsilon^\varepsilon, \\ s &\xrightarrow{x=ab} \left\{ \begin{array}{l} y = [\![x = \varepsilon]\!]\, \varepsilon : x_h \to \varphi(x_t) \\ x = ab \end{array} \right\}_x^y = \\ &= \{y = a \to \varphi(b)\}_{ab}^y = \left\{ \begin{array}{l} y = a \to p \\ p = \varphi(b) \end{array} \right\}_{ab}^y. \end{aligned}$$

Одна из окрестностей, соответствующих данной развертке, имеет вид



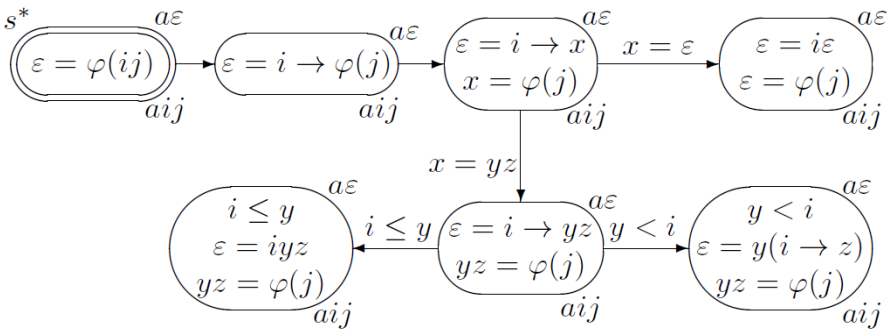
(23)

Одна из окрестностей состояния s_2 в (23) имеет вид



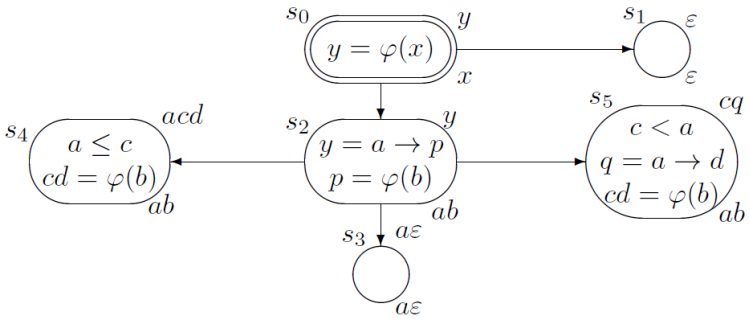
(24)

Одна из окрестностей состояния s^* в (24) имеет вид



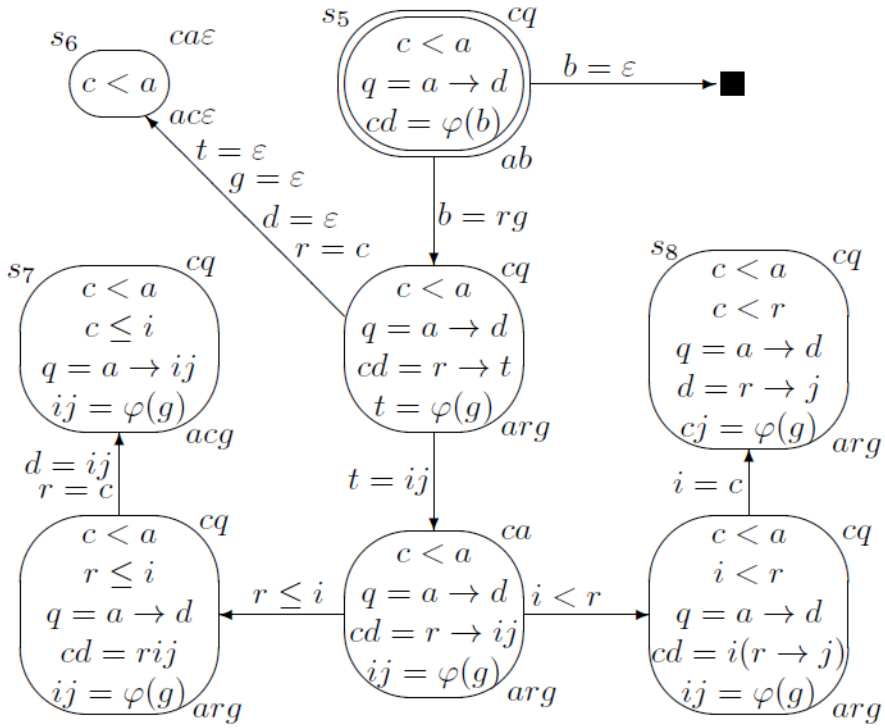
Все листья последней окрестности противоречивы, т.к. среди конъюнктивных членов, входящих в их условия, содержатся равенства вида $\varepsilon = uv$, которые равны терму $\bot$. Следовательно, по сказанному выше, противоречивой является и s^*.

Объединяя окрестности (23) и (24), с учетом вышесказанного заключаем, что одна из окрестностей состояния $s_0 \stackrel{\text{def}}{=} \{y = \varphi(x)\}_x^y$ имеет вид

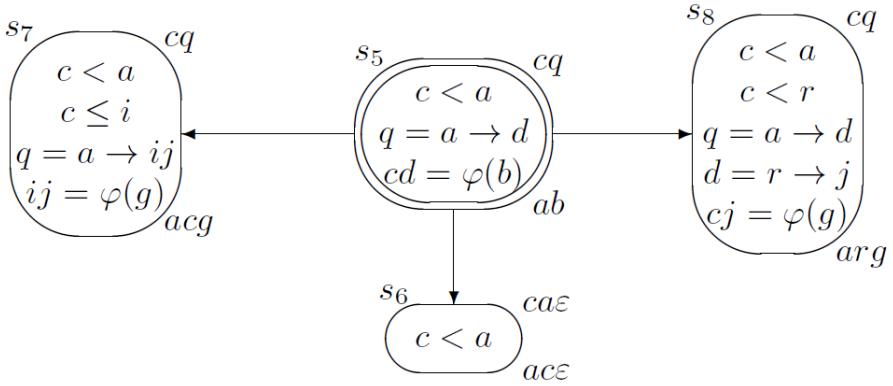


(25)

Другой пример окрестности связан с состоянием s_5. Одна из окрестностей этого состояния имеет вид



Данную окрестность можно редуцировать, и получить окрестность



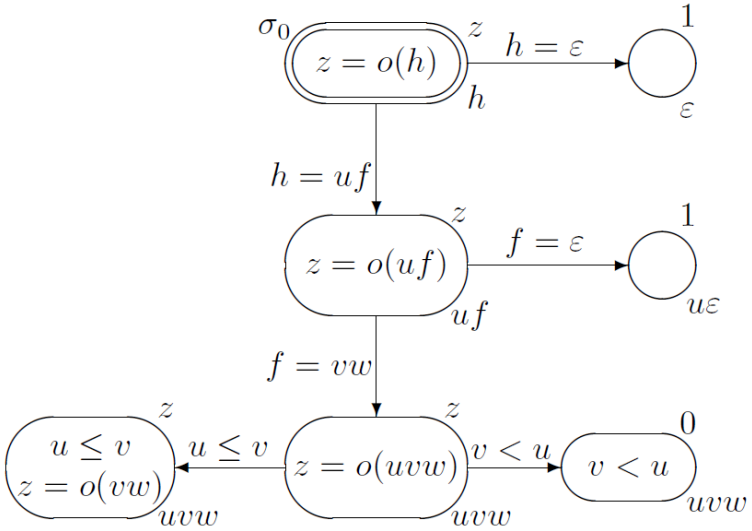
(26)

5.3.2. Примеры окрестностей для программы проверки упорядоченности строк

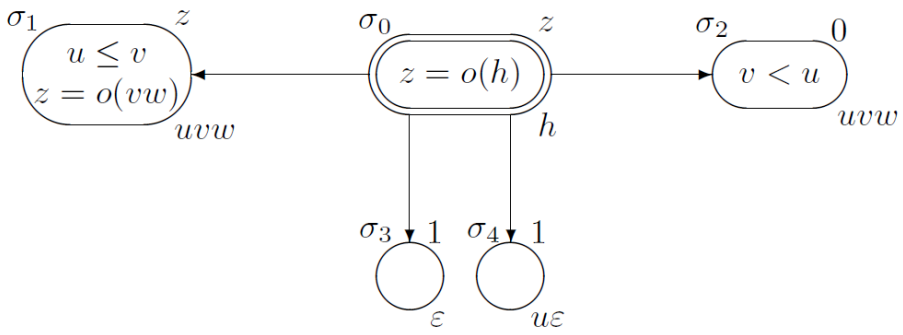
Другие примеры окрестностей состояний связаны с ФП (7) проверки упорядоченности строк. Перепишем эту ФП, используя более короткое обозначение для входящей в нее функциональной переменной:

$$o(x) = [\![x = \varepsilon]\!]1 : \Big([\![x_t = \varepsilon]\!]1 : [\![x_h \le (x_t)_h]\!] o(x_t) : 0\Big) \tag{27}$$

Одна из окрестностей состояния $\sigma_0 \stackrel{\text{def}}{=} \{z = o(h)\}_h^z$ этой ФП имеет вид



Используя определение понятия окрестности состояния ФП, данную окрестность можно преобразовать в окрестность



(28)

6. Вложения состояний

6.1. Явные, условные и обоснованные вложения состояний

Пусть задана пара состояний $s, s' \in \mathcal{S}_\Sigma$.

- **Явное вложение** s в s' – это запись вида

$$\theta : s \hookrightarrow s',$$

где θ – уточнение, и $\beta_s = \beta_{s'}[\theta] \wedge \beta$, причем $\Phi_\beta = \emptyset$.

- **Условное вложение** s в s' – это запись вида

$$\left\{ \begin{array}{l} \eta : r \hookrightarrow r' \\ u[\theta] \hookrightarrow u'[\theta'] \end{array} \right\} : s \hookrightarrow s', \tag{29}$$

где $\eta : r \hookrightarrow r'$ – явное вложение, u, $u' \in \mathcal{S}_\Sigma$, θ и θ' – уточнения, и

$$\beta_s = \beta_{u[\theta]} \wedge \beta_r, \quad \beta_{s'} = \beta_{u'[\theta']} \wedge \beta_{r'}.$$

Посылкой условного вложения (29) называется запись $u \hookrightarrow u'$, где u и u' – соответствующие состояния, входящие в (29).

- **Обоснованное** вложение s в s' – это запись вида

$$s \overset{!}{\hookrightarrow} s' \tag{30}$$

если $\exists U \in \mathcal{U}_s, \exists U' \in \mathcal{U}_{s'}$: для каждого нетерминального листа $r \in U$

– либо существует явное вложение r в некоторое $r' \in U'$,
– либо существует условное вложение r в некоторое $r' \in U'$, причем его посылка имеет вид $s \hookrightarrow s'$, где s и s' – состояния в (30).

- Будем говорить, что s **вложено** в s', если существует либо явное вложение s в s', либо условное вложение s в s' с обоснованной посылкой.

  Запись $s \subseteq s'$ обозначает, что s вложено в s'.

Отметим, что каждое обоснованное вложение можно рассматривать как условное вложение с обоснованной посылкой (компонента "явное вложение" в этом условном вложении является тривиальной).

6.2. Примеры вложений состояний

6.2.1. Примеры явных вложений

1) Для состояний $s_4 = \left\{ \begin{array}{l} a \le c \\ cd = \varphi(b) \end{array} \right\}^{acd}_{ab}$ и $s_0 = \{y = \varphi(x)\}^y_x$, входящих в окрестность (25), имеется явное вложение

$$[cd/y, b/x] : s_4 \hookrightarrow s_0. \tag{31}$$

2) Для состояний $s_7 = \left\{ \begin{array}{l} c < a,\ c \le i \\ q = a \to ij \\ ij = \varphi(g) \end{array} \right\}^{cq}_{acg}$ и $s_2 = \left\{ \begin{array}{l} y = a \to p \\ p = \varphi(b) \end{array} \right\}^{y}_{ab}$, входящих в окрестности (26) и (23) соответственно, имеется явное вложение

$$[q/y, ij/p, g/b] : s_7 \hookrightarrow s_2. \tag{32}$$

3) Для состояний $\sigma_1 = \left\{ \begin{array}{l} r \le v \\ z = o(vw) \end{array} \right\}^{z}_{rvw}$ и $\sigma_0 = \{z = o(h)\}^z_h$, входящих в окрестность (28), имеется явное вложение

$$[vw/h] : \sigma_1 \hookrightarrow \sigma_0. \tag{33}$$

6.2.2. Пример условного вложения

Для состояний $s_8 = \left\{ \begin{array}{l} c < a,\ c < r \\ q = a \to d \\ d = r \to j \\ cj = \varphi(g) \end{array} \right\}^{cq}_{arg}$ и $s_2 = \left\{ \begin{array}{l} y = a \to p \\ p = \varphi(b) \end{array} \right\}^{y}_{ab}$, входящих в окрестности (26) и (23) соответственно, имеется условное вложение

с посылкой $s_5 \hookrightarrow s_0$:

$$\left\{ \begin{array}{l} [q/y, d/p] : \left\{ \begin{array}{l} c < a \\ q = a \to d \end{array} \right\}_{acd}^{cq} \hookrightarrow \{y = a \to p\}_{ap}^{y} \\ s_5[\theta] = \left\{ \begin{array}{l} c < r \\ d = r \to j \\ cj = \varphi(g) \end{array} \right\}_{rg}^{cd} \hookrightarrow s_0[\theta'] = \{p = \varphi(b)\}_{b}^{p} \end{array} \right\} : s_8 \hookrightarrow s_2, \quad (34)$$

где $\theta = [r/a, d/q, j/d, g/b]$, $\theta' = [p/y, b/x]$.

6.2.3. Пример обоснованного вложения

Один из примеров обоснованного вложения: $s_5 \overset{!}{\hookrightarrow} s_0$, где s_5 и s_0 – состояния из (25). В данном случае $U =$ (26) и $U' =$ (23). В окрестности (26)

- состояние s_6 является терминальным,
- существует явное вложение (32) состояния s_7 в состояние s_2, и
- существует условное вложение (34) состояния s_8 в состояние s_2 с посылкой $s_5 \hookrightarrow s_0$.

7. Диаграммы состояний

7.1. Понятие диаграммы состояний

Пусть задана ФП Σ.

Диаграмма состояний (ДС) ФП Σ – это тройка

$$D = (U, N, I), \quad (35)$$

компоненты которой имеют следующий смысл:

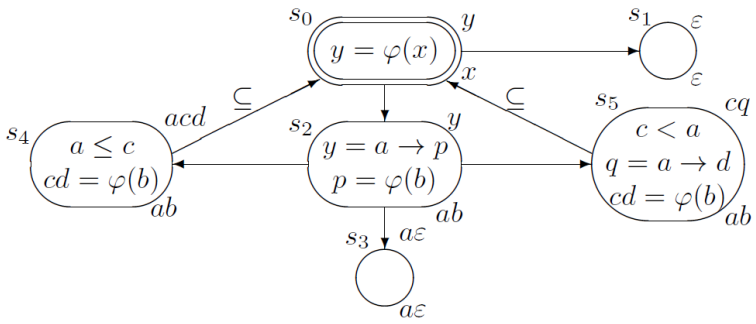
- U – некоторая окрестность начального состояния s_Σ^0,
- N – совокупность всех нетерминальных листьев окрестности U, и
- I состоит из пар вида (s, s'), где $s \in N$, s' – предок s, и $s \subseteq s'$, причем $\forall\, s \in N \;\; \exists\, s' : (s, s') \in I$.

При графическом изображении ДС (35) мы будем обозначать пары из I помеченными стрелками на окрестности U: пара $(s, s') \in I$ будет изображаться стрелкой с началом s, концом s' и меткой $\subseteq$.

7.2. Примеры диаграмм состояний

7.2.1. Диаграмма состояний для программы сортировки

ДС для ФП (22) строится на основе окрестности (25) и имеет вид

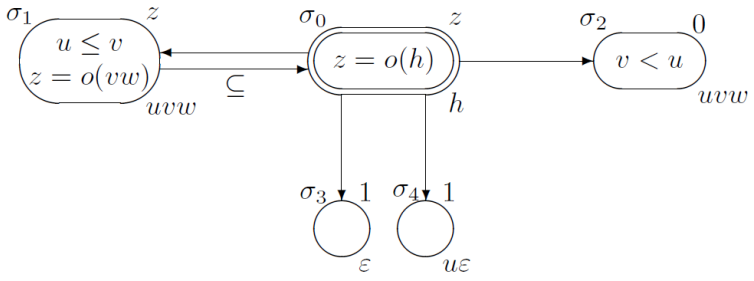


(36)

В данной ДС ребра с меткой $\subseteq$ соответствуют явному вложению (31) и обоснованному вложению $s_5 \overset{!}{\hookrightarrow} s_0$, рассмотренному в пункте 6.2.3.

7.2.2. Диаграмма состояний для программы проверки упорядоченности строк

ДС для ФП (27) строится на основе окрестности (28) и имеет вид



(37)

В данной ДС ребро с меткой $\subseteq$ соответствует явному вложению (33).

8. Верификация функциональных программ на основе понятия диаграммы состояний

8.1. Суперпозиция функциональных программ

8.1.1. Понятие суперпозиции функциональных программ

Пусть заданы ФП Σ и Σ', причем главные термы в Σ и Σ' имеют вид $\varphi(\bar{x})$ и $\varphi'(u)$ соответственно, где $\tau(\varphi(\bar{x})) = \tau(u)$, и $X\Phi_\Sigma \cap X\Phi_{\Sigma'} = \emptyset$.

В этом случае можно определить ФП $\Sigma'(\Sigma)$, которая называется **суперпозицией** ФП Σ и Σ', и представляет собой совокупность равенств,

- первое из которых имеет вид $\psi(\bar{x}) = \varphi'(\varphi(\bar{x}))$, где ψ – новая функциональная переменная соответствующего типа, и
- другие равенства – это все равенства, входящие в Σ и Σ'.

Нетрудно видеть, что $\forall\, \bar{d} \in \mathcal{D}_{\bar{x}} \quad f_{\Sigma'(\Sigma)}(\bar{d}) = f_{\Sigma'}(f_\Sigma(\bar{d}))$.

8.1.2. Окрестности начального состояния суперпозиции функциональных программ

Пусть заданы

- ФП Σ и Σ', удовлетворяющие условиям в начале пункта 8.1.1, и
- окрестности $U \in \mathcal{U}_{s^0_\Sigma}$, $U' \in \mathcal{U}_{s^0_{\Sigma'}}$, причем выполнено условие

$$\forall\, s \in U,\ \forall\, s' \in U' \quad X_s \cap X_{s'} = \emptyset.$$

$\forall\, s \in U$, $\forall\, s' \in U'$ будем обозначать записью ss' состояние ФП $\Sigma'(\Sigma)$, определяемое следующим образом: пусть $s = \{\beta\}^y_{\bar{x}}$, $s' = \{\beta'\}^z_{y'}$, тогда

$$ss' \stackrel{\text{def}}{=} \{\beta \wedge \beta' \wedge (y = y')\}^z_{\bar{x}}.$$

Нетрудно видеть, что если $s^0_\Sigma = \{y = \varphi(\bar{x})\}^y_{\bar{x}}$, $s^0_{\Sigma'} = \{z = \varphi'(y')\}^z_{y'}$, то

$$s^0_{\Sigma'(\Sigma)} = \{z = \psi(\bar{x})\}^z_{\bar{x}} = \{z = \varphi'(\varphi(\bar{x}))\}^z_{\bar{x}} = \left\{ \begin{array}{l} z = \varphi'(y) \\ y = \varphi(\bar{x}) \end{array} \right\}^z_{\bar{x}} = s^0_\Sigma s^0_{\Sigma'}.$$

Обозначим записью UU' дерево, вершины которого имеют метки вида ss', где $s \in U$, $s' \in U'$, указанием недетерминированного алгоритма его построения. Данный алгоритм состоит из нескольких этапов. Дерево, построенное на каждом из этих этапов, обозначается той же записью UU'.

- На первом этапе UU' определяется как дерево из одной вершины, которая имеет метку $s^0_\Sigma s^0_{\Sigma'}$.

- Каждый последующий этап заключается в том, что если построенное к текущему моменту дерево UU' содержит лист v с меткой ss', где либо s не является листом в U, либо s' не является листом в U' то выполняется одна из двух следующих операций:

  – если s не является листом в U, и список его последователей имеет вид $s_1, \ldots, s_n$, то к построенному дереву UU' добавляются последователи вершины v с метками s_1s', ..., s_ns',

  – если s' не является листом в U', то вместо предыдущей может выполняться аналогичная операция для последователей s'.

Теорема 1

Определенное выше дерево UU' является окрестностью начального состояния ФП $\Sigma'(\Sigma)$. ■

8.1.3. Диаграмма состояний суперпозиции функциональных программ

Теорема 2

Пусть ФП Σ и Σ' имеют ДС, и определена суперпозиция $\Sigma'(\Sigma)$.
Тогда ФП $\Sigma'(\Sigma)$ тоже имеет ДС. ■

8.2. Задача верификации функциональных программ

Задача верификации ФП Σ заключается в построении доказательства утверждения о том, что ФП Σ удовлетворяет свойству, выражаемому некоторой формальной спецификацией *Spec*.

Ниже будет использоваться следующее обозначение: запись вида $f = 1$, где f – функция, обозначает утверждение:

функция f принимает значение 1 на всех своих аргументах.

В ряде случаев

- формальная спецификация *Spec* выражается другой ФП Σ', и

- корректность Σ относительно *Spec* выражается утверждением

$$f_{\Sigma'(\Sigma)} = 1. \tag{38}$$

Например, одно из свойств корректности ФП сортировки в пункте 3.1 выражается утверждением вида (38) (а именно, утверждением (8)).

Теорема 3

Пусть ФП Σ имеет ДС, в которой для каждого терминального состояния s терм y_s является константой 1. Тогда $f_\Sigma = 1$. ■

Приведенные выше теоремы являются теоретической основой метода верификации ФП на основе построения ДС

- для анализируемой ФП Σ, и
- для ФП Σ', представляющей проверяемое свойство.

Если эти ФП имеют ДС, то, согласно теореме 2, $\Sigma'(\Sigma)$ тоже имеет ДС. Если эта ДС обладает свойством, описанным в теореме 3, то будет верно (38).

В следующем пункте излагается пример применения данного метода.

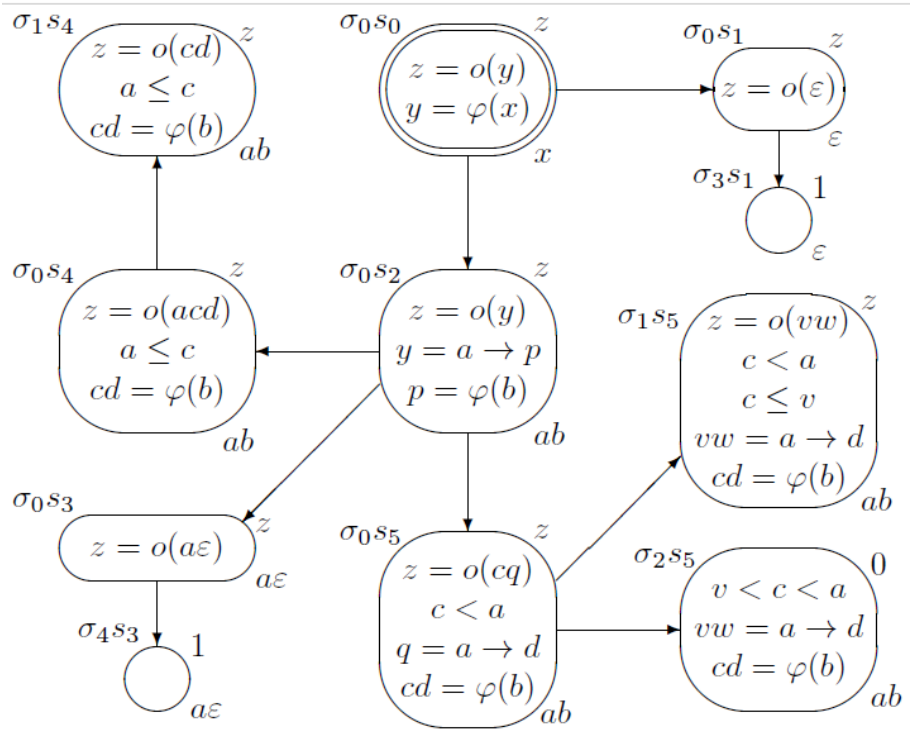
8.3. Пример верификации функциональной программы сортировки при помощи диаграммы состояний

В этом пункте мы иллюстрируем описанный выше метод верификации примером доказательства утверждения (8) для ФП, определенной в пункте 3.1.

Для доказательства равенства (38), где $\Sigma =$ (22) и $\Sigma' =$ (27), построим окрестность начального состояния $s^0_{\Sigma'(\Sigma)}$ ФП $\Sigma'(\Sigma)$ как окрестность вида UU', в соответствии с алгоритмом в пункте (8.1.2) используя

- в качестве U – окрестность (25) состояния s^0_Σ и
- в качестве U' – окрестность (28) состояния $s^0_{\Sigma'}$.

Некоторые состояния получившейся окрестности будут противоречивыми. После их удаления получим следующую окрестность:



(39)

Окрестность (39) содержит 5 листьев. Нетрудно видеть, что

- выходной терм двух из этих листьев ($\sigma_3 s_1$ и $\sigma_4 s_3$) равен 1,
- существует явное вложение листа $\sigma_1 s_4$ в состояние $\sigma_0 s_0$:

$$[b/x, cd/y] : \sigma_1 s_4 \hookrightarrow \sigma_0 s_0,$$

- существует условное вложение $\sigma_1 s_5$ в $\sigma_0 s_0$ с обоснованной посылкой $s_5 \hookrightarrow s_0$:

$$\left\{ \begin{array}{l} [vw/y] : \left\{ \begin{array}{l} c \le v \\ z = o(vw) \end{array} \right\}^z_{vw} \hookrightarrow \{z = o(y)\}^z_y \\ s_5[vw/q] \hookrightarrow s_0[\,] \end{array} \right\} : \sigma_1 s_5 \hookrightarrow \sigma_0 s_0.$$

Построим окрестность листа $\sigma_2 s_5$. Рассмотрим последователей состояния $\sigma_2 s_5$, соответствующих последователям s_6, s_7, s_8 состояния s_5.

- $\sigma_2 s_6 = \{v < u, c < a, uvw = ca\varepsilon\}^0_{ac\varepsilon}$, это состояние противоречиво.
- $\sigma_2 s_7 = \left\{ \begin{array}{l} v < c < a, c \leq i \\ vw = a \to ij \\ ij = \varphi(b) \end{array} \right\}^0_{acg}$. Из данного состояния возможны два комплементарных перехода в состояния, в условии одного из которых будет конъюнктивный член $v = a$, а в условии другого – $v = i$. Нетрудно видеть, что оба эти состояния противоречивы.
- $\sigma_2 s_8 = \left\{ \begin{array}{l} v < c < a, c < r \\ vw = a \to d \\ d = r \to j \\ cj = \varphi(g) \end{array} \right\}^0_{arg}$. Из данного состояния возможны два комплементарных перехода в состояния s, s', где
  - в β_s будет конъюнктивный член $d = \varepsilon$, поэтому в β_s возможен также коньюнктивный член $v = a$, откуда нетрудно получить конъюнктивный член $a < c < a$ в β_s, т.е. s противоречиво,
  - в $\beta_{s'}$ будет конъюнктивный член $d = pq$, где p, q – новые переменные. Из s' возможны два комплементарных перехода в состояния $\tilde{s}, \tilde{s}'$, где
    - в $\beta_{\tilde{s}}$ будет конъюнктивный член $a \leq p$, откуда следует, что в $\beta_{\tilde{s}}$ возможен также коньюнктивный член $v = a$, откуда нетрудно доказать противоречивость $\tilde{s}$, и
    - в $\beta_{\tilde{s}'}$ будет конъюнктивный член $p < a$, и

$$\tilde{s}' = \left\{ \begin{array}{l} v < c < a, c < r \\ w = a \to q \\ vq = r \to j \\ cj = \varphi(g) \end{array} \right\}^0_{arg} = \left\{ \begin{array}{l} v < c < a, c < r \\ vq = r \to j \\ cj = \varphi(g) \end{array} \right\}^0_{arg}.$$

Таким образом, одна из окрестностей $\sigma_2 s_5$ имеет вид

$$\sigma_2 s_5 \to \tilde{s}'. \tag{40}$$

Нетрудно видеть, что имеется явное вложение

$$[q/w, r/a, j/d, g/b] : \tilde{s}' \hookrightarrow \sigma_2 s_5.$$

Объединяя (39) и (40), получаем окрестность с пятью листьями, причем

- два из этих листьев терминальны и их выходной терм равен 1, и
- остальные три листа нетерминальны, и каждый из них вложен в некоторого своего предка.

Таким образом, объединение окрестностей (39) и (40), вместе с указанными выше вложениями нетерминальных листьев, является ДС ФП $\Sigma'(\Sigma)$.

На основании теоремы 3, заключаем, что верно равенство (38). ■

9. Заключение

В настоящей статье было введено понятие диаграммы состояний функциональной программы, и предложен метод верификации, основанный на понятии диаграммы состояний.

Главным преимуществом предложенного метода верификации является возможность его полной автоматизации: построение ДС может быть выполнено автоматически при помощи достаточно простого алгоритма.

Одна из проблем для дальнейших исследований, связанная с понятием диаграммы состояний, имеет следующий вид: найти достаточное условие (по возможности наиболее сильное) на функциональную программу, такое, что если функциональная программа удовлетворяет этому условию, то она имеет диаграмму состояний.

Список литературы

[1] А.М.Миронов: Теория функциональных программ. Часть 1. Москва, издательство ИПИ РАН, 2013. - 160 с.
`http://is.ifmo.ru/verification/2013/mironov-functional.pdf`

[2] R.W. Floyd: Assigning meanings to programs. In J.T. Schwartz, editor, Proceedings Symposium in Applied Mathematics, Mathematical Aspects of Computer Science, pages 19-32. AMS, 1967.

[3] C. A. R. Hoare: An axiomatic basis for computer programming. Communications of the ACM, 12(10): 576–580, 583, October 1969.

[4] R. Milner: A Calculus of Communicating Systems. Number 92 in Lecture Notes in Computer Science. Springer Verlag, 1980.

[5] R. Milner: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, 1999.

[6] Hoare, C. A. R.: Communicating sequential processes. Communications of the ACM 21 (8): 666–677, 1978.

[7] Separation Logic: A Logic for Shared Mutable Data Structures. John C. Reynolds. LICS 2002.

[8] Clarke, E.M., Grumberg, O., and Peled, D.: Model Checking. MIT Press, 1999.

[9] J.A. Bergstra, A. Ponse, and S.A. Smolka, editors: Handbook of Process Algebra. North-Holland, Amsterdam, 2001.

[10] C.A. Petri: Introduction to general net theory. In W. Brauer, editor, Proc. Advanced Course on General Net Theory, Processes and Systems, number 84 in LNCS, Springer Verlag, 1980.

[11] N. D. Jones and N. Andersen. Flow analysis of lazy higher-order functional programs. Theoretical Computer Science, 375:120–136, 2007.

[12] Ranjit Jhala, Rupak Majumdar, Andrey Rybalchenko: HMC: Verifying Functional Programs Using Abstract Interpreters, `http://arxiv.org/abs/1004.2884`

[13] N. Kobayashi and C.-H. L. Ong. A type theory equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In Proceedings of LICS 2009. IEEE Computer Society, 2009.

[14] C.-H. L. Ong. On model- checking trees generated by higher order recursion schemes. In Proceedings 21st Annual IEEE Symposium on Logic in Computer Science, Seattle, pages 81–90. Computer Society Press, 2006.

[15] N. Kobayashi, N. Tabuchi, and H. Unno. Higher-order multiparameter tree transducers and recursion schemes for program verification. In POPL, pages 495–508, 2010.

Verification of functional programs by state diagrams

Andrew M. Mironov

In the paper we introduce graphical objects (called **state diagrams**) related to functional programs. It is shown that the state diagrams can be used to solve problems of verification of functional programs. The proposed approach is illustrated by an example of verification of a sorting program.

Keywords: program verification, functional programs, state diagrams