О проблеме эквивалентности последовательных программ с процедурами

Казбекова Д. О.
Научный руководитель: д.ф.-м.н., проф. В. А. Захаров

2020 г.

План

- Стандартные схемы программ
- Логико-термальная эквивалентность
- Схемы программ с процедурами
- Эквивалентность программ с процедурами

Об истории исследований

Подход к решению проблемы эквивалентности программ, опирающийся на выделении модели, формализующей понятие программы, и проверку эквивалентности методами заданной модели, впервые был предложен Ляпуновым[1] и Яновым[2]. Понятие схемы программы ввел А. А. Ляпунов. Идеи Ляпунова были развиты в конце 50-х и в 60-х гг. его учениками и коллегами — Яновым, Ершовым[3], Криницким[4], Калужниным, Подловченко[5].

---

[1] Ляпунов А. А. О логических схемах программ, 1958

[2] Янов Ю. И. О логических схемах алгоритмов, 1958

[3] Ершов А. П. Об операторных схемах Янова, 1967

[4] Криницкий Н. А. Равносильные преобразования алгоритмов и программирование, 1970

[5] Подловченко Р. И. От схем Янова к теории моделей программ, 1998

Пример схемы программы

Программа вычисления факториала

$in(x)$
$y \leftarrow 1$
l : **if** $(x == 0)$ **then goto m**
$y \leftarrow x * y$
$x \leftarrow x - 1$
goto l
m : **return** y

Схема программы

$in(x)$
$y \leftarrow a$
l : **if** $p(x)$ **then goto m**
$y \leftarrow g(x, y)$
$x \leftarrow h(x)$
goto l
m : **return** y

Еще пример схемы программы

Программа вычисления значения $y = ax^2 + bx + c$

$in(a, b, c, x)$
$y \leftarrow a * x * x + b * x + c$
return y

Другая программа вычисления значения $y = ax^2 + bx + c$

$in(a, b, c, x)$
$y \leftarrow x^2$
$y \leftarrow a * y$
$y \leftarrow y + b * x$
$y \leftarrow y + c$
return y

Схема программы $\mathbf{S}'$

$in(a, b, c, x)$
$y \leftarrow h(a, b, c, x)$
return y

Схема программы $\mathbf{S}''$

$in(a, b, c, x)$
$y \leftarrow pow(x, m)$
$y \leftarrow mul(a, y)$
$y \leftarrow add(y, mul(b, x))$
$y \leftarrow add(y, c)$
return y

Еще пример схемы программы

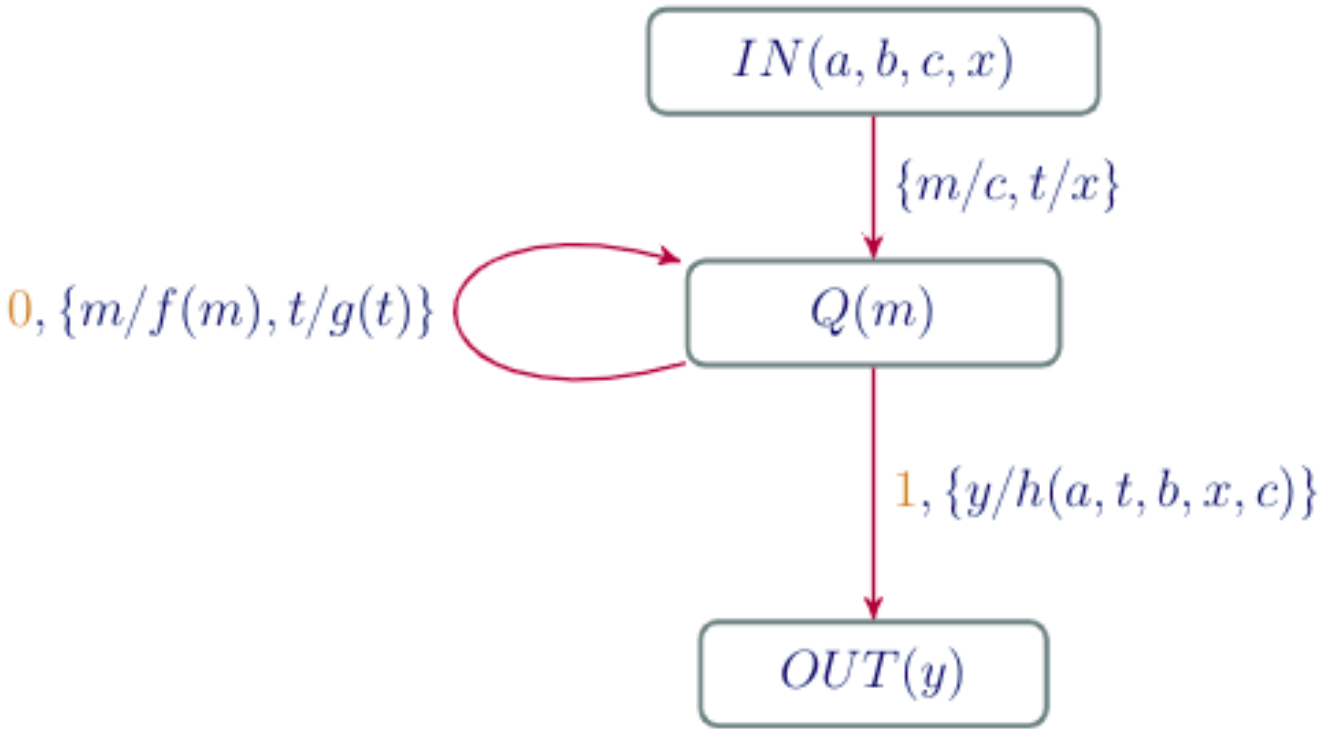
Программа вычисления значения $y = ax^2 + bx + c$

$m \leftarrow 2$
$t \leftarrow x$
p : **if** $m == 1$ **goto l**
$m \leftarrow m - 1$
$t \leftarrow t * t$
goto P
l : $y \leftarrow a * t + b * x + c$
return y

Схема программы **S**

$m \leftarrow c$
$t \leftarrow x$
p : **if** $Q(m)$ **goto l**
$m \leftarrow f(m)$
$t \leftarrow g(t)$
goto P
l : $y \leftarrow h(a, t, b, x, c)$
return y

Схема программы S в виде графа



Подстановки

Пусть заданы множество переменных Var, множество функциональных символов $Func = \{f_i^{(n_i)}, \dots\}$, n_i — местность, множество предикатных символов $Pred = \{P_j^{(k_j)}, \dots\}$, k_j — местность.
Под множеством термов $Term(Var, Func)$ будем понимать наименьшее множество, удовлетворяющее двум условиям:

- $Var \subseteq Term(Var, Func)$;
- если $f^{(n)} \in Func$ и $t_1, \dots, t_n \in Term(Var, Func)$, то $f^{(n)}(t_1, \dots, t_n) \in Term(Var, Func)$,

а элементы этого множества $t_i \in Term(Var, Func)$ будем называть *термами*.

Подстановки

k-местной атомарной формулой, или *атомом*, назовем всякое выражение вида $P^{(k)}(t_1, \ldots, t_k)$, где $P^{(k)}$ — предикатный символ, а $t_1, \ldots, t_k$ – термы. Множество атомарных формул обозначим как $Atom(Var, Func)$.

Пусть X и Y — два конечных множества переменных, $X \subseteq Var, Y \subseteq Var$. $X - Y$-*подстановкой* назовем всякое отображение $\theta : X \to Term(Y, Func)$, сопоставляющее каждой переменной из X некоторый терм из множества $Term(Y, Func)$.

Множество $Dom_\theta = \{x : \theta(x) \neq x\}$ называется *областью подстановки*.

Пустой подстановкой назовем подстановку ϵ с $Dom_\epsilon = \emptyset$

Схема программы, или программа

Definition

Программой над множеством переменных Var назовем размеченную систему переходов $\pi = \langle Var, V, v_{in}, v_{out}, B, \Longrightarrow \rangle$, где

- V — конечное множество вершин,
- v_{in}, v_{out} — соответственно входная и выходная вершины,
- $B : V \to Atom(Var, Func, Pred)$ — функция привязки, которая ставит в соответствие каждой вершине v программы π атом из $Atom(Var, Func, Pred)$, вершины v_{in}, v_{out} помечены специальными атомами $IN^{(k)}(x_1, \ldots, x_k), OUT^{(l)}(t_1, \ldots, t_l)$
- $\Longrightarrow$:
  $V \backslash \{v_{out}, v_{in}\} \times \{0, 1\} \to V \times Subst(Var, Var, Func)$
  $v_{in} \to V \times Subst(Var, Var, Func)$ — функция переходов

Вычисление

Вычисления и эквивалентность программ определяются в интерпретациях первого порядка. Интерпретация сигнатуры σ — это алгебраическая система $I = (D, \overline{Func}, \overline{Pred})$, где

- D — область интерпретации;
- $\overline{Func} \ : \ Func \to (D^n \to D)$ — оценка функциональных символов;
- $\overline{Pred} \ : \ Pred \to (D^m \to \{0, 1\})$ — оценка предикатных символов.

Оценкой множества переменных Var в интерпретации I назовем всякое отображение $\overline{d} : Var \to D$, ставящее в соответствие каждой переменной элемент из области интерпретации.
Под записью $Val(Var, I)$ будем понимать множество всех оценок перменных Var в интерпретации I.

Вычисление

Под *вычислением* программы $\pi = \langle Var, V, v_{in}, v_{out}, B, \Longrightarrow \rangle$ в интерпретации I для оценки переменных $\overline{d}$ будем понимать максимальную последовательность вида

$$comp(\pi, I, \overline{d}) = (v_0, \overline{d_0}), (v_1, \overline{d_1}), \ldots, (v_s, \overline{d_s}), ...,$$

которая удовлетворяет условиям:

- v_0 — входная вершина, $v_0 = v_{in}$,
- для любого $i, i \geq 0$, в программе существует переход $v_i \stackrel{\delta,\theta}{\Longrightarrow} v_{i+1}$, где $\delta = B(v_i)(\overline{d_i})$ и при этом $\overline{d_{i+1}} = \theta(\overline{d_i})$.

Результатом вычисления $comp(\pi, I, \overline{d})$ будет являться набор $\overline{d_N}(x_{i_1}), \ldots, \overline{d_N}(x_{i_k})$ для вершины $v_N = v_{out}$ и атома, приписанного вершине v_{out}, $OUT(x_{i_1}, \ldots, x_{i_k})$.

Эквивалентность

Для каждой интерпретации I может быть задана частичная функция $\Phi_{\pi,I} : Val(Var, I) \to D^k$, которая отображает оценки значений переменных в наборы значений переменных на выходе программы. Указанная частичная функция является значением заданной программы π в интерпретации I.
Программы π', π'' являются *функционально эквивалентными*, если для любой интерпретации I их частичные функции совпадают, то есть $\Phi_{\pi',I} = \Phi_{\pi'',I}$.

Эквивалентность

Эрбрановсиким универсумом будем называть множество всевозможных термов, не содержащих переменных, которые могут быть построены из множества функциональных символов $Func$ и множества $\mathbf{C}$, где $\mathbf{C}$— множество констант $\mathbf{x}$, введенных для каждой переменной $x \in Var$.

Свободной интерпретацией называется всякая интерпретация $I = (D, \overline{Func}, \overline{Pred})$, в которой

- областю интерпретации служит эрбрановский универсум H,
- оценкой $\overline{c}$ каждой константы $c \in \mathbf{C}$ является терм из множества H,
- значением каждой функции $\overline{f}$ на наборе основных термов $t_1, \ldots, t_n$ из H является основной терм $f^{(n)}(t_1, \ldots, t_n)$,
- оценка предикатных символов может быть произвольной.

Эквивалентность

Theorem

Программы π_1 и π_2 функционально эквивалентны тогда и только тогда, когда они эквивалентны на множестве всех свободных интерпретаций

Theorem

Проблема функциональной эквивалентности программ не является частично разрешимой

Luckham, Park, Paterson. On Formalized Computer Programs, 1970

Об истории исследований

В. Э. Иткин ввел новый вид эквивалентности стандартных схем программ — логико-термальную эквивалентность —, и предложил алгоритм распознавания этой эквивалентности[6].
После работ Иткина для проверки логико-термальной эквивалентности программ были разработаны полиномиальные по времени алгоритмы в работах В. К. Сабельфельда[7], В. А. Захарова, Т. А. Новиковой[8].

[6] Иткин В. Э. Логико-термальная эквивалентность схем программ, 1972

[7] Котов В. Е., Сабельфельд В. К. Теория схем программ, 1991

[8] Захаров В. А., Новикова Т. А. Полиномиальный по времени алгоритм проверки логико-термальной эквивалентности программ, 2012

Путь, трасса

Путь ω в программе π — это последовательность переходов вида:

$$\omega = v_0 \overset{\delta_0,\theta_0}{\Longrightarrow} v_1 \overset{\delta_1,\theta_1}{\Longrightarrow} v_2 \dots v_n \overset{\delta_n,\theta_n}{\Longrightarrow} v_{n+1},$$

где $\delta_i \in \{0, 1\}$ — пометка, соответствующая переходу из вершины v_i в вершину v_{i+1}.
Трассой в программе назовем конечный путь, начинающийся в точке входа: $v_0 = v_{in}$. Если трасса tr такова, что

$$tr = v_{in} \overset{\delta_0,\theta_0}{\Longrightarrow} v_1 \overset{\delta_1,\theta_1}{\Longrightarrow} v_2 \dots v_n \overset{\delta_n,\theta_n}{\Longrightarrow} v_{out},$$

будем называть ее *полной трассой*.

Логико-термальная история

Definition

Логической историей $lh(\omega)$ пути ω называется последовательность пар

$$lh(\omega) = (A_{v_1}, \delta_1)(A_{v_2}, \delta_2)\dots(A_{v_k}, \delta_k),$$

где δ_i — пометка перехода $v_j \overset{\delta_j,\theta_j}{\Longrightarrow} v_{j+1}$, а $A_{v_j} = B(v_j)$ — атом, приписанный вершине v_j.

Логико-термальная история

Пусть задана некоторая полная трасса ω в программе π:

$$\omega = v_{in} \overset{\delta_0,\theta_0}{\Longrightarrow} v_1 \overset{\delta_1,\theta_1}{\Longrightarrow} v_2 \overset{\delta_2,\theta_2}{\Longrightarrow} \dots v_{n-1} \overset{\delta_{n-1},\theta_{n-1}}{\Longrightarrow} v_{out}.$$

Definition

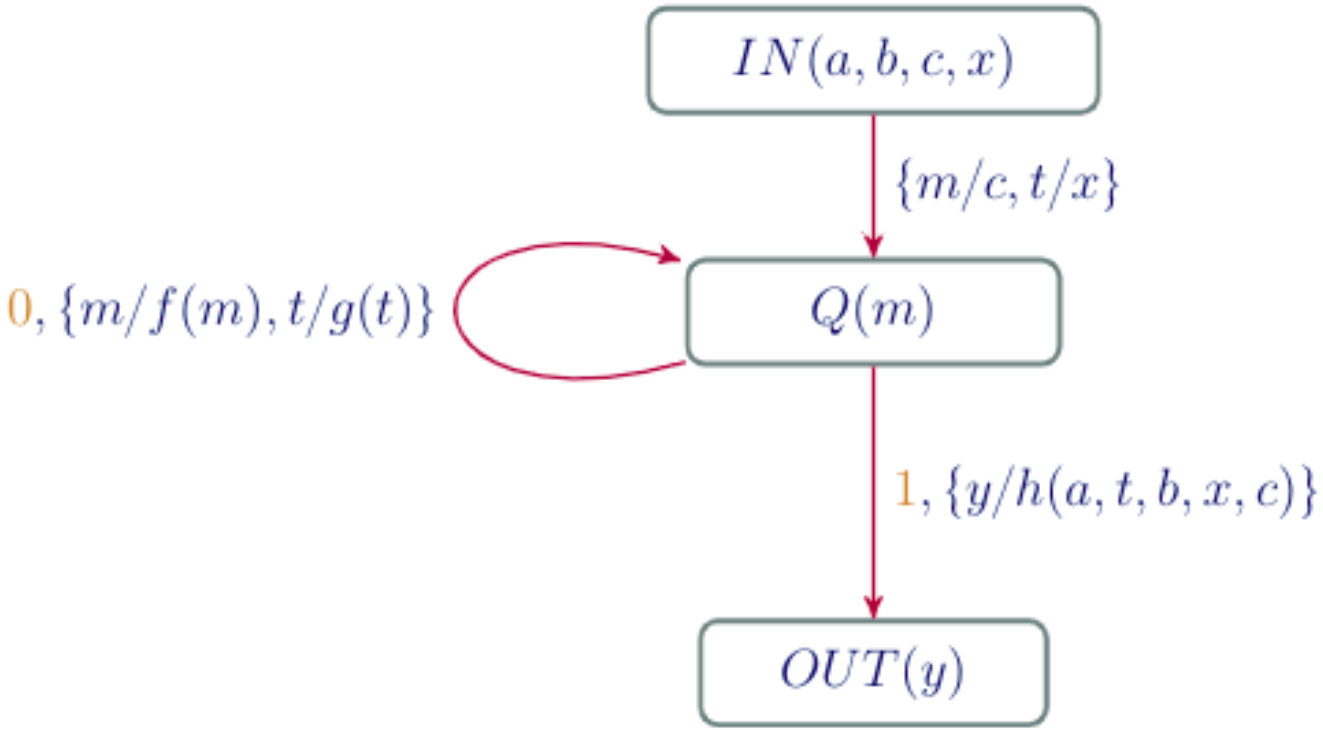
Логико-термальной историей полной трассы ω назовем последовательность пар:

$$lth(\omega) = (A_{v_{in}}\mu_0, \delta_0)(A_{v_1}\mu_1, \delta_1)\dots(A_{v_{n-1}}\mu_{n-1}, \delta_{n-1})(A_{v_{out}}\mu_n, 1),$$

где $\mu_0 = \epsilon,\ \mu_i = \theta_{i-1}\theta_{i-2}\dots\theta_0$, для любого $i, 1 \leq i \leq n$.

Множество всех логико-термальных историй программы π будем называть ее *детерминантом* и обозначать $det(\pi)$.

Возвращаясь к схеме S...



Возвращаясь к схеме S...

Рассмотрим **полную трассу** в схеме **S**

$$\omega = v_{in} \overset{\{m/c,t/x\}}{\Longrightarrow} v_1 \overset{0,\{m/f(m),t/g(t)\}}{\Longrightarrow} v_1 \overset{1,\{y/h(a,t,b,x,c)\}}{\Longrightarrow} v_{out},$$

где $B(v_{in}) = IN(a,b,c,x)$, $B(v_1) = Q(m)$, $B(v_{out}) = OUT(y)$.
Тогда **логической историей** трассы ω будет следующая последовательность

$$lh(\omega) = (IN(a,b,c,x),1)(Q(m),1)(Q(m),0)(OUT(y),1)$$

Будем считать $\delta_1 = 1$ для $v_1 = v_{in}$.
А **логико-термальная история** трассы ω — это последовательность

$$\begin{aligned} lth(\omega) = &(IN(a,b,c,x)\{m/c,t/x\},1) \\ &(Q(m)\{m/f(m),t/g(t)\}\{m/c,t/x\},1) \\ &(Q(m)\{m/f(m),t/g(t)\}\{m/c,t/x\},0) \\ &(OUT(y)\{y/h(a,t,b,x,c)\}\{m/f(m),t/g(t)\}\{m/c,t/x\},1) \end{aligned}$$

Логико-термальная эквивалентность

Definition

Две схемы программ π_1 и π_2 *логико-термально эквивалентны*, если $det(\pi_1) = det(\pi_2)$

Theorem

Для любых π_1, π_2 *справедливо*

$$\pi_1 \overset{lt}{\sim} \pi_2 \implies \pi_1 \sim \pi_2$$

Для схем программ существует эффективный ($O(n^6)$, где n — число вершин схемы программы в графовом представлении) алгоритм проверки логико-термальной эквивалентности[9][10]. Однако логико-термальная эквивалентнось была определена и изучена только для программ без вызовов процедур.

[9] Сабельфельд В. К. Полиномиальная оценка сложности распознавания логико-термальной эквивалентности, 1979

[10] Захаров В. А., Новикова Т. А. Полиномиальный по времени алгоритм проверки логико-термальной эквивалентности программ, 2012

Программы с процедурами

Пусть заданы множество именных символов $S = \{N_1^{(l_1,r_1)}, \ldots, N_m^{(l_m,r_m)}\}$, где l_i и r_i — местность i-го именного символа.
Вызовом процедуры с именем $N^{(l,r)}$, $N^{(l,r)} \in S$, назовем следующий набор $\langle N^{(l,r)}, T_N, X_N \rangle$, где $T_N = (t_1, \ldots, t_l)$ — набор входных параметров процедуры длины l, $X_N = (x_1, \ldots, x_r)$ — набор выходных переменных длины r. Обозначать вызов процедуры N будем как $N(T_N, X_N)$.
Пусть $Call(S, T, X)$ — множество всех вызовов процедур, где

$$T = \bigcup_{N^{(l,r)} \in S} T_N,\ X = \bigcup_{N^{(l,r)} \in S} X_N$$

Программы с процедурами

Будем представлять программу с процедурами как набор процедур с выделенной главной процедурой.

Программы с процедурами

Будем представлять программу с процедурами как набор процедур с выделенной главной процедурой.

Под ***процедурой*** же будем понимать размеченную систему переходов

$$\pi = \langle N, Var_N, V_N, v_{in}^N, v_{out}^N, B, \Longrightarrow, S \rangle, \text{ где}$$

- N — имя процедуры;
- Var_N — множество переменных процедуры;
- $V_N = V_R^N \cup V_C \cup \{v_{in}^N, v_{out}^N\}$ — множество вершин в системе переходов, V_R^N — вершина-распознаватель, V_C — вершина-вызов;
- v_{in}^N, v_{out}^N — входная и выходная вершины;
- B — функция привязки;
- $\Longrightarrow$ — функция переходов;
- S — множество имен процедур.

Программы с процедурами

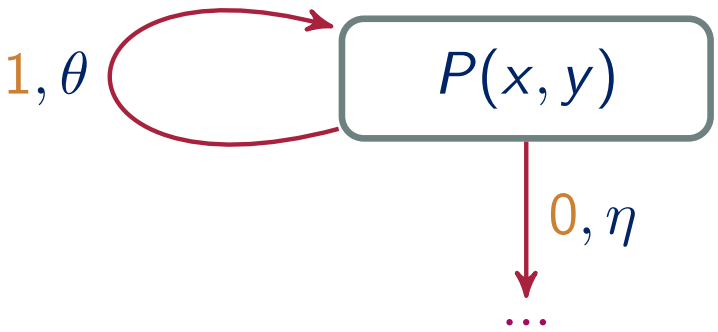
Функция привязки

$$
\begin{aligned}
B : V_R^N &\to Atom(Var_N, Func, Pred),\\
V_C &\to Call(S, T, X),\\
v_{in}^N &\to N(T'),\ T' = (t_1, \ldots, t_l), N(T') \in Env(S, T, X),\\
v_{out}^N &\to N(X'),\ X' = (x_1, \ldots, x_r), N(X') \in Env(S, T, X).
\end{aligned}
$$

Функция переходов

$$
\begin{aligned}
\Longrightarrow : V_R^N \times \{0,1\} &\to V \backslash \{v_{in}\} \times Subst(Var_N, Var_N, Func),\\
V_C \cup \{v_{in}\} &\to V \backslash \{v_{in}\}.
\end{aligned}
$$

Программы с процедурами




```
if x >= y then l else m
```

Рис. 1: Вершина-распознаватель в схеме программы и условный оператор

Рис. 2: Вершины v^N_{in} и v^N_{out}

Программы с процедурами

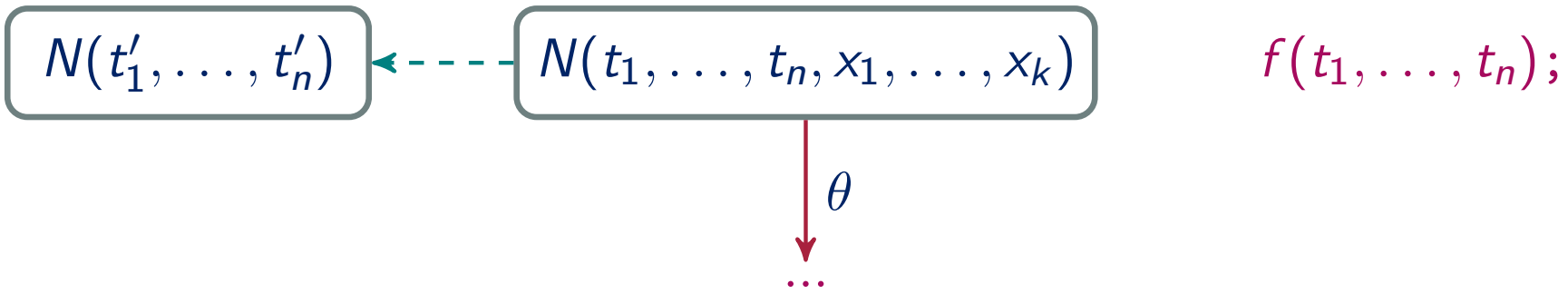


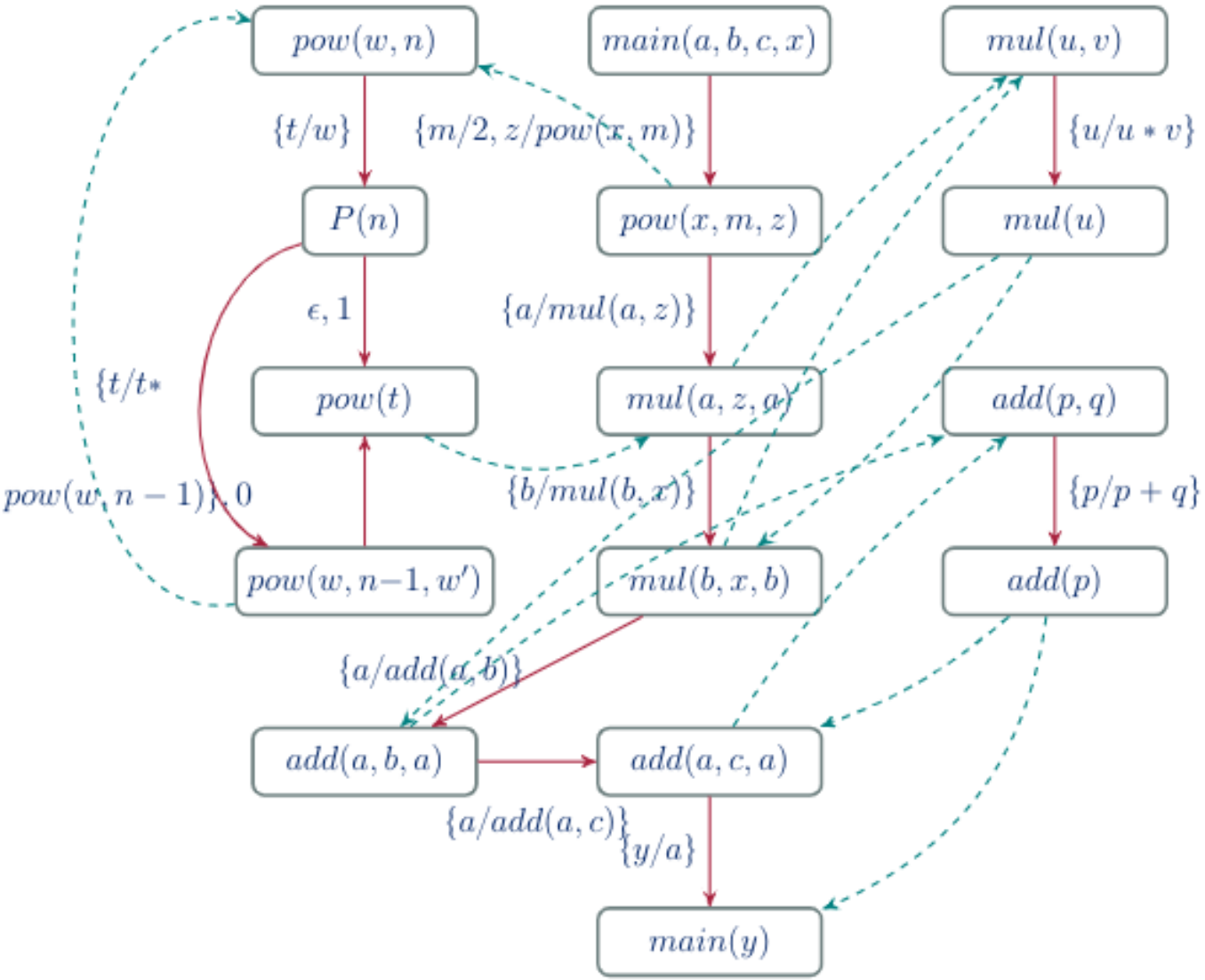
Рис. 3: Вершина-вызов в схеме программы и вызов функции

Пример программы с процедурами

Программой с процедурами назовем систему $\Pi = \langle \pi_{N_0}, \pi_{N_1}, \pi_{N_2}, \ldots, \pi_{N_m} \rangle$, где $N_0, N_1, N_2, \ldots, N_m$ — имена процедур из S, причем, будем полагать, что π_{N_0} главная процедура, $Var_{N_i} \cap Var_{N_j} = \varnothing$ и $V_{N_i} \cap V_{N_j} = \varnothing$, для любых $i \neq j,\ 0 \leq i, j \leq m$.

Пусть задана программа с процедурами $\Pi = \langle \pi_{main}, \pi_{mul}, \pi_{add}, \pi_{pow} \rangle$.
$Var_{main} = \{a, b, c, x, y, z\}$, $Var_{mul} = \{u, v\}$, $Var_{add} = \{p, q\}$,
$Var_{pow} = \{n, w, w', t\}$.
Результат вычисления программы — значение $ax^2 + bx + c$.

Программа Π в виде графа



Путь

Путь ω из вершины v_0 в вершину v_{n+1} в программе с процедурами Π — это последовательность переходов вида:

$$\omega = v_0 \overset{E_1}{\Longrightarrow} v_1 \overset{E_2}{\Longrightarrow} v_2 \ldots v_n \overset{E_n}{\Longrightarrow} v_{n+1}, \tag{1}$$

где $E_i = \begin{cases} N_i \\ (\theta_i, \delta_i) \end{cases}$, либо имя процедуры из S, если $v_i \in V_C$, либо пара (θ_i, δ_i), если $v_i \in V_R^N$ для некоторого имени N, $\delta_i \in \{0, 1\}$, $\theta_i \in Subst(Var_N, Var_N, Func)$.

Логико-термальная история программ с процедурами

Трассой в программе с процедурами назовем конечный путь из вершины $v_{in}^{N_0}$ процедуры π_{N_0}. Если трасса tr такова, что:

$$tr = v_{in}^{N_0} \stackrel{E_1}{\Longrightarrow} v_1 \stackrel{E_2}{\Longrightarrow} v_2 \ldots v_n \stackrel{E_n}{\Longrightarrow} v_{out}^{N_0},$$

будем называть ее *полной трассой*.
Логической историей пути ω назовем последовательность

$$lh(\omega) = L_{i_1} \ \ldots \ L_{i_k},$$

где $L_{i_j} = \begin{cases} N_i \\ (P_i, \delta_j) \end{cases}$, $j \in [1, \ldots, k]$, $N_i \in S$, если $v_i \in V_C$, и $P_i \in Atom(Var_N, Func, Pred)$, если $v_i \in V_R^N$, $\delta_j \in \{0, 1\}$.

Логико-термальная история программ с процедурами

Definition

Логико-термальной историей полной трассы tr назовем последовательность

$$lth(tr) = N_0(T')U_{i_1} \dots U_{i_n} N_0(X'),$$

где

$$T' = (t_{j_1}, \dots, t_{j_k}), X' = (x_{j_1}, \dots, x_{j_l}), N_0 = N_0^{(k,l)}, U_{i_j} = \begin{cases} N_{i_j} \\ N_{i_j}(T'') \\ N_{i_j}(X'') \\ (P_i\mu_j, \delta_j) \end{cases},$$

T'', X'' — соответственно входные параметры и выходные переменные процедуры N_{i_j}, $P_i \in Atom(Var_N, Func, Pred)$ для некоторого $N \in S$, $\mu_j = \theta_{j-1} \dots \theta_0$, $\delta_j \in \{0, 1\}$.

Грамматики

Грамматикой называется четверка $\mathcal{G} = \langle \mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S} \rangle$, где

- $\mathcal{N}$ — конечное множество *нетерминальных символов*,
- $\mathcal{T}$ — конечное множество *терминальных символов*, $\mathcal{N} \cap \mathcal{T} = \varnothing$,
- $\mathcal{P} \subset (\mathcal{N} \cup \mathcal{T})^* \mathcal{N} (\mathcal{N} \cup \mathcal{T})^* \times (\mathcal{N} \cup \mathcal{T})^*$ — *правила вывода*,
- $\mathcal{S} \in \mathcal{N}$ — начальный символ.

Элемент (α, β) множества $\mathcal{P}$ называется *правилом* и записывается в виде $\alpha \mapsto \beta$.

Логико-термальная история программ с процедурами

Пусть задана программа с процедурами $\Pi = \langle \pi_{N_0}, \pi_{N_1}, \ldots, \pi_{N_m} \rangle$.
Построим грамматику $\mathcal{G}^{\Pi} = \langle \mathcal{N}, \mathcal{T}, \mathcal{P}, Start \rangle$, где

- $\mathcal{N} = \{(v, \theta, \eta) : v \in V_{\pi_{N_i}},$
  $\theta, \eta \in Subst(Var_{\pi_{N_i}}, Var_{\pi_{N_i}}, Func)\} \cup \{Start\}$, $V_{\pi_{N_i}}$ — множество вершин процедуры π_{N_i}, $0 \leq i \leq m$,
- $\mathcal{T} = \{\langle N_j \theta \rangle : N_j^{(l,r)} \in S\} \cup \{\langle P_i \theta, \delta \rangle \ :$
  $P_i \in Atom(Var_{\pi_{N_i}}, Func, Pred), \delta \in \{0, 1\},$
  $\theta \in Subst(Var_{\pi_{N_i}}, Var_{\pi_i}, Func)\}$,
- $Start$ — стартовый нетерминал.

Множество правил вывода грамматики $\mathcal{G}^{\Pi}$ выглядит следующим образом:

$$Start \mapsto (v_{in}^{N_0}, \epsilon, \mu), \text{ для вершины } v_{in}^{N_0} \in V_{\pi_{N_0}},$$
$$\epsilon \text{ — пустая подстановка},$$
$$\mu \in Subst(Var_{\pi_{N_0}}, Var_{\pi_{N_0}}, Func)$$

$$(v_{in}^{N_i}, \theta, \mu) \mapsto \langle N_i(T')\theta \rangle (u, \eta\theta, \mu), \text{ для перехода вида } v_{in}^{N_i} \stackrel{\eta}{\Longrightarrow} u,$$
$$v_{in}^{N_i}, u \in V_{\pi_{N_i}}, \mu \in Subst(Var, Var, Func),$$
$$\eta, \theta \in Subst(Var_{\pi_{N_i}}, Var_{\pi_{N_i}}, Func),$$
$$N_i(T') \in In(S, T), N_i^{(l,r)} \in S,$$
$$T' = (t_1, \ldots, t_l)$$

Логико-термальная эквивалентность

Правила вывода грамматики

$(v^{N_i}_{out}, \mu, \mu) \mapsto \langle N_i(X')\mu \rangle$, для вершины $v^{N_i}_{out}$,

$v^{N_i}_{out} \in V_{\pi_{N_i}}, \mu \in Subst(Var, Var, Func)$

$N_i(X') \in Out(S, X), N_i^{(l,r)} \in S,$

$X' = (x_1, \ldots, x_r)$

$(v, \theta, \mu) \mapsto \langle A\theta, \delta \rangle (u, \eta\theta, \mu)$, для перехода вида $v \overset{\eta,\delta}{\Longrightarrow} u$ и

$B(v) = A, A \in Atom(Var_{\pi_{N_i}}, Func, Pred)$

$v, u \in V_{\pi_{N_i}}, \delta \in \{0, 1\},$

$\theta, \eta \in Subst(Var_{\pi_{N_i}}, Var_{\pi_{N_i}}, Func),$

$\mu \in Subst(Var, Var, Func)$

Логико-термальная эквивалентность

Правила вывода грамматики

$(v, \theta, \mu) \mapsto (v_{in}^{N_i}, \theta', \mu)(u, \theta'', \mu)$, для перехода вида $v \stackrel{\eta}{\Longrightarrow} u$ и

$$B(v) = N_i(T', X'), N_i^{(l,r)} \in S,$$

$$v, u \in V_{\pi_{N_j}},$$

$$\theta' \in Subst(Var_{\pi_{N_i}}, Var_{\pi_{N_i}}, Func),$$

$$\theta'' \in Subst(Var_{\pi_{N_j}}, Var_{\pi_{N_j}}, Func),$$

$$\mu \in Subst(Var, Var, Func).$$

Логико-термальная эквивалентность

Первый способ задания логико-термальных историй:

$$main(a,b,c,x)pow(x,m,y)pow(z,n)\{w/z\}(P(n),0)$$
$$pow(z,n-1,w')\ldots(P(n),1)pow(w)mul(a,y,a)mul(u,v)$$
$$\{u/u*v\}mul(u)mul(b,x,b)\ldots add(a,c,a)$$
$$add(p,q)\{p/p+q\}add(p)main(r)$$

Второй способ:

Построить грамматику, которая порождала бы все логико-термальные истории программы.

$$Start \mapsto (v_{in}^{main},\theta,\mu),\theta=\{a/\mathbf{a}',b/\mathbf{b}',c/\mathbf{c}',x/\mathbf{x}'\},$$
$$\mu=\{a/\mathbf{a}',b/\mathbf{b}',c/\mathbf{c}',x/\mathbf{x}',y/y,m/m,z/z\}$$
$$(v_{in}^{main},\theta,\mu)\mapsto\langle main(a,b,c,x)\theta\rangle(pow(x,m,z),\eta\theta,\mu),$$
$$\eta=\{m/\mathbf{c}_1,z/pow(x,m)\},$$
$$\mu=\{a/\mathbf{a}',b/\mathbf{b}',c/\mathbf{c}',x/\mathbf{x}',y/y,m/\mathbf{c}_1,z/pow(x,m)\}$$

Результаты

Две схемы программ **логико-термально эквивалентны**, если у них одинаковые множества логико-термальных историй.

Theorem

Если две программы логико-термально эквивалентны, то для любой интерпретации предикатных и функциональных символов эти программы функционально эквивалентны, то есть вычисляют одну и ту же функцию.

Theorem

Для любой программы с процедурами соответствующая ей грамматика порождает все логико-термальные истории истории программ.

Дальнейшие исследования нацелены на

- разработку алгоритма проверки логико-термальной эквивалентности программ с процедурами,
- обоснование корректности этого алгоритма и оценку его сложности,
- реализацию прототипа программно-инструментальной системы проверки логико-термальной эквивалентности программ с процедурами

Спасибо за внимание!

$$Start \mapsto (v_{in}^{main}, \theta, \mu),$$

$$\theta = \{a/\mathbf{a}', b/\mathbf{b}', c/\mathbf{c}', x/\mathbf{x}'\},$$

$$\mu = \{a/\mathbf{a}', b/\mathbf{b}', c/\mathbf{c}', x/\mathbf{x}', y/y, m/m, z/z\}$$

$$(v_{in}^{main}, \theta, \mu) \mapsto \langle main(a, b, c, x)\theta\rangle(pow(x, m, z), \eta\theta, \mu),$$

$$\eta = \{m/\mathbf{c}_1, z/pow(x, m)\},$$

$$\mu = \{a/\mathbf{a}', b/\mathbf{b}', c/\mathbf{c}', x/\mathbf{x}', y/y, m/\mathbf{c}_1, z/pow(x, m)\}$$

$$(pow(x, m, z), \theta, \mu) \mapsto (v_{in}^{pow}, \theta', \mu')(mul(a, z, a), \{a/mul(a, z)\}\theta, \mu')$$

$$\theta' = \{w/x, n/m\}\theta, \mu' = \{w/x, n/m, t/t, w'/w'\}\mu,$$

$$(v_{in}^{pow}, \theta, \mu) \mapsto \langle pow(w, n)\theta\rangle(P(n), \{t/w\}\theta, \mu),$$

$$\mu = \{t/\mathbf{x}', w/\mathbf{x}', n/\mathbf{c}_1, w'/w', a/\mathbf{a}', b/\mathbf{b}', c/\mathbf{c}', x/\mathbf{x}', y/y, m/\mathbf{c}_1,$$

$$z/pow(x, m)\}$$

$$(P(n), \theta, \mu) \mapsto \langle P(n)\theta, 1\rangle(v_{out}^{pow}, \theta, \mu)$$

$$(P(n), \theta, \mu) \mapsto \langle P(n)\theta, 0\rangle(pow(w, n-1, w'), \eta\theta, \mu),$$

$$\eta = \{t/t * pow(w, n-1)\},$$

$$\mu = \{t/\mathbf{x}' * pow(\mathbf{x}', \mathbf{c}_1 - 1), w/\mathbf{x}', n/\mathbf{c}_1, w'/w', a/\mathbf{a}', b/\mathbf{b}', c/\mathbf{c}',$$

$$x/\mathbf{x}', y/y, m/\mathbf{c}_1, z/pow(x, m)\}$$

$$(v_{out}^{pow}, \theta, \mu) \mapsto \langle pow(t)\mu \rangle$$

$$(pow(w, n-1, w'), \theta, \mu) \mapsto \ldots,$$

аналогично нетерминалу $(pow(x, m, z), \theta, \mu)$

$$(mul(a, z, a), \theta, \mu) \mapsto (v_{in}^{mul}, \theta', \mu')(mul(b, x, b), \{b/mul(b, x)\}\theta, \mu'),$$

$$\theta' = \{u/a, v/z\}\theta, \mu' = \{u/\mathbf{a}', v/pow(\mathbf{x}', \mathbf{c}_1)\}\mu$$

$$(v_{in}^{mul}, \theta, \mu) \mapsto \langle mul(u, v)\theta \rangle (v_{out}^{mul}, \{u/u * v\}\theta, \mu)$$

$$(v_{out}^{mul}, \theta, \mu) \mapsto \langle mul(u)\mu \rangle$$

$$(mul(b, x, b), \theta, \mu) \mapsto (v_{in}^{mul}, \theta', \mu')(add(a, b, a), \{a/add(a, b)\}\theta, \mu')$$

$$(add(a, b, a), \theta, \mu) \mapsto (v_{in}^{add}, \theta', \mu')(add(a, c, a), \{a/add(a, c)\}\theta, \mu')$$

$$(v_{in}^{add}, \theta, \mu) \mapsto \langle add(p, q)\theta \rangle (add(p), \{p/p + q\}\theta, \mu)$$

$$(add(a, c, a), \theta, \mu) \mapsto (v_{in}^{add}, \theta', \mu')(v_{out}^{main}, \{y/a\}\theta, \mu')$$

$$(v_{out}^{main}, \{y/a\}\theta, \mu) \mapsto \langle main(y)\mu \rangle$$