О суперкомпиляции неветвящихся программ

Андрей П. Немытых
Институт программных систем РАН
г. Переславль-Залесский

Второе совместное рабочее совещание ИПС РАН и МГТУ имени Н.Э. Баумана
по функциональному языку программирования Рефал
11 июня 2019 г., Москва

«Так ринемся скорей из области томленья —
По мановению эфирного гонца —
В край, где слагаются заоблачные звенья
И башни высятся заочного дворца! ...»

Осип Мандельштам, 1911 г.

Об одном невыученном уроке

11 июня 2019 г., Москва

Неветвящиеся программы

Определение:

Неветвящейся программой называется программа, которая не содержит явных и неявных операторов условного перехода.

Примеры операторов (синтаксических конструкторов) условного перехода:

- **if, while, for;**
- **образец в Рефале (левая часть предложения);**
- **оператор «,» в Рефале.**

Примеры неветвящихся программ

– **В базисе функций:** $+,\ \times,\ g$**:**

$f(x_1, x_2, x_3)\ \{$
$\quad y := {x_1}^2 + 2 \times x_2;$
$\quad z := y + x_3;$
$\quad u := g(z^3 + 5 + z + y \times x_1);$
$\quad \text{return } u;$
$\}$

– **Выражение (правая часть предложения) в Рефале (в базисе функций:** $f,\ g$**, приписывание):**

$H(e.x, e.y)\ \{$
$\quad e.u := \langle f\ e.x\rangle\ \ e.x\ \ \langle g\ e.y\ \ \langle f\ e.x\rangle\rangle\ \langle g\ e.y\rangle;$
$\quad \text{return } e.u;$
$\}$

Вычисление значения многочлена в данной точке (в базисе функций: $+$, $\times$)

$$\mathbf{f(x) = a_0 + a_1 \times x + \ldots + a_n \times x^n;}$$

Тривиальная неветвящаяся программа

$$\begin{aligned}
&\mathbf{p(a_0, \ldots, a_n, x)\ \{}\\
&\quad \mathbf{y_1 := x;\ y_2 := y_1 \times x;\ y_3 := y_2 \times x;\ \ldots,\ y_n := y_{n-1} \times x;}\\
&\quad \mathbf{z_1 := a_1 \times y_1;\ z_2 := a_2 \times y_2;\ z_3 := a_3 \times y_3;\ \ldots,\ z_n := a_n \times y_n;}\\
&\quad \mathbf{return\ a_0 + z_1 + z_2 + \ldots + z_n;}\\
&\mathbf{\}}
\end{aligned}$$

Всего потребовалось примерно $2n$ умножений.

Схема Горнера

$$\mathbf{a_0 + a_1 \times x + \ldots + a_n \times x^n = a_0 + x \times (a_1 + x \times (\ldots + x \times (a_{n-1} + x \times a_n) \ldots))}$$

Неветвящаяся программа

$\mathbf{p(a_0, \ldots, a_n, x)}$ {

$\mathbf{y_0 := a_{n-1} + x \times a_n;\ y_1 := a_{n-2} + x \times y_0;\ \ldots,\ y_{n-1} := a_0 + x \times y_{n-2};}$

return $\mathbf{y_{n-1}}$;

}

Потребовалось $\mathbf{n}$ умножений.

Параметризованные конфигурации
в дереве развертки суперкомпилируемой программы
суть неветвящиеся программы.

В схеме Горнера

$$a_0 + a_1 \times x + \ldots + a_n \times x^n = a_0 + x \times (a_1 + x \times (\ldots + x \times (a_{n-1} + x \times a_n) \ldots)))$$

потребовалось n умножений.

Схема Горнера является оптимальной среди всех схем вычисления значения многочленов в точке без предварительной обработки коэффициентов.

Теорема: Каждая схема вычисления значения многочленов от одной переменной в точке без предварительной обработки коэффициентов содержит не менее n $\pm$ операций и не менее n $\times$ операций.

Схема Горнера для класса всех многочленов от одной переменной является неулучшаемой.

Специализация неветвящихся программ

Для вычисления многочлена

$$p(x) = x^{15} + x^{14} + \ldots + x + 1 = \frac{x^{16}-1}{x-1}$$

требуется лишь четыре операции умножения и одно деление, т.е. 5 операций $*$, вместо 14 умножений по схеме Горнера.

Ветвящаяся программа:

$p(x)$ {
    if$(x = 1)$ then { return 16; };
    $y_1 := x \times x;\ y_2 := y_1 \times y_1;\ y_3 := y_2 \times y_2;\ y_4 := y_3 \times y_3;$
    $y_5 := x - 1;\ y_6 := y_4 - 1;$
    return $\frac{y_6}{y_5}$;
}

Вычисление значения монома в данной точке (в базисе функций: $+$, $\times$)

$\mathbf{f(x) = x^n;}$

Тривиальная неветвящаяся программа

$\mathbf{p(x)}$ {

$\quad \mathbf{y_1 := x;\ y_2 := y_1 \times x;\ y_3 := y_2 \times x;\ \ldots,\ y_n := y_{n-1} \times x;}$

$\quad \mathbf{return\ y_n;}$

}

Всего потребовалось $n - 1$ умножение.

Специализация неветвящихся программ

$f(x) = x^{2019};$

$$2019 = 11111100011_2;$$

Что означает:

$$2019 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^1 + 2^0;$$

Следовательно:

$$x^{2019} = x^{2^{10}} x^{2^9} x^{2^8} x^{2^7} x^{2^6} x^{2^5} x^{2^1} x;$$

Число умножений K не больше, чем число единиц в записи 11111100011_2, умноженное на два.

Следовательно:

$$K \leq 2\log_2(2019)$$

Специализация неветвящихся программ

$f(x) = x^n$;

При вычислении посредством наивной неветвящейся программы потребовалось $n - 1$ умножение.

Специализация наивной программы дает неветвящуюся программу, использующую не более $2m$ умножений, где $m = \log_2(n)$.

Ускорение по числу умножений от размера (длины) входного данного: $O(\frac{2^m}{m})$.

Вычисление определителя матрицы (в базисе функций: $\pm, *$)

$$\mathbf{f(x_{11}, \ldots, x_{1n}, \ldots, x_{n1}, \ldots, x_{nn})} = \det \begin{pmatrix} x_{11} & \ldots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \ldots & x_{nn} \end{pmatrix} =$$

$$= \sum_{\sigma \in S_n} \mathbf{sign}(\sigma) \mathbf{x_{1\sigma(1)}} \times \ldots \times \mathbf{x_{n\sigma(n)}};$$

Здесь $\mathbf{sign}(\sigma) = (-1)\hat{}$(число элементов следующего множества) :

$\{\mathbf{i, j} \in \{1, \ldots, \mathbf{n}\} \mid \mathbf{i} < \mathbf{j},\ \sigma(\mathbf{i}) > \sigma(\mathbf{j})\}$

Тривиальная неветвящаяся программа

$\mathbf{p(x_{11}, \ldots, x_{1n}, \ldots, x_{n1}, \ldots, x_{nn})}$ {

$\quad \mathbf{z} := 0;$

Следующие две строчки строятся для каждой перестановки $\sigma \in \mathbf{S_n}$

$\quad \mathbf{y_1} := \mathbf{x_{1\sigma(1)}};\ \mathbf{y_2} := \mathbf{y_1} \times \mathbf{x_{2\sigma(2)}};\ \ldots,\ \mathbf{y_n} := \mathbf{sign}(\sigma)\mathbf{y_{n-1}} \times \mathbf{x_{n\sigma(n)}};$

$\quad \mathbf{z} := \mathbf{z} + \mathbf{y_n};$

$\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

$\quad \mathbf{return\ z};$

}

Всего $\mathbf{n!}$ слагаемых, в каждом из которых $\mathbf{n-1}$ умножение.

Алгоритм Гаусса (в базисе функций: $\pm, *$)

$$\det\begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{nn} \end{pmatrix} = (-1)^t \det\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1(n-1)} & a_{1n} \\ 0 & a_{22} & \dots & a_{2(n-1)} & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & a_{nn} \end{pmatrix} =$$

$$= (-1)^t a_{11} \times a_{22} \cdots \times a_{nn}$$

Ветвящаяся программа без циклов

$\mathbf{p(x_{11}, \dots, x_{1n}, \dots, x_{n1}, \dots, x_{nn})}$ {

$\mathbf{if(x_{11} \neq 0)\{\ b_2 := x_{21}/x_{11};\ b_3 := x_{31}/x_{11};\ \dots,\ b_n := x_{n1}/x_{11};}$

Следующая строчка строится для каждого $i \geq 0$.

$\mathbf{x_{(2+i)2} := x_{(2+i)2} - b_2 \times x_{12};\ \dots,\ x_{(2+i)n} := x_{(2+i)n} - b_2 \times x_{1n};}$

$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$

}

Повторяем вышепостроенную конструкцию для подматрицы M_{jj}, $m_{ks} = x_{(k+j-1)(s+j-1)}, j > 1$ в правом нижнем углу, начинающейся с первого $x_{jj} \neq 0$.

$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$

$\mathbf{return\ (-1)^t x_{11} \times x_{22} \cdots \times x_{nn};}$

}

Алгоритм Гаусса (в базисе функций: $\pm, *$)

Ветвящаяся программа без циклов

$\mathrm{p}(\mathrm{x_{11}}, \ldots, \mathrm{x_{1n}}, \ldots, \mathrm{x_{n1}}, \ldots, \mathrm{x_{nn}})$ {

$\mathbf{if}(\mathrm{x_{11}} \neq 0)\{\ \mathrm{b_2} := \mathrm{x_{21}}/\mathrm{x_{11}};\ \mathrm{b_3} := \mathrm{x_{31}}/\mathrm{x_{11}};\ \ldots,\ \mathrm{b_n} := \mathrm{x_{n1}}/\mathrm{x_{11}};$

Следующая строчка строится для каждого $\mathrm{i} \geq 0$.

$\mathrm{x_{(2+i)2}} := \mathrm{x_{(2+i)2}} - \mathrm{b_2} \times \mathrm{x_{12}};\ \ldots,\ \mathrm{x_{(2+i)n}} := \mathrm{x_{(2+i)n}} - \mathrm{b_2} \times \mathrm{x_{1n}};$

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

}

Повторяем вышепостроенную конструкцию для подматрицы $\mathrm{M_{jj}}$, $\mathrm{m_{ks}} = \mathrm{x_{(k+j-1)(s+j-1)}}, \mathrm{j} > 1$ в правом нижнем углу, начинающейся с первого $\mathrm{x_{jj}} \neq 0$.

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

$\mathbf{return}\ (-1)^{\mathrm{t}}\mathrm{x_{11}} \times \mathrm{x_{22}} \cdots \times \mathrm{x_{nn}};$

}

Всего в алгоритме Гаусса $\mathrm{O}(\mathrm{n}^3)$ умножений и $\mathrm{O}(\mathrm{n}^2)$ делений.

Алгоритм Гаусса (в базисе функций: $\pm, \cancel{*}$)

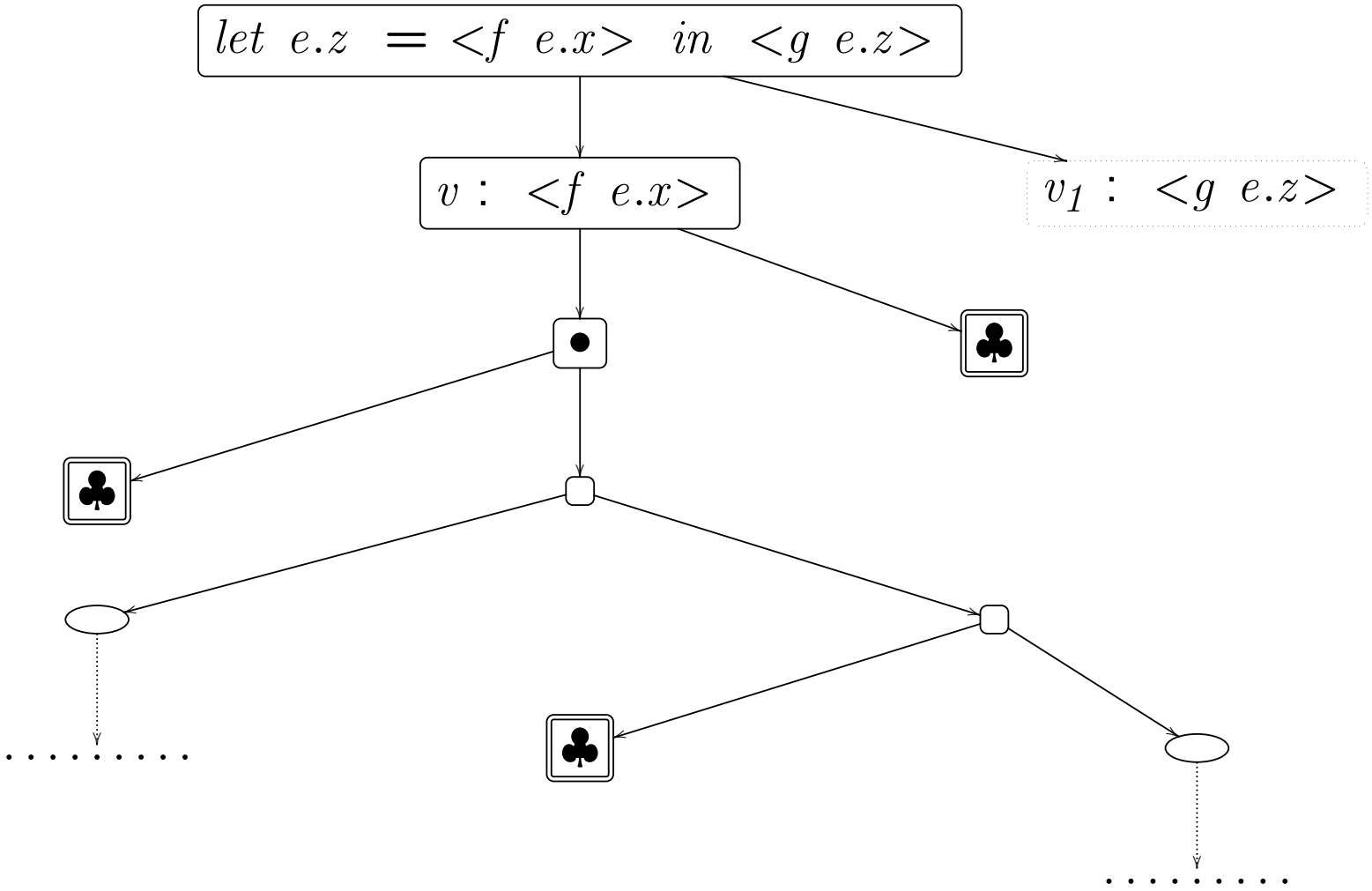
В алгоритме Гаусса потребовалось $O(n^3)$ умножений и $O(n^2)$ делений.

Теорема (Штрассен, 1973): Если есть алгоритм, использующий деление, для вычисления значения в точке многочлена от k переменных $P(x_1, \ldots, x_k)$, тогда есть алгоритм без делений, вычисляющий $P(x_1, \ldots, x_k)$, причем его сложность лишь незначительно больше сложности исходного алгоритма.

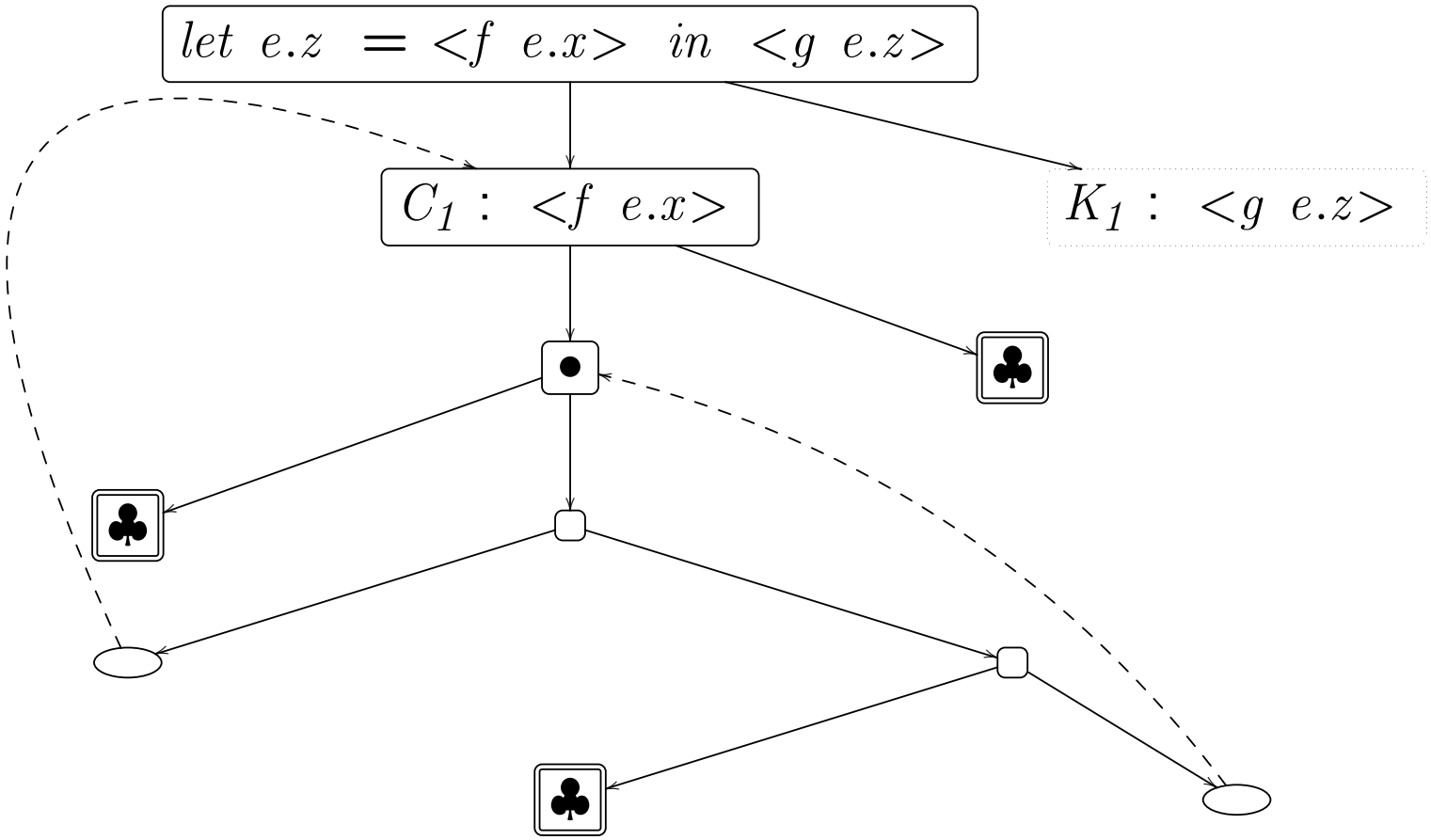
$$O(n^3) + O(n^2) \ll O(n!)$$

Следствие: Существует алгоритм для вычисления определителя матрицы размера $n \times n$ (в базисе функций $\pm, \times$), сложность которого много меньше, чем $n!$ – сложности наивного алгоритма.

Развертка



Свертка



Промежуточное состояние графа развертки-свертки программы: конфигурация K_1 еще не разворачивалась.

Схема Горнера для класса всех многочленов от одной переменной является неулучшаемой.

Специализация неветвящихся программ

Для вычисления многочлена

$$\mathrm{p(x)} = \mathrm{x}^{15} + \mathrm{x}^{14} + \ldots + \mathrm{x} + 1 = \frac{\mathrm{x}^{16}-1}{\mathrm{x}-1}$$

требуется лишь четыре операции умножения и одно деление, т.е. 5 операций $*$, вместо 14 по схеме Горнера.

Следствие: Существует алгоритм для вычисления

$$\mathrm{p(x)} = \mathrm{x}^{15} + \mathrm{x}^{14} + \ldots + \mathrm{x} + 1 = \frac{\mathrm{x}^{16} - 1}{\mathrm{x} - 1}$$

в базисе функций $\pm, \times$, который использует 6 операций умножения, вместо 14 по схеме Горнера.

Схема Горнера для класса всех многочленов от одной переменной является неулучшаемой.

Специализация неветвящихся программ

Следствие: Существует алгоритм для вычисления

$$p(x) = x^{15} + x^{14} + \ldots + x + 1 = \frac{x^{16} - 1}{x - 1}$$

в базисе функций $\pm, \times$, который использует 6 операций умножения, вместо 14 по схеме Горнера.

Доказательство:

$$\frac{x^{2^4}-1}{x-1} = 1 + x + \cdots + x^{2^4-1} = (1 + x + \cdots + x^{2^3-1})(1 + x^{2^3}) =$$
$$= (1 + x + x^2 + x^{2^2-1})(1 + x^{2^2})(1 + x^{2^3}) =$$
$$= (1 + x)(1 + x^{2^1})(1 + x^{2^2})(1 + x^{2^3})$$

требуется 6 операций $\times$, вместо 14 по схеме Горнера.

Специализация неветвящихся программ

Следствие: Существует алгоритм для вычисления

$$p(x) = x^{15} + x^{14} + \ldots + x + 1 = \frac{x^{16} - 1}{x - 1}$$

в базисе функций $\pm, \times$, который использует 6 операций умножения, вместо 14 по схеме Горнера.

Доказательство:

$\frac{x^{2^4}-1}{x-1} = (1 + x)(1 + x^{2^1})(1 + x^{2^2})(1 + x^{2^3})$

требуется 6 операций $\times$, вместо 14 по схеме Горнера:

Неветвящаяся программа:

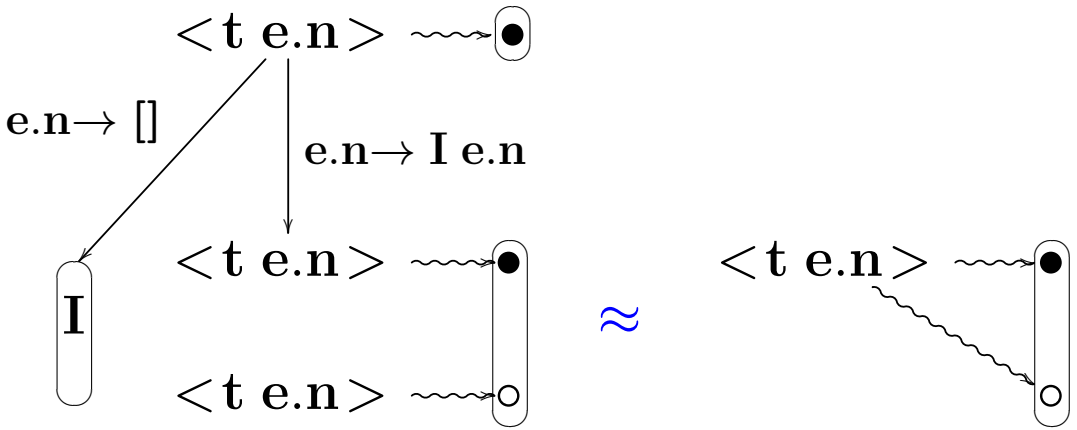
```
p(x) {
    y1 := x × x;  y2 := y1 × y1;  y3 := y2 × y2;
    z := (1 + x) × (1 + y1);  z := z × (1 + y2);  z := z × (1 + y3);
    return z;
}
```

Задача №1: Написать модельный суперкомпилятор с прогонкой, параллельно разворачивающий вызовы функций и копирующий значения повторных вызовов.

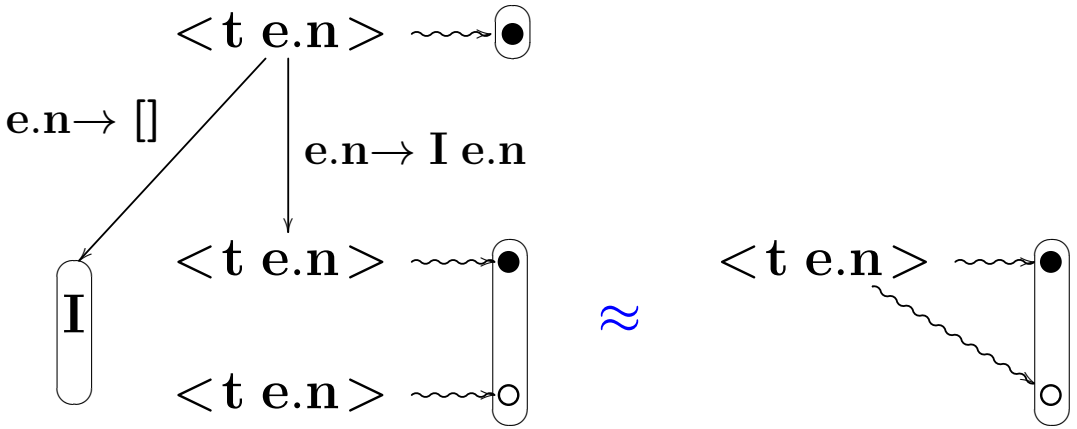
Пример

```
t {
      = I;
I e.n = <t e.n> <t e.n> ;
}
```

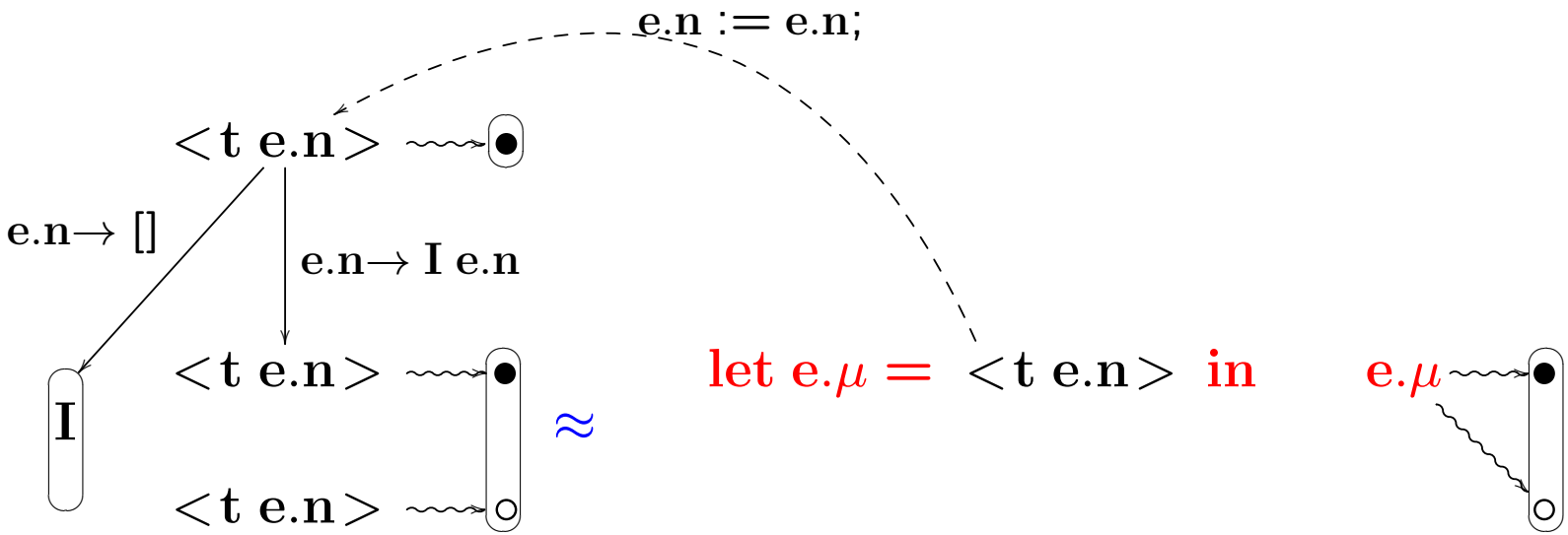
Прогонка:



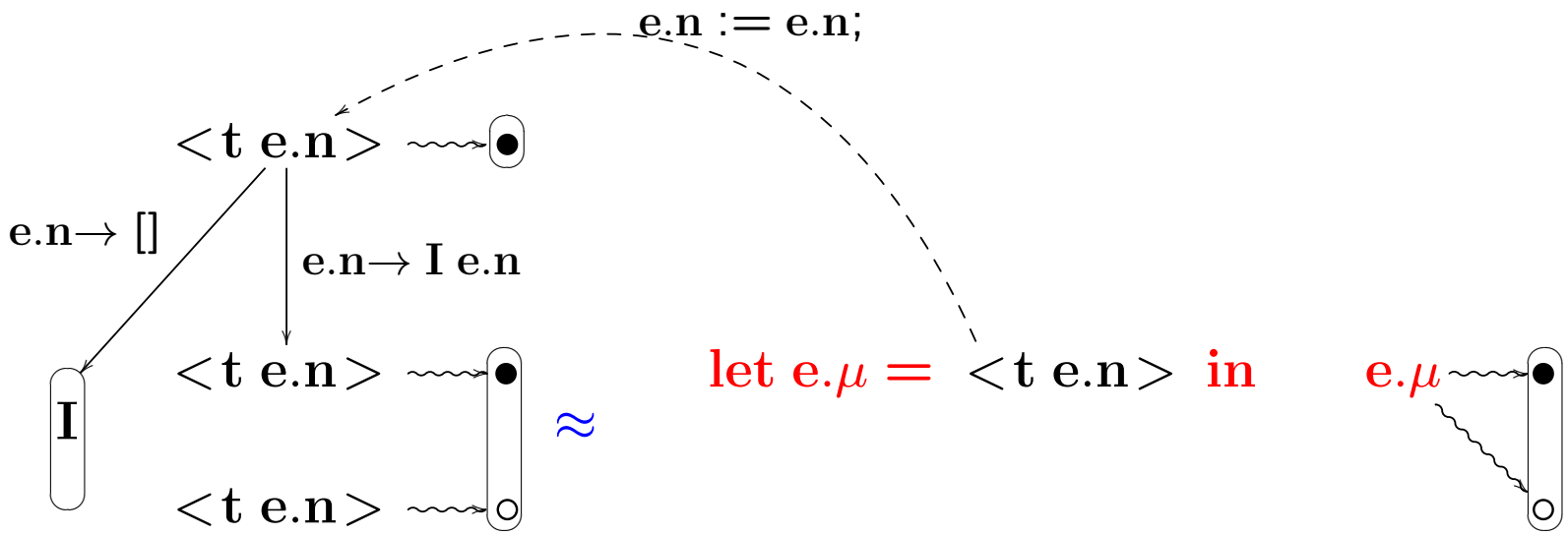
Прогонка:



Свертка:



Свертка:



Остаточная программа:

```
T {
                          = I;
I e.n , <T e.n>: e.μ = e.μ e.μ;
}
```

Задача №1: Написать модельный суперкомпилятор с прогонкой, параллельно разворачивающий вызовы функций и копирующий значения повторных вызовов.

- входной язык должен допускать операторы присваивания в левых частях предложений (безусловные запятые);
- в случае явной композиции функций структура конфигураций (графов) будет сложнее;
- проблемы с вложением и обобщением конфигураций.

Нужно изучать хорошие бинарные отношения на множествах деревьев и графов.

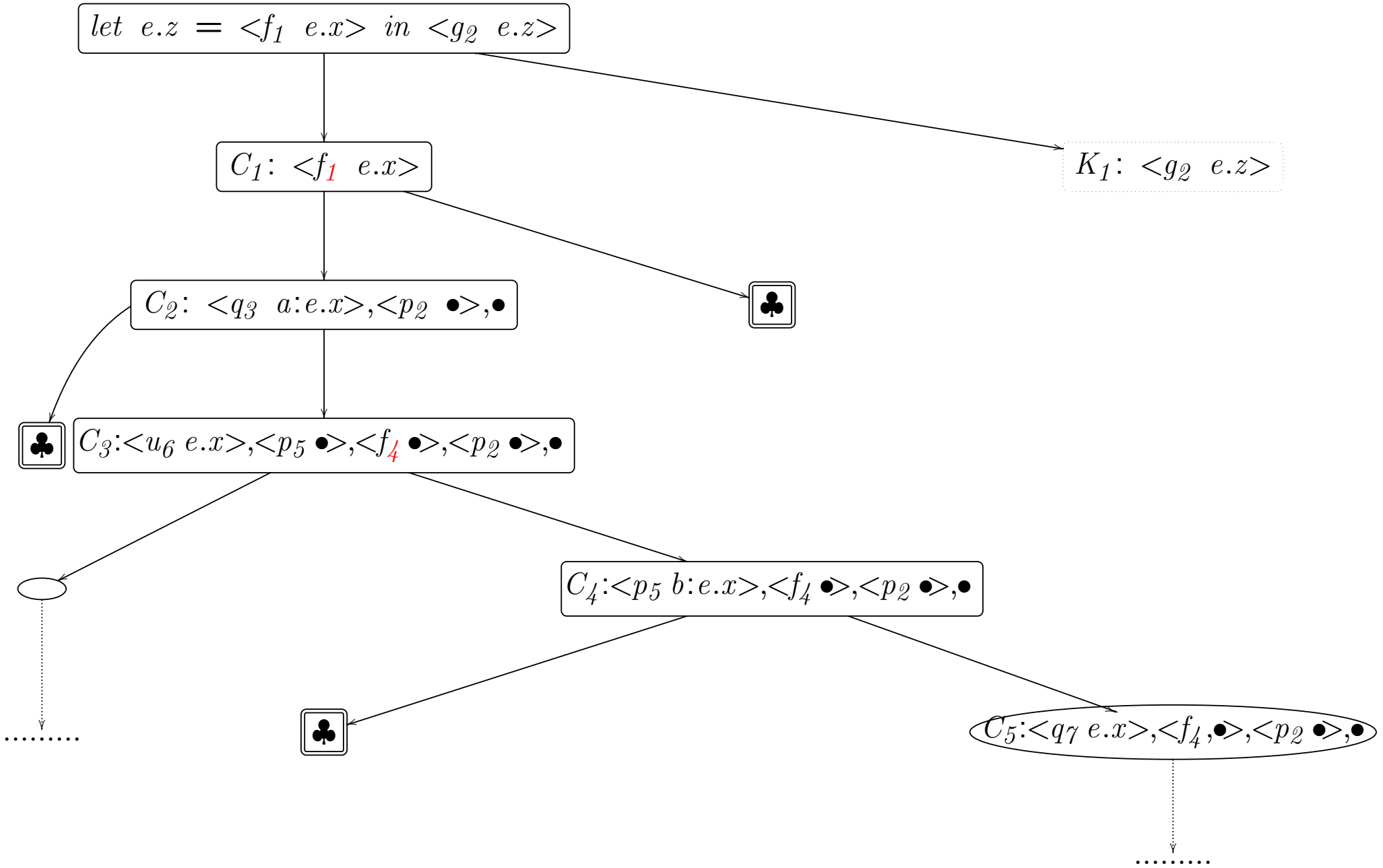
Хорошие бинарные отношения

Определение: **Бинарное отношение $\beta(\bullet,\bullet)$ на множестве $\mathfrak{M}$ назовем хорошим отношением на $\mathfrak{M}$, если для любой бесконечной последовательности $\{\mathrm{a_n}\}$, где $\mathrm{a_i} \in \mathfrak{M}$, существует пара $\mathrm{j,k} \in \mathbb{N}$ такая, что $\mathrm{j} < \mathrm{k}$ и $\beta(\mathrm{a_j}, \mathrm{a_k})$.**

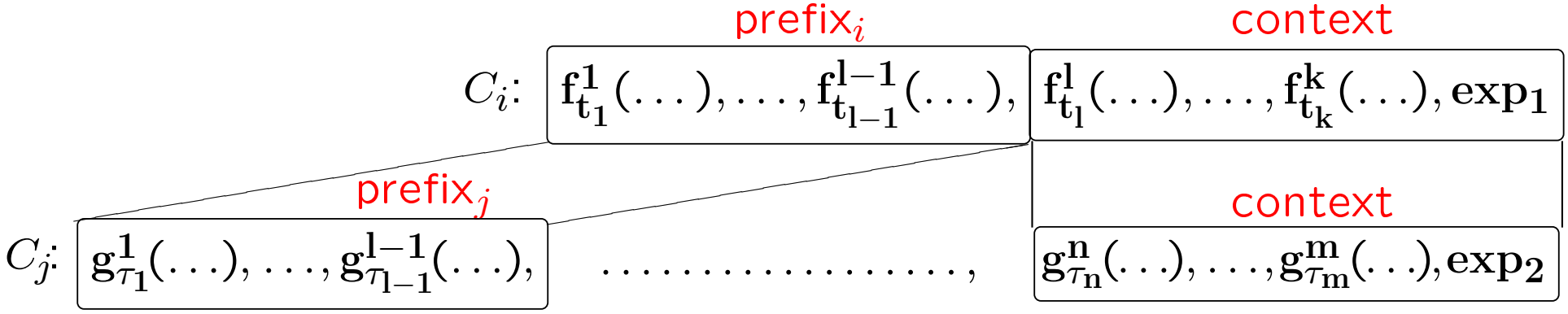
Определение: **Бинарное отношение $\preccurlyeq$ на множестве $\mathfrak{M}$ называется предпорядком, если $\forall \mathrm{a} \in \mathfrak{M} \,.\, \mathrm{a} \preccurlyeq \mathrm{a}$ и $\forall \mathrm{a,b,c} \in \mathfrak{M} \,.(\mathrm{a} \preccurlyeq \mathrm{b} \,\&\, \mathrm{b} \preccurlyeq \mathrm{c}) \Rightarrow (\mathrm{a} \preccurlyeq \mathrm{c})$.**

Хорошие отношения, определенные на элементах последовательностей конфигураций на путях дерева развертки программы, являются инструментами остановки процесса развертки программы вдоль этих путей.

Временны́е конфигурации



Отношение Турчина

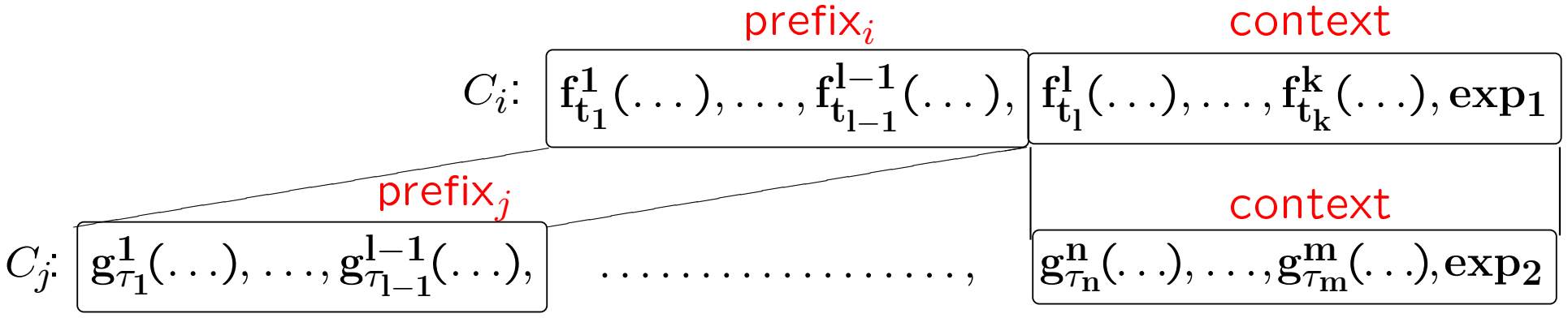


- $\forall \mathbf{s}\,.\,(0 < \mathbf{s} < \mathbf{l})\ \mathbf{f^s_{t_s}} \simeq \mathbf{g^s_{\tau_s}}$ **(т.е.,** $f^s = g^s$**);**
- $\mathbf{t_{l-1}} \neq \tau_{\mathbf{l-1}}$;
- $\mathbf{f^l_{t_l} = g^n_{\tau_n}, f^{l+1}_{t_{l+1}} = g^{n+1}_{\tau_{n+1}},\ldots,f^k_{t_k} = g^m_{\tau_m}}$
  (т.е., $\mathbf{f^l = g^n,\ldots}$ **и** $\mathbf{t_l = \tau_n,\ldots}$**), где** $\mathbf{k - l = m - n}$**.**

Говорят, что конфигурации $\mathbf{C_i}$**,** $\mathbf{C_j}$ **находятся в отношении Турчина** $\mathbf{C_i \triangleleft C_j}$**.**

На любом бесконечном пути $\mathbf{C_1, C_2, \ldots, C_n, \ldots}$ **найдутся две временны́х конфигурации** $\mathbf{C_i, C_j}$ **такие, что** $\mathbf{i < j}$ **и** $\mathbf{C_i \triangleleft C_j}$**.**

Отношение Турчина не является транзитивным отношением

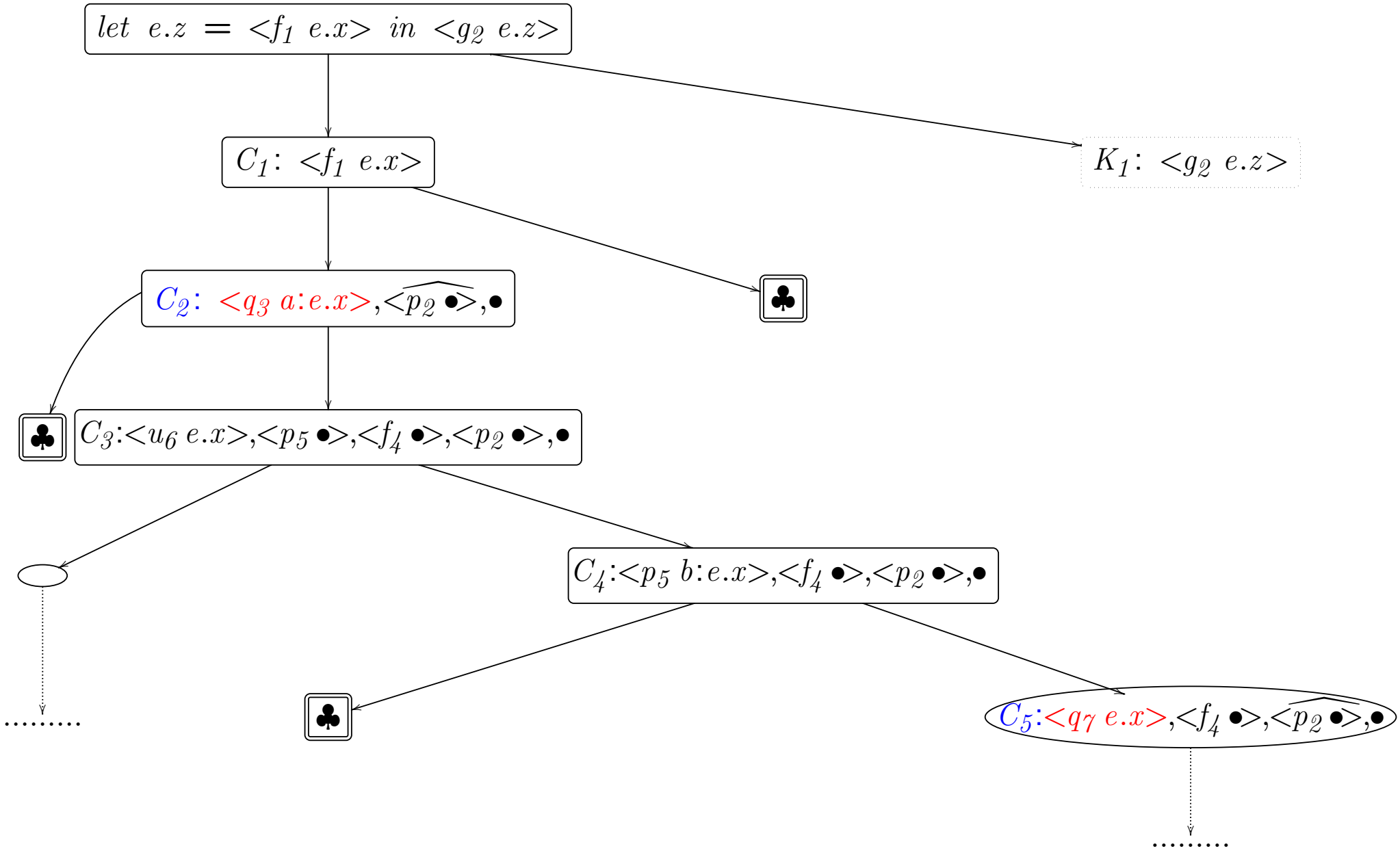


Говорят, что конфигурации C_i, C_j находятся в отношении Турчина $C_i \triangleleft C_j$.

Идея:

- **на данном отрезке пути вызовы функций из контекста не участвуют в вычислении конфигурации C_j;**
- **каждый вызов функции из префикса C_i участвует в вычислении конфигурации C_j.**

Пример: $C_2 \triangleleft C_5$



Задача №N: **Теория сложности вычислений и методы преобразования программ.**

«Несозданных миров отмститель будь, художник, —
Несуществующим существованье дай;
Туманным облаком окутай свой треножник
И падающих звезд пойми летучий рай!»

Осип Мандельштам, 1911 г.

Список литературы

- W. N. Chin. Towards an Automated Tupling Strategy // In Proc. of PEPM'93, pp: 119–132, 1993.
- Н. В. Верещагин, А. Х. Шень. Логические формулы и схемы // Математическое просвещение. Третья серия. Вып. 4. С. 53–80. МЦНМО, 2000.
- A. Pettorossi. A Powerful Strategy for Deriving Efficient Programs by Transformation // In: Proc. of the 1984 ACM Symposium on LISP and Functional Programming, pp: 273–281, 1984.
- A. Pettorossi, A. Skowron. Higher Order Generalization in Program Derivation // LNCS, vol. 250, pp: 182–196, 1987.
- В. Я. Пан. О способах вычисления значений многочленов // Успехи математических наук. 1966. Т. 21, вып. 1(127). С. 103–134.
- А. А. Разборов. Алгебраическая сложность // Летняя школа современной математики. МЦНМО, 2016.