О некоторых открытых задачах суперкомпиляции

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

Совместное рабочее совещание ИПС РАН и МГТУ имени Н.Э. Баумана
по функциональному языку программирования Рефал
5 июня 2018 г., Москва

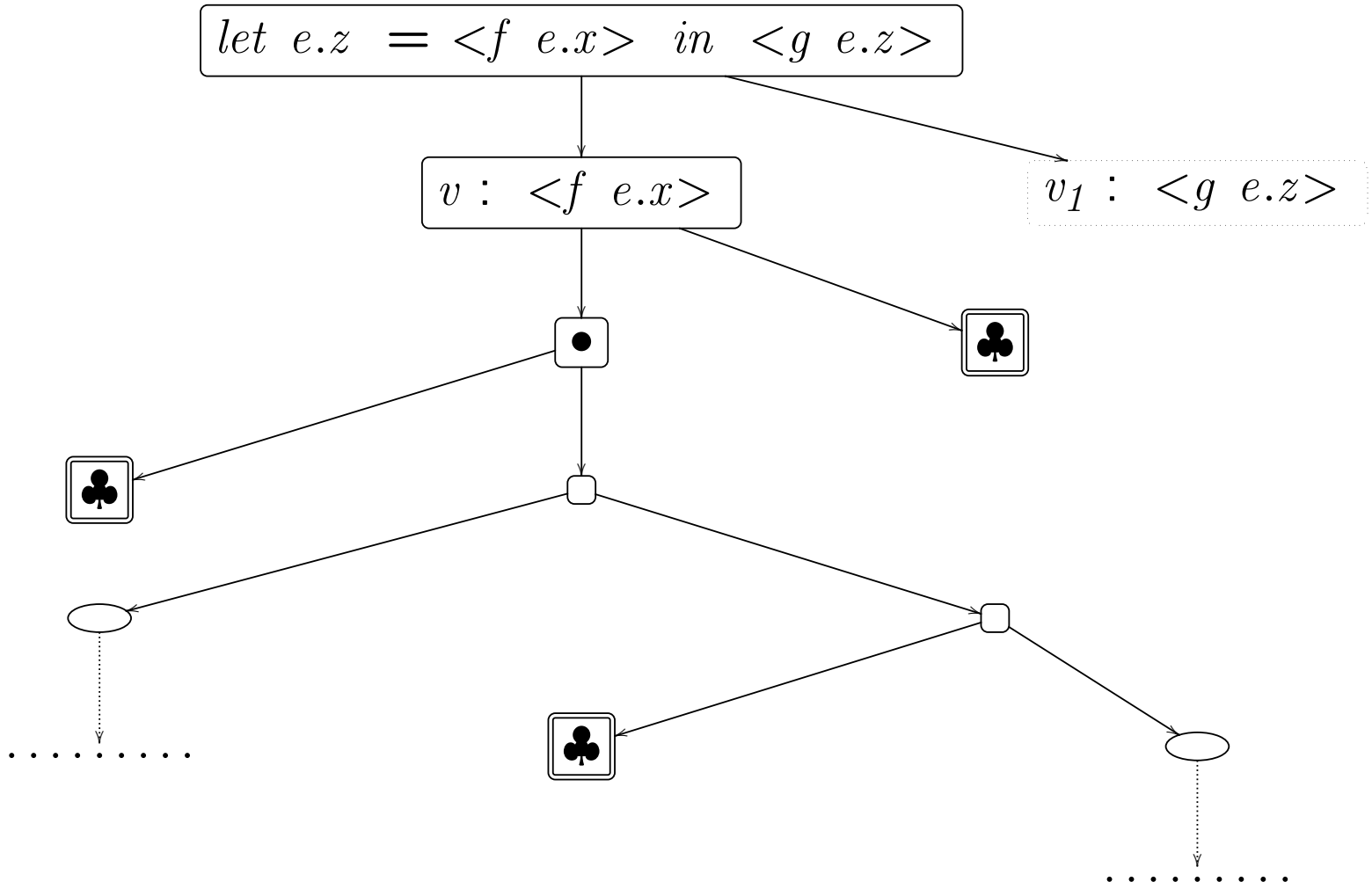
«Задачник огромных корней»

Осип Мандельштам, 1930-е гг.

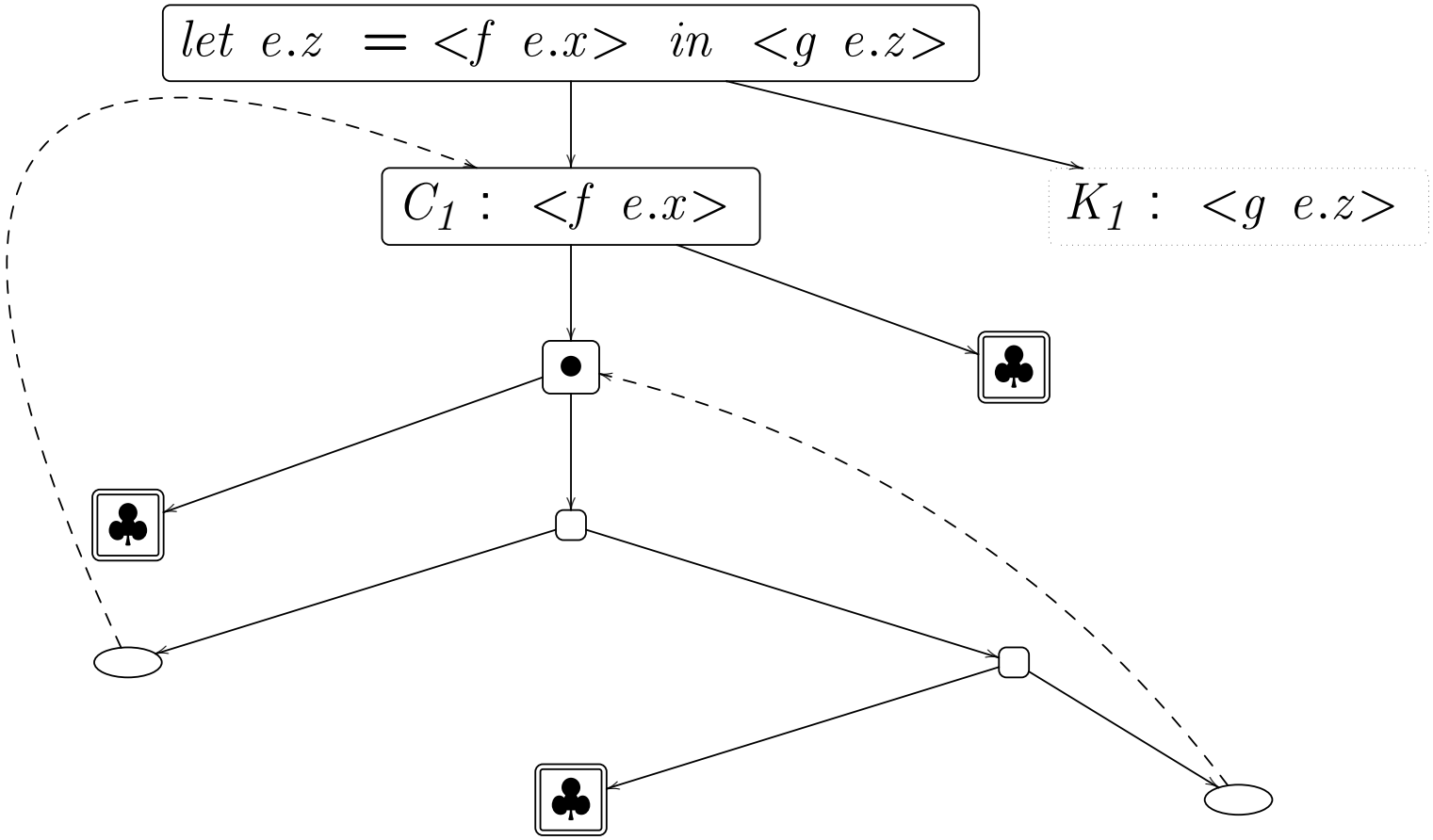
Задачи развития методов анализа и преобразования программ

5 июня 2018 г., Москва

Развертка



Свертка



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

Цели суперкомпиляции

– **Свернуть дерево развертки исходной программы P в конечный граф, который представляет программу Q – результат суперкомпиляции. Частичная функция $\mathfrak{F}_{\mathrm{Q}}$, определяемая программой Q, является расширением частичной функции $\mathfrak{F}_{\mathrm{P}}$, определенной программой P на О.Д.З. $\mathbb{D}_{\mathrm{P}}$:**
$\mathbb{D}_{\mathrm{P}} \subseteq \mathbb{D}_{\mathrm{Q}}$ $\&\forall \mathrm{d} \in \mathbb{D}_{\mathrm{P}}. \mathfrak{F}_{\mathrm{P}}(\mathrm{d}) = \mathrm{P}(\mathrm{d}) = \mathrm{Q}(\mathrm{d}) = \mathfrak{F}_{\mathrm{Q}}(\mathrm{d})$.

– $\forall \mathrm{d} \in \mathbb{D}_{\mathrm{P}}$ **«время»** T **вычисления** $\mathrm{T}[\mathrm{Q}(\mathrm{d})]$, **как правило, должно быть меньше, чем** $\mathrm{T}[\mathrm{P}(\mathrm{d})]$: $\mathrm{T}[\mathrm{Q}(\mathrm{d})] \leq \mathrm{T}[\mathrm{P}(\mathrm{d})]$.

Эти две цели часто противоречивы.

Цели суперкомпиляции

– **Обнаружить (и обрезать) как можно больше недостижимых путей в дереве развертки. Большее число итераций развертки часто позволяет обнаружить большее число недостижимых путей.**

– **Свернуть дерево развертки исходной программы P в конечный граф, который представляет программу Q – результат суперкомпиляции такой, что:**
$\mathbb{D}_{\mathbf{P}} \subseteq \mathbb{D}_{\mathbf{Q}}$ $\&\forall \mathrm{d} \in \mathbb{D}_{\mathbf{P}}. \mathfrak{F}_{\mathbf{P}}(\mathrm{d}) = \mathrm{P}(\mathrm{d}) = \mathrm{Q}(\mathrm{d}) = \mathfrak{F}_{\mathbf{Q}}(\mathrm{d})$.

– $\forall \mathrm{d} \in \mathbb{D}_{\mathbf{P}}. \mathrm{T}[\mathrm{Q}(\mathrm{d})]$, **как правило, должно быть меньше, чем** $\mathrm{T}[\mathrm{P}(\mathrm{d})]$:
$\mathrm{T}[\mathrm{Q}(\mathrm{d})] \leq \mathrm{T}[\mathrm{P}(\mathrm{d})]$.

Эти две цели часто противоречивы.

Причины существования бесконечных путей в дереве развертки

Длина описаний параметризованных состояний программы может неограничено расти вдоль данного пути:

- **размер описания параметризованных данных;**
- **число вызовов функций в стеке функций.**

Декомпозиция параметризованного стека функций

$\mu_v(exp)$ **обозначает кратность переменной в выражении** exp**.**

Определение: Параметризованной конфигурацией называется конечная последовательность вида

$$\texttt{let e.h = <}\mathrm{f_1}\ \mathrm{exp_{11}},\ \ldots,\ \mathrm{exp_{1m}}\texttt{>}\ \texttt{in}\ \ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ \texttt{let e.h = <}\mathrm{f_k}\ \mathrm{exp_{k1}},\ \ldots,\ \mathrm{exp_{kj}}\texttt{>}\ \texttt{in}\ \mathrm{exp_{n+1}}$$

где $\mathrm{exp_{n+1}}$ **есть пассивное выражение,**
$\forall \mathrm{i} > 1.\ \mu_{\mathrm{e.h}}(\texttt{<}\mathrm{f_i}\ \ldots\texttt{>}) = \mu_{\mathrm{e.h}}(\mathrm{exp_{n+1}}) = 1,\ \mu_{\mathrm{e.h}}(\texttt{<}\mathrm{f_1}\ \ldots\texttt{>}) = 0;$
$\forall \mathrm{i,j}$ **переменная** $\mathrm{e.h}$ **не входит в вызовы функций, являющиеся подвыражениями выражения** $\mathrm{exp_{ij}}$**.**

Декомпозиция параметризованного стека функций

$$\texttt{let e.h = } \langle f_1\ exp_{11},\ \ldots,\ exp_{1m}\rangle\ \texttt{in}\ \ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\quad \texttt{let e.h = } \langle f_k\ exp_{k1},\ \ldots,\ exp_{kj}\rangle\ \texttt{in}\ exp_{n+1}$$

Т.к. в стеке значение переменной $e.h$ переприсваивается в каждом `let`, то используем представление:

$$\langle f_1\ exp_{11},\ \ldots,\ exp_{1m}\rangle,\ \ldots,\ \langle f_k\ exp_{k1},\ \ldots,\ exp_{kj}\rangle,\ exp_{n+1}$$

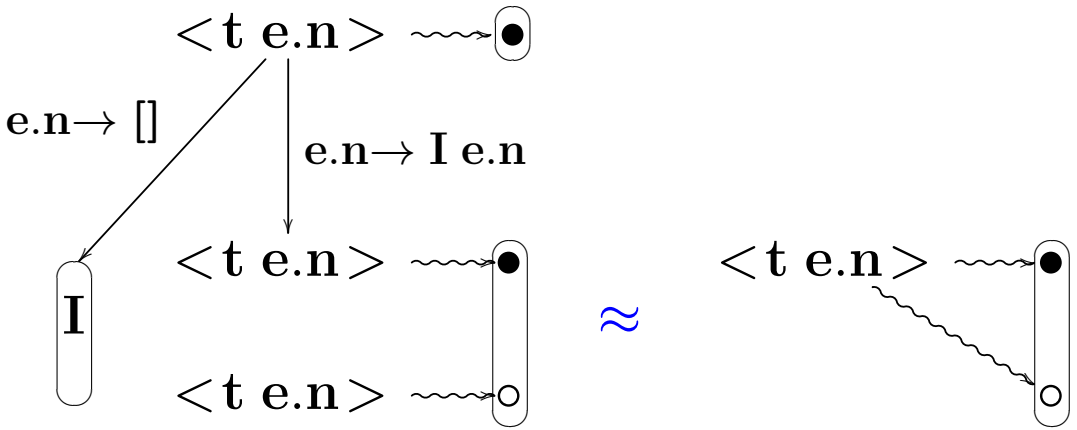
- **$e.h$ заменяется знаком $\bullet$**
- **последнее выражение может быть опущено, если оно равно $\bullet$**

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

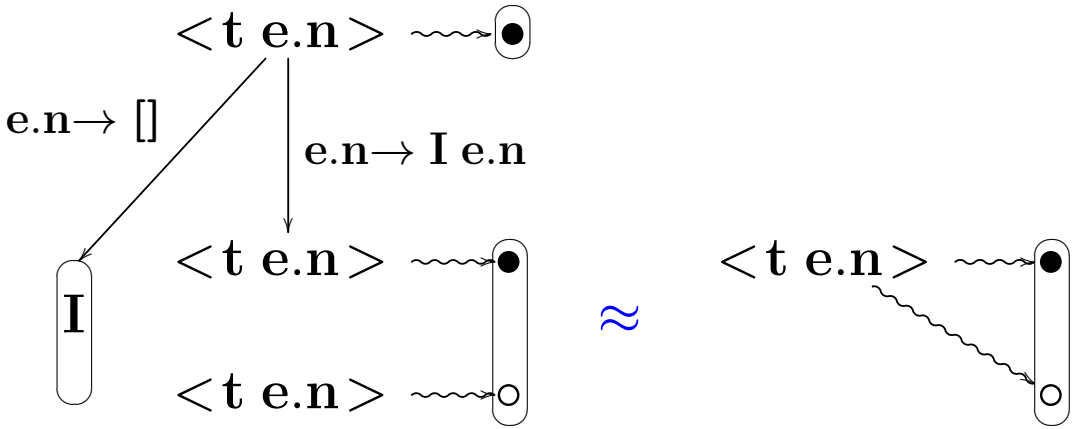
Пример

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

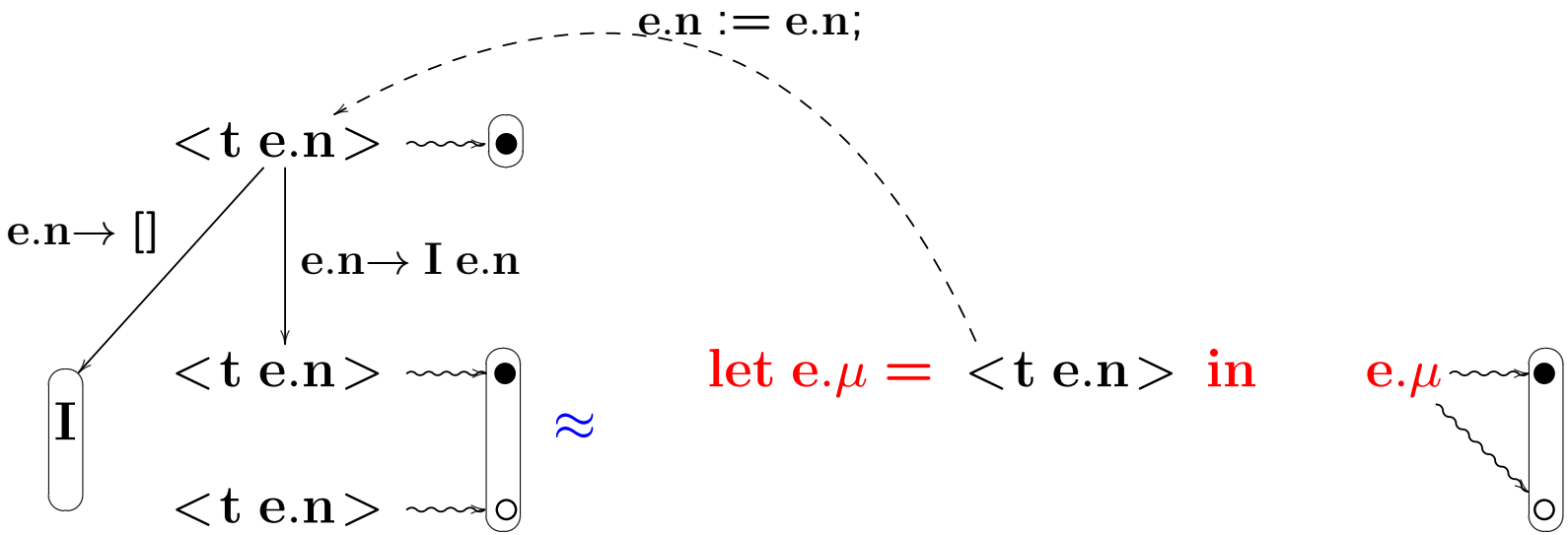
Прогонка:



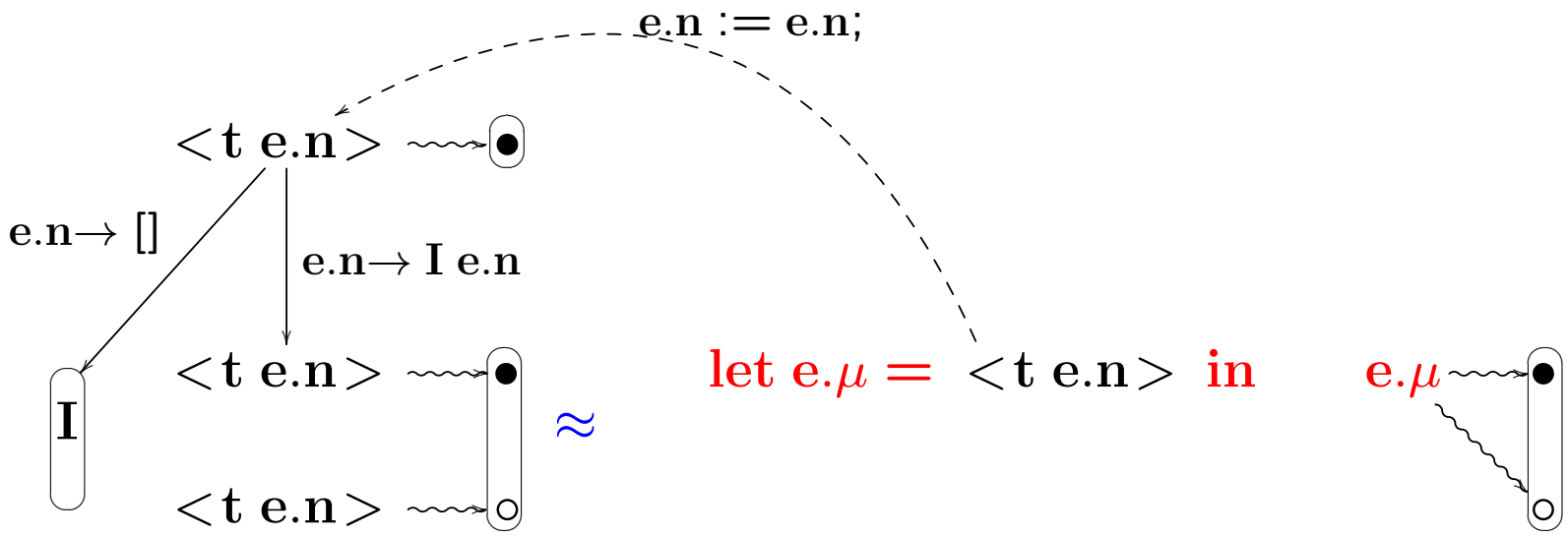
Прогонка:



Свертка:



Свертка:



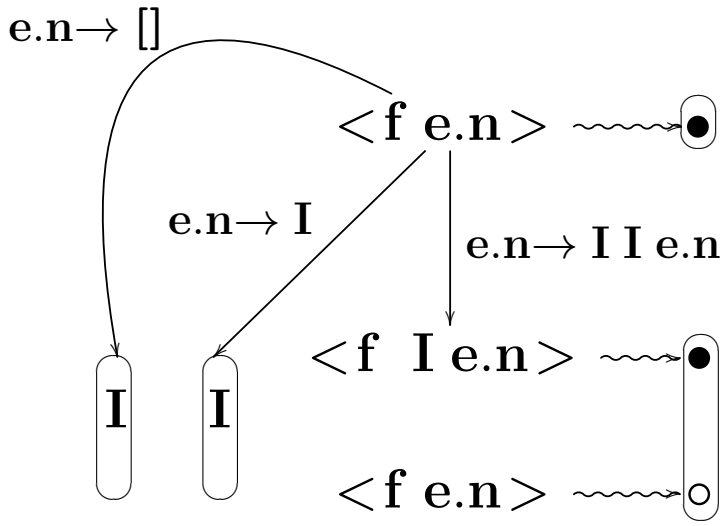
Остаточная программа:

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

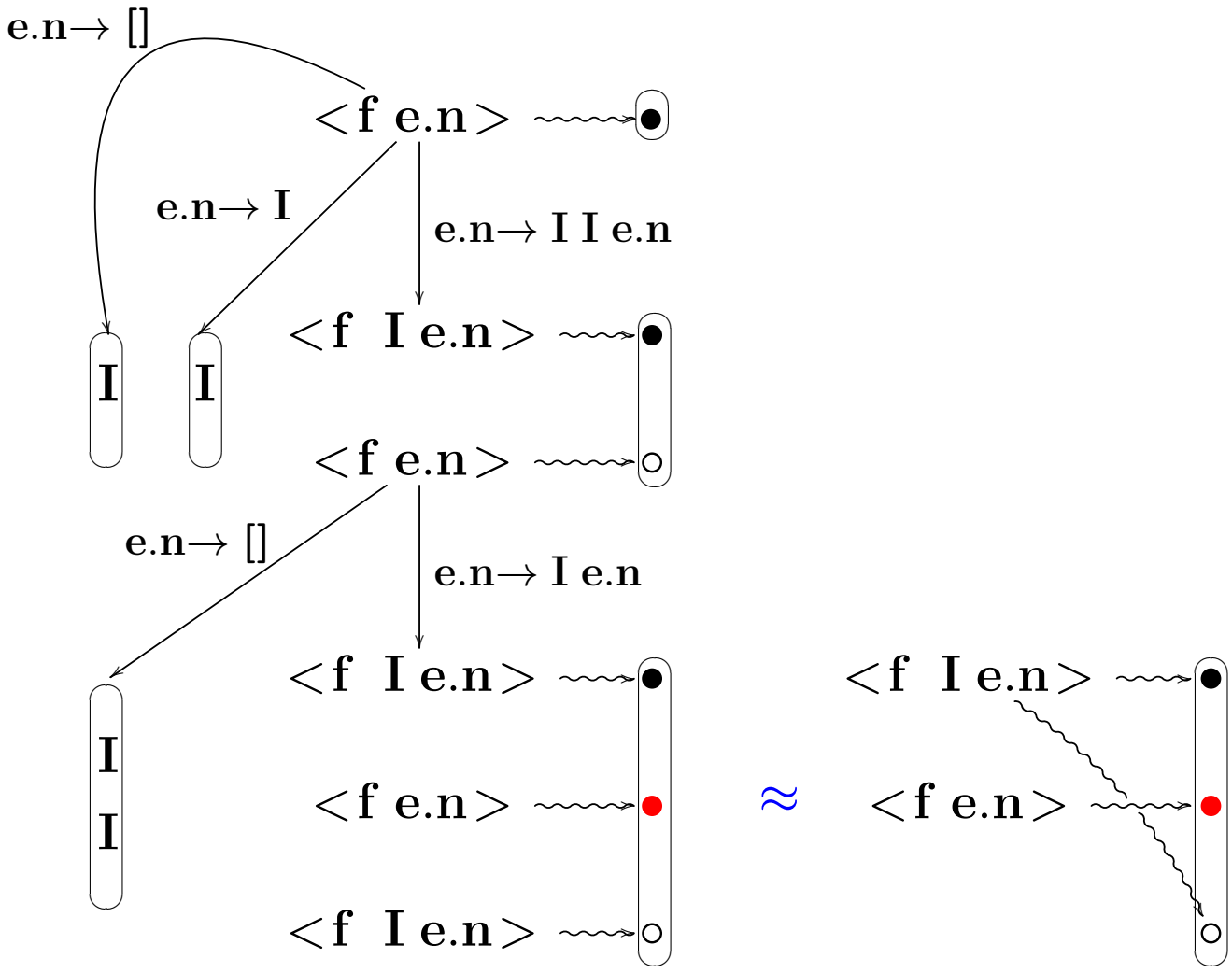
Пример №2

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

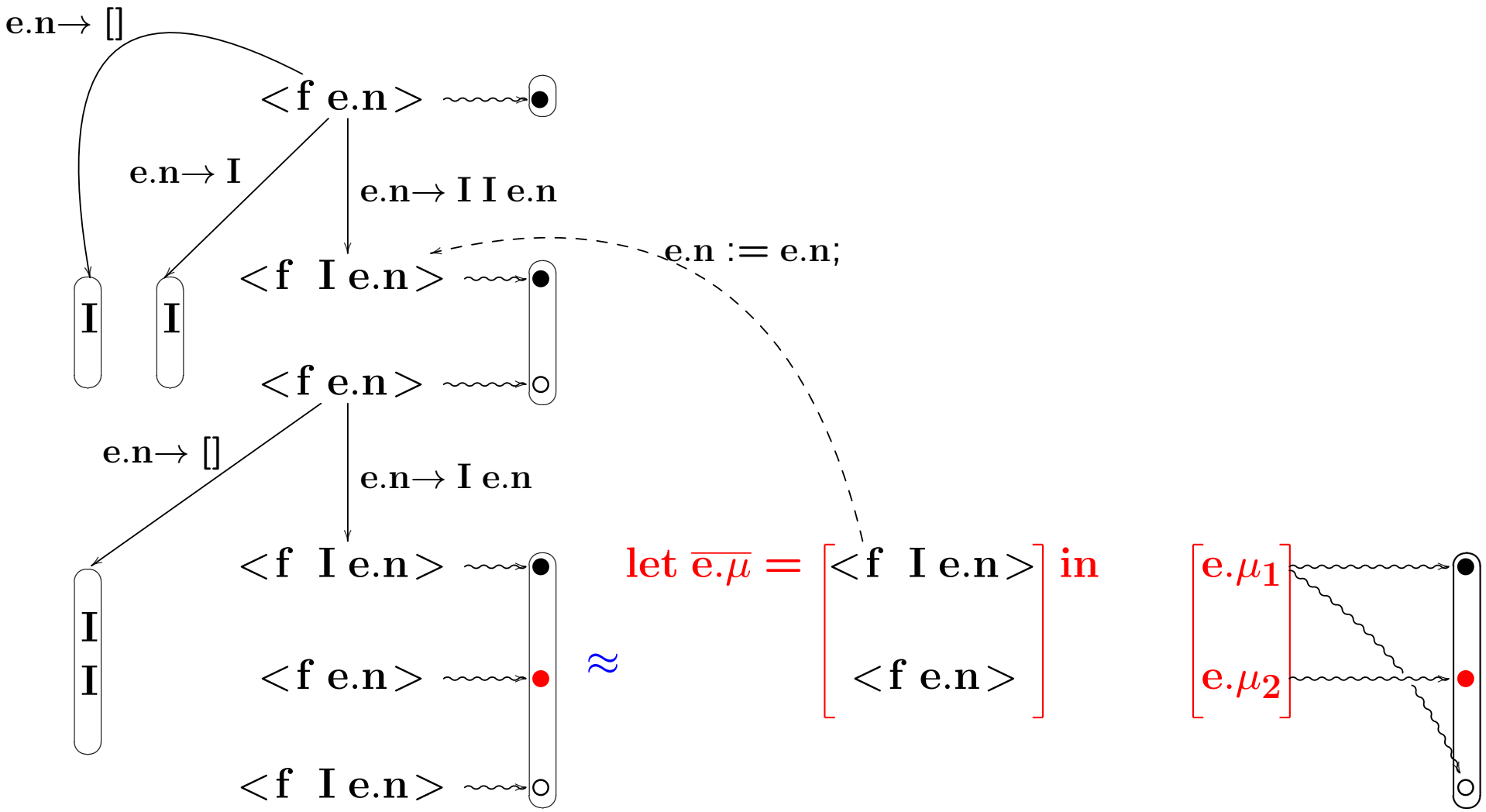
Прогонка:



Прогонка:


e.n→ []
<f e.n>
e.n→ I
e.n→ I I e.n
I
I
<f I e.n>
<f e.n>
e.n→ []
e.n→ I e.n
I
I
<f I e.n>
<f e.n>
<f I e.n>
≈
<f I e.n>
<f e.n>

Свертка:


e.n→ []
<f e.n>
e.n→ I
e.n→ I I e.n
e.n := e.n;
I
I
<f I e.n>
<f e.n>
e.n→ []
e.n→ I e.n
I
I
<f I e.n>
<f e.n>
<f I e.n>
≈
let e.μ = [<f I e.n> <f e.n>] in
[e.μ1 e.μ2]

Входная программа:

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

Остаточная программа:

```
F {
        = I;
   I    = I;
   I I  = I I;
 I I I e.n, <F  I e.n>: e.μ1 = e.μ1  <F  e.n>  e.μ1;
}
```

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.

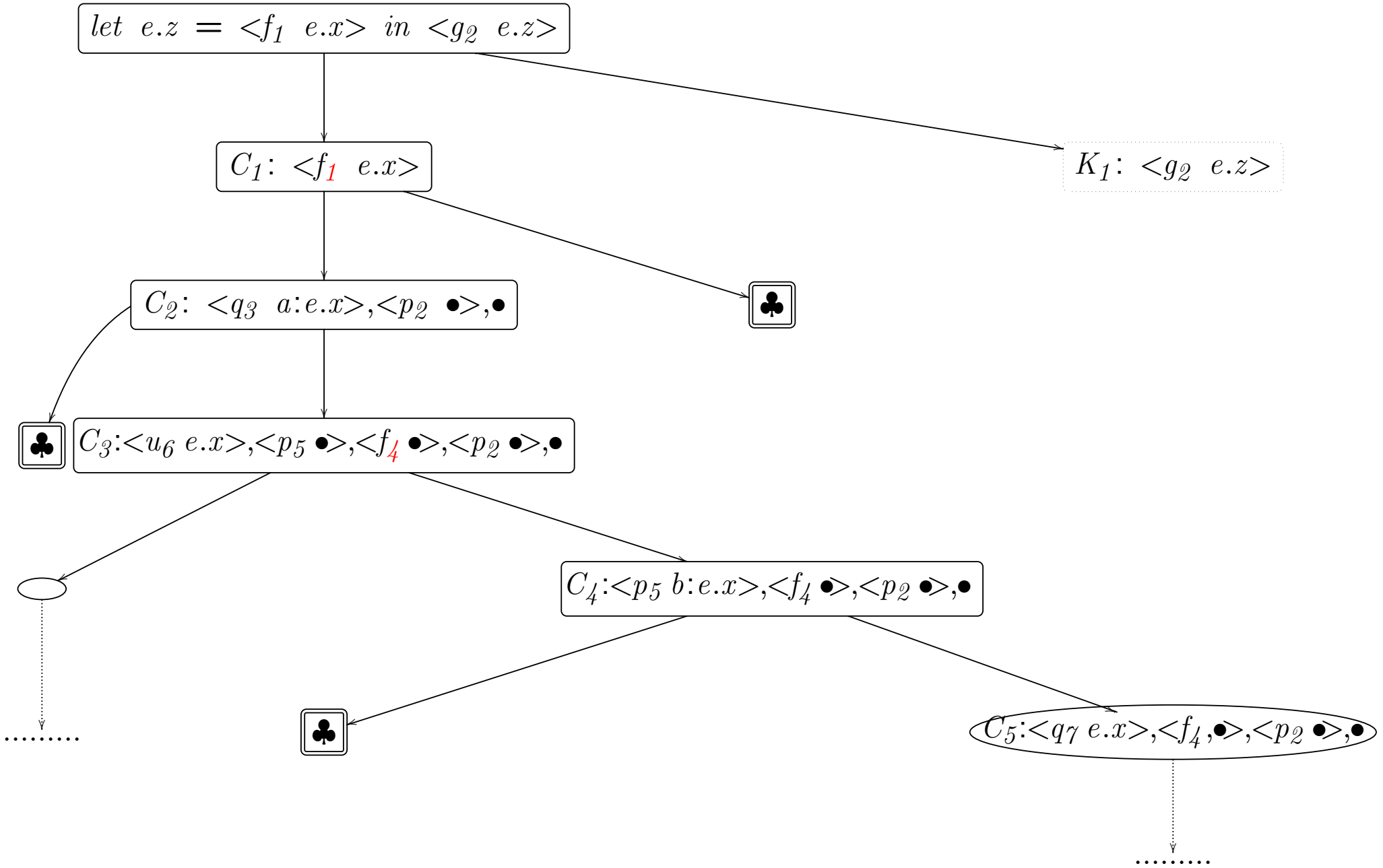
Хорошие бинарные отношения

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

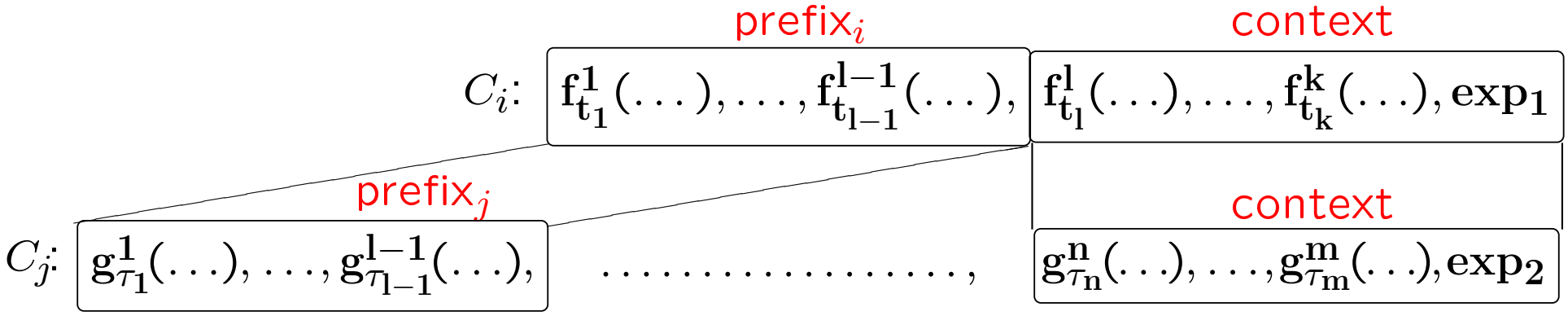
Определение: Бинарное отношение $\preccurlyeq$ на множестве $\mathfrak{M}$ называется предпорядком, если $\forall a \in \mathfrak{M}\,.\,a \preccurlyeq a$ и $\forall a,b,c \in \mathfrak{M}\,.(a \preccurlyeq b \,\&\, b \preccurlyeq c) \Rightarrow (a \preccurlyeq c)$.

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

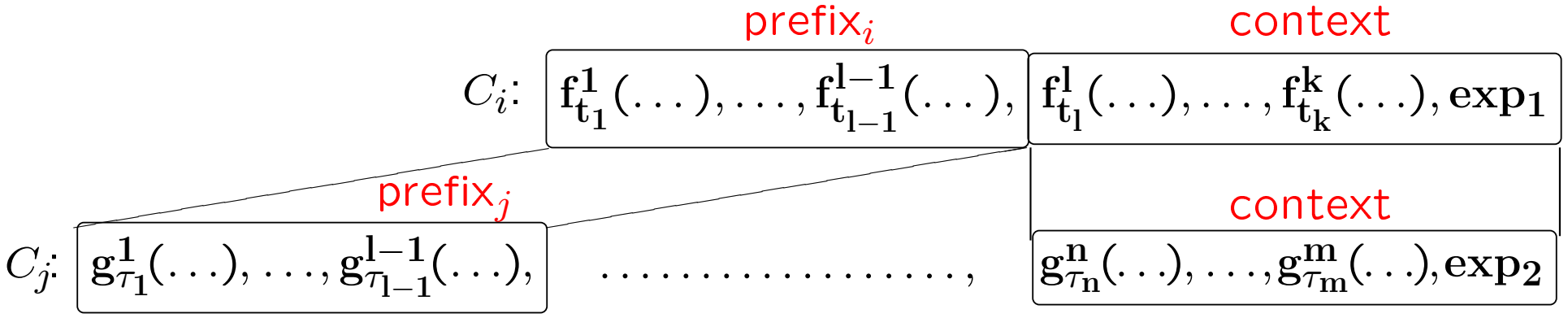


- $\forall \mathbf{s}\,.\,(0 < \mathbf{s} < \mathbf{l})\ \mathbf{f^s_{t_s}} \simeq \mathbf{g^s_{\tau_s}}$ **(т.е.,** $f^s = g^s$**);**
- $\mathbf{t_{l-1}} \neq \tau_{\mathbf{l-1}}$;
- $\mathbf{f^l_{t_l} = g^n_{\tau_n}, f^{l+1}_{t_{l+1}} = g^{n+1}_{\tau_{n+1}},\ldots,f^k_{t_k} = g^m_{\tau_m}}$
  (т.е.,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

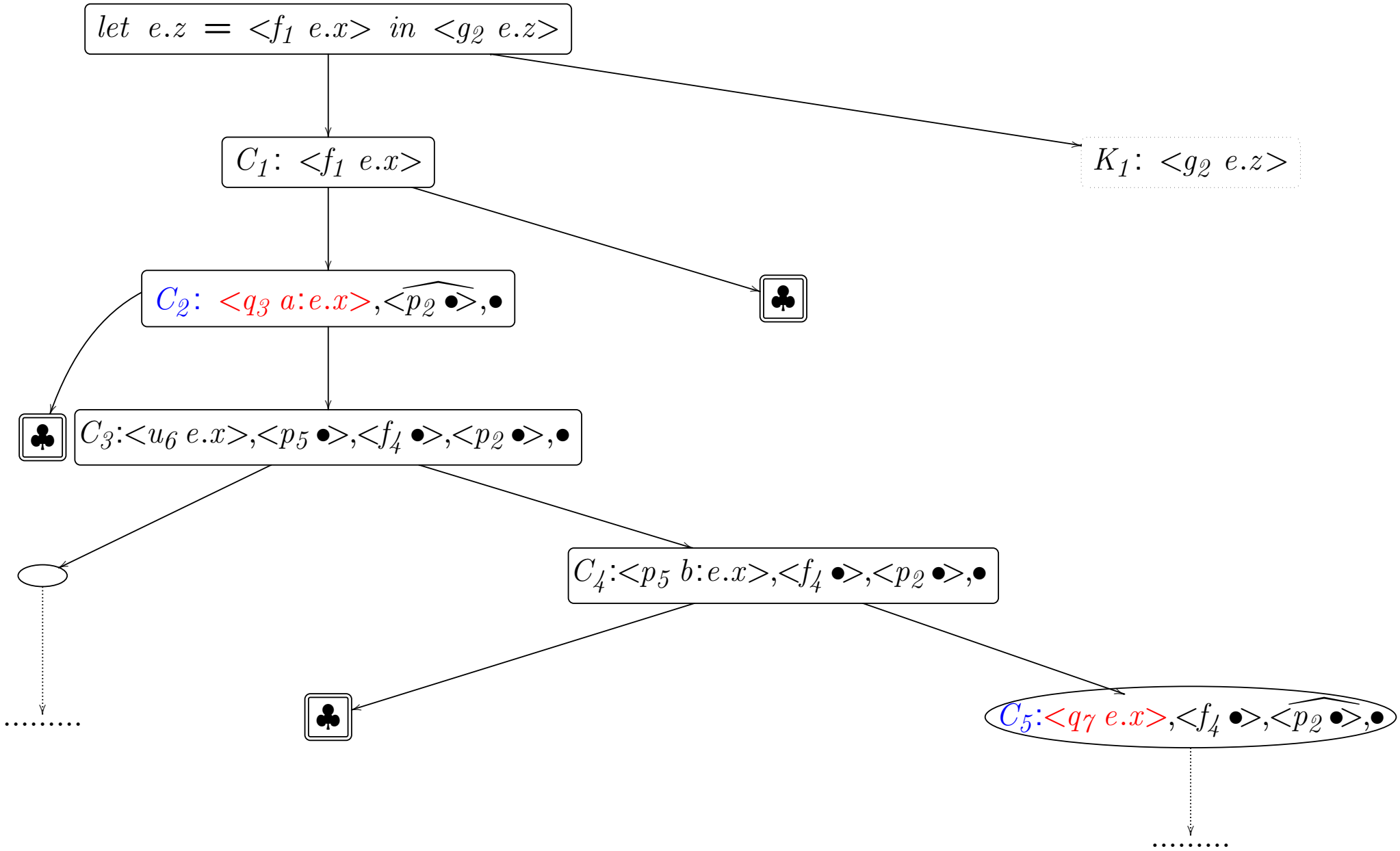


Говорят, что конфигурации C_i, C_j находятся в отношении Турчина $C_i \triangleleft C_j$.

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 №2 (В.Ф. Турчин, для студентов): Экспериментальное исследование возможностей отношения Турчина при обобщении структур данных – деревьев.

- Унарный конструктор построения дерева (•) может моделироваться, как одношаговая функция.
- Рассматриваем только программы с конечным алфавитом символов-констант.
- Инструменты исследования: суперкомпиляторы SCP4 и MSCP-A.

$$\mathrm{Bracket}\,\{\quad \mathrm{s.name_0}\ \mathrm{e.x} = <\mathrm{s.name_0}\ \mathrm{e.x}>;\quad \}$$
$$\mathrm{s.name_0}\,\{\quad \mathrm{e.x} = (\mathrm{s.name_0}\ \mathrm{e.x});\quad \}$$

Задача №3: **Разработка и реализация языка описания стратегий суперкомпиляции.**

Задача №4: **Внедрение алгоритмов, опробованных в суперкомпиляторах, в оптимизирующие компиляторы языка Рефал.**

Задача №5: Реализация модельного SCP-C с языком описания свойств конфигураций программ, невыразимых посредством подстановок параметров («рестрикций»).

- Основанном на языке диофантовых уравнений и неравенств.
- Как альтернатива SMT (satisfiability modulo theories) «решателям», но при изучении последних.

Уточнение понятия параметризованной(ого) конфигурации (состояния) программы

Пусть дана программа P, написанная на Рефале.

Определение: Параметризованным(ой) состоянием (конфигурацией) программы P называется пара

$$[\Delta(\mathrm{expr}(\mathrm{p_1}, \ldots, \mathrm{p_n})), \Phi(\mathrm{p_1}, \ldots, \mathrm{p_n})]$$

Где $\mathrm{expr}(\mathrm{p_1}, \ldots, \mathrm{p_n})$ – Рефал выражение, зависящее от параметров $\mathrm{p_1}, \ldots, \mathrm{p_n}$ и, возможно, декомпозированое $\Delta(\mathrm{expr}(\mathrm{p_1}, \ldots, \mathrm{p_n}))$ в последовательность вызовов, в порядке которой они будут вычисляться. Параметры $\mathrm{p_1}, \ldots, \mathrm{p_n}$ суть неизвестные ($\mathrm{e}-$, $\mathrm{t}-$, $\mathrm{s}-$)типизированные данные. $\Phi(\mathrm{p_1}, \ldots, \mathrm{p_n})$ – свойство значений параметров, описанное в некотором языке L.

Язык L называется языком рестрикций.

Примеры языков рестрикций

– **простейший язык, содержащий связки отрицания и конъюнкции (**$\#\mathrm{s.x} \neq \#\mathrm{s.y}$**, SCP4);**

– **язык уравнений в словах (**$\mathrm{a}\ \#\mathrm{e.y} = \#\mathrm{e.y}\ \mathrm{a}$**, MSCP-A);**

– **алгоритмически полный язык, например, Рефал-5.**

Задача №6 (В.Ф. Турчин): **Реализация модельного SCP-D, входным языком которого и языком рестрикций является Рефал-5.**

Рекурсивные вызовы самого себя для анализа свойств конфигураций:

$$\text{<SCP-D } [\Delta(\text{expr}(p_1,\ldots,p_n)),\text{<SCP-D } \Phi(p_1,\ldots,p_n)\text{>}]\text{>}$$

$$\begin{array}{l}\text{<SCP-D } [\Delta(\text{expr}(p_1,\ldots,p_n)),\\ \qquad \text{<SCP-D } [\Delta(\text{expr}_1(p_{11},\ldots,p_{1n})),\text{<SCP-D } \Phi_1(p_{11},\ldots,p_{1n})\text{>}]\text{>}\\ \qquad]\\ \text{>}\end{array}$$

. . . . . .

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

«Я знаю, что обман в видении немыслим
И ткань моей мечты прозрачна и прочна;
Что с дивной легкостью мы, созидая, числим
И достигает звезд полет веретена ...»

Осип Мандельштам, 1911 г.