

# **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**

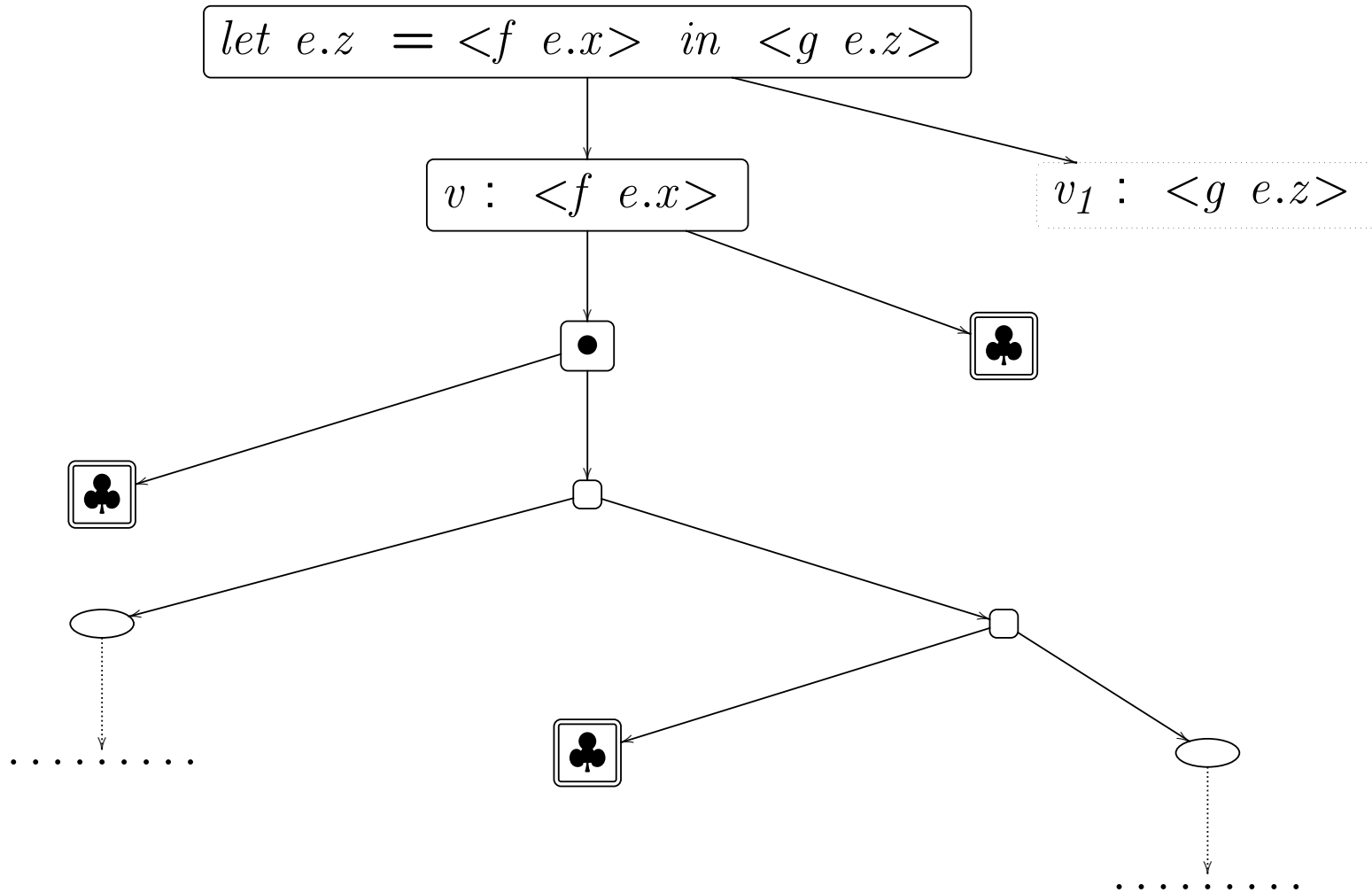
**«Задачник огромных корней»**

**Осип Манделъштам, 1930-е гг.**

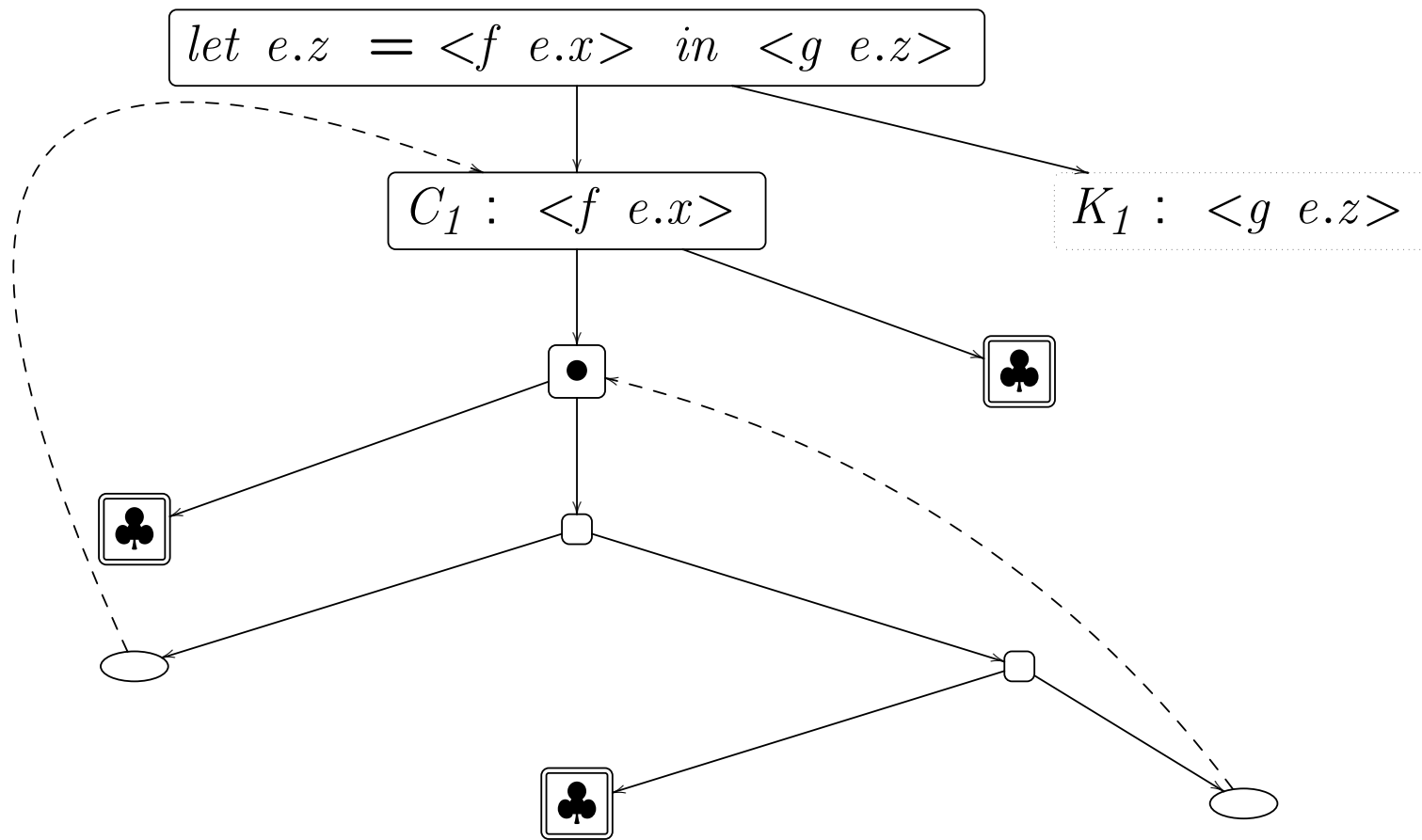
**Задачи развития методов  
анализа и преобразования программ**

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  $\tilde{\mathfrak{f}}_Q$ , определяемая программой  $Q$ , является расширением частичной функции  $\tilde{\mathfrak{f}}_P$ , определенной программой  $P$  на О.Д.З.  $\mathbb{D}_P$ :  
$$\mathbb{D}_P \subseteq \mathbb{D}_Q \ \& \ \forall d \in \mathbb{D}_P. \tilde{\mathfrak{f}}_P(d) = P(d) = Q(d) = \tilde{\mathfrak{f}}_Q(d).$$
- $\forall d \in \mathbb{D}_P$  «время»  $T$  вычисления  $T[Q(d)]$ , **как правило**, должно быть меньше, чем  $T[P(d)]$ :  $T[Q(d)] \leq T[P(d)]$ .

Эти две цели часто противоречивы.

## Цели суперкомпиляции

- Обнаружить (и обрезать) как можно больше недостижимых путей в дереве развертки. Большое 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:  
$$\mathbb{D}_P \subseteq \mathbb{D}_Q \ \& \ \forall d \in \mathbb{D}_P. \ \mathfrak{F}_P(d) = P(d) = Q(d) = \mathfrak{F}_Q(d).$$
- $\forall d \in \mathbb{D}_P. \ T[Q(d)],$  **как правило**, должно быть меньше, чем  $T[P(d)]:$   
 $T[Q(d)] \leq T[P(d)].$

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

$\mu_v(\text{exp})$  обозначает кратность переменной в выражении  $\text{exp}$ .

**Определение:** Параметризованной конфигурацией называется конечная последовательность вида

let e.h =  $\langle f_1 \text{exp}_{11}, \dots, \text{exp}_{1m} \rangle$  in ...

..... let e.h =  $\langle f_k \text{exp}_{k1}, \dots, \text{exp}_{kj} \rangle$  in  $\text{exp}_{n+1}$

где  $\text{exp}_{n+1}$  есть пассивное выражение,

$\forall i > 1. \mu_{e.h}(\langle f_i \dots \rangle) = \mu_{e.h}(\text{exp}_{n+1}) = 1, \mu_{e.h}(\langle f_1 \dots \rangle) = 0;$

$\forall i, j$  переменная e.h не входит в вызовы функций, являющиеся подвыражениями выражения  $\text{exp}_{ij}$ .

## Декомпозиция параметризованного стека функций

let e.h =  $\langle f_1 \text{ exp}_{11}, \dots, \text{exp}_{1m} \rangle$  in ...  
 ..... let e.h =  $\langle f_k \text{ exp}_{k1}, \dots, \text{exp}_{kj} \rangle$  in  $\text{exp}_{n+1}$

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

$\langle f_1 \text{ exp}_{11}, \dots, \text{exp}_{1m} \rangle, \dots, \langle f_k \text{ exp}_{k1}, \dots, \text{exp}_{kj} \rangle, \text{exp}_{n+1}$

- e.h заменяется знаком •
- последнее выражение может быть опущено, если оно равно •

### Пример

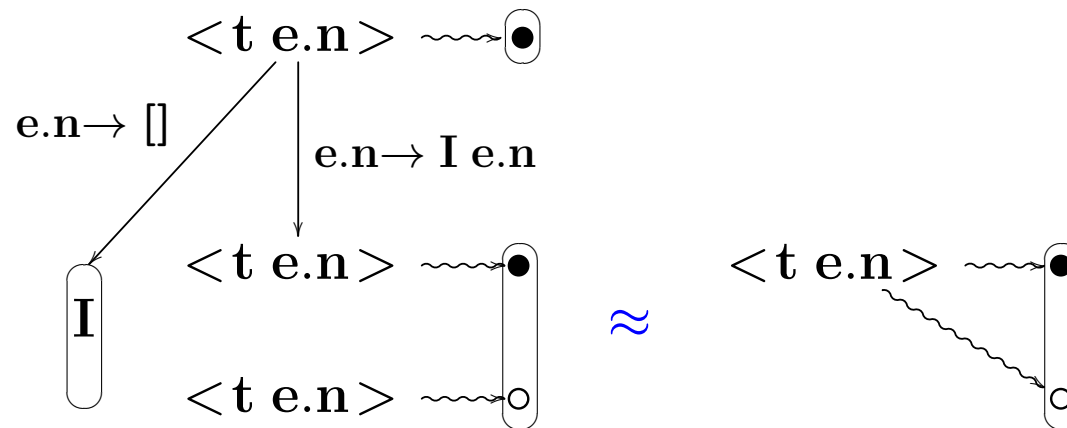
$\langle f \ a : e.xs \ ++ \ e.ys \rangle, \langle g \ \bullet \ ++ \ e.ys, (\text{Var } b \ c), [] \rangle, \langle f \ s.x : \bullet \rangle,$   
 $s.x : \bullet \ ++ \ \langle t \ s.x : e.zs \rangle, \bullet$   
 $= s.x \ \langle f \ s.x \ \langle g \ \langle f \ a \ e.xs \ e.ys \rangle \ e.ys, (\text{Var } b \ c), [] \rangle \ \langle t \ s.x \ e.zs \rangle \rangle$

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

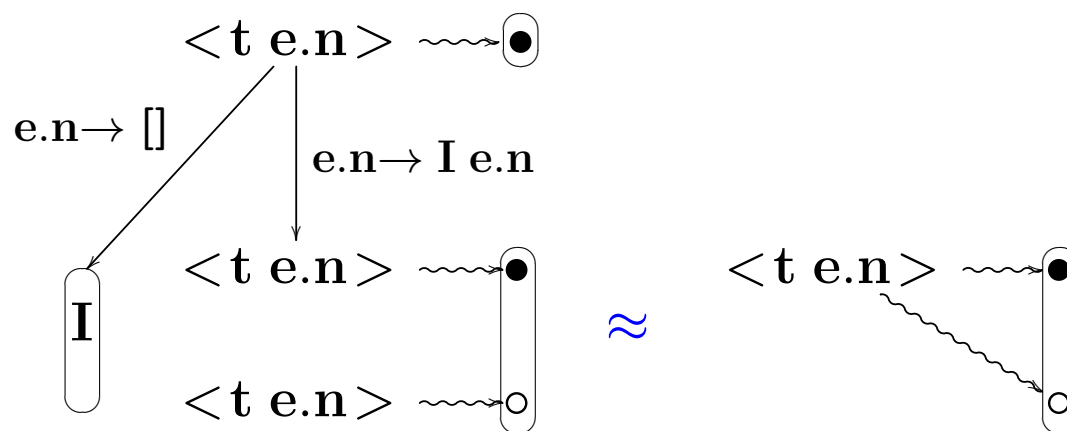
### Пример

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

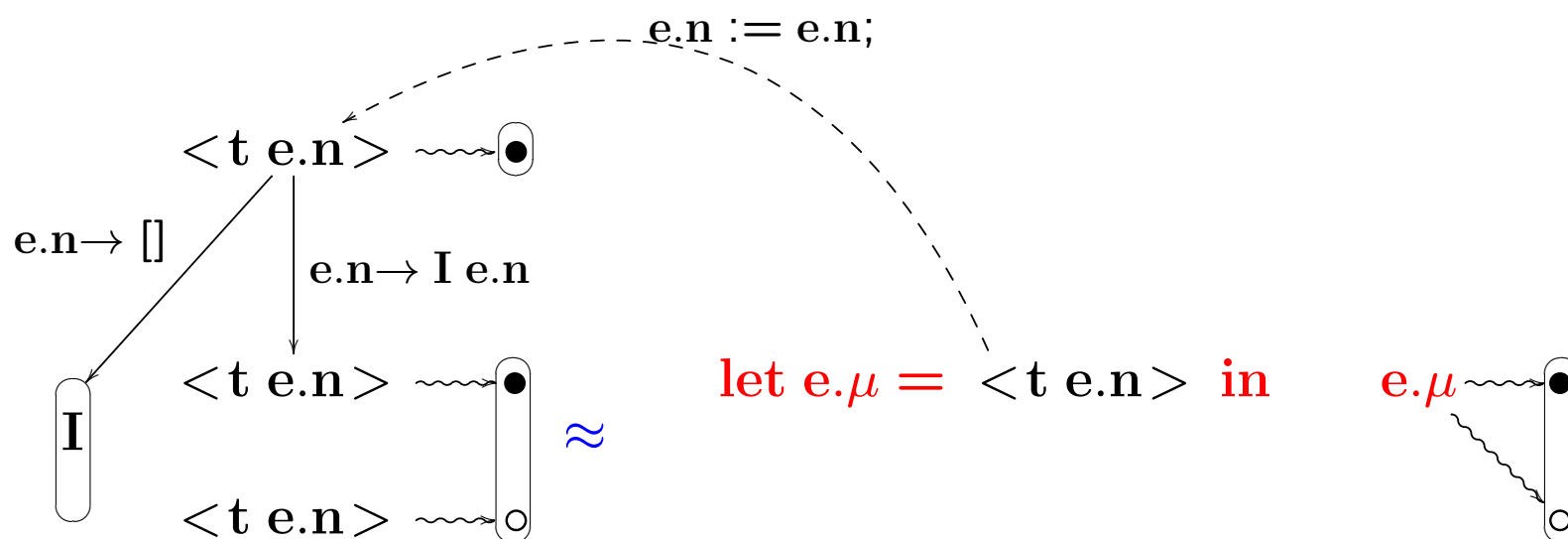
**Прогонка:**



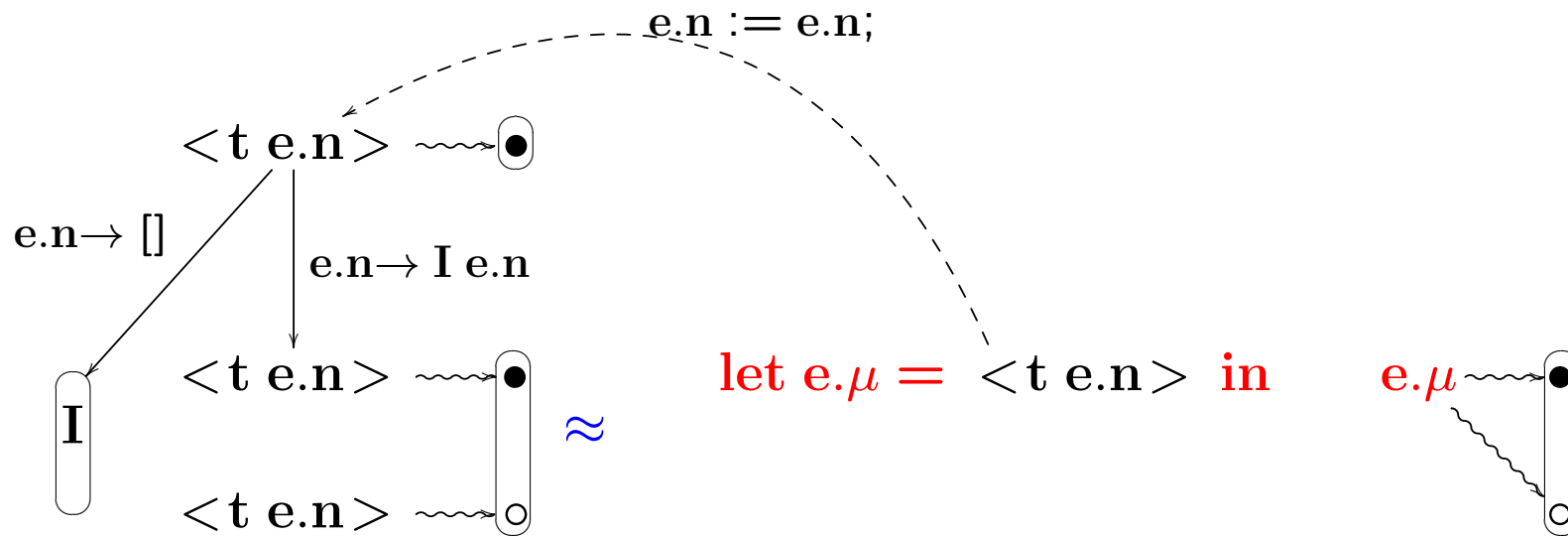
Прогонка:



Свертка:



Свертка:



Остаточная программа:

$T \{$   
 $\quad \quad \quad = I;$   
 $I e.n, \langle T e.n \rangle: e.\mu = e.\mu e.\mu;$   
 $\}$

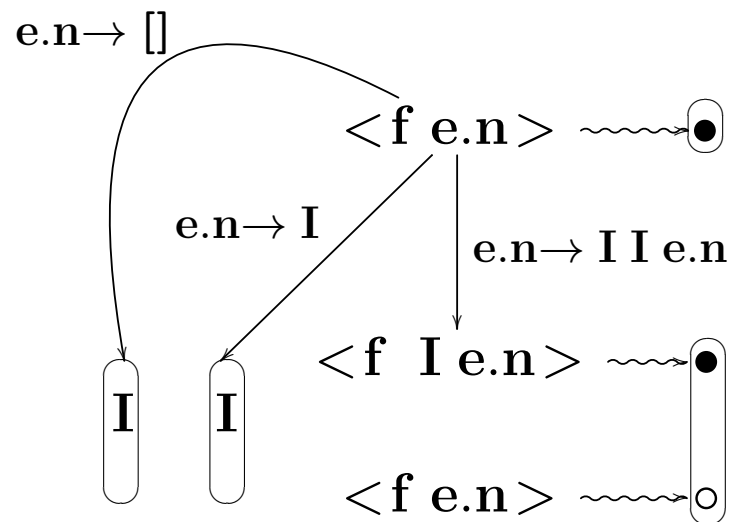
## Пример №2

```

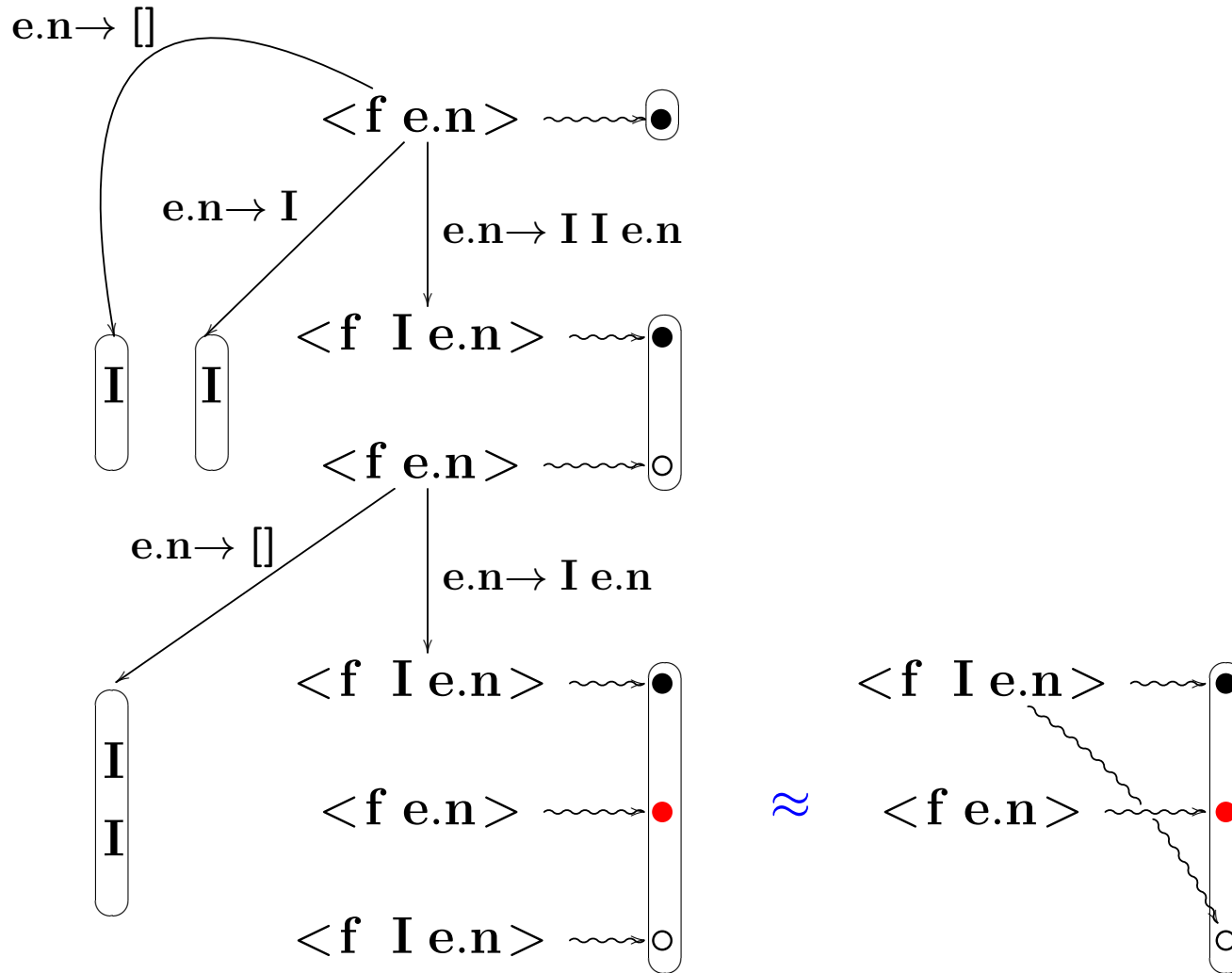
f {
    = I;
    I = I;
    II e.n = <f I e.n> <f e.n> ;
}

```

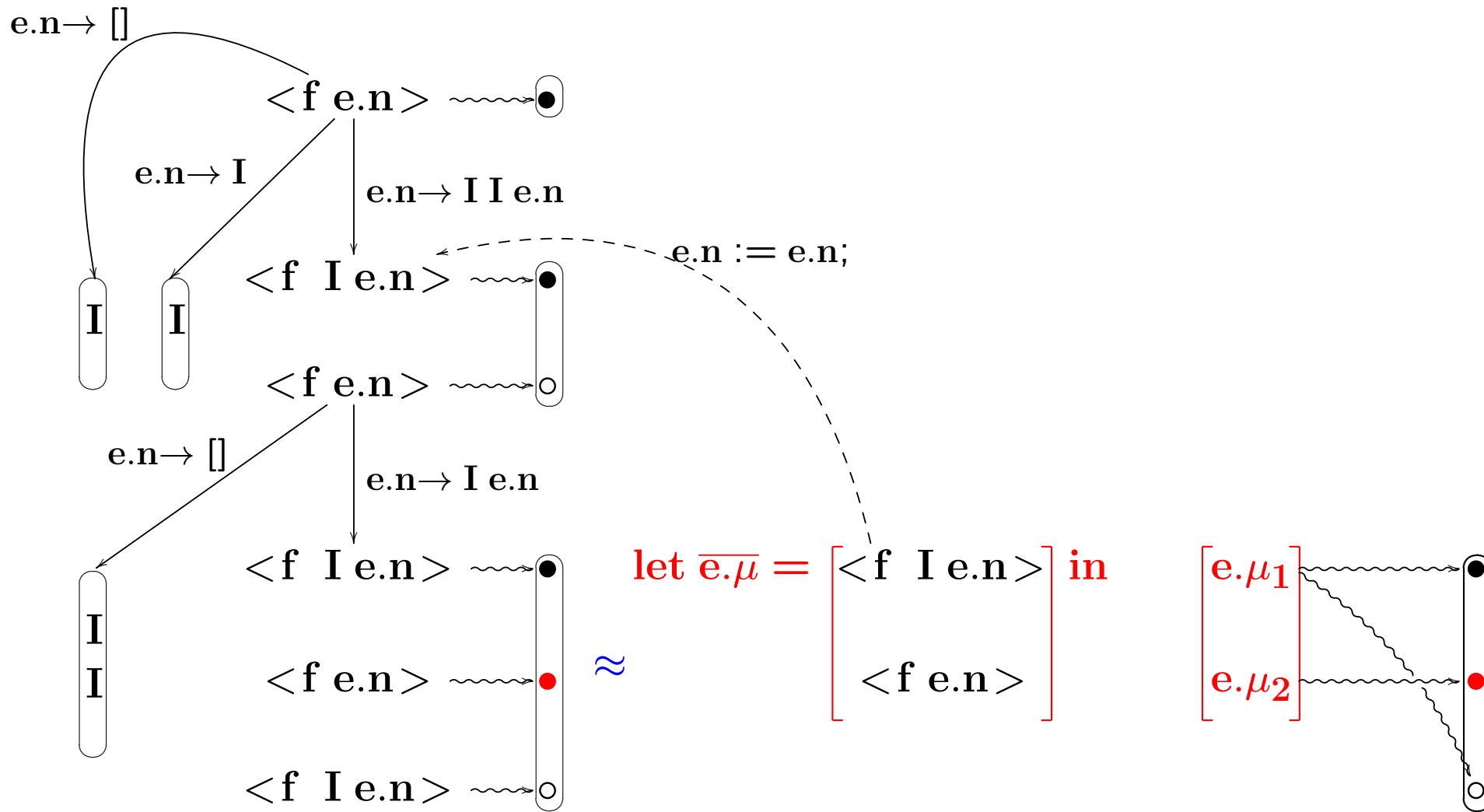
## Прогонка:



Прогонка:



Свертка:



## Входная программа:

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

## Остаточная программа:

```
F {
    = I;
    I = I;
    II = II;
    III 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 запяты);
- 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.**

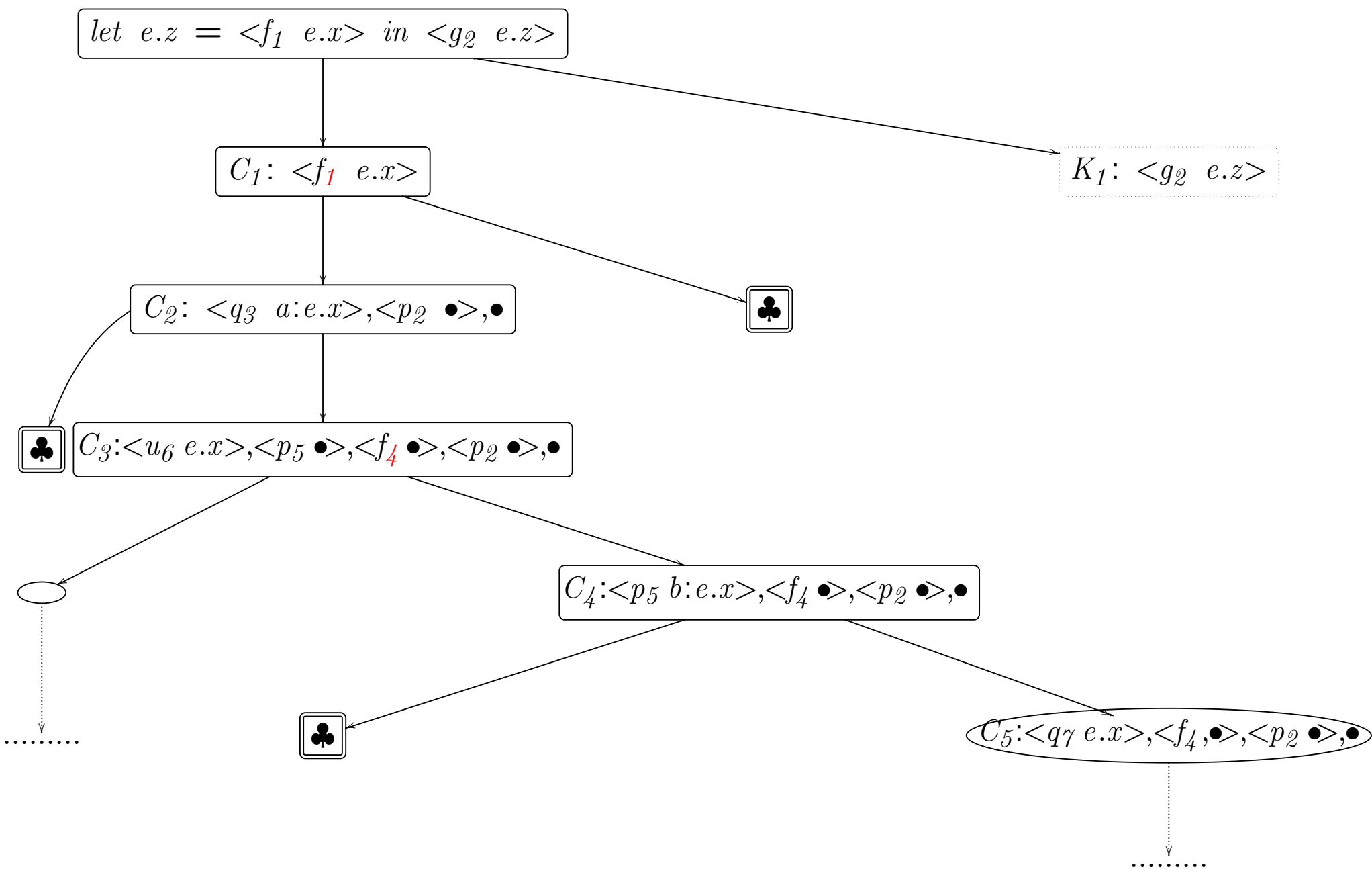
## Хорошие бинарные отношения

**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  $\beta(a_j, a_k)$ .

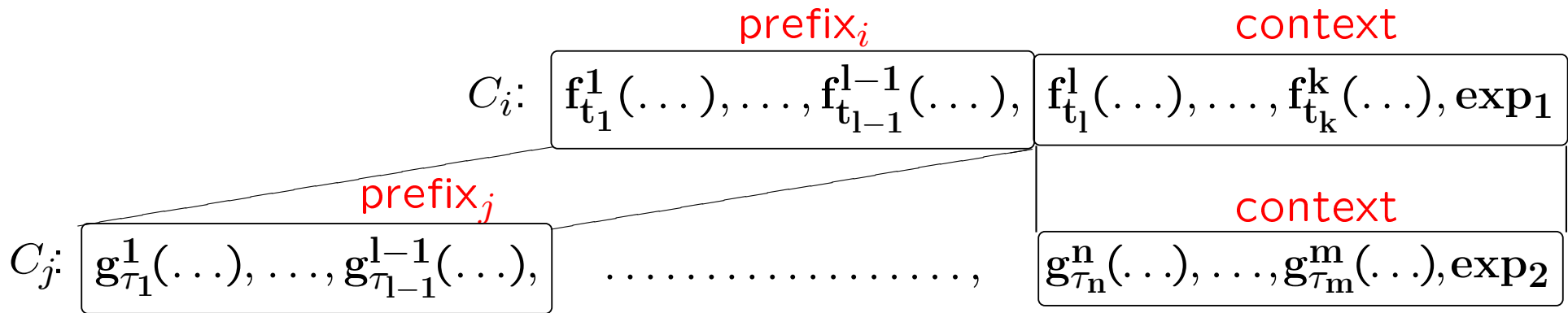
**Определение:** Бинарное отношение  $\preceq$  на множестве  $\mathfrak{M}$  называется предпорядком, если  $\forall a \in \mathfrak{M} . a \preceq a$  и  $\forall a, b, c \in \mathfrak{M} . (a \preceq b \ \& \ b \preceq c) \Rightarrow (a \preceq c)$ .

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

### Временные конфигурации



### Отношение Турчина



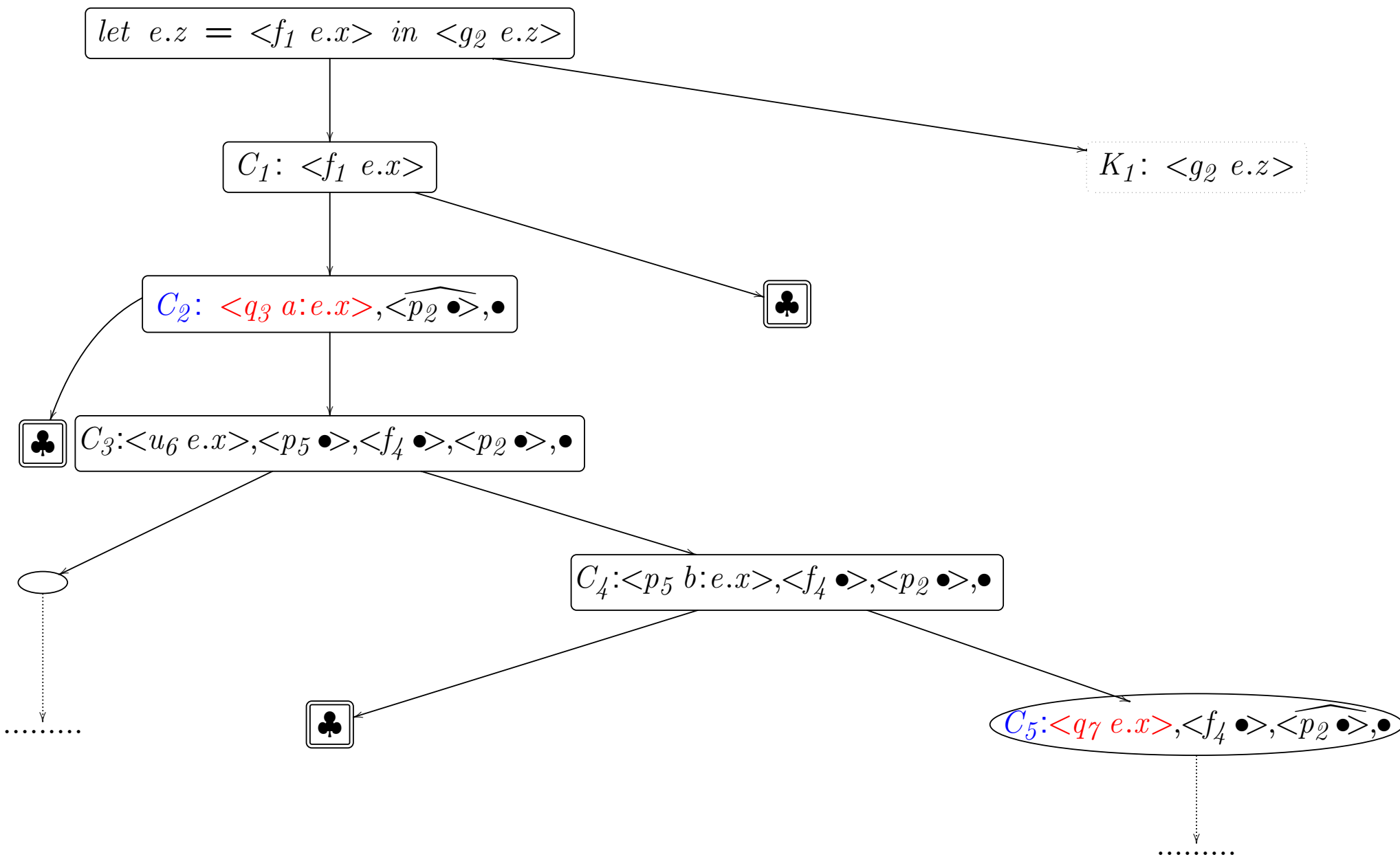
- $\forall s. (0 < s < l) f_{t_s}^s \simeq g_{\tau_s}^s$  (т.е.,  $f^s = g^s$ );
- $t_{l-1} \neq \tau_{l-1}$ ;
- $f_{t_l}^l = g_{\tau_n}^n, f_{t_{l+1}}^{l+1} = g_{\tau_{n+1}}^{n+1}, \dots, f_{t_k}^k = g_{\tau_m}^m$   
(т.е.,  $f^l = g^n, \dots$  и  $t_l = \tau_n, \dots$ ), где  $k - l = m - n$ .

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

На любом бесконечном пути  $C_1, C_2, \dots, C_n, \dots$  найдутся две временных конфигурации  $C_i, C_j$  такие, что  $i < j$  и  $C_i \triangleleft C_j$ .



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 ( $\bullet$ ) может моделироваться, как одношаговая функция.
- Рассматриваем только программы с конечным алфавитом символов-констант.
- Инструменты исследования:  
суперкомпиляторы SCP4 и MSCP-A.

Bracket { s.name<sub>0</sub> e.x = <s.name<sub>0</sub> e.x>; }  
s.name<sub>0</sub> { e.x = (s.name<sub>0</sub> e.x); }

**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

$$[\Delta(\text{expr}(p_1, \dots, p_n)), \Phi(p_1, \dots, p_n)]$$

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

Язык  $L$  называется языком рестрикций.

## Примеры языков рестрикции

- простейший язык, содержащий связки отрицания и конъюнкции ( $\#s.x \neq \#s.y$ , SCP4);
- язык уравнений в словах ( $a \#e.y = \#e.y a$ , 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:

$$\langle \text{SCP-D } [\Delta(\text{expr}(p_1, \dots, p_n)), \langle \text{SCP-D } \Phi(p_1, \dots, p_n) \rangle] \rangle$$

$$\langle \text{SCP-D } [\Delta(\text{expr}(p_1, \dots, p_n)),$$

$$\quad \langle \text{SCP-D } [\Delta(\text{expr}_1(p_{11}, \dots, p_{1n})), \langle \text{SCP-D } \Phi_1(p_{11}, \dots, p_{1n}) \rangle] \rangle$$

$$\quad ]$$

$$\rangle$$

.....

**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.**