Теория вычислительных процессов и структур

Лекция №2. Стандартные схемы программ

Содержание лекции

Программа как объект исследования

- *Программа* выступает как промежуточное звено при общении человека с машиной, связывая поставленную задачу с процессом поиска ее решения на машине.
- Понятие *задачи* ОЧЕНЬ широкое. Будем представлять задачу как функцию над множеством данных — объектов, кодирующих заданную и искомую информацию. $Y = F(X)$
- Будучи посредником между задачей и машиной, программа задает некоторую механическую процедуру решения задачи, являясь одним из способов задания алгоритма.

Программа как объект исследования

Программа (как и алгоритм) обладает следующими основными свойствами:

- Конструктивность
- Конечность
- Массовость
- Однозначность

Программа как объект исследования I

Конструктивность:

- Проявляется в том, что программа представляет собой конечный текст на некотором формализованном языке, который можно изучать и преобразовывать методами формальной лингвистики.
- Синтаксическая сторона программы: написание программы, первые фазы трансляции
- Семантическая сторона программы:
  - Программа задает последовательность действий машины.
  - Каждый шаг осуществляет преобразование данных и вызывается соответствующими синтаксическими конструкциями.
  - Каждая отдельная конструкция и их совокупности несут от дельную смысловую нагрузку
- Задачи, связанные с конструктивностью:
  - Проблема распознавания синтаксической правильности программы
  - Нахождение оценок сложности синтаксического анализа

Программа как объект исследования II

Конструктивность:

- Конструирование и исследование классов формальных языков, которые могут служить основой для разработки языков программирования с новыми синтаксическими свойствами и возможностями.

Программа как объект исследования

Схема программы —– это математическая модель программы, отвечающая следующим требованиям:

- Модель позволяет изучать свойства достаточно широких классов программ, а не отдельных конкретных программ.
- Сохраняет все интересующие исследователя свойства и особенности рассматриваемого класса программ.
- Позволяет игнорировать несущественные для данной проблемы свойства.
- Дает возможность определить разрешимые и неразрешимые свойства программ и классов программ.
- Удобство (когда структура модели “графически” подобна структуре программ).

Программа как объект исследования

- Схемы, так же как и программы, представляют собой тексты на некоторых формализованных языках, а эти языки являются упрощенными репликациями языков программирования.
- Главное отличие схем от программ состоит в следующем:
  - Арифметические, логические и другие выражения программы образуются с помощью символов операций и указателей функций, семантика которых фиксирована языком программирования или выводима по определенным правилам композиции из семантики базовых операций.
  - Поэтому каждый оператор программы единственным образом задает некоторую функцию над данными, а из этих функций образуется вычислимая функция, задаваемая программой в целом.

Программа как объект исследования

В схемах же:

- Индивидуальные операции и функции заменены абстрактными символами переменных-операций и переменных-предикатов (функциональные и предикатные символы)
- Если вместо каждого такого символа подставить конкретную функцию или, соответственно, предикат (говорят задать интерпретацию схемы), то схема превращается в программу.
- Схема не задает алгоритм в отличие от программы, и с помощью схемы нельзя вычислить что-либо.
- Схема описывает строение множества программ, получающихся из схемы в результате задания интерпретации.

Программа как объект исследования

Поясним различия на примере:

Программа на алг. языке:

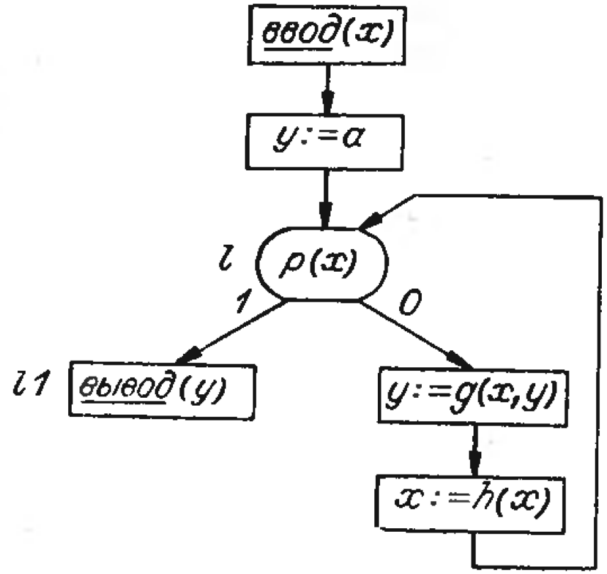
```
Начало целые x, y;
    ввод(x);
    y := 1;
L:  если x=0 то на l1;
    y := x * y;
    x := x - 1;
    на l;
L1: вывод(y);
Конец
```

Схема программы:

```
Начало
    ввод(x);
    y := a;
L:  если p(x) то на l1;
    y := g(x, y);
    x := h(x);
    на l;
L1: вывод(y);
Конец
```

Программа как объект исследования

Схему программы можно изобразить графически в виде известной нотации блок-схем.



В структуре схемы можно выделить:

- Символы-переменные (константы)
- Функциональные символы (функции)
- Предикатные символы (предикаты)
- Операторы условного и безусловного переходами

Программа как объект исследования

- Проведем следующую интерпретацию данной схемы:
  - Выберем область интерпретации: целые неотрицательные числа
  - Константу a заменим на символ 1
  - Функциональный символ g — операцией умножения чисел
  - Функциональный символ h — операцией вычитания единицы
  - Предикатный символ p — операцией сравнения с 0
- В результате получим программу вычисления факториала
- Схема программы вместе с некоторой интерпретацией образует программу

Программа как объект исследования

- Проведем следующую интерпретацию данной схемы:
  - Выберем область интерпретации: множество всех слов в некотором алфавите V
  - Константу a заменим на пустой символ ϵ
  - Функциональный символ g — операцией приписывания первый буквы первого слова ко второму слову: $CONSCAR(a\alpha, \beta) = a\beta$
  - Функциональный символ h — операцией стирания первого символа у слова: $CDR(a\alpha) = \alpha$
  - Предикатный символ p — операцией сравнения с пустым словом
- В результате получим программу выполняющую переворачивание символов у входного слова
- Интерпретируя по-разному переменные, функциональные и предикатные символы, можно получить совершенно разные программы.

Программа как объект исследования

- Но все такие программы будут устроены одинаково (одни и те же переменные, операторы, логическая структура)
- Все эти программы имеют общую схему
- Анализ схемы позволяет сделать заключения, относящиеся ко всем программам, полученным из этой схемы с помощью всех возможных интерпретаций.
- Например, схема имеет цикл и может никогда не завершится

Определение класса стандартных схем

- *Схема программы* представляет собой текст на некотором формальном языке. Этот язык можно рассматривать как абстрактный язык программирования, для которого не задана семантика данных и операций
- Из списка программных примитивов, используемых в языках программирования, для стандартных схема (и программ) отобран небольшой набор: константы, простые переменные, выражения, оператор присваивания, условный оператор, метки и переходы на метки
- *Класс стандартных схем* характеризуется *базисом* и *структурой схемы*.
- Базис фиксирует символы, из которых строятся схемы, указывает их роль (переменные, и т.п.), задает вид выражений и операторов схемы.

Определение класса стандартных схем

Базис

Базис класса стандартных схем состоит из четырех непересекающихся счетных множеств символов и множества операторов - слов, построенных из этих символов.

1. Множество символов, называемых *переменными*: $\{x, x_1, \ldots, y, y_1, \ldots, z, z_1, \ldots\}$;
2. Множество *функциональных символов*: $\{f^{(0)}, f^{(1)}, f^{(2)}, \ldots, g^{(0)}, g^{(1)}, g^{(2)}, \ldots, h^{(0)}, h^{(1)}, h^{(2)}, \ldots\}$, где верхний индекс задает *местность символа*; нульместные символы $\{f^{(0)}, g^{(0)}, h^{(0)}, \ldots\}$ будем называть *константами* и обозначать начальными буквами латинского алфавита $a, b, c, \ldots$;
3. Множество *предикатных символов*: $\{p^{(0)}, p^{(1)}, p^{(2)}, \ldots, q^{(0)}, q^{(1)}, q^{(2)}, \ldots\}$, где верхний индекс также задает местность символа; нульместные символы $\{p^{(0)}, q^{(0)}, \ldots\}$ будем называть *логическими константами*.
4. Множество специальных символов $\{\text{старт}, \text{стоп}, (,), :=\}$.

Определение класса стандартных схем

Термы

Термами (*функциональными выражениями*) называют слова, построенные из переменных, функциональных и специальных символов по следующим правилам:

1. односимвольные слова, состоящие из переменных или констант, являются термами;
2. слово τ вида $f^{(n)}(\tau_1, \tau_2, \ldots, \tau_n)$, где $\tau_1, \tau_2, \ldots, \tau_n$ — термы, является термом;
3. те и только те слова, о которых говорится в пунктах (1) и (2), являются термами.

Примеры термов: $x, f^{(0)}, a, f^{(1)}(x), g^{(2)}(x, h^{(3)}(y, f^{(1)}(x), a))$.

Определение класса стандартных схем

Тесты

- *Логическими выражениями* (или *тестами*) называют логические константы и слова вида $p^{(n)}(\tau_1, \tau_2, \ldots, \tau_n)$, где $p^{(n)}$ — предикатный символ, а $\tau_1, \tau_2, \ldots, \tau_n$ — термы.
- Примеры тестов: $p^{(0)}, p^{(1)}(x), q^{(3)}(x, y, z), p^{(1)}(f^{(2)}(x, y))$.
- Условимся в функциональных и логических выражениях опускать индексы местности, когда это не приводит к двусмысленности или противоречию.

Определение класса стандартных схем

Операторы

Множество операторов состоит из операторов четырех типов:

1. *начальный оператор* — односимвольное слово вида старт;
2. *заключительные операторы* — слова вида стоп$(x, \ldots, y)$, где $x, \ldots, y$ — переменные;
3. *операторы присваивания* (*преобразователи*) — слова вида $x : \tau$, где x — переменная, τ — терм;
4. *условные операторы* (или *распознаватели*) — логические выражения.

Среди операторов присваивания выделим специальный случай, когда терм τ — переменная, и назовем такие операторы *пересылками* (например, $x := y$), а также случай, когда τ — константа, и назовем такие операторы *засылками констант* (например, $x := a$).

Класса стандартных схем

Линейная форма

Пусть задан некоторый базис $\mathcal{B}$. Расширим множество специальных символов дополнительными символами $\{:, \text{на}, \text{если}, \text{то}, \text{иначе}\}$. *Обрамленным оператором* в базисе $\mathcal{B}$ назовем объект одного из следующих видов:

1. $k : s$, где k — число, называемое *меткой*, : — дополнительный специальный символ, s — оператор присваивания или заключительный оператор из базиса $\mathcal{B}$;
2. $k : s\text{на}l$, где k — метка, s — оператор присваивания из $\mathcal{B}$, на — дополнительный специальный символ, l — число, называемое *переходом*;
3. k : еслиsтоlиначеm, где k — метка, s — условный оператор из $\mathcal{B}$, если, то, иначе — дополнительные специальные символы, l и m — числа, называемые, соответственно, *левым* и *правым переходами*.

Класса стандартных схем

Линейная форма

Стандартной схемой в базисе $\mathcal{B}$ называется конечная последовательность $(o_0, o_1, \ldots, o_i, \ldots, o_n)$ обрамленных операторов в базисе $\mathcal{B}$, где:

1. o_0 — начальный оператор старт (считаем, что он всегда имеет неявную метку 0 и неявный переход 1);
2. o_i $(1 \leq i \leq n)$ — обрамленный оператор присваивания или условный оператор с меткой i;
3. o_n — обрамленный заключительный оператор с меткой n;
4. любой переход в схеме — положительное число, не превышающее n.

Конечное множество всех переменных, входящих в операторы схемы S, образует ее *память* X_S. Набор Z_S переменных, перечисленных в заключительном операторе схемы S, назовем набором *выходных переменных* схемы S.

Класса стандартных схем

Линейная форма

Пример стандартной схемы программы

(старт,
1 : $y := a$,
2 : если $p(x)$ то 5 иначе 3,
3 : $y := g(x, y)$,
4 : $x := h(x)$ на 2,
5 : стоп(y))

Класса стандартных схем

Графовая форма

- В практике программирования широко используется так называемое блок-схемное представление алгоритмов и программ, достоинством которого является наглядность изображения логической (или управляющей) структуры программы.
- Стандартная схема изображается как помеченный граф, вершинами которого служат метки схемы, дуги отмечают связь между переходами и метками, в качестве пометок вершин используются слова-операторы схемы.

Класса стандартных схем I

Графовая форма

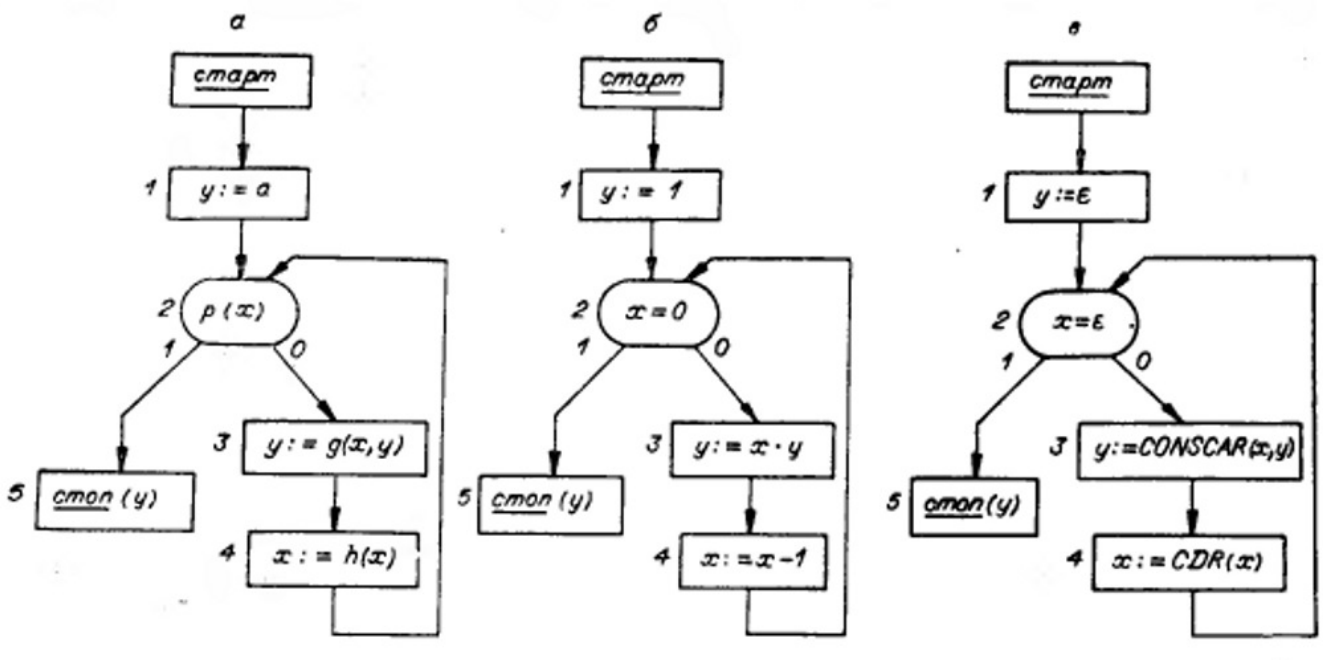
Более точно, пусть $S = (o_0, o_1, o_2, \ldots, o_n)$ — стандартная схема программы с $n+1$ обрамленным оператором. Та же схема в графовой форме — это помеченный ориентированный граф со множеством вершин $\{0, 1, 2, \ldots, n\}$. *Начальная вершина* 0 помечена начальным оператором старт, *заключительная вершина* n — заключительным оператором, вершина k $(1 \leq k \leq n)$ — словом-оператором, входящим в обрамленный оператор o_k. Из вершины k $(k \neq n)$ в вершину l $(l \neq 0)$ ведет дуга в том и только в том случае, если:

1. $l = k + 1$ и o_k — обрамленный оператор присваивания без переходов или начальный оператор;
2. o_k — обрамленный оператор присваивания с переходом l;
3. o_k — обрамленный условный оператор с левым переходом l, в этом случае дуга (k, l) помечается дополнительным символом-цифрой 1 и называется 1-дугой.
4. o_k — обрамленный условный оператор с правым переходом l, в этом случае дуга (k, l) помечается дополнительным символом-цифрой 0 и называется 0-дугой.

Класса стандартных схем

Графовая форма

Таким образом, в начальную вершину 0 графа не заходит ни одна дуга; из заключительной вершины не исходит ни одна дуга; из *вершины-преобразователя*, помеченной оператором присваивания, исходит ровно одна дуга; из *вершины-распознавателя*, помеченной условным оператором, исходят ровно две дуги, одна помечена символом 1, другая — символом 0.



Класс стандартных схем I

Понятие интерпретации и программы

Пусть задан некоторый базис $\mathcal{B}$, в котором определен класс стандартных схем. *Интерпретацией базиса $\mathcal{B}$ в области интерпретации D* называется функция I, которая сопоставляет:

1. каждой переменной x базиса — элемент $d = I(x)$ из области интерпретации D;
2. каждой константе a базиса — элемент $d = I(a)$ из D;
3. каждому функциональному символу $f^{(n)}$, где $n \geq 1$, — всюду определенную функцию $F^{(n)} = I(f^{(n)}) : D^n \to D$;
4. каждой логической константе — один из символов множества $\{0, 1\}$;
5. каждому предикатному символу $p^{(n)}$, где $n \geq 1$, — всюду определенный предикат $P^{(n)} = I(p^{(n)}) : D^n \to \{0, 1\}$.

Определение интерпретации не уточняет природы объектов множества D. Это может быть множество всех чисел, множество всех слов в некотором алфавите и т.п. Если D — конечное множество, то интерпретация называется *конечной*.

Класс стандартных схем

Понятие интерпретации и программы

Пара (S, I), где S — схема в базисе $\mathcal{B}$, а I — интерпретация этого базиса, называется *интерпретированной стандартной схемой* или (*стандартной*) *программой*.
Далее определим понятие *выполнения программы* с помощью следующих понятий:

- состояние памяти;
- значение терма;
- конфигурация;
- и протокол.

Класс стандартных схем

Состояние памяти

- *Состоянием памяти* программы (S, I) называется функция $W : X_S \to D$, которая каждой переменной x из памяти схемы S сопоставляет элемент $W(x)$ из области интерпретации D.
- Элемент $W(x)$ — *значение переменной* x;
- Запись $W \overset{x}{=} W'$ означает, что два состояния памяти W и W' совпадают с точностью до значений переменной x.
- Если $X' = (x_1, \ldots, x_n)$ — некоторый набор переменных из памяти X_S, то $W(X')$ — набор $(W(x_1), \ldots, W(x_n))$ значений переменных $x_1, \ldots, x_n$.
- *Начальным состоянием* памяти X_S при интерпретации I назовем состояние W_0 такое, что для любой переменной x из памяти $W_0(x) = I(x)$.

Класс стандартных схем

Значение терма

- *Значение терма* τ при интерпретации I и состоянии памяти W (обозначается $\tau_I(W)$) определим следующим образом:
  1. если $\tau = x$, где x — переменная, то $\tau_I(W) = W(x)$;
  2. если $\tau = a$, где a — константа, то $\tau_I(W) = I(a)$;
  3. если $\tau = f^{(n)}(\tau_1, \tau_2, \ldots, \tau_n)$, то $\tau_I(W) = I(f^{(n)})(\tau_{1I}(W), \tau_{2I}(W), \ldots, \tau_{nI}(W))$.

Класс стандартных схем

Конфигурация

- *Конфигурацией программы* (S, I) назовем пару $u = (o, W)$, где o — обрамленный оператор схемы S, W — состояние памяти схемы S.
- Выполнение программы описывается конечной или бесконечной последовательностью конфигураций, которую назовем *протоколом выполнения программы*.

Класс стандартных схем I

Протокол

Протокол $(u_0, u_1, \ldots, u_i, u_{i+1}, \ldots)$ выполнения программы (S, I) определим следующим образом (пусть k_i — метка обрамленного оператора в i-й конфигурации протокола):

1. $u_0 = (o_0, W_0)$, где o_0 — начальный оператор схемы S, W_0 — начальное состояние памяти схемы S при интерпретации I.
2. Пусть $u_i = (o_{k_i}, W_i)$ — i-я конфигурация в протоколе. Если o_{k_i} — заключительный оператор в схеме S, то u_i — последняя конфигурация протокола (и протокол конечен). В противном случае в протоколе имеется $(i+1)$-я конфигурация $u_{i+1} = (o_{k_{i+1}}, W_{i+1})$, в которой $o_{k_{i+1}}$ — обрамленный оператор схемы S с меткой k_{i+1}, равна:
   - 2.1 1, если $i = 0$;
   - 2.2 $k_i + 1$, если $i > 0$ и оператор o_{k_i} — преобразователь без перехода;
   - 2.3 l, если $i > 0$ и оператор o_{k_i} — преобразователь с переходом l;
   - 2.4 l, если o_{k_i} — распознаватель вида k_i : если $p(\tau_1, \ldots, \tau_n)$ то l иначе m и $I(p)(\tau_{1I}(W_{i-1}), \ldots, \tau_{nI}(W_{i-1})) = 1$;

Класс стандартных схем II

Протокол

2.5 m, если o_{k_i} — распознаватель вида k_i : если $p(\tau_1, \ldots, \tau_n)$ то l иначе m и $I(p)(\tau_{1l}(W_{i-1}), \ldots, \tau_{nl}(W_{i-1})) = 0$;

Что же касается второй компоненты конфигурации u_{i+1} состояния W_{i+1}, то

2.1 $W_{i+1} = W_i$, если $o_{k_{i+1}}$ — распознаватель;

2.2 $W_{i+1} \stackrel{x}{=} W_i$ и $W_{i+1}(x) = \tau_l(W_i)$, если $o_{k_{i+1}}$ — преобразователь вида k : $x := \tau$ на l (переход может отсутствовать).

Понятие интерпретации и программы

- Из определения стандартной схемы непосредственно следует, что протокол выполнения любой стандартной программы конечен тогда и только тогда, когда он содержит конфигурацию с заключительным оператором, причем эта конфигурация обрывает протокол.
- Если протокол программы (S, I) конечен, то говорят, что *программа останавливается*, в противном случае — *зацикливается*.
- Пусть схема S имеет m выходных переменных $Z_S = (x_i, \ldots, x_m)$. Если программа (S, I) останавливается и W — состояние памяти в последней конфигурации протокола выполнения этой программы, то $W(Z_S) \in D^m$ — *результат* $val(S, I)$ выполнения программы. Если же программа зацикливается, то ее результат не определен.

Примеры программ

Рассмотрим две программы (S, I_1) и (S, I_2), где S — схема, рассмотренная на предыдущих слайдах.

$$S = \langle \{x, y\}, \{a, g, h\}, \{p\}, \{\text{старт}, \text{стоп}, (,), :=\} \rangle$$

I_1 задается следующим образом:

- Область интерпретации D_1 — множество всех целых неотрицательных чисел;
- $I_1(x) = 4$; $I_1(y) = 0$; $I_1(a) = 1$;
- $I_1(g) = G$, где $G : D_1^2 \to D_1$ — функция умножения чисел $G(d_1, d_2) = d_1 \times d_2$;
- $I_1(h) = H$, где $H : D_1 \to D_1$ — функция вычитания единицы $H(d) = d - 1$;
- $I_1(p) = P_1$, где $P_1 : D_1 \to \{0, 1\}$ — предикат “равно 0”, т.е.

$$P_1(d) = \begin{cases} 1, & \text{если} d = 0, \\ 0, & \text{если} d \neq 0. \end{cases}$$

- Программ (S, I_1) вычисляет 4!.

Примеры программ

I_2 задается следующим образом:

- Область интерпретации $D_2 = V^*$, где $V = \{a, b, c\}$;
- $I_2(x) = abc$; $I_2(y) = \epsilon$; $I_2(a) = \epsilon$;
- $I_2(g) = CONSCAR$, где $CONSCAR : D_2^2 \to D_2$ — приписывает первую букву первого слова к началу второго слова, т.е. $CONSCAR(d\delta, \gamma) = d\gamma$, где $d \in V$, $\delta, \gamma \in V^*$;
- $I_2(h) = CDR$, где $CDR : D_2 \to D_2$ — функция, стирающая первую букву слова, т.е. $CDR(d\delta) = \delta$, где $d \in V$, $\delta \in V^*$
- $I_2(p) = P_2$, где $P_2 : D_2 \to \{0, 1\}$ — предикат “равно ϵ”, т.е.

$$P_2(\alpha) = \begin{cases} 1, & \text{если } \alpha = \epsilon, \\ 0, & \text{если } \alpha \neq \epsilon. \end{cases}$$

- Программ (S, I_2) переворачивает исходное слово.

Примеры программ

Протокол программы (S, I_1):

Конфигурация	u_0	u_1	u_2	u_3	u_4	...	u_{14}	u_{15}
Метка операторов	0	1	2	3	4	...	2	5
Значения:								
x	4	4	4	4	3	...	0	0
y	0	1	1	4	4	...	24	24

Протокол программы (S, I_2):

Конф-ция	u_0	u_1	u_2	u_3	u_4	...	u_{11}	u_{12}
Метка оп-ов	0	1	2	3	4	...	2	5
Значения:								
x	abc	abc	abc	abc	bc	...	ϵ	ϵ
y	ϵ	ϵ	ϵ	a	a	...	cba	cba