Схемы программ

- Цель программирования - описание процессов обработки данных .
- Данные, информация, носители данных, информационная среда
- Описать процесс - определить последовательность состояний заданной информационной среды.
- Программа?

Программы и схемы программ

- Схемы программ - это математические модели программ, описывающие строение программы, то есть строение множества программ, где конкретные операции и функции заменены абстрактными функциональными и предикатными символами.

Стандартные схемы программ (ССП)

Полный *базис В класса стандартных схем* состоит из 4-х непересекающихся, счетных множеств символов и множества операторов.

Множества символов полного базиса:

- $X = \{x, x_1, x_2..., y, y_1\ y_2..., z, z_1, z_2...\}$ - *переменные*;
- $F = \{f^{(0)}, f^{(1)}, f^{(2)}..., g^{(0)}, g^{(1)}, g^{(2)}..., h^{(0)}, h^{(1)}, h^{(2)}...\}$ - множество *функциональных символов*;
- $P = \{p^{(0)}, p^{(1)}, p^{(2)}...; q^{(0)}, q^{(1)}, q^{(2)}...; \}$ - множество *предикатных символов*;
- {**start, stop, ...,:=** и т. д.} - множество специальных символов.

Стандартные схемы программ (ССП)

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

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

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

- *Тестами (логическими выражениями)* называются логические константы и слова вида $p^{(n)}(\tau_1, \tau_2, ..., \tau_n)$.

Примеры: $p^{(0)}$, $p^{(0)}(x)$, $g^{(3)}(x, y, z)$, $p^{(2)}(f^{(2}(x, y))$.

Стандартные схемы программ (ССП)

- Множество операторов включает пять типов:

1. *начальный оператор* - слово вида **start**($x_1, x_2...x_k$), где k ≥0, а $x_1, x_2...x_k$ - переменные, называемые результатом этого оператора;
2. *заключительный оператор* - слово вида **stop**($\tau_1, \tau_2...\tau_n$), где n ≥0, а $\tau_1, \tau_2...\tau_n$ - термы; вхождения переменных в термы τ называются *аргументами* этого *оператора*;
3. *оператор присваивания* - слово вида x := τ, где x – переменная (*результат оператора*), а τ - терм; вхождения переменных в термы называются *аргументами* этого *оператора*;
4. *условный оператор* (тест) - логическое выражение; вхождения переменных в логическое выражение называются *аргументами* этого *оператора*;
5. *оператор петли* - односимвольное слово **loop**.

Графовая форма (ССП)

- *Стандартной схемой* в базисе B называется конечный (размеченный ориентированный) граф без свободных дуг и с вершинами следующих пяти видов:

1. *Начальная вершина* (ровно одна) помечена начальным оператором. Из нее выходит ровно одна дуга. Нет дуг, ведущих к начальной вершине.
2. *Заключительная вершина* (может быть несколько). Помечена заключительным оператором. Из нее не выходит ни одной дуги.
3. *Вершина-преобразователь.* Помечена оператором присваивания. Из нее выходит ровно одна дуга.
4. *Вершина-распознаватель.* Помечена условным оператором (называемым условием данной вершины). Из нее выходит ровно две дуги, помеченные 1 (левая) и 0 (правая).
5. *Вершина-петля.* Помечена оператором петли. Из нее не выходит ни одной дуги.

Линейная форма (ССП)

- СПП в линейной форме представляет собой последовательность *инструкций*, которая строится следующим образом:
- если выходная дуга начальной вершины ведет к вершине L, то начальной вершине соответствует инструкция:

$$0: \textbf{start}(x_1,..., x_n)\ \textbf{goto}\ L;$$

- если вершина L - преобразователь (x:=τ) и выходная дуга ведет к L1, то

$$L: x: =\tau\ \textbf{goto}\ L1;$$

- если вершина L – заключительная вершина, то

$$L: \textbf{stop}(\tau_1,..., \tau_m);$$

- если вершина с меткой L - распознаватель, причем 1-дуга ведет к вершине L1, а 0-дуга - к вершине L0, то:

$$L: \textbf{if}\ \ p(\tau_1,...\tau_k)\ \textbf{then}\ \ L1\ \textbf{else}\ L0;$$

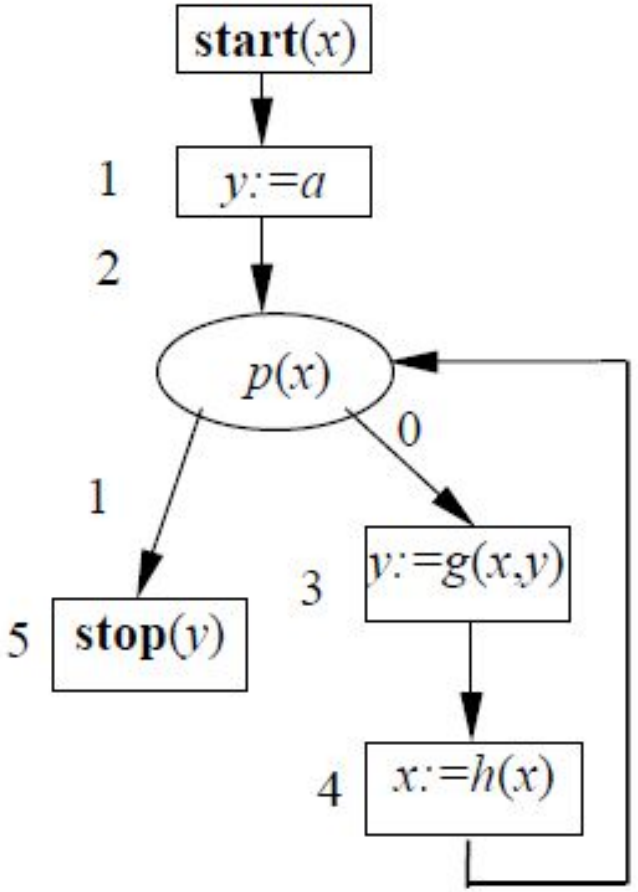
- если вершина с меткой L - петля, то ей соответствует инструкция

$$L: \textbf{loop}.$$

Пример

Вычисление $n!$

0: **start**(x) **goto** 1,
1: y: = a **goto** 2,
2: **if** p(x) **then** 5 **else** 3,
3: y: = g(x,y) **goto** 4,
4: x: = h(x) **goto** 2,
5: **stop**(y).



Интерпретация стандартных схем программ

Пусть в некотором базисе B определен класс ССП. *Интерпретацией базиса* B *в области интерпретации* D называется функция I, которая сопоставляет:

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

Пара (S,I) называется *интерпретированной стандартной схемой* (ИСС), или *стандартной программой* (СП).

Выполнение программы

Состоянием памяти программы *(S,I)* называют функцию $W: X_S \rightarrow D$, которая каждой переменной x из памяти схемы *S* сопоставляет элемент *W(x)* из области интерпретации *D*.

Значение терма т при *I* и состоянии памяти W ($\tau_I(W)$) определяется :

1. если $\tau = x$, *x* – переменная, то $\tau_I(W) = W(x)$;
2. если $\tau = a$, *a* – константа, то $\tau_I(W) = I(a)$;
3. если $\tau = f(n)(\tau_1, \tau_2 ..., \tau_n)$, то
$$\tau_I(W) = I(f(n))(\tau_{1I}(W), \tau_{2I}(W), ..., \tau_{nI}(W)).$$

Значение теста π при интерпретации *I* и состоянии памяти W или $\pi_I(W)$:

если $\pi = \pi^{(n)}(\tau_1, \tau_2 ..., \tau_n)$,

то $p_I(W) = I(\pi^{(n)})(\tau_{1I}(W), \tau_{2I}(W), ... \tau_{nI}(W))$, $n \geq 0$.

Конфигурация программы

- Конфигурация программы - пара $U=(L,W)$, где L - метка вершины схемы S, а W - состояние ее памяти.
- Выполнение программы описывается конечной или бесконечной последовательностей конфигураций, которую называют протоколом выполнения программы (ПВП).

Протокол выполнения программы

Протокол $(U_0, U_1, ..., U_i, U_{i+1}, ...)$ выполнения программы (S,I) определяем следующим образом ($U_i=(k_i, W_i)$):

$U_0=(0, W_0)$, W_0 – начальное состояние памяти схемы S при интерпретации I.

Пусть $U_i=(k_i, W_i)$ - i-я конфигурация ПВП, а O - оператор схемы S в вершине с меткой k_i.

Если O - **stop**$(\tau_1, \tau_2 ... \tau_n)$, то U_i - последняя конфигурация. В этом случае считают, что, программа (S,I) *останавливается*, а последовательность значений $\tau_{1I}(W)$, $\tau_{2I}(W)$,..., $\tau_{nI}(W)$ объявляют *результатом* val(S,I) *выполнения программы* (S,I).

Протокол выполнения программы

Если *O* - не заключительный оператор, в протоколе имеется следующая, (i+1)-я конфигурация $U_{i+1} = (k_{i+1}, W_{i+1})$, причем

a) если *O* - начальный оператор, а выходящая из него дуга ведет к вершине с меткой L, то $k_{i+1} = L$ и $W_{i+1} = W_i$;

b) если *O* - оператор присваивания x:=τ, а выходящая из него дуга ведет к вершине с меткой L, то $k_{i+1} = L, W_{i+1} = W_i, W_{i+1}(x) = \tau_1(W_i)$;

c) если *O* - условный оператор *p* и $p_1(W_i) = \Delta$, где $\Delta \in \{0,1\}$, а выходящая из него дуга ведет к вершине с меткой *L*, то $k_{i+1} = L$ *и* $W_{i+1} = W_i$;

d) если *O* - оператор петли, то $k_{i+1} = L$ и $W_{i+1} = W_i$, так что протокол бесконечен.

Пример

Программа *(S,I)* вычисляет *4!*

Интерпретация *(S, I)* задана так:

1. область интерпретации D_I - подмножество множества Nat целых неотрицательных чисел;
2. *I(x)=4; I(y)=0; I(a)=1;*
3. *I(g)=G*, где *G* - функция умножения чисел, т. е. *G(d1,d2)= d1*d2*;
4. *I(h)=H*, где *H* - функция вычитания единицы, т. е. *H(d)= d-1*;
5. *I(p)=P*, где *P* - предикат «равно 0», т.е. *P(d)=1*, если *d=0*.

Пример

Программа *(S,I)* вычисляет *4!*

Конфигурация		U_0	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}	U_{13}
Метка		0	1	2	3	4	2	3	4	2	3	4	2	3	
Значения	x	4	4	4	4	3	3	3	2	2	2	1	1	1	0
	y	0	1	1	4	4	4	12	12	12	24	24	24	24	24

Свойства и виды ССП

- ССП S в базисе B *тотальна (пуста),* если для любой интерпретации I базиса B программа *(S, I)* останавливается (зацикливается).

- Стандартные схемы S_1 и S_2 в базисе B *функционально эквивалентны* ($S_1 \sim S_2$), если либо обе зацикливаются, либо обе останавливаются с одинаковым результатом, т. е. val (S_1, I) » val (S_2, I).

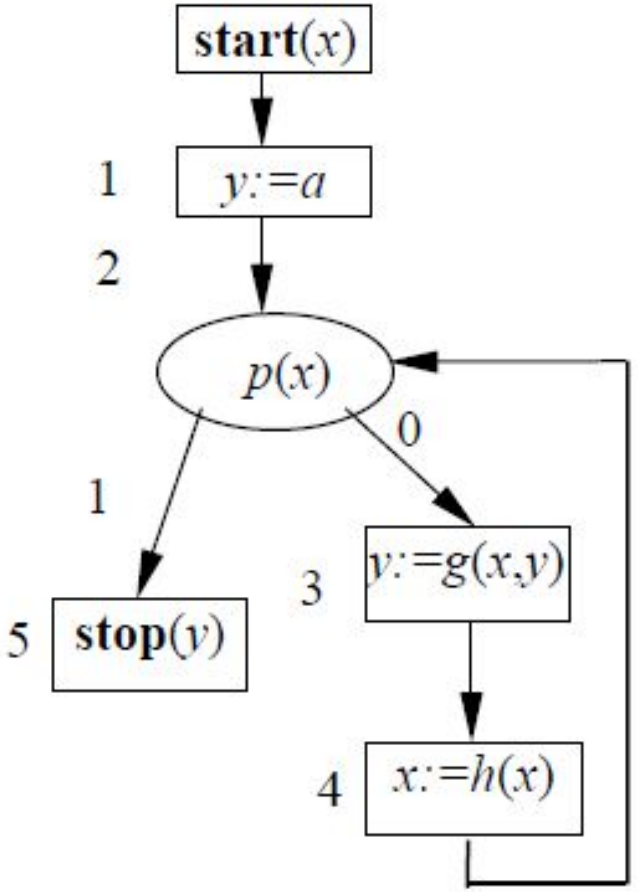
Свойства и виды ССП

Цепочкой стандартной схемы (ЦСС) называют:

1. конечный путь по вершинам
   начальной вершины к заключ
2. бесконечный путь по вершин
   начальной вершиной схемы

В случае, когда вершина-р
дополнительно указывается вер
определяющий 1-дугу или 0
вершины.

Примеры: (0, 1, 2^1, 5); (0, 1, 2^0, 3, 4,

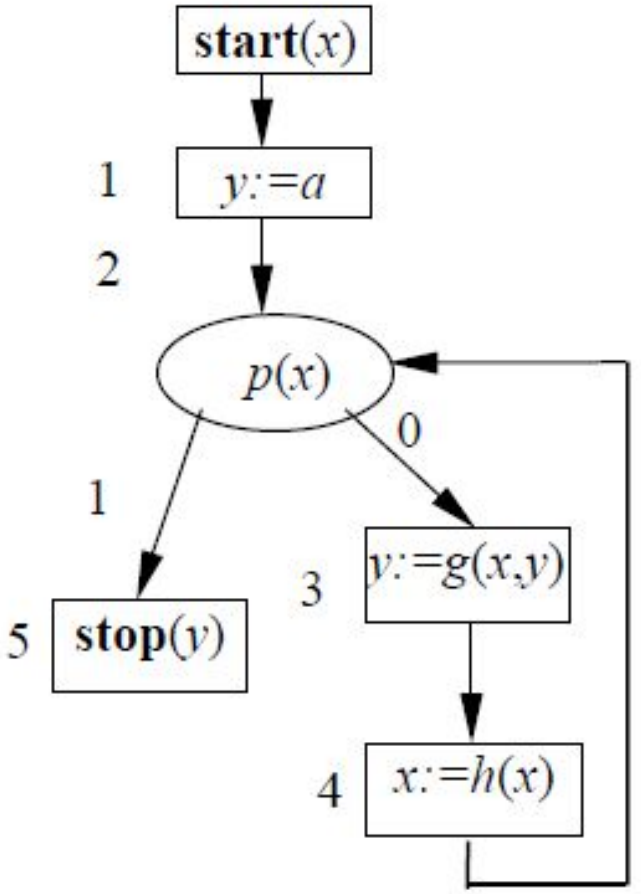


Свойства и виды ССП

Цепочкой операторов (ЦО) назы
последовательность операторов
некоторой цепочки схемы.

Пример: (**start**(x), y:=a, $p^1(x)$, **stop**
$p^0(x)$, y:=g(x, y), x:=h(x), $p^0(x)$, y:=g(
y:=g(x, y), x:=h(x), …)) и т. д.

Предикатные символы ЦО обоз
вершины распознавателей в ЦС



Свойства и виды ССП

- ЦСС в базисе B называют *допустимой*, если она подтверждается хотя бы одной интерпретацией этого базиса.

- ССП *свободна*, если все ее цепочки допустимы.

- *Допустимая цепочка операторов* - это цепочка операторов, соответствующая допустимой цепочке схемы. В тотальной схеме все допустимые цепочки (и допустимые цепочки операторов) конечны. В пустой схеме - бесконечны.

Свободные интерпретации (СИ)

Все СИ базиса B имеют одну и ту же область интерпретации, которая совпадает со множеством *T* всех термов базиса B. Все СИ одинаково интерпретируют переменные и функциональные символы, а именно:

1. для любой переменной x из базиса B и для любой СИ I_h этого базиса $I_h(x)=x$;
2. для любой константы a из базиса B $I_h(a)=a$;
3. для любого функционального символа $f^{(n)}$ из базиса B, где $n^3 1$, $I_h(f^{(n)})=F^{(n)}: T^n \to T$, где $F^{(n)}$ - словарная функция такая, что

$$F^{(n)}(t_1, t_2, ..., t_n)= f^{(n)} (t_1, t_2, ... t_n),$$

т. е. функция $F^{(n)}$ по термам $t_1, t_2, ...t_n$ из T строит новый терм, используя функциональный смысл символа $f^{(n)}$.

Интерпретации предикатных символов - полностью *свободна,* т.е. разные СИ различаются лишь интерпретаций предикатных символов.

Пример

Пусть I_h -СИ базиса, схема S, $P=I_h(p)$ задан так: P(t) = 1, если число функциональных символов в t больше двух; P(t) = 0, в противном случае.

Конфигурация	Метка	Значения	
		X	y
U_0	0	`x`	`y`
U_1	1	`x`	`a`
U_2	2	`x`	`a`
U_3	3	`x`	`g(x,a)`
U_4	4	`h(x)`	`g(x,a)`
U_5	2	`h(x)`	`g(x,a)`
U_6	3	`h(x)`	`g(h(x),g(x,a))`
U_7	4	`h(h(x))`	`g(h(x),g(x,a))`
U_8	2	`h(h(x))`	`g(h(x),g(x,a))`
U_9	3	`h(h(x))`	`g(h(h(x)),g(h(x),g(x,a)))`
U_{10}	4	`h(h(h(x)))`	`g(h(h(x)),g(h(x),g(x,a)))`
U_{11}	2	`h(h(h(x)))`	`g(h(h(x)),g(h(x),g(x,a)))`
U_{12}	5	`h(h(h(x)))`	`g(h(h(x)),g(h(x),g(x,a)))`

Пример

- $g^{(2)}(h^{(1)}(x), g^{(2)}(x,y))$ - бесскобочный терм ghxgxy.

Правила восстановления терма по бесскобочной записи аналогичны правилам восстановления арифметических по их прямой польской записи.

Примеры A*B => AB* A*B+C =>AB*C+ A*(B+C/D) =>ABCD/+* A*B+C*D =>AB*CD*+

Правила представления в польской записи:

1. Идентификаторы следуют в том же порядке, что и в инфиксной записи
2. Операторы следуют в том же порядке, в каком они должны вычисляться (слева направо)
3. Операторы располагаются непосредственно за своими операндами.

Согласованные свободные интерпретации

Интерпретация I и СИ I_h (того же базиса B) согласованы, если для любого логического выражения p справедливо $I_h(p)=I(p)$.

Если интерпретация I и свободная интерпретация I_h согласованы, то программы (S, I) и (S, I_h) либо зацикливаются, либо обе останавливаются и $I(val(S,I_h))=val(S,I)$, т.е. каждой конкретной программе можно поставить во взаимно-однозначное соответствие свободно интерпретированную (стандартную) согласованную программу

Согласованные свободные интерпретации

Теорема Лакхэма – Парка – Паттерсона. Стандартные схемы S_1 и S_2 в базисе B функционально эквивалентны тогда и только тогда, когда они функционально эквивалентны на множестве всех свободных интерпретаций базиса B, т.е. когда для любой свободной интерпретации I_h программы (S_1,I_h) и (S_2,I_h) либо обе зацикливаются, либо обе останавливаются и

$$val(S_1,I) = \{I(val(S_1\ I_h)) = I(val(S_2\ I_h))\} = val(S_2,I).$$

Согласованные свободные интерпретации

- Стандартная схема S в базисе B пуста (тотальна, свободна) тогда и только тогда, когда она пуста (тотальна, свободна) на множестве всех свободных интерпретаций этого базиса, т.е. если для любой свободной интерпретации I_h программа (S, I_h) зацикливается (останавливается).

- Стандартная схема S в базисе B свободна тогда и только тогда, когда она свободна на множестве всех свободных интерпретаций этого базиса, т. е. когда каждая цепочка схемы подтверждается хотя бы одной свободной интерпретацией.

Логико-термальная эквивалентность

- Отношение эквивалентности E, заданное на парах стандартных схем, называют *корректным,* если для любой пары схем S_1 и S_2 из S_1 ~E~ S_2 следует, что S_1~ S_2, т. е. S_1 и S_2 функционально эквивалентны.

Моделирование ССП

Автоматы

- Одноленточные автоматы
- Многоленточные автоматы
- Двухголовочные автоматы

Одноленточный автомат (ОКА)

- ОКА задается набором A = { V, Q, R, q_0, #, I } и правилом функционирования.

  V - алфавит;

  Q - *множество состояний* (Q∩V=∅);

  R - множество *заключительных состояний* (R∈Q);

  q_0 - выделенное *начальное* состояние;

  I - *программа автомата*;

  # - «пустой» символ.

- Программа автомата I представляет собой множество *команд* вида $qa \rightarrow q'$, в которой $q, q' \in Q$, a *V* и для любой пары (q, a) существует единственная команда, начинающаяся этими символами.

Одноленточный автомат (ОКА)

Особенности одноленточного автомата:

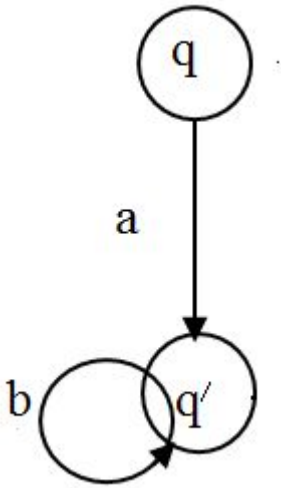
- выделены заключительные состояния;
- машина считывает символы с ленты, ничего не записывая на нее;
- на каждом шаге головка автомата, считав символ с ленты и перейдя согласно программе в новое состояние, обязательно передвигается вправо на одну клетку;
- автомат останавливается в том и только в том случае, когда головка достигнет конца слова, т.е. символа #.

Одноленточный автомат (ОКА)

- Автомат допускает слово *a* в алфавите V, если, начав работать с лентой, содержащей это слово, он останавливается в заключительном состоянии.

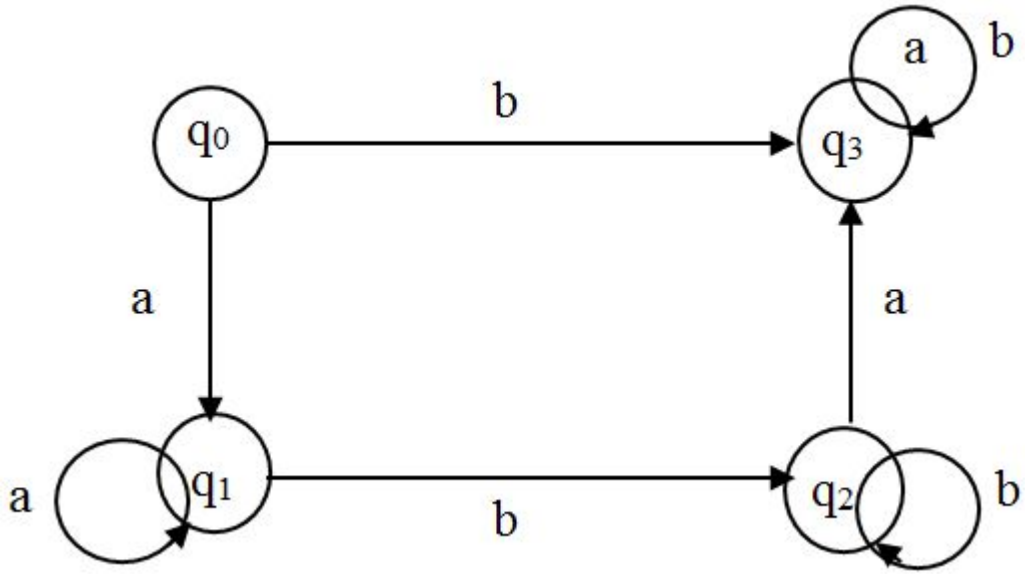
- Автомат A задает характеристическую функцию множества M_A *допускаемых им слов* в алфавите V, т. е. он распознает, принадлежит ли заданное слово множеству M_A если связать с остановкой в заключительном состоянии символ 1, а с остановкой в незаключительном состоянии – 0.

Одноленточный автомат (ОКА)



- Множество вершин – множество состояний Q;
- Из вершины q в вершину q’ ведет дуга, помеченная символом *a*, тогда и только тогда, когда программа автомата содержит команду $qa \rightarrow q'$.
- Работе автомата над заданным словом соответствует путь из начальной вершины q_0.
- Последовательность проходимых вершин этого пути – это последовательность принимаемых автоматом состояний, образ пути по дугам – читаемое слово.
- Любой путь в графе автомата, начинающийся в вершине q_0 и заканчивающийся в вершине $q' \in R$, порождает слово, допустимое автоматом.

Приме



ОКА A = ({a, b}, {q_0, q_1, q_2, q_3}, {q_2}, q_0, #, I), допускающего слова {a^n b^m | n=1,2, ...; m=1,2, ...}, задается графом. Программа I содержит команды:

$q_0a \to q_1$; $q_0b \to q_3$; $q_1a \to q_1$; $q_1b \to q_2$; $q_2a \to q_3$; $q_2b \to q_2$;
$q_3a \to q_3$; $q_3b \to q_3$.

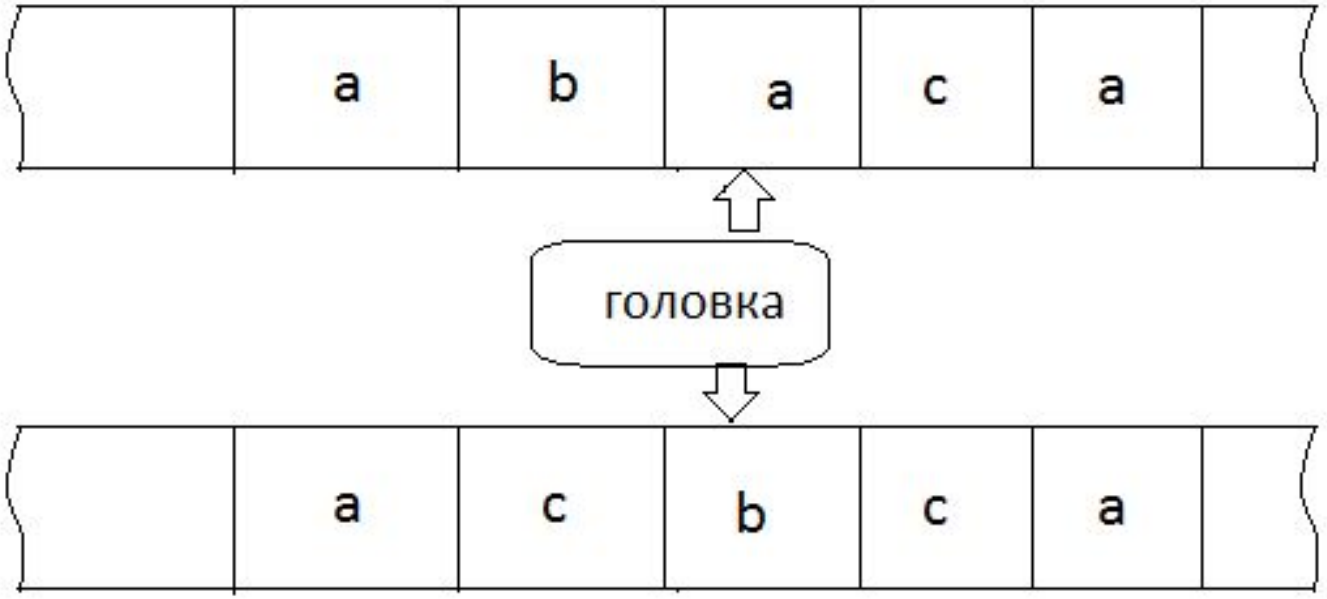
Одноленточный автомат (ОКА)

- Автомат называется ***пустым***, если $M_A = \varnothing$.
- Автоматы A1 и A2 ***эквивалентны***, если $M_{A1} = M_{A2}$.

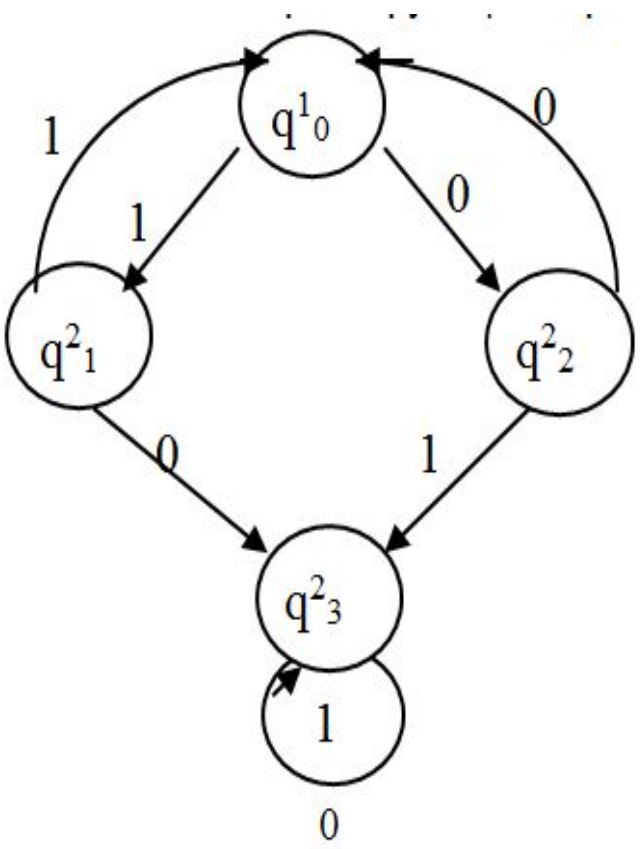
- Для ОКА доказано:
  - Проблема пустоты ОКА разрешима.
  - Предположение о том, что минимальная длина допускаемого слова больше n отвергается на том основании, что оно может быть сведено к слову меньшей длины, путем выбрасывания участков между двумя повторяющимися в пути узлами.
  - Проблема эквивалентности ОКА разрешима.

Многоленточные автоматы (МКА)

- МКА задается $A = \{ V, Q, R, q_0, \#, I \}$, где множество состояний Q разбивается на $n \geq 2$ непересекающихся подмножеств $Q_1, ..., Q_n$.



Пример

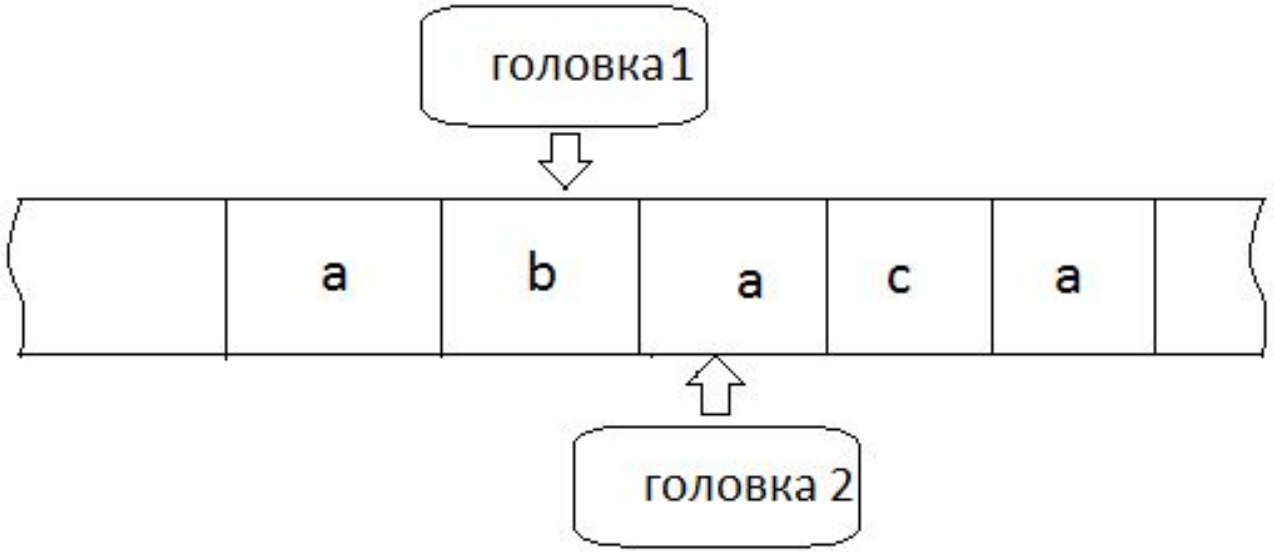


$Q=Q_1 \cup Q_2$, $Q_1=\{q_0^1\}$; $Q_2=\{q_1^2, q_2^2, q_3^2\}$; $R=\{q_0^1\}$; $V=\{0, 1\}$, начальное состояние - q_0^1.

МКА обрабатывает (U_1, U_2), где слово U_1 записано на первой ленте, а U_2 - на второй.

Допустимое множество наборов M_A -это все возможные пары одинаковых слов, т.е. наборы, где $U_1 = U_2$. Например, набор может быть (1101, 1101) и т.п.

Двухголовочные автоматы



Множество состояний Q разбито на два непересекающихся множества. В состояниях Q_1 активна первая головка, а в состояниях Q_2 - вторая.

Рекурсивные схемы программ

Вычисление факториала

Рекурсивно определяемая функция

FACT(x) = 1,если x = 0,
FACT(x) = x *FACT(x — 1), если x > 0.

Программа

FACT(a),

FACT(x) = **if** x = 0 **then** 1 **else** x ´ FACT(x - 1),

где a — некоторое целое неотрицательное число.

Рекурсивная схема

- *Полный базис РС* включает четыре счетных множества символов: *переменные, функциональные символы, предикатные символы, специальные символы.*
- Множество специальных символов : {**if, то, else**, **(**, **)**, **,**}.
- Отличие множества функциональных символов разбито на два непересекающиеся подмножества*: множество базовых функциональных символов и множество определяемых функциональных символов* (обозначаются прописными буквами, например, $F^{(1)}$,$G^{(2)}$, и т.д.).
- В базисе РС нет множества операторов, вместо него – множество логических выражений и множество термов.

Термы в РС

- *Простые термы*
- *Базовые термы* - не содержат определяемых функциональных символов
- *Вызовы-термы* вида $F^{(n)}(t_1,t_2,\ldots t_n)$, где $t_1,t_2,\ldots t_n$ - простые термы, $F^{(n)}$ - определяемый функциональный символ.
- *Логическое выражение* - слово вида $p^{(n)}(t_1,t_2,\ldots t_n)$,
- *Терм* - это *простой терм*, или *условный терм*, т.е. слово вида

if p **then** t_1 **else** t_2,

где p - логическое выражение, t_1, t_2 - простые термы, называемые *левой* и соответственно *правой альтернативой.*

- Примеры термов:
  - f(x, g(x, y)); h(h(a)) - базовые термы;
  - f(F(x), g(x, F(y))); H(H(a)) - простые термы;
  - F(x); H(H(a)) - вызовы;
  - **If** p(x, y) **then** h(h(a)) **else** F(x) - условный терм.

Рекурсивное уравнение

Рекурсивным уравнением, или *определением функции* F назовем слово вида $F^{(n)}(x_1,x_2,\ldots x_n) = t(x_1,x_2,\ldots x_n)$,
где $t(x_1,x_2,\ldots x_n)$ - терм, содержащий переменные, называемые *формальными параметрами* функции F.

Рекурсивной схемой называется пара (t, M), где t - терм, называемый *главным термом схемы* (или ее *входом*). M - такое множество рекурсивных уравнений, что все определяемые функциональные символы в левых частях уравнений различны и всякий определяемый символ, встречающийся в правой части некоторого уравнения или в главном терме схемы, входит в левую часть некоторого уравнения.

Рекурсивная схема

- Пример РС:

  R_{S1}: F(x); F(x) = **if** p(x) **then** a **else** g(x, F(h(x))).

  R_{S2}: A(b, c); A(x, y) = **if** p(x) **then** f(x) **else** B(x, y);

  B(x, y) = **if** p(y) **then** A(g(x), a) **else** C(x, y);

  C(x, y) = A(g(x), A(x, g(y))).

  R_{S3}: F(x); F(x) = **if** p(x) **then** x **else** f(F(g(x)), F(h(x))).

- Пара (R_S, I), где R_S - РС в базисе B, а I - интерпретация этого базиса, называется *рекурсивной программой.* При этом заметим, что определяемые функциональные символы не интерпретируются.

Пример

Программа *(S,I)* вычисляет *4!*

Интерпретация *(S, I)* задана так:

1. область интерпретации D_I - подмножество множества Nat целых неотрицательных чисел;
2. *I(x)=4; I(y)=0; I(a)=1;*
3. *I(g)=G*, где *G* - функция умножения чисел, т. е. *G(d1,d2)= d1*d2*;
4. *I(h)=H*, где *H* - функция вычитания единицы, т. е. *H(d)= d-1*;
5. *I(p)=P*, где *P* - предикат «равно 0», т.е. *P(d)=1*, если *d=0*.

Пример

№ п/п	Значение терма для (R_{S1}, I_1)
1	F(4)
2	4*F(3)
3	4*(3*F(2))
4	4*(3*(2*F(1)))
5	4*(3*(2*(1*F(0))))
6	4*(3*(2*(1*1)))=24

Схемы с процедурами

- Схемы с процедурами
  - *главная схема*
  - *схема процедуры*

Главная схема - это стандартная схема, в которой имеются операторы присваивания специального вида $x := F^{(n)}(y_1, y_2, \ldots y_n)$, называемые *операторами вызова процедур*

Схема процедуры состоит из *заголовка и тела процедуры,* разделенных символом равенства. Заголовок имеет тот же вид, что и левая часть рекурсивных уравнений. Тело процедуры - это стандартная схема того же вида, что и главная схема.

Схемы с процедурами

Главная схема	Множество схем процедур
(**start**(x), 1: z:=x, 2: u:=a, 3: x:=F(x, z, u), 4: u:=b, 5: z:=F(z, x, u) 6: **stop**(z))	F(y, v, w) = **start**, 1: **if** p(y) **then** 2 **else** 4, 2: y:=h(y), 3: v:=G(v, w) **goto** 1, 4: **if** q(w) **then** 5 **else** 6, 5: y:= v, 6: **stop**(y)) G(t, r) = (**start**, 1: **if** q(r) **then** 2 **else** 3, 2: t := f(t), 3: **stop**(t);