Лекция 1. Математические модели программ

1. Модели программ курса теории программирования
2. Модели памяти программ. Алгебры данных программ
3. Модель вычислений
4. Модель управления.
5. Выражения в алгоритмической алгебре В.М.Глушкова
6. U-Y – программы
7. Семантика моделей программ
8. Семантика недетерминированных моделей управления

Текст программы на формально определенном языке программирования (с помощью **формул Бэкуса-Наура** или **синтексических диаграмм Н.Вирта**) по существу представляет математическую модели вычислений, т.е. модель программы. Текст прогрммы представляет собой формулу (выражение).

В практике программирования и обучении программированию широко используются **блок-схемы программ**. Программа представлена графом особого вида, вершины которого отмечены операторами, представляющими вычисления.

В теоретических работах по программированию используются как **графовые модели программ**, так и **алгебраические модели программ**.

Математические модели программ, в отчичие от собственно программ, представляют собой абстракции, ориентированные на решение конкретных теоретических задач. В таких моделях в абсолют возводятся одни средства и свойства программ, и игнорируются другие их средства и свойства.

Модели программ курса теории программирования

В настоящем курсе мы рассмотрим

- **Операторные схемы программ** (задача экономного распределения памяти)
- **Схемы программ Ю.Янова** (задача эквивалентного преобразования условий)
- ***U-Y* – схемы программ** над памятью (задачи статического анализа программ)
- **Выражения в алгоритмической алгебре В.М.Глушкова** (задачи генерации программных инвариантов)
- **Формулы пропозициональной динамической логики** (ПДЛ) (недетерминированные модели программ)

Модель программы это:

- **Модель памяти,**
- **Модель данных,**
- **Модель вычислений,**
- **Модель управления.**

Модели памяти программ. Алгебры данных программ

Общими понятиями для моделей программ являются:

Память программы – это модель ее хранилища данных.

Следует различать *неструктурированную* память и *структурированную* память. Структурированная память является предметом изучения теории типов данных.

Неструктурированная память – модель памяти программы с переменными простых типов.

В модель программы включают:

- Понятие (неструктурированной) памяти X программы P как конечного множества (вектора) переменных $X = (x_1, \ldots, x_n)$.
- Понятие алгебры данных D как области определения всех переменных (памяти) программы. Алгебра данных определяет множество операций над данными, допустимыми при программировании.
- Вообще говоря, D следует рассматривать как **многосортную алгебру**. К базовым алгебрам относятся алгебры (сорты) *Variable*, *Natural*, *Boolean*.
- Во многих задачах, кроме базовых, определен еще только один сорт – сорт, над которым с помощью операторов присваивания определены вычисления.

Модель вычислений

- Состояние $\bar{d}$ памяти программы P - вектор $\bar{d} = (d_1,...,d_n)$. Если память $X = (x_1,...,x_n)$ программы находится в состоянии $\bar{d}$, это означает, что $x_1 = d_1,...,x_n = d_n$. Таким образом, мы говорим о пространстве состояний памяти как о множестве D^X.
- Операторы присваивания во многих задачах интерпретируются как векторные. Мы будем называть их вычисляющими операторами. Оператор $y = (x_1 := F_1(X),..., x_n := F_n(X))$ преобразует пространство состояний памяти: y: $D^X \to D^X$.

Это означает, что их компоненты выполняются одновременно (параллельно). Множество операторов присваивания обозначим через Y. Отметим, что в некоторых задачах анализа программ следует рассматривать последовательное выполнение компонент.

- Условия преобразуют терм F алгебры D в ($True$, $False$). Множество условий обозначим через U. В многих моделях программ существенную роль играет язык условий L. Возможно, например, уточнение этого языка как языка алгебры предикатов:

  Сигнатура атомарных предикатов: "=","<","≤";

  Сигнатура логических связок: "&","∨","¬".

Модель управления.

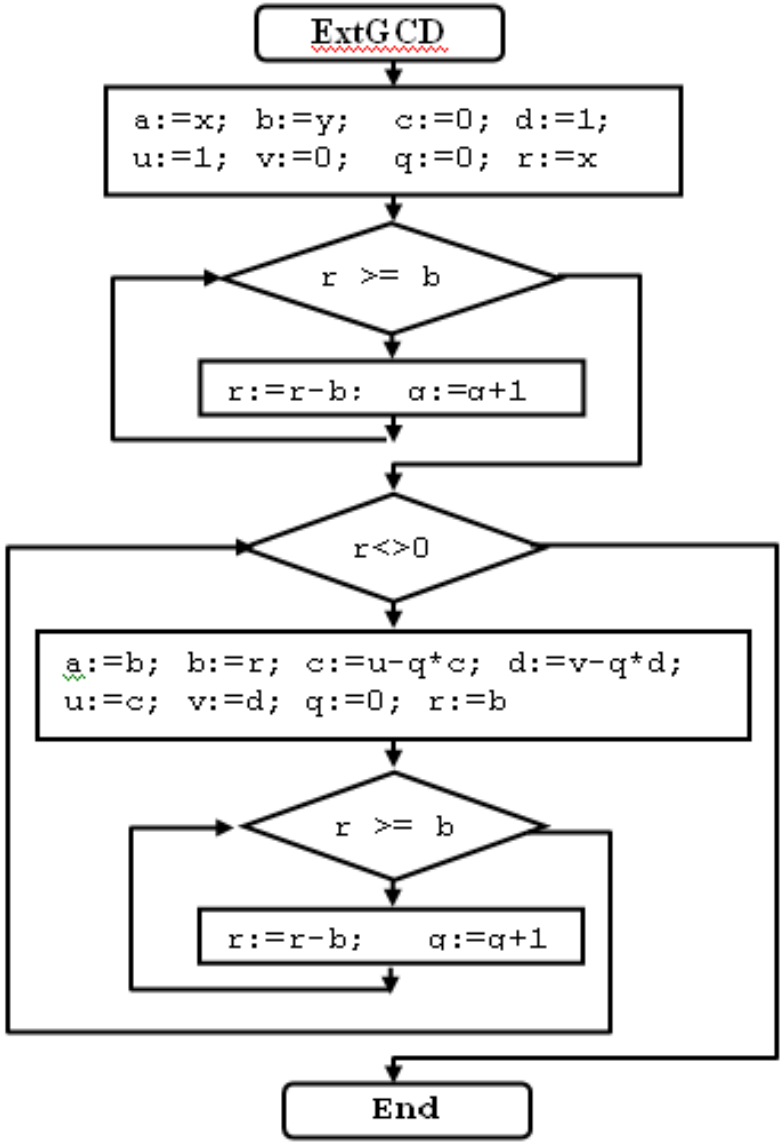
Рассмотрим в качестве примера программы и ее алгебраической модели расширенный алгоритм Евклида, который находит b = GCD(x, y) и такие числа c, d, что c*x + d*y = b.

Процедура реализует расширенный алгоритм Евклида

1. Исходный текст

```
Procedure ExtGCD(x, y: Integer; var b, c, d: Integer);
Var
  a, u, v, q, r : Integer;
Begin
  a:=x; b:=y; c:=0; d:=1; u:=1; v:=0; q:=0; r:=x
  While r >= b do begin
    r:=r-b;   q:=q+1
  End;
  While r<>0 do begin
    a:=b; b:=r; c:=u-q*c; d:=v-q*d;
    u:=c; v:=d; q:=0; r:=b
    While r >= b do begin
      r:=r-b; q:=q+1
    End
  End
End;
```

2. Блок-схема


ExtGCD
a:=x; b:=y; c:=0; d:=1;
u:=1; v:=0; q:=0; r:=x
r >= b
r:=r-b; q:=q+1
r<>0
a:=b; b:=r; c:=u-q*c; d:=v-q*d;
u:=c; v:=d; q:=0; r:=b
r >= b
r:=r-b; q:=q+1
End

Выражения в алгоритмической алгебре В.М.Глушкова

Алгебраические модели императивных структурированных программ используют представление программ формулами специальной программной алгебры, операциями которой являются операторы управления: последовательное выполнение, ветвление, повторение. Например:

$P;Q$ - последовательное выполнение;

$P \underset{u}{\vee} Q$ - ветвление с условием;

$\underset{u}{\{P\}}$ - повторение с предусловием;

$\{P\}_{u}$ - повторение с постусловием.

Алгебраическая модель ExtGCD в алгоритмической алгебре В.М.Глушкова

Для представления модели программы нужно сформировать:

$E = (x, y, b, c, d)$ - множество входных параметров программы;

$X = (a, b, c, d, q, r, u, v, x, y)$ - память программы;

$U = (u_1, u_2)$ - словарь условий программы;

$u_1 = (r >= b)$;

$u_2 = (r <> 0)$;

$Y = (y_1, y_2, y_3)$ - словарь операторов присваивания программы;

$y_1 = (a := x, b := y, c := 0, d := 1, u := 1, v := 0, q := 0, r := x)$;

$y_2 = (r := r - b,\ q := q + 1)$;

$y_3 = (a := b, b := r, c := u - q * c, d := v - q * d, u := c, v := d,\ q := 0, r := b)$.

Алгебраическая модель программы теперь определяется формулой:

$$P = y_1; \underset{u_1}{\{ y_2 \}}; \underset{u_2}{\{ y_3; \underset{u_1}{\{ y_2 \}} \}}$$

Модель программы – формула пропозициональной динамической логики

В качестве моделей программ будут использоваться формулы пропозициональной динамической логики (ПДЛ), интерпретированные над областью данных. Сокращенное описание:

Синтаксис ПДЛ включает два сорта выражений:

- программы **p, q, r**, …;
- высказывания (или формулы) $\varphi, \psi, \ldots$.

Если φ, ψ формулы, а **p**, **q** - программы, то

$\varphi \vee \psi, \neg \varphi, < p > \varphi$ - формулы;

p;q, **p|q**, **p***, φ**?** – программы.

Операция **p;q** означает последовательную композицию, **p|q** – недетерминированный выбор, **p*** - итерацию и φ**?** - проверку условия. Интуитивные значения этих конструкций таковы:

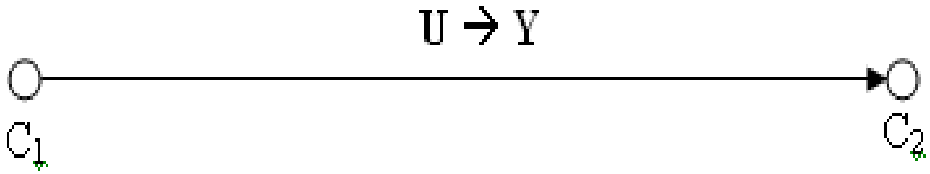
- **p;q** = "Выполнить **p**, а потом выполнить **q**",
- **p|q** = "Выбрать недетерминировано **p** или **q** и выполнить его",
- **p*** = "Выполнить p повторяя его недетерминированное число раз",
- φ**?** = "Проверка условия φ, вывести true, если она истинна или false, если она ложна".

Определение U-Y – программы

U-Y – схемой программы над памятью (*U-Y* – программой) называют инициальный связный граф *P* с выделенным подмножеством заключительных вершин (состояний), дуги которого отмечены парами (условие-оператор).

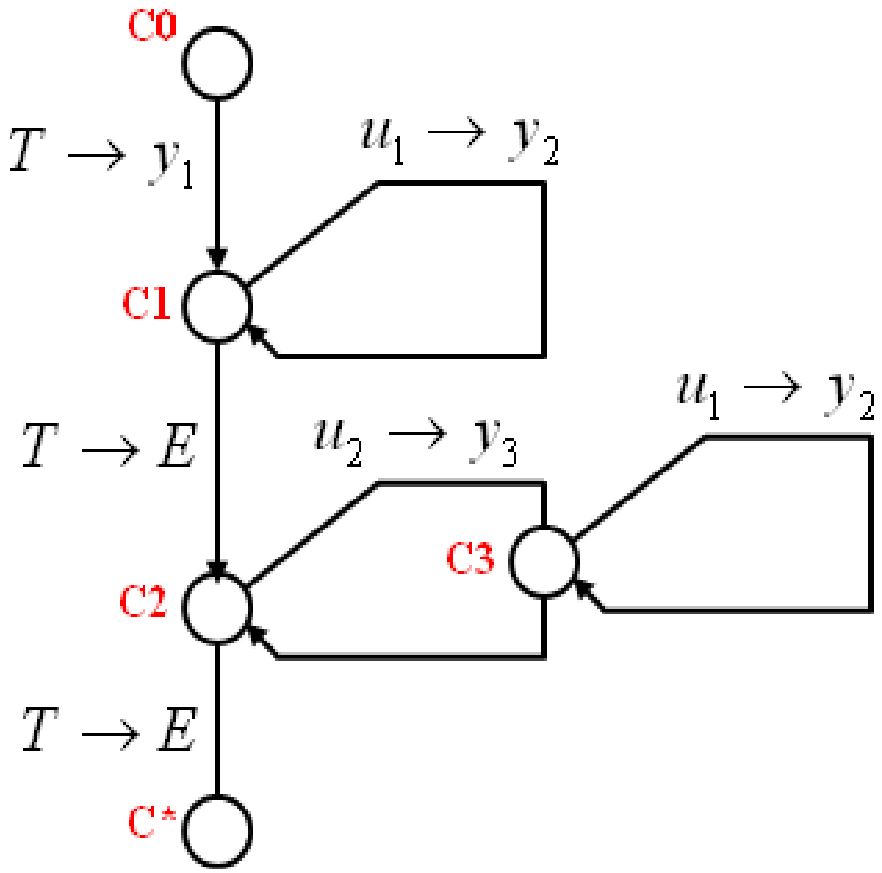
$$P = < S, E, s_0 S^*, \delta >, \ \delta : E \to (U, Y)$$

Вершины графа *P* – суть контрольные точки программы. Переход от одной точки к другой сопровождается выполнением оператора y, если u в текущем состоянии памяти программы принимает значение *True*.



Выполнение программы P начинается в состоянии s_0 , осуществляется недетерминировано и заканчивается в одном из заключительных состояний S^*.

Пример: U-Y схема программы ExtGCD



Семантика моделей программ

Модели программ, рассмотренные выше, представляют только статические (синтаксические) аспекты компьютерных программ.

Однако, наиболее важными являются динамические (вычислительные) аспекты.

Для каждого типа моделей нужно определить семантику, представленную моделями этого типа.

Базовые семантические определения – определения **семантики данных**.

$X = (x_1, x_2, ..., x_n)$ - память;

$$D = < Sign, Sort_1, ..., Sort_k >$$

$$Sign = < \varphi_1, \varphi_2, ..., \varphi_m >$$

$$\varphi_j : Sort_{j1} \times Sort_{j2} \times ... Sort_{jk_j} \to Sort_j$$

Примеры

$$y = a^x \qquad Pow : \mathrm{Re}al \times Natural \to \mathrm{Re}al$$

$$a < b \qquad Less : Number \times Number \to Boolean$$

$$Number = Natural \cup Integer \cup \mathrm{Re}al$$

Процесс исполнения программы

В примере ExtGCD

$$b = GCD(x, y) \text{ и } c*x + d*y = b.$$

$$X = (a, b, c, d, q, r, u, v, x, y)$$

Пространство состояний памяти

$$D^X = < Nat \times Nat \times Int \times Int \times Nat \times Nat \times Int \times Int \times Nat \times Nat >$$

Пусть `x = 28, y = 12.` Тогда исходное состояние памяти

$$S_0 = (\omega, \omega, \omega, \omega, \omega, \omega, \omega, \omega, 28, 12)$$

После выполнения оператора y_1

```
a:=x; b:=y;  c:=0; d:=1;  u:=1; v:=0;  q:=0; r:=x
```

$$S_1 = (28, 12, 0, 1, 0, 28, 1, 0, 28, 12)$$

Функциональная семантика программы

Процесс исполнения программы – последовательность состояний памяти

$\sigma = < S_0, S_1, S_2, \ldots S_m, \ldots, S^* >$ - завершившийся

$S_0, S_1, S_2, \ldots S_m, \ldots$ - не завершающийся

$\sigma(S_0) = S_0 P = S^* \qquad \sigma(S_0) = P(S_0) = S^*$

$\Sigma_P = \{S^* : \sigma - завершившийся\}$

$\Delta_P \subseteq D^X = \{S_0 : \sigma(S_0) - завершившийся\}$

$P : \Delta_P \to \Sigma_P$ функциональная семантика программы

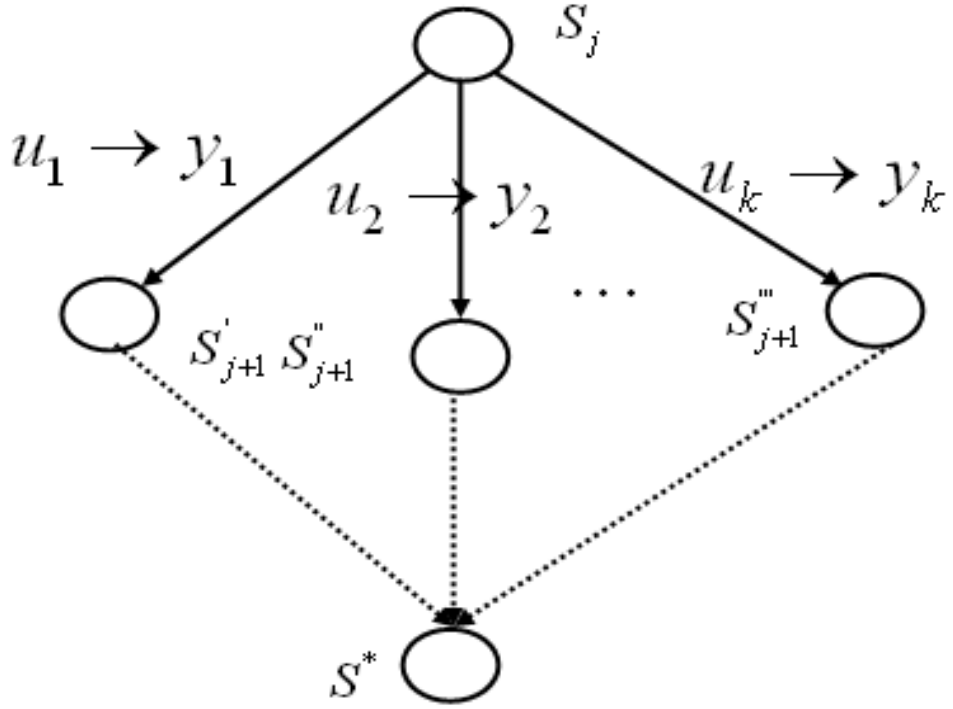
Отметим, что проблема $S \in \Delta_P$ - **проблема остановки** – алгоритмически неразрешима.

Семантика моделей управления

1. Семантика вычисляющего оператора $Y = F(X)$ нами установлена: $S_{j+1} = F(S_j)$. Она является общей для моделей U-Y программ и выражений в алгоритмической алгебре В.М.Глушкова.
2. Семантика «охраняемого» вычисляющего оператора $U(X) \to Y := F(X)$ определена естественным образом: $U(S_j)\ \&\ (S_{j+1} = F(S_j))$.

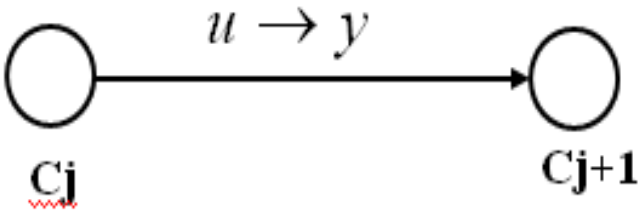
Семантика недетерминированных моделей управления

3. Семантика U-Y программы определяется недетерминированным ветвлением



В модели U-Y программы на условия $u_1, u_2, \ldots, u_k$ не накладывается никаких ограничений. Предпологжим, что $u_1(S_j) = True, u_2(S_j) = True$. Тогда должны выполняться вычисляющие операторы $y_1(S_j), y_2(S_j)$. Это и есть **недетерминизм.**

Определение 1. Исходное состояние памяти S_0 программы P называется допустимым, если в графе управления P существует путь $(c_0, c_1, ..., c_k, c^*)$ такой, что для любого перехода



$$(u(S_j) = True)\ \&\ (S_{j+1} = y(S_j))$$

Определение 2. U-Y программа P называется детерминированной, если в любом состоянии ветвления $U_i \,\&\, U_j = \varnothing$ и $U_1 \vee ... \vee U_k = True$.

Теорема о детерминизации. Для любой U-Y программы P существует детерминированная программа Q такая, что $\Delta_P = \Delta_Q, \Sigma_P = \Sigma_Q$ (функциональные семантики совпадают).

Выводы

- Модель компьютерной программы – математический объект, отражающий те свойства программы, которые подлежат изучению и игнорирующий остальные свойства программы.
- Как правило, модель программы описывается либо алгебрайческой формулой, либо графом (алгебраические и графические модели)
- Модель программы определена ее синтаксисом и семантикой. Семантика программы определяет ее динамические свойства (исполнение).

Спасибо за внимание