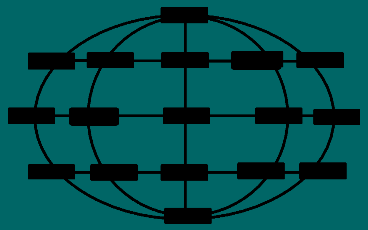
Методы построения и анализа алгоритмов



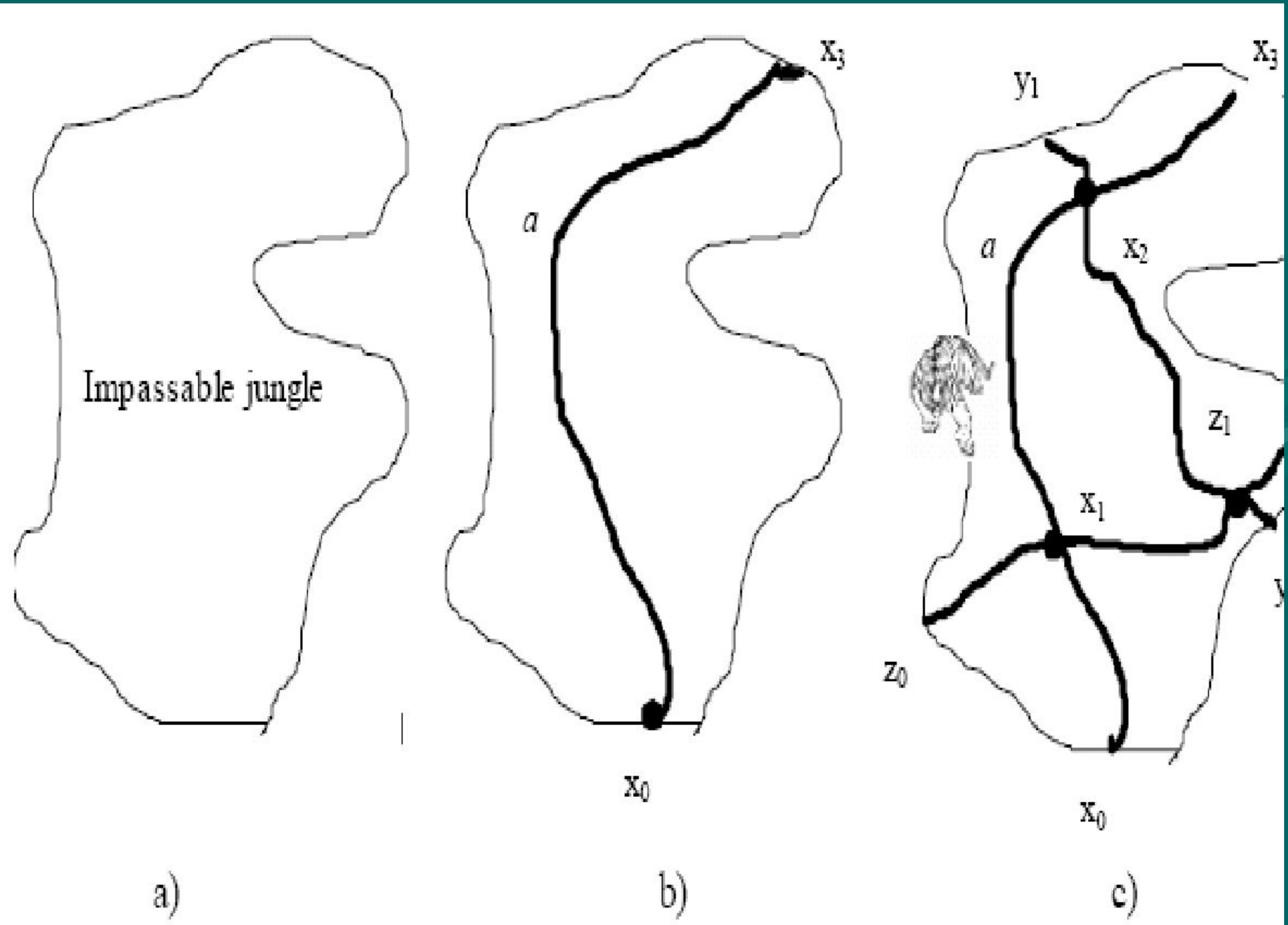
Малышкин Виктор Эммануилович

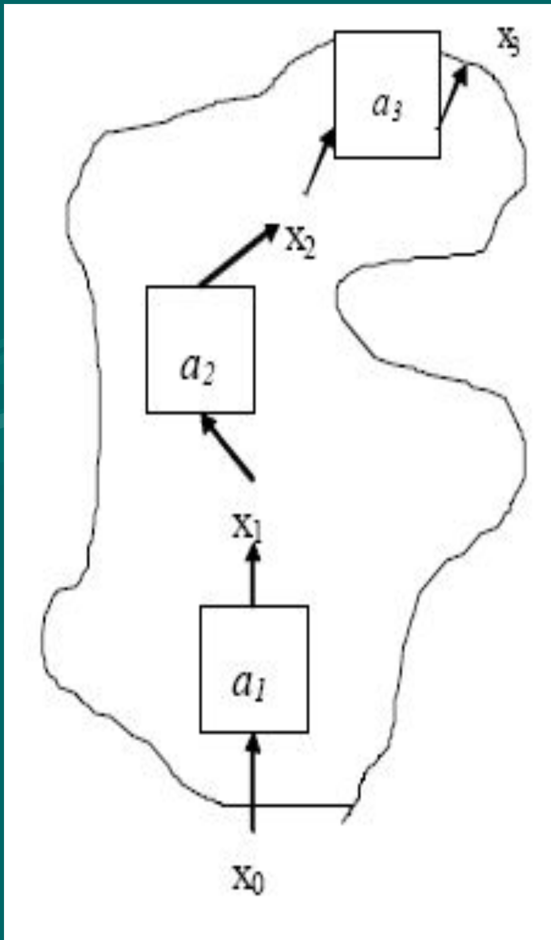
Кафедра Параллельных Вычислительных Технологий
Новосибирский государственный технический университет

E_mail: **malysh@ssd.sscc.ru**
Телефон: **3308 994**

Новосибирск

Общая идея структурного синтеза программ



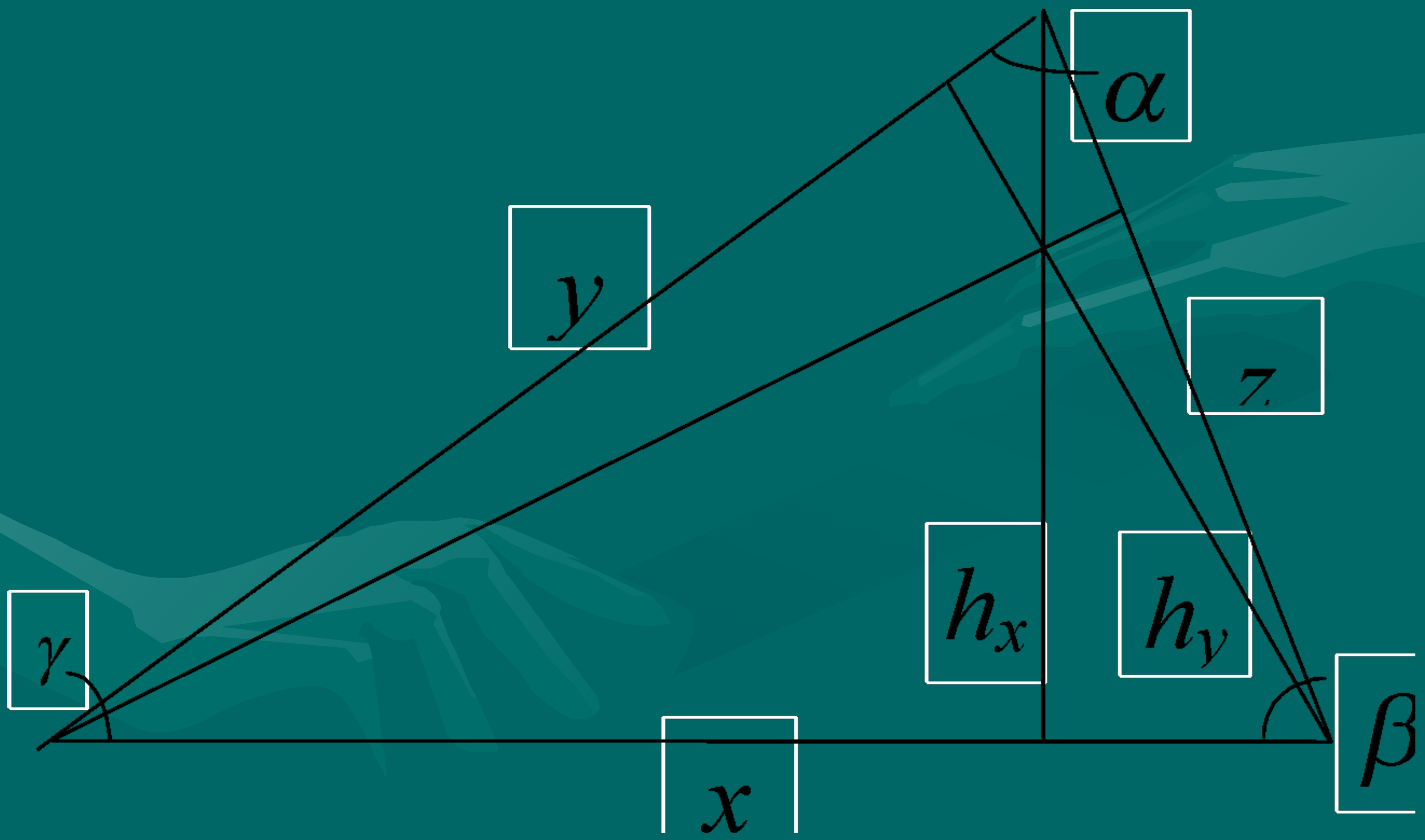


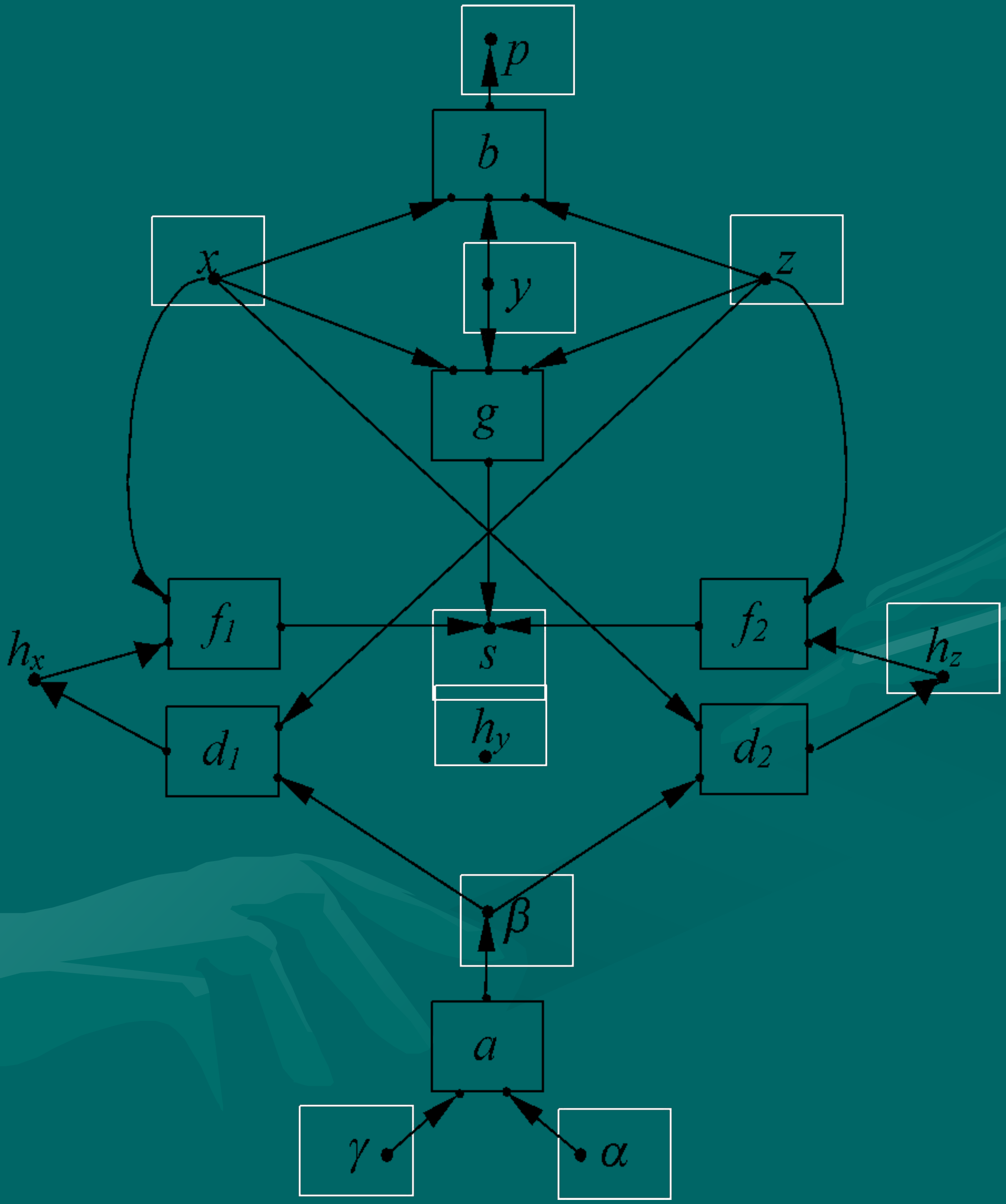
1. Базой знаний в вычислительных моделях является множество алгоритмов, причем хороших алгоритмов (как тропинки в джунглях не прокладываются плохо, так и в вычислительных моделях накапливаются только хорошие алгоритмы). И комбинации хороших алгоритмов (путь *x0x1z1x2x3* в джунглях) тоже могут быть хороши. Они хотя и не обязательно оптимальны, но и не самые худшие. Задача вывода приемлемого алгоритма становиться простой и сводится к ограниченному управляемому перебору на графе.

В дополнению к этому, так же как массив джунглей разбиваются тропинками на фрагменты, так и описание предметной области разбивается в вычислительных моделях на множество меньших предметных областей, для которых построение более или менее полных теорий (для каждой подобласти своей) более вероятно. Таким образом, в вычислительных моделях формализованное описание предметной области строится как система теорий, связанных соотношениями модели.

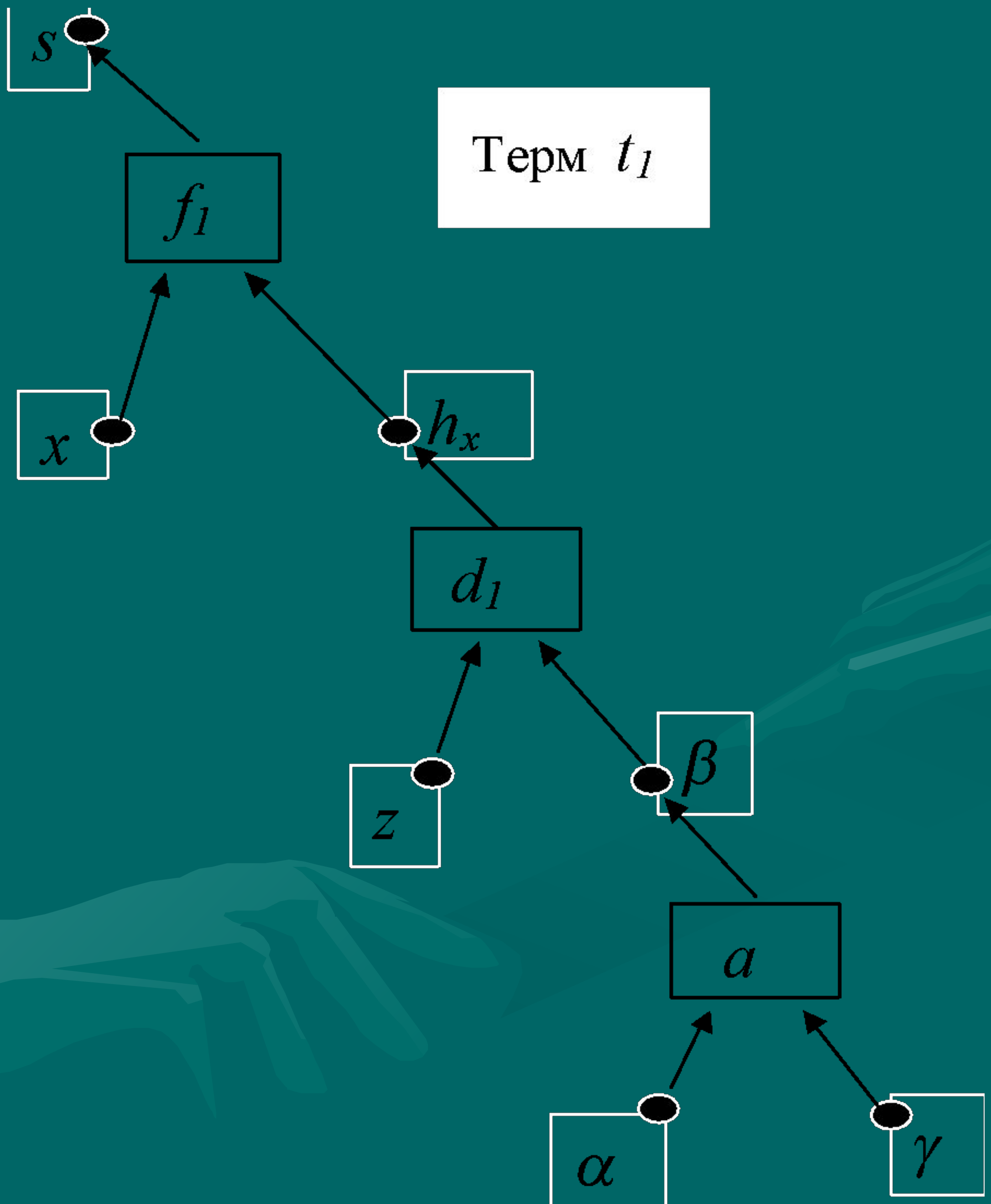
Метод синтеза программ на вычислительных моделях применяется тогда, когда достаточно полная модель предметной области еще не создана, а считать уже надо - обычная на практике ситуация.

X$=\{x, y, ..., z\}$ конечное множество переменных, **F**$=\{a, b, ..., c\}$ конечное множество функциональных символов. Пара **C**=(**X**,**F**) называется *вычислительной моделью*.





p
b
x
y
z
g
f1
s
f2
hx
hz
hy
d1
d2
β
a
γ
α



Терм t_1

Множества термов из $T(V,F)$ обозначается $T1$, $T1 \subseteq T(V,F)$. Впредь будем работать только с термами из $T1$. Это конечные множества.

Множество термов $T_V^W = \{t \in T1 \mid \text{out}(t) \cap W \neq \varnothing\}$. Это множество задает все вычисления, которые основаны на V и завершаются в W.

Множество термов $R \subseteq T_V^W$ такое, что $\forall x \in W \, \exists t \in R(x \in \text{out}(t))$ называется (V,W)-планом вычислений. Ясно, что (V,W)-план задает детерминант вычислимой функции, которая вычисляет переменные W из переменных V

Планирование алгоритма

Разработано много различных алгоритмов планирования. Здесь рассматривается хорошо реализуемый алгоритм, который позволяет строить все термы из T_V^W и имеет линейную временную сложность относительно числа дуг в графическом представлении ПВМ.

Представление графа

Пусть задана вычислительная модель **C**=(**X**,**F**), которая после трансляции представлена в виде двух таблиц *ТХ* и *О*П. Каждая строка таблицы *ТХ* имеет вид (x,A(x),comp(x)), а таблицы *О*П - (a,in(a),out(a)).

Здесь $x \in \mathbf{X}$, $a \in \mathbf{F}$, $\mathrm{comp}(x)=\{a \in \mathbf{F} \mid x \in \mathrm{out}(a)\}$, $A(x)=\{a \in \mathbf{F} \mid x \in \mathrm{in}(a)\}$.

Алгоритм планирования состоит из двух частей: восходящей и нисходящей.

В *восходящей* части алгоритма строятся множества переменных и операций, используемых в термах из множества $T_V=T(V,F)$.

Обозначим $V_0=V$, тогда

$$F_0=\{a\in \mathbf{F} \mid \mathrm{in}(a)\subseteq V_0\}=\bigcup_{x\in V_0}\{a\in A(x) \mid \mathrm{in}(a)\subseteq V_0\}$$

содержит все операции ПВМ такие, что $\mathrm{in}(a)\subseteq V_0$. Далее формируется множество $V_1=\{x\in \mathbf{X} \mid x\in \mathrm{out}(a)\wedge a\in F_0\}\cup V_0$, на основе V_1 строится множество

$$F_1=\bigcup_{x\in V_1\setminus V_0}\{a\in A(x) \mid \mathrm{in}(a)\subseteq V_1\}$$

и т. д. до тех пор, пока при некотором целом положительном k не окажется, что $F_k=\varnothing$. На этом завершается восходящая часть алгоритма планирования.

Множества V_i и F_i, $i=0,\ldots,k$, содержат все переменные и операции, используемые в термах из множества T_V

Если $W \nsubseteq V_k$, то планирование можно прекращать, так как в этом случае существует переменная в W, которая не вычисляется никаким термом из множества T_V, и, следовательно, не существует алгоритма решения сформулированной задачи на основе имеющихся знаний о ПО. В этом случае говорим, что сформулированная задача синтеза *неразрешима*. В противном случае можно начать строить множества переменных и операций, используемых в термах из T_V^W.

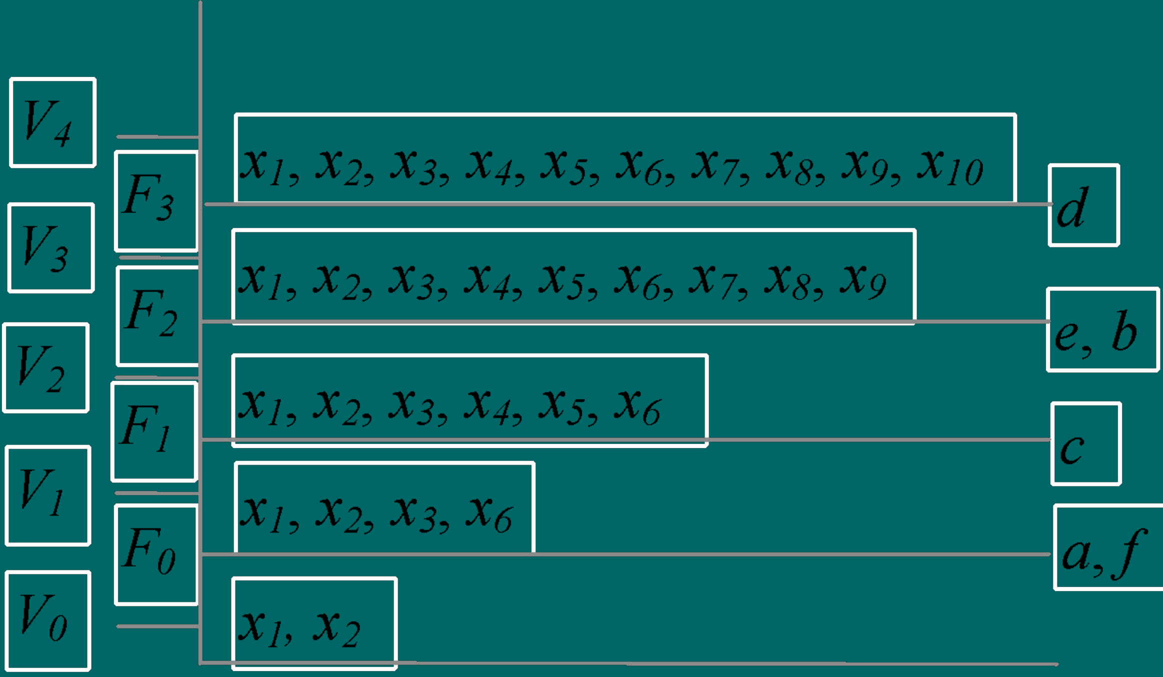
Обозначим $F^* = \bigcup_{i=0} Fi$, и определим множества:

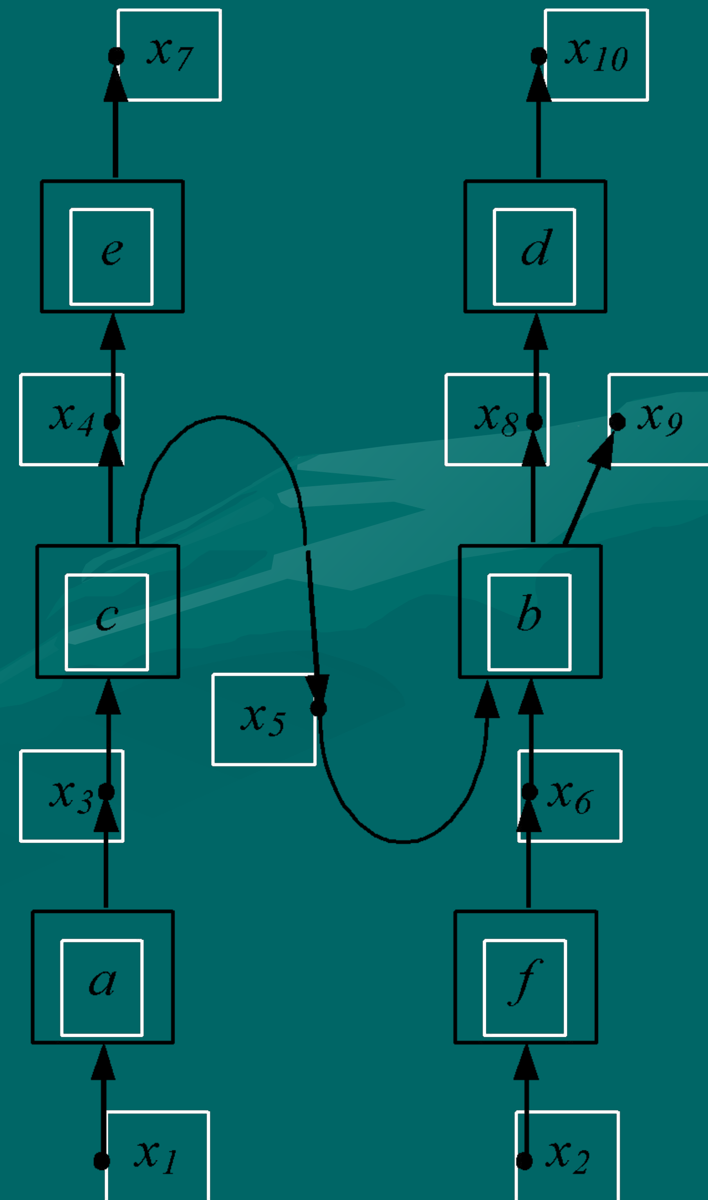
$$G_i = \bigcup_{x \in H_{i-1}} \left\{ a \in F^* \;\middle|\; a \in \mathrm{comp}(x) \wedge a \notin \bigcup_{m=1}^{i-1} G_m \right\},\ H_i = \bigcup_{a \in G_i} \mathrm{in}(a).$$

Построение множеств G_i и H_i завершается, когда при некотором целом положительном r окажется $G_r = \varnothing$.

Множества G_i и H_i, $i = 1, \ldots, r$, содержат все переменные и операции, используемые в термах из множества T_V^W

Построение множеств G_i и H_i завершается, когда при некотором целом положительном r окажется $G_r = \varnothing$. Множества G_i и H_i, $i = 1, ..., r$, содержат все переменные и операции, используемые в термах из множества .

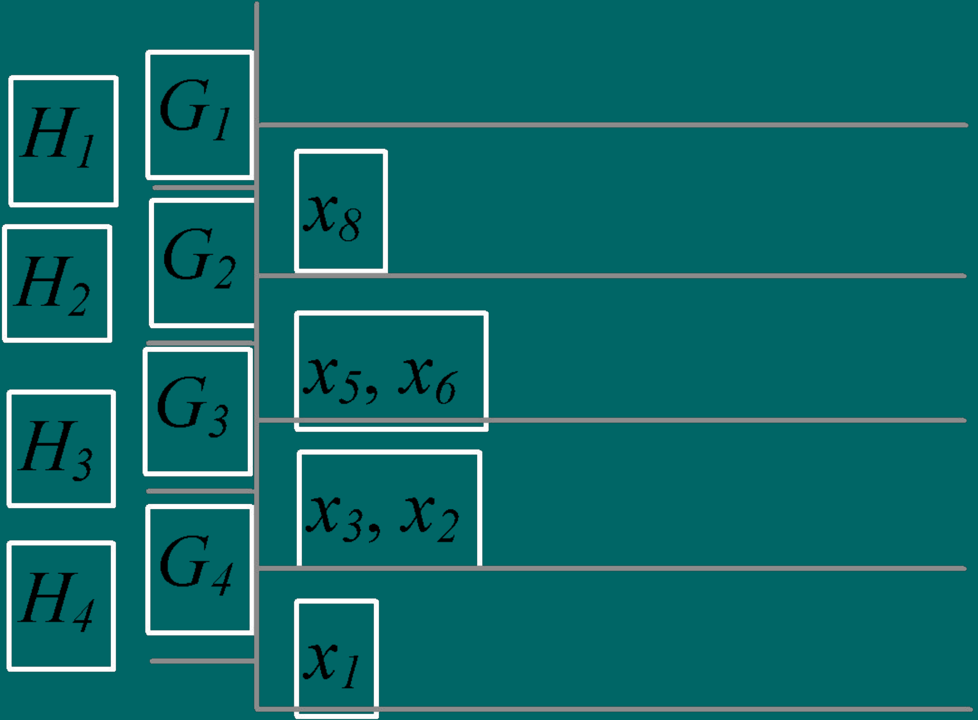




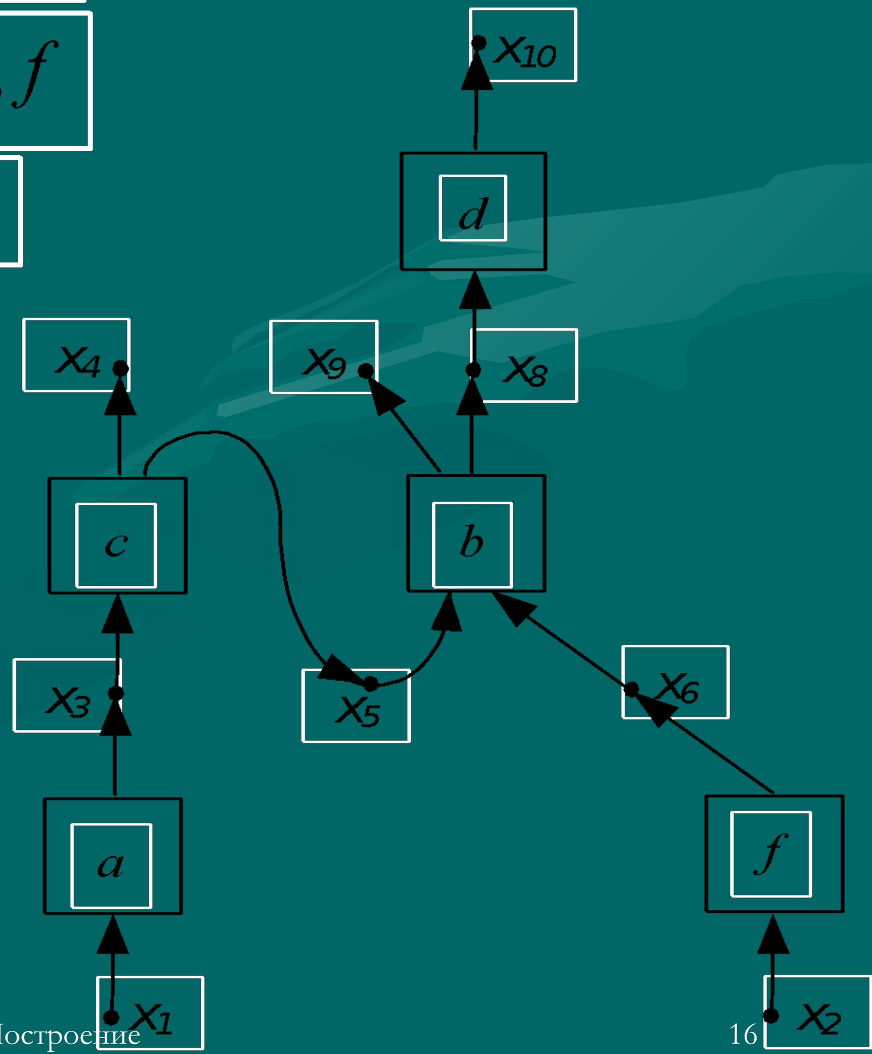
Сверху множества F_i и V_i, образовавшиеся в результате восходящей части алгоритма планирования на ПВМ справа

$$V = \{x_1, x_2\},$$

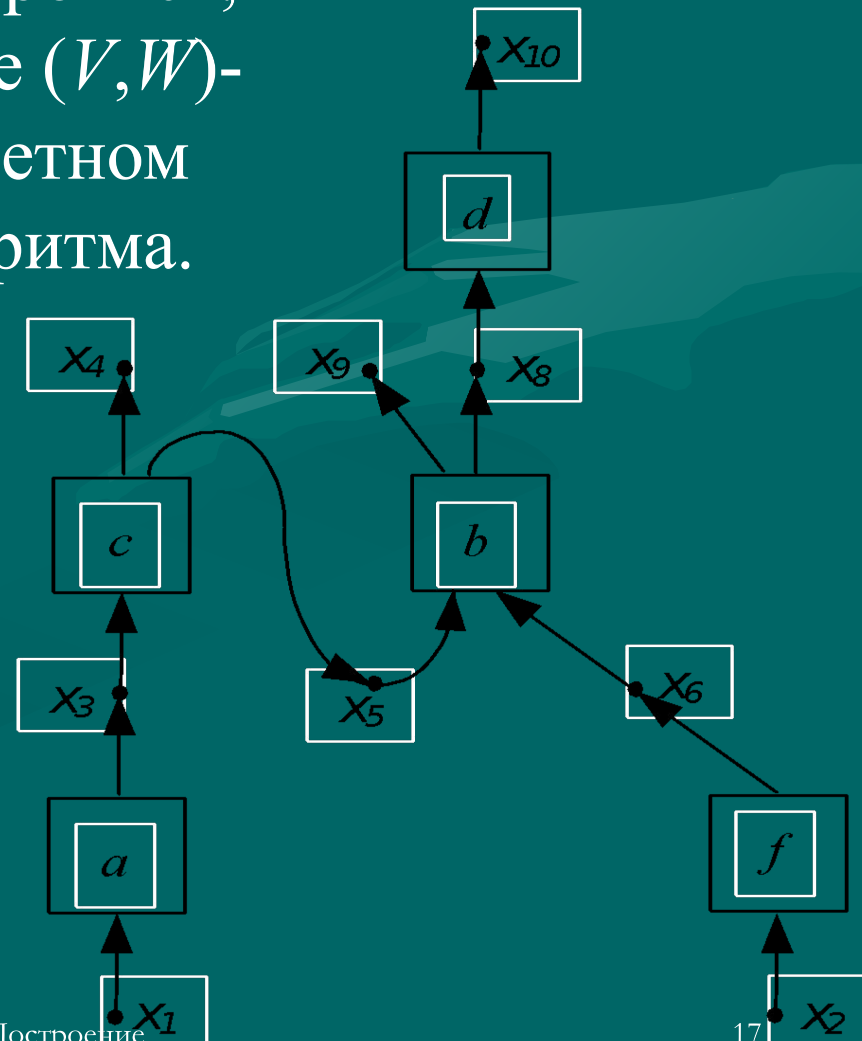
$$W = \{x_{10}\}$$



Множества G_i и H_i (сверху) сформировались в нисходящей части алгоритма планирования. После завершения планирования остаются лишь переменные и операции из множеств Gi и Hi, остальные удаляются (справа).



Таким образом, результатом планирования является ПВМ, оставшаяся от ***C*** после удаления “лишних” переменных и операций. Множество T_V^W не строится, подходящий в некотором смысле (V,W)-план T строится в каждом конкретном случае процедурой выбора алгоритма.



В случае, когда $W \not\subseteq Vk$, сформулированная задача синтеза оказывается неразрешимой и необходимо изменить формулировку задачи, т. е. либо уменьшить W, удалив из него невычислимые переменные, либо расширить V, включив в него такие новые переменные, что станут вычислимыми все переменные из W. Для уменьшения затрат на расширение V может быть использован алгоритм планирования. Для этого необходимо выполнить его *нисходящую* часть из множества переменных $W'=W \setminus Vk$ с использованием всех операций из F. Все переменные из построенных при этом множеств Hi, $i=1, 2, ..., r$, являются кандидатами на включение в V. Из них человек может выбрать те переменные, значения которых ему доступны.

Из описания алгоритма следует, что проверка условия in(a)$\subseteq V_i$ делается не более одного раза для каждой входной дуги произвольно взятой операции a, а проверка условия out(a)$\cap H_{i\text{-}1} \neq \varnothing$ - не более одного раза для каждой выходной дуги a. Понятно, что алгоритм планирования имеет линейную относительно числа дуг в графическом представлении ПВМ временную сложность, если в качестве элементарных шагов алгоритма взять проверки in(a)$\subseteq V_i$ и out(a)$\cap H_{i\text{-}1} \neq \varnothing$.

При реализации алгоритма переменные и операции в *ТХ* и *ОП* могут кодироваться целыми положительными числами. Для представления всевозможных множеств переменных — $A(x)$, $\mathrm{in}(a)$, Vi, Fi и т. д., — можно использовать битовые шкалы. Шкала V_i, к примеру, содержит в k-й позиции единицу, если переменная номер k принадлежит Vi. Применение битовых шкал сводит проверку условий $\mathrm{in}(a) \subseteq Vi$ и $\mathrm{out}(a) \cap Hi\text{-}1 \neq \varnothing$ к двум логическим операциям.

....

Рекомендуемые учебники

- Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д. *Структуры данных и алгоритмы*. : Пер. с англ. : Уч. пос. — М. : Издательский дом "Вильяме", 2000. — 384 с.
- Кормен Т., Лейзерсон Ч., Риверс Р., Штайн К. *Алгоритмы. Построение и анализ* – М.: «Вильямс», 2012
- В.Э.Малышкин, В.Д.Корнеев. *Параллельное программирование мультикомпьютеров*. – В серии «Учебники НГТУ», Новосибирск, изд-во НГТУ, 2011, 296 стр. (есть в библиотеке)

ВОПРОСЫ

1. Что мы называем алгоритмом? Почему?
2. Сколько существует алгоритмов и программ, вычисляющих вычислимую функцию?
3. Задача, ее модель, алгоритм решения
4. Задача управления движением на перекрестке и ее модель
5. Три подхода к решению комбинаторной задачи
6. Задача раскраски графа. Жадный алгоритм раскраски графа
7. Абстрактные типы данных. Что такое?

ВОПРОСЫ

8.Что такое вычислительная сложность алгоритма?

9. Время работы алгоритма. От чего зависит? Верхняя оценка сложности.

10. Общая схема решения *переборных* задач .Какие алгоритмы называются эвристическими?

11. Задача/проблемы построения расписания

12, Формулировки задачи построения расписания.

13. Способы сокращения перебора.

14. Стратегии построения субоптимплльных расписаний