МГТУ им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

УДК 519.876.5

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОПИСАНИЯ ПРОЦЕССОВ ФУНКЦИОНИРОВАНИЯ ДИСКРЕТНЫХ СИСТЕМ

Электронное учебное издание

Учебное пособие по дисциплине
«Описание параллельных процессов»

Автор Черненький В.М.

Рекомендуется НМС МГТУ им. Н.Э. Баумана в качестве учебного пособия

МОСКВА
2012

Аннотация

Пособие предназначено для студентов, изучающих проблемы описания взаимодействующих последовательно-параллельных процессов. Пособие содержит теорию описания совокупности процессов. В основу построения формализованной схемы описания функционирования АСУ положено понятие процесса. Выполнено исследование таких операций над процессами, как свертка, развертка, проецирование, объединение. Материал опирается на способ задания процесса, названный алгоритмической моделью процесса (АМП), включающий такие понятия, как элементарный оператор, трек операторов и инициатор. Дальнейшие исследования описания процессов на основе АМП позволили строго определить понятия структуры, локальных сред, обобщенных и объединенных операторов, блоков, выполнить их классификацию, определить понятие ресурса, конфликтов на ресурсах, предложить способы их разрешения. Полученные в итоге результаты позволяют предложить языковую метасистему, обладающую достаточно высоким уровнем абстракции.

1. Определение процесса

Система отличается многочисленными характеристиками, определяемыми аспектами ее описания. Нас будет интересовать сейчас и в дальнейшем анализ процесса ее функционирования. Под функционированием системы понимается процесс изменения ее состояния во времени. В данном курсе рассматривается способ описания такого процесса с учетом того, что система имеет высокую размерность, разделяется на множество объектов, различным способом связанных между собой, руководствуется сложными алгоритмами, описывающими переход из одного состояния в другое.

Всю совокупность параметров системы, определяющих процесс функционирования или участвующих в нем, назовем *параметрическим множеством системы* $Q = \{q_i\}_{i=1}^{n}$, где q_i – некоторый параметр. Каждый параметр q_i принимает множество значений, обозначаемое в дальнейшем как $\sigma(q_i)$.

Например, выберем в качестве системы центральный процессор (ЦП) и жесткий диск (Д).

Для ЦП определим следующие параметры:

количество процессоров (КП)

производительность одного процессора (ПОП)

состояние процессора (СП)

стоимость процессора (СтП).

Для Д определим следующие параметры:

скорость записи и считывания (СЗС)

состояние диска (СД).

Тогда множества значений параметров можно определить следующим образом:

σ(КП) = {1, 2};

σ(ПОП) = {10, 50, 100, 500}, эти значения могут соответствовать какой-либо системе оценки производительности, например, обозначать среднее количество тысяч операций за единицу времени;

σ(СП) = {“работает”, “простаивает”, “тестируется”};

σ(СтП) = {2 - 20}, эти значения стоимости могут быть выражены, например, в некоторых условных единицах;

σ(СЗС) = {10 - 50}, эти показатели скорости операций на диске могут быть заданы аналогично параметру ПОП;

σ(СД) = {“работает”, “простаивает”, “ремонтируется”}.

Определим *пространство состояний системы* как декартово произведение $S=\prod_{\forall i}\sigma(q_i)$. В этом пространстве каждый параметр выступает в роли координаты, а размерность пространства равна мощности множества *Q*. Элемент пространства *S* есть возможное *состояние системы*. В рассмотренном выше примере размерность пространства состояний равна 6, а само пространство состояний имеет вид:

$$S = \sigma(\text{КП}) \times \sigma(\text{ПОП}) \times \sigma(\text{СП}) \times \sigma(\text{СтП}) \times \sigma(\text{СЗС}) \times \sigma(\text{СД})$$

Возможным состоянием системы $\mathbf{s} \in S$ может быть следующий кортеж:

$$\mathbf{s} = <2,\ 50,\ \text{“простаивает”},\ 15,\ 40,\ \text{“ремонтируется”}>$$

В дальнейшем нас будет интересовать процесс изменения состояний системы во времени. Примем за основу определение процесса, предложенное в работе [2].

Процесс Z есть четверка:

$$Z=< S,\ T,\ F,\ \alpha > \qquad (1)$$

где:

S – пространство состояний системы, определенное ранее;

T – множество моментов времени *изменения* состояний системы;

F – график процесса, определяемый как отображение $T \to S$, причем это отображение должно быть функциональным (однозначным);

α – отношение линейного порядка на *T*.

Если множество *T* задано как упорядоченное, то в определении процесса α может быть опущено.

В общем случае множества *T* и *S* могут быть как дискретными, так и непрерывными.

Интервал времени $[t_\text{Н},\ t_\text{К}]$, где $t_\text{Н}=\min\{\text{T}\}$, $t_\text{К}=\max\{\text{T}\}$, назовем *интервалом определения процесса*.

Поскольку пространство S координатного типа, то в случае необходимости подчеркнуть систему координат Q, на которой оно определено, будем обозначать его также S_Q.

В этих обозначениях, если множество T задано, как упорядоченное, а пространство S определено на множестве параметров Q, определить процесс можно как: $Z=< S_Q, T, F >$.

Кортеж $< \tilde{x}_1 ... \tilde{x}_i ... \tilde{x}_n >$, где $\tilde{x}_i$ - значения элементов множества X , будем обозначать как $< ... \underset{X}{\tilde{x}} ... >$.

Определим фазовое пространство Φ процесса Z как $\Phi = T \times S$.

Тогда график F есть подмножество Φ.

Если $f \in F$, то $f = <t, < ... \underset{Q}{\tilde{q}} ... >>$, где $t \in T$.

Если $Q_1 \subseteq Q$, то определим понятие проекции f на пространство S_{Q_1} как:

$Пр_{S_{Q_1}} f = <t, < ... \underset{Q_1}{\tilde{q}} ... >>$. Проекцией f на T является t.

Введем понятие подпроцесса Z^i как плотное во времени подмножество процесса Z на интервале $[t_i; t_j]$ при условии, что $[t_i; t_j] \subseteq [t_H, t_K]$. Плотность по времени означает, что на интервале $[t_i; t_j]$ нет ни одной точки, принадлежащей T и не относящейся к подпроцессу Z^i . Этот интервал назовем *интервалом определения подпроцесса.* Понятие подпроцесса позволяет рассматривать процесс в виде последовательности подпроцессов и производить операции разделения и объединения фрагментов процесса.

2. Операции над процессами

Операция свертки процесса

Пусть задан процесс $Z=<S, T, F, \alpha>$

Процесс $Z_1=<S_1, T_1, F_1, \alpha_1>$ является сверткой процесса Z, если он получен в результате следующих преобразований:

а) произведено полное разбиение интервала определения процесса Z на n непересекающихся подинтервалов $[\tau_j, \tau_{j+1}]$, где j=1..n, причем $\tau_1=t_H$, $\tau_{n+1}=t_K$. В результате получим разбиение процесса Z на n подпроцессов Z^j (j=1..n);

б) поставим в соответствие каждому подпроцессу Z^j одно значение состояния s_1^j из множества S_1 и одно значение времени β^j из интервала $[\tau_j, \tau_{j+1}]$. В результате получим дискретное множество $T_1 = \{\beta^j\}_{j=1}^n$, график $F_1 = \{<\beta^j, s_1^j>\}_{j=1}^n$, отношение $\alpha_1 \subset \alpha$.

Таким образом, получим новый процесс Z_1, который и называется сверткой процесса Z. Очевидно, процесс Z_1 дискретен во времени. Никаких ограничений на характер пространства состояний S_1 не накладывается. Однако на практике при проведении операции свертки пространство S_1, как правило, оказывается значительно меньшей мощности, чем исходное пространство S.

Рассмотрим для примера процесс решения задачи с использованием CPU и жесткого диска HD (рисунок 1).

Пусть исходный процесс Z описан в пространстве S, имеющем следующие состояния: {обращение к памяти, загрузка регистров, операции сумматора, головка диска, запись-считывание на диске, печать результата}.

Множество моментов времени изменения состояния T = {0, t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11}.

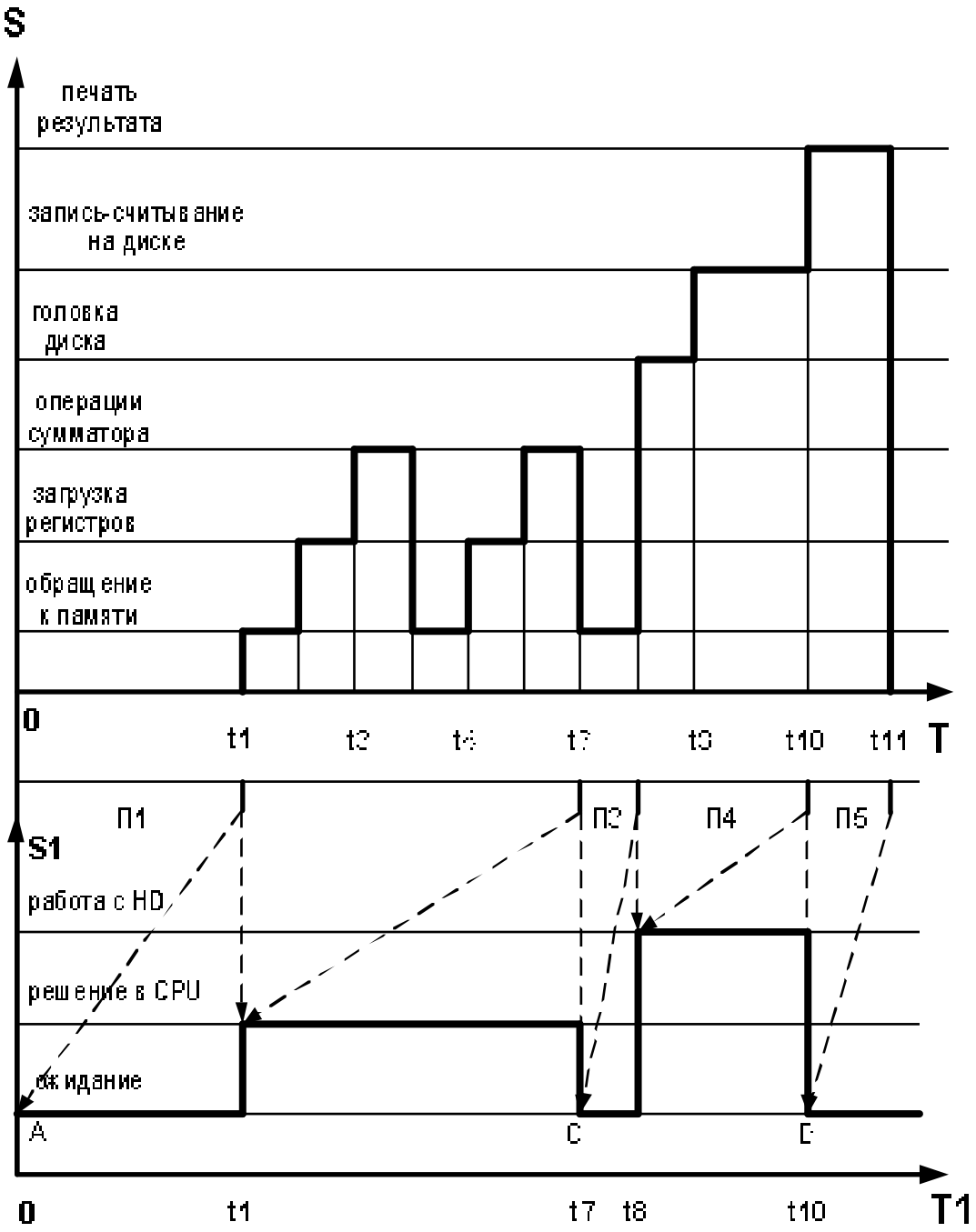


Рисунок 1. Пример операции свертки процесса

Допустим, что нам необходимо построить процесс Z1, отражающий лишь длительность занятия задачей CPU и жесткого диска HD. Для этого зададим новое пространство S1, имеющее следующие состояния: {ожидание, решение в CPU, работа с HD}. Разобьем интервал определения процесса Z на подинтервалы: П1, П2, П3, П4, П5. Это разбиение определяет также

соответствующие подпроцессы процесса Z. Выполним отображение подпроцессов процесса Z на фазовую плоскость процесса Z1: П1 отображается в точку A, П2 – в точку B, П3 – в точку C, П4 – в точку D, П5 – в точку E. В результате получим процесс Z1 с пространством состояний S1, множеством моментов времени изменения состояния T1={0, t1, t7, t8, t10}. График этого процесса приведен на этом же рисунке.

Как видно из примера, операция свертки порождает новый процесс, дискретный во времени, поскольку подпроцессы процесса Z, имеющие конечную длительность, отображаются лишь на одну точку фазовой плоскости нового процесса Z1.

Операция свертки относится к классу операций анализа.

Операция развертки

Операция развертки обратна по отношению к операции свертки: процесс Z является разверткой процесса Z_1. При выполнении этой операции необходимо каждую точку $\langle \beta^j, s_1^j \rangle$ процесса Z_1 развернуть в подпроцесс Z^j.

Поставим каждому β^j в соответствие интервал $[\tau_j , \tau_{j+1}]$, при условии, что:

$$\tau_j \le \beta^j \le \tau_{j+1} \qquad \text{и} \qquad \bigcap_{\forall j} [\tau_j , \tau_{j+1}] = \varnothing. \tag{2}$$

Зададим отображение B_j: $[\tau_j, \tau_{j+1}] \to S$. Отображение B_j позволяет получить фазовую траекторию подпроцесса Z^j. Для построения процесса Z в целом необходимо задать все B_j $(j=1,..,n)$. Операция развертки позволяет восстановить исходный процесс на основе некоторых представлений о свернутых процессах.

Операция развертки относится к классу операций синтеза.

Операция проецирования

Процесс Z_1 является *проекцией* процесса Z на координатное пространство S_{Q_1} (обозначение $Z_1 = Пр_{S_{Q_1}} Z$), если $Q_1 \subseteq Q$ и процесс построен по следующей процедуре:

1) каждую точку графика F проецируем на пространство S_{Q_1}. В результате получаем множество $\overline{F}$. Мощность множества $\overline{F}$ равна мощности множества F;
2) упорядочиваем множество $\overline{F}$ в соответствие с α. Результат действий 1) и 2) будем называть *отображением* процесса Z на пространство S_{Q_1};
3) вводим отношение эквивалентности на множестве $\overline{F}$ такое, что **r** *подряд расположенных* точек f_{i+1} ,… f_{i+r} множества $\overline{F}$ (f_{i+1}= <t^{i+1}, $s_1{}^{i+1}$> ,… f_{i+r}=<t^{i+r}, $s_1{}^{i+r}$>) считаются эквивалентными, если $s_1{}^{i+1}$= $s_1{}^{i+2}$ = …=$s_1{}^{i+r}$

r может принимать любые положительные целые значения, начиная от 1. Таким образом формируются классы эквивалентных значений $K_Э$. При **r**=1 класс содержит одну единственную точку.

4) каждому классу эквивалентности $K_Э$ на $\overline{F}$ ставим в соответствие одну точку $f_{экв}$= <t_{min}, $s_{экв}$> где $t_{min} = \underset{\forall t_i \in K_Э}{min}\{t\}$, $s_{экв}$=$s_1{}^{i}$=… для всего класса $K_Э$.
5) формируем множество F_1 из элементов $f_{экв}$ по всем классам эквивалентности на $\overline{F}$, мощность F_1 равна количеству классов эквивалентности на $\overline{F}$.

6) проецируя F_1 на T, получим множество T_1. Очевидно, что $T_1 \subseteq T$, сужение отношения α на T_1 обозначим α_1.

В результате выполнения вышеуказанных операций получим процесс $Z_1 = Пр_{S_{Q_1}} Z$:

$$Z_1 = < S_{Q_1}, T_1, F_1, \alpha_1 >.$$

Пользуясь операцией проецирования, можно переопределить понятие подпроцесса:

$$Z_{[t_1,t_2]} = Пр_{[t_1,t_2]} Z$$

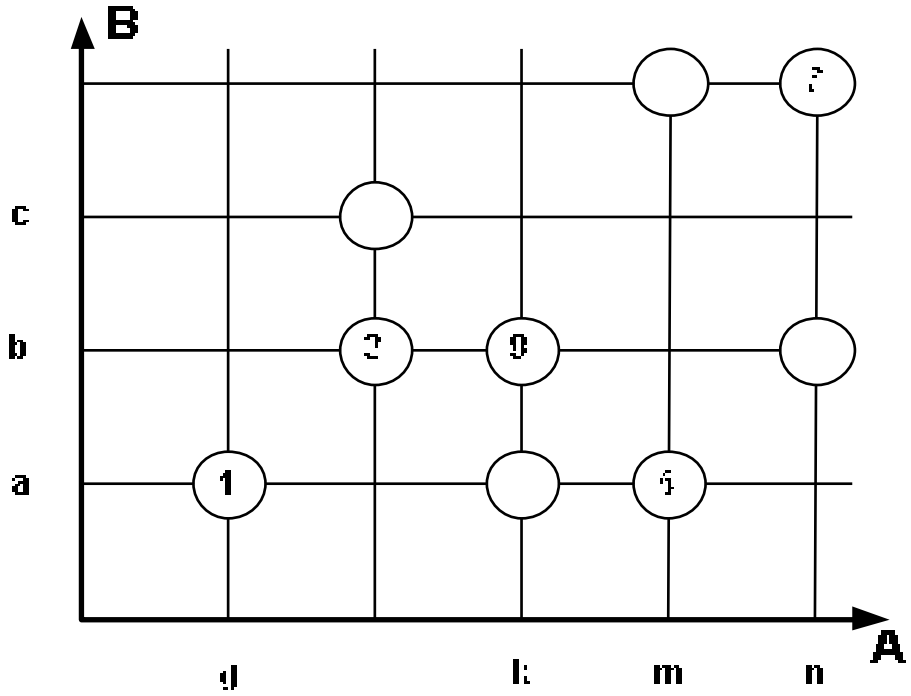


Рисунок 2. Процесс Z в двухпараметрическом пространстве

Пример операции проецирования приведен на рисунке 2. На нем показан исходный процесс Z, заданный в двухпараметрическом пространстве $S=\sigma(A)\times\sigma(B)$, где $\sigma(A)=\{g, h, k, m, n\}$, $\sigma(B)=\{a, b, c, d\}$. Ось времени не показана, однако значения моментов времени изменения состояний указаны в

кружочках, обозначающих соответствующее состояние. Таким образом, как видно из рисунка, множество T={1, 2, 3, 4, 5, 6, 7, 8, 9}.

График процесса F={<1,<g,a>, <2,<h,c,>, <3,<h,b>, <4,<k,a>, <5,<m,a>, <6,<m,d>, <7,<n,d>, <8,<n,b>, <9,<k,b>}

Построим процесс $Z_A = Пр_{S_A} Z$, являющийся проекцией процесса Z на пространство σ(A), в соответствие с алгоритмом выполнения операции проецирования.

1) Строим график $\overline{F}_A$={<1,g>, <2,h>, <3,h>, <4,k>, <5,m>, <6,m>, <7,n>, <8,n>, <9,k>}

2) Множество $\overline{F}_A$ уже упорядочено по времени. Получаем отображение процесса Z на пространство σ(A), показанное на рисунке 3.

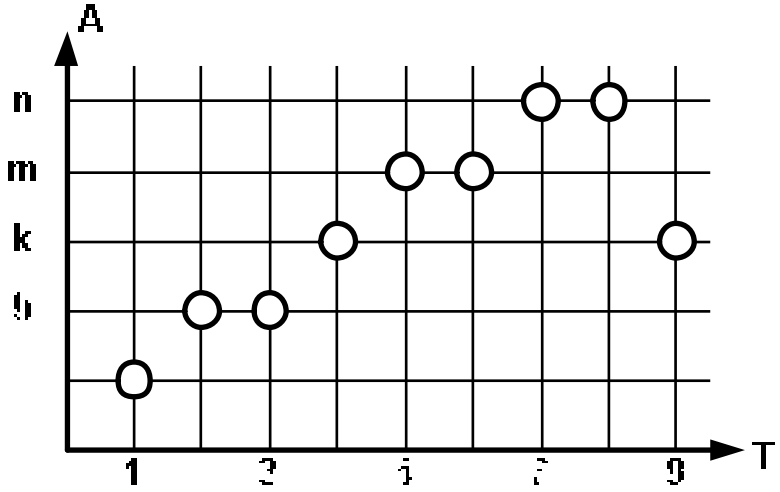


Рисунок 3. Отображение процесса Z на параметр A

3) На множестве $\overline{F}_A$ определяем классы эквивалентности: $К_1$ – точка 1, $К_2$ – точки 2,3, $К_3$ – точка 4, $К_4$ – точки 5,6, $К_5$ – точки 7,8, $К_6$ – точка 9.

4) Ставим в соответствие классу $К_2$ точку <2,h>, классу $К_4$ точку <5,m>, классу $К_5$ точку <7,n>.

5) Формируем график F_A={<1,g>, <2,h>, <4,k>, <5,m>, <7,n>, <9,k>}

6) Формируем множество времен изменения состояний T_A=<1,2,4,5,7,9>

Полученный в результате проведенных операций процесс $Z_A = Пр_{S_A} Z$ показан на рисунке 4.

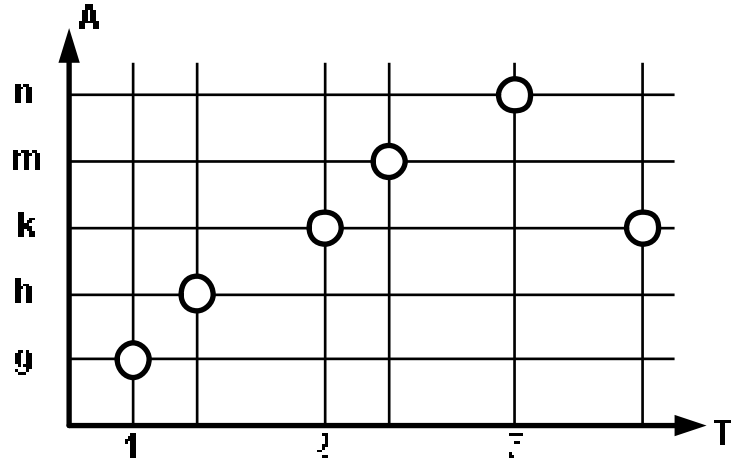


Рисунок 4. Процесс Z_A – проекция процесса Z на параметр A

Аналогично строится и проекция процесса Z на параметр B (пространство, состоящее из единственного параметра). Процесс Z_B показан на рисунке 5.

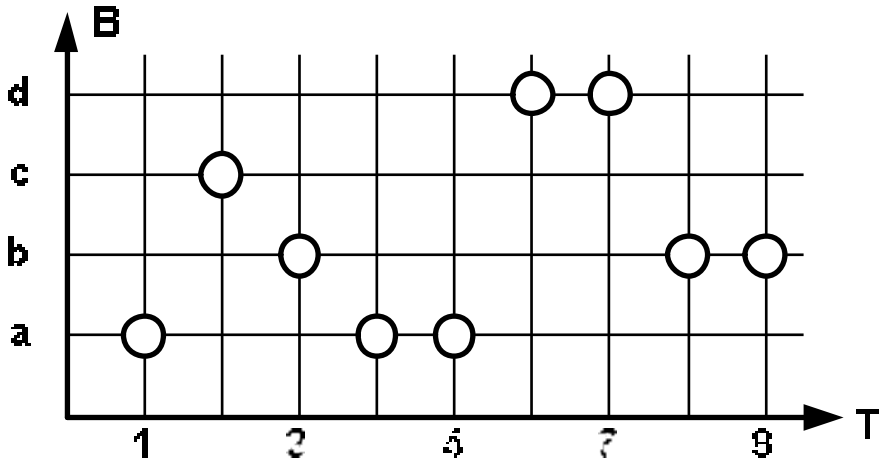


Рисунок 5. Отображение процесса Z на параметр B

После выполнения пункта 5 алгоритма получим результат, представленный на рисунке 6.

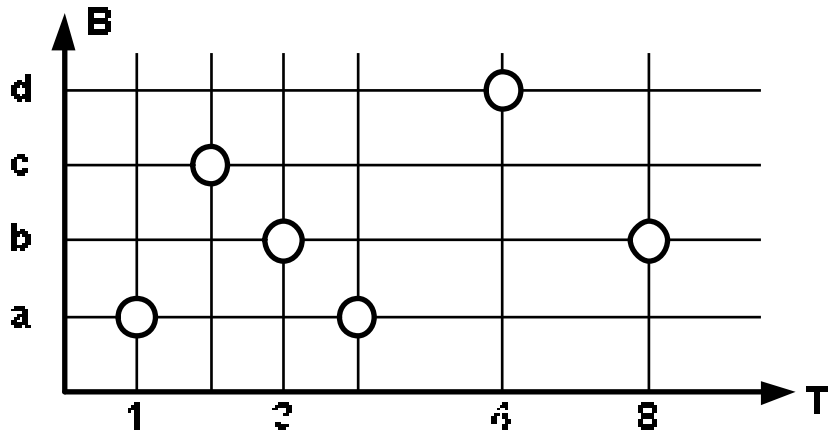


Рисунок 6. Процесс Z_B - проекция процесса Z на параметр B

Операция объединения

Пространство S_Q называется *склейкой* пространств S_{Q_1} и S_{Q_2}, если $Q=Q_1 \cup Q_2$. Интерес представляет случай непустого пересечения Q_1 и Q_2., когда пространства S_{Q_1} и S_{Q_2} имеют общую область.

Пусть кортеж s_1 принадлежит пространству S_{Q_1}, кортеж s_2 принадлежит пространству S_{Q_2}.Обозначим значения параметра q из кортежа s_1 как $\tilde{q}^{s_1}$, а значение параметра q из кортежа s_2 как $\tilde{q}^{s_2}$.

Тогда $s_1=<\ldots \underset{Q_1}{\tilde{q}^{s_1}} \ldots>$, $s_2=<\ldots \underset{Q_2}{\tilde{q}^{s_2}} \ldots>$.

Кортеж s является *левой склейкой* кортежей s_1 и s_2, когда:

$s_л= <\ldots \underset{Q_1}{\tilde{q}^{s_1}} \ldots \underset{Q \setminus Q_1}{\tilde{q}^{s_2}} \ldots>$.

Правая склейка определяется как:

$s_п= <\ldots \underset{Q \setminus Q_2}{\tilde{q}^{s_1}} \ldots \underset{Q_2}{\tilde{q}^{s_2}} \ldots>$.

Если $s_л=s_п$, то склейка называется *функциональной*.

Пусть заданы процессы $Z_1=<S_{Q_1}, T_1, F_1, \alpha_1>$ и $Z_2=<S_{Q_2}, T_2, F_2, \alpha_2>$. Процесс $Z=<S_Q, T, F, \alpha>$ является *объединением* процессов Z_1 и Z_2 (обозначение $Z= Z_1 \cup Z_2$), если:

- S_Q является склейкой пространств S_{Q_1} и S_{Q_2}
- $T= T_1 \cup T_2$
- для каждого $t \in T$ строится: $f_t=<t, s_t>$, где s_t – склейка кортежей s^1_t ($s^1_t \in S_{Q_1}$) и s^2_t ($s^2_t \in S_{Q_2}$), кортежи s^1_t и s^2_t принадлежат соответственно графикам F_1 и F_2 для значения t
- все склейки кортежей s^1_t и s^2_t для всех $t \in T$ являются функциональными
- совокупность f_t для всех $t \in T$ формирует график F
- отношение α строится как транзитивное замыкание на $\alpha_1 \cup \alpha_2$.

Процессы Z_1 и Z_2, допускающие операцию объединения, называются *согласованными*.

Можно доказать ряд утверждений (проведение доказательства предлагается выполнить самостоятельно).

Утверждение 1.

Два процесса Z_1 с пространством состояний S_{Q_1} и Z_2 с пространством состояний S_{Q_2} согласованы, если $Q_1 \cap Q_2 = \varnothing$.

Утверждение 2.

Пусть задан процесс Z_1 с пространством состояний S_{Q_1} и Z_2 с пространством состояний S_{Q_2}. Пусть в некоторой момент времени t состояние Z_1 равно $s_1 \in S_{Q_1}$, а состояние Z_2 равно $s_2 \in S_{Q_2}$. В общем случае будем полагать, что $Q_1 \cap Q_2 \neq \varnothing$. Обозначим $Q_3=Q_1 \cap Q_2$.

Если для всех моментов времени $t \in T$ *значения* $<...\underset{Q_3}{\tilde{q}}^{s_1}...>=<...\underset{Q_3}{\tilde{q}}^{s_2}...>$,

то процессы Z_1 *и* Z_2 *–согласованы.*

Утверждение 3.

Если $Z_1 = Пр_{S_{Q_1}} Z$ *и* $Z_2 = Пр_{S_{Q_2}} Z$, *то процессы* Z_1 *и* Z_2 *согласованы.*

Утверждение 4.

Пусть заданы процесс Z_1, определенный на интервале $\left[t_H^1, t_K^1\right]$, и Z_2, определенный на интервале $\left[t_H^2, t_K^2\right]$.

Если $\left[t_H^1, t_K^1\right] \cap \left[t_H^2, t_K^2\right] = \varnothing$, *то процессы* Z_1 *и* Z_2 *согласованы.*

Сформулированные выше утверждения позволяют выработать практически важные рекомендации по управлению процессами с тем, чтобы осуществить их объединение.

3. Система, объекты, задание процесса

Как уже было отмечено выше, мы полагаем заданным параметрическое множество системы Q, состоящее из параметров q_i (i=1..n). Если $\sigma(q_i)$ - множество значений, принимаемых параметром q_i., то пространство состояний системы S_Q определяется как $S_Q = \prod_{q_i \in Q} \sigma(q_i)$.

Будем рассматривать *объект*, как составную часть системы, характеризующуюся *параметрическим множеством объекта* $O_l \subset Q$. В дальнейшем для краткости, когда это не вызывает двусмысленности, вместо понятий *параметрическое множество системы* и *параметрическое множество объекта* будем говорить *система* и *объект* соответственно.

Пространство состояний S_{O_l} объекта O_l определяется аналогично системе, как $S_{O_l} = \prod_{q_i \in Q_l} \sigma(q_i)$. Будем предполагать, что система всегда имеет

полное разбиение на объекты. Таким образом: $\bigcup_{\forall l} O_l = Q$. Разбиение является *непересекающимся*, если $O_m \bigcap_{m \neq l} O_l = \varnothing$. В противном случае разбиение произведено на *пересекающиеся* объекты.

Если задан процесс Z_Q в системе, то процесс в объекте O_l может быть определен, как:

$$Z_{O_l} = \underset{S_{O_l}}{\boldsymbol{Пр}} Z_Q . \tag{3}$$

Как следует из утверждения 2: $\bigcup_{\forall O_l} Z_{O_l} = Z_Q$.

Пусть имеем объект O_l в системе Q. Тогда генерация процесса Z_{O_l} может быть выполнена путем задания оператора H^{O_l} [2]:

$$s_{t_i}^{O_l} = H^{O_l}\left(A^{O_l}, t_i, \omega\right), \tag{4}$$

где: $t_i \in T_{O_l}$;

A - множество аргументов: $A \subseteq Q$;

ω- случайное число.

Включение параметра ω позволяет задавать оператор от случайных значений аргументов, а также случайный выбор операторов.

Необходимо обратить внимание на то, что, если пространство состояния объекта определяется на параметрах $O_l \subset Q$, то множество аргументов является самостоятельным подмножеством Q : $A^{O_l} \subseteq Q$. Анализ соотношений между O_l, A^{O_l} и Q позволяют произвести следующую классификацию:

если $A^{O_l} \subseteq O_l$, то в объекте O_l развивается *локальный* процесс;

если $A^{O_l} \subset Q$, то процесс в O_l *частично зависимый от системы*;

если $A^{O_l} = Q$, то процесс в O_l *полнозависимый от системы*.

В ходе развития процесса совокупность аргументов во множестве A^{O_l} может изменяться. Таким образом, состав элементов множества A^{O_l} в общем случае зависит от времени. Обозначим эту зависимость как $A^{O_l}_{t_i}$.

Рассмотрим два объекта O_l и O_m в системе Q. Пусть $O_l \bigcap O_m = \varnothing$, а процессы в них заданы следующими операторами:

$$s^{O_l}_{t_i} = H^{O_l}\left(A^{O_l}_{t_i}, t_i, \omega\right); \qquad s^{O_m}_{t_i} = H^{O_m}\left(A^{O_m}_{t_i}, t_i, \omega\right). \tag{5}$$

Если $O_m \bigcap A^{O_l}_{t_i} = \varnothing$ и $O_l \bigcap A^{O_m}_{t_i} = \varnothing$, то такие процессы и объекты называются *не сцепленными* в момент времени t_i.

Если $O_l \bigcap A^{O_m}_{t_i} \neq \varnothing$, то объект O_m *сцеплен* с объектом O_l в момент времени t_i. То же относится и к их процессам. Это означает, что для определения состояния объекта O_m в момент времени t_i, необходимо знание состояния объекта O_l в это же время. Обозначим отношение сцепления как $O_\text{l} \to O_\text{m}$. Если $O_m \bigcap A^{O_l}_{t_i} \neq \varnothing$, то объект O_l *сцеплен* с объектом O_m в момент времени t_i: $O_\text{m} \to O_l$. Если одновременно $O_m \bigcap A^{O_l}_{t_i} \neq \varnothing$ и $O_l \bigcap A^{O_m}_{t_i} \neq \varnothing$, то объекты O_m и O_l *взаимно-сцеплены* в момент времени t_i: $O_\text{m} \leftrightarrow O_\text{l}$. При операторном способе описания процессов всегда нежелательна модель, приводящая к появлению взаимно - сцепленных объектов, поскольку возникающую неопределенность приходится раскрывать путем решения в общем случае систем нелинейных уравнений, что может привести к непреодолимым трудностям. *В дальнейшем будем стремиться создавать модели, не приводящие к взаимному сцеплению объектов.*

Не следует смешивать отношение сцепления и зависимости. Так, если $O_\text{m} \to O_\text{l}$ и $O_\text{l} \to O_\text{k}$, то вовсе не обязательно, чтобы $O_\text{m} \to O_\text{k}$. Таким образом, отношение сцепления не является транзитивным.

Если к отношению сцепления добавить полное транзитивное замыкание, то получим *отношение зависимости*. Т.е. если O_l зависит от O_m, а O_k зависит от O_l, то O_k зависит и от O_m. Таким образом, *отношение сцепления можно определить как отношение непосредственной зависимости.*

4. Алгоритмическая модель процесса

Конечно, было бы желательно иметь возможность задания процесса в виде единого оператора (4), однако это, как правило, либо затруднительно, либо невозможно. Ниже предлагается описать оператор *H* в виде некоторой алгоритмической структуры.

Рассмотрим *дискретный во времени* процесс Z. Пространство состояний S может быть как непрерывным, так и дискретным. Поставим в соответствие каждой i-ой точке процесса (момент времени изменения состояния t_i) некоторый оператор h_i^c. Оператор h_i^c вычисляет значение состояния $s_i \in S$ в момент времени t_i:

$$s_i = h_i^c\left(A_i, t_i, \omega\right). \tag{6}$$

Оператор h_i^c описывает вычисление только одной i-й точки процесса *Z*. В силу этого условия будем в дальнейшем называть этот оператор *элементарным*.

Таким образом, если график процесса содержит *n* точек, то мы должны задать линейную последовательность элементарных операторов:

$$h_1^c, h_2^c \ldots, h_i^c, \ldots h_n^c. \tag{7}$$

Введем новый элемент модели - *инициатор*. Первоначально будем полагать, что инициатор - это объект, обладающий следующими фундаментальными свойствами:

а) *независимостью*: может существовать самостоятельно без операторов;

б) *динамичностью*: инициатор имеет возможность перемещаться от оператора к оператору; будем называть попадание инициатора на оператор *сцеплением инициатора с элементарным оператором*;

в) *инициативностью*: в момент сцепления инициатора с оператором происходит *выполнение (инициирование) элементарного оператора*, что соответствует вычислению нового состояния процесса.

Будем в дальнейшем полагать, что выполнение элементарного оператора происходит *мгновенно*. Это ограничение не сужает применимости предлагаемой модели, поскольку, если необходимо описать процесс, где вычисление нового состояния требует затрат реального времени, то можно ввести два элементарных оператора, ограничивающих начальный и конечный момент времени интервала вычислений. Таким образом, описание процесса может быть выполнено путем задания линейной последовательности операторов $\left\langle h_i^c \right\rangle_{i=1}^{n}$ и перемещения по этой последовательности инициатора ***I***, сцепляющегося с элементарными операторами h_i^c в заданные моменты времени t_i изменения состояния процесса.

Предлагаемая модель описания процесса предполагает, что *моменты сцепления инициатора с элементарными операторами определяют сами элементарные операторы.* С этой целью введем *оператор условия сцепления инициатора* h_i^y, который определяет условие, при выполнении которого инициатор сцепляется со следующим оператором h_{i+1}^c. Возможны следующие варианты задания такого условия:

а) указание момента времени сцепления инициатора с оператором h_{i+1}^c;

б) определение логического условия, при выполнении которого инициатор сцепляется с оператором h_{i+1}^{c};

в) комбинированная форма, включающая варианты а) и б).

Таким образом:

$$h_i^y \in \left\{h_i^t, h_i^л, h_i^{t,л}\right\}, \tag{8}$$

где: h^y_{i} - оператор условия сцепления;

h^t_{i} - оператор временного условия (соответствует варианту а);

$h^л_{\text{i}}$ - оператор логического условия (соответствует варианту б);

$h^{t,л}_{\text{i}}$ - оператор комбинированного условия (соответствует варианту в).

Расширим понятие элементарного оператора, добавив к нему помимо оператора h^c_{i} оператор h^y_{i}. Таким образом, определим элементарный оператор h_{i}, как двойку:

$$h_i = \left\langle h_i^c, h_i^y \right\rangle. \tag{9}$$

При сцеплении инициатора с элементарным оператором h_{i} происходит мгновенное выполнение его обеих составных частей: выполнение h^c_{i} позволяет вычислить новое состояние s_{i} процесса Z, а выполнение оператора h^y_{i} дает возможность определить момент времени, либо логическое условие сцепления инициатора со следующим элементарным оператором h_{i+1}.

Теперь можно определить понятие *алгоритмической модели процесса* (в дальнейшем АМП), в виде тройки:

$$\text{АМП} = \left\langle \{h_i\}_{i=1}^{n}, \beta, I \right\rangle, \tag{10}$$

где: $\{h_i\}_{i=1}^{n}$ - множество элементарных операторов;

β - линейный порядок на $\{h_i\}_{i=1}^{n}$;

I - инициатор.

Следует обратить внимание, что АМП содержит один и только один инициатор, т.е. *каждому процессу соответствует один инициатор*. В этом смысле инициатор является представителем процесса, при его потери либо отсутствии развитие процесса прекращается. Если в системе развивается одновременно *m* процессов, то в модели присутствует *m* инициаторов.

Линейную последовательность элементарных операторов назовем *треком TR*:

$$TR = \left\langle \{h_i\}_{i=1}^{n}, \beta \right\rangle. \qquad (11)$$

Таким образом, можно АМП определить также как двойку:

$$\text{АМП} = <TR, I>. \qquad (12)$$

Процесс задан, если задан трек элементарных операторов и инициатор.

5. Понятие структуры

Пусть задан некоторый трек TR. В реальных приложениях трек содержит достаточно много элементарных операторов, выполняющих одни и те же операции над аргументами. Согласно [3] операторы эквивалентны, если при одних и тех же значениях аргументов они вычисляют одинаковые результаты. Введение понятия эквивалентных операторов позволяет задать отношение эквивалентности на множестве $\{h_i\}_{i=1}^{n}$ элементарных операторов трека *TR*.

Структурой назовем свертку трека *TR* по отношению эквивалентности элементарных операторов.

Пример. Пусть задан некоторый трек TR (рисунок 7).

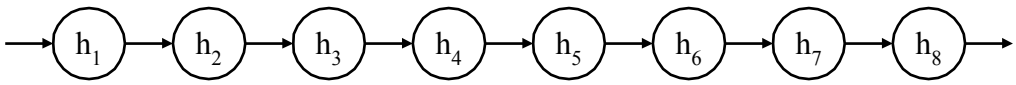


Рисунок 7. Пример трека

Пусть отношение эквивалентности элементарных операторов имеет вид:

$$\{(h_1, h_3),(h_2, h_5, h_6, h_8),(h_4, h_7)\}. \quad (13)$$

Тогда структура имеет вид графа (рисунок 8).

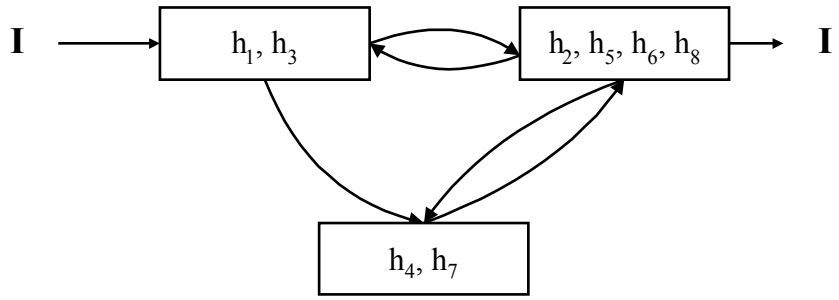


Рисунок 8. Свертка трека

Очевидно, если заданы трек и отношение эквивалентности операторов, то всегда возможно построение структуры. Однако обратное восстановление трека по структуре является неоднозначной операцией. Эта операция относится к классу операций развертки. С тем, чтобы операцию построения трека из структуры сделать однозначной, введем еще один тип элементарного оператора - *навигационный* элементарный оператор. Навигационный оператор определяется так же, как и элементарный оператор, однако в результате его выполнения определяется тот элементарный оператор в структуре, который должен выполняться следующим. Выполнение навигационного оператора инициируется инициатором. Поскольку время на выполнение навигационного оператора, как и всех элементарных операторов, равно нулю, то использование его не сказывается на времени реализации процесса. В общем случае навигационный оператор должен следовать за каждым элементарным оператором в структуре.

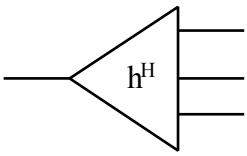


Рисунок 9. Обозначение навигационного оператора

Если навигационный оператор обозначить, как показано на рисунке 9, то структура из вышеописанного примера будет иметь вид (рисунок 10):

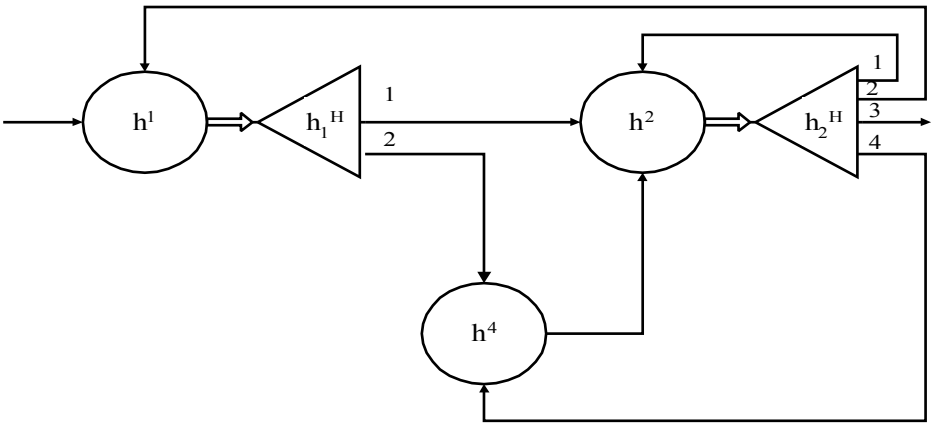


Рисунок 10. Структура трека

Здесь операторы h_1, h_2, h_4 являются представителями своего класса эквивалентности. Как видно, после операторов h_1 и h_2 стоят навигационные операторы $h^{н}_1$ и $h^{н}_2$, в то время как после оператора h_4 нет необходимости в использовании навигационного оператора. Навигационный оператор используется также и для организации циклов (оператор $h^{н}_2$, выход 1).

Использование структуры по сравнению с треком позволяет значительно снизить размерность описания процесса. Однако необходимо иметь в виду, что процесс определен только в случае задания трека, а поэтому структура есть лишь способ более компактного описания трека, генерация самого трека остается необходимой операцией. На практике задание структуры с

навигационными операторами для последующей генерации трека используется часто и повсеместно, где необходима генерация процесса.

Определение элементарного оператора в составе структуры можно представить в виде: $h=<h^c,h^y,h^н>$.

Особый интерес представляет случай, когда структура имеет вид полнодоступного графа (рисунок 11).

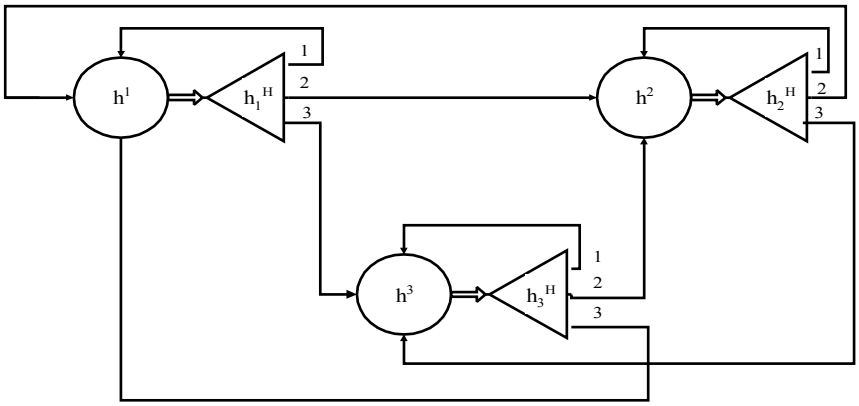


Рисунок 11. Пример полнодоступной структуры

Здесь возможна генерация любого трека на базе эквивалентных классов элементарных операторов h_1, h_2, h_3. Если объединить все навигационные операторы $h^н_1$, $h^н_2$, $h^н_3$ в один $h^н$, то получим граф вида (рисунок 12).

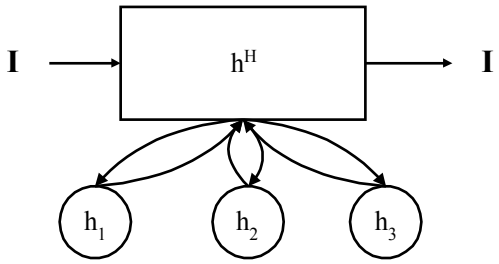


Рисунок 12. Свертка полнодоступной структуры

Как видно из примера, полнодоступная структура может быть описана двухуровневым деревом, в котором верхний уровень представляет собой

объединенный навигационный оператор, а второй уровень содержит не связанные между собой элементарные операторы. Простота полнодоступной структуры приводит часто к ее использованию для описания структур, не имеющих вид полнодоступного графа. Некоторые проблемы при таком описании возникают лишь при задании алгоритма объединенного навигационного оператора $h^н$, однако можно предложить достаточно много способов его реализации.

6. Операторно-параметрическая схема

Элементарный оператор h оперирует с параметрами и изменяет состояние объекта. По отношению к оператору параметры могут быть входными, выходными и рабочими (рисунок 13).

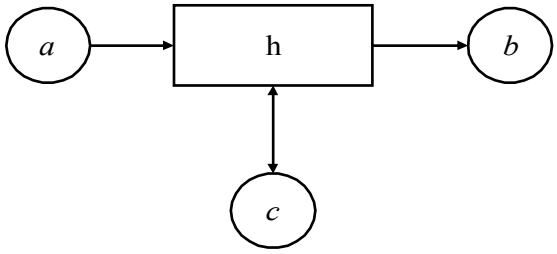


Рисунок 13. Отношение параметра к оператору

где: *a* - входной параметр;

b - выходной параметр;

c - рабочий параметр.

Входной параметр означает его принадлежность к множеству *A*, выходной - к формированию состояния *s* , что говорит о его принадлежности к множеству O, рабочий - к тому и другому множеству одновременно.

Если на треке операторов указать используемые этими операторами параметры и их взаимосвязи, то *получим операторно-параметрическую схему*. Пример такой схемы приведен на рисунке 14. Двойными линиями показан путь инициатора, а одинарными - отношение параметров к

операторам. Такие схемы дают наглядную картину взаимодействия параметров в ходе реализации процесса.

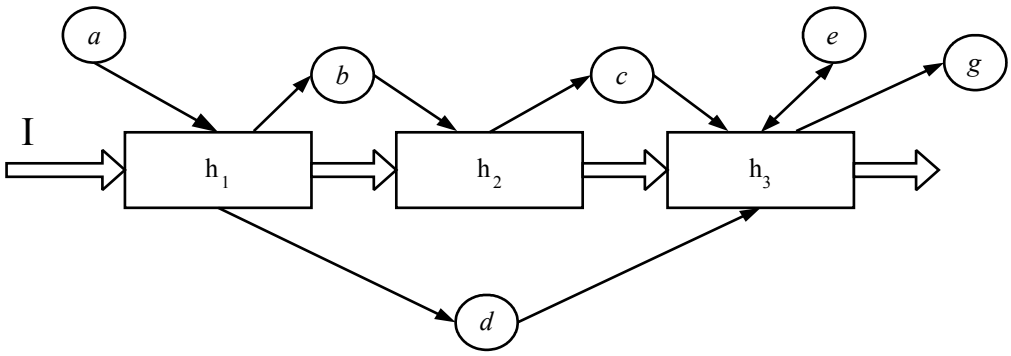


Рисунок 14. Операторно-параметрическая схема

Из схемы видно, что трек состоит из трех элементарных операторов и, следовательно, описывает три точки процесса. Из схемы следует, что для оператора h1 множества A1={a}, O1={b,d}; для оператора h2 множества A2={b}, O2={c}; для оператора h3 множества A3={d,c,e}, O3={e,g}. Оператор h2 сцеплен с оператором h1, поскольку параметр b является выходным для h1 (принадлежит A1) и входным для h2 (принадлежит O2), h3 сцеплен с h1 и h2 через параметры d и c соответственно.

Как видно из приведенного примера, операторно-параметрическая схема дает достаточно полное представление о способе описания процесса с использованием АМП.

7. Описание подобных процессов

Рассмотрим случай задания двух достаточно близких по описанию процессов Z_1 (трек A) и Z_2 (трек B) (рисунок 15). И в том и в другом треке используются операторы h_1 и h_2, но они взаимодействуют с разными параметрами как входными, так и выходными. Было бы желательно найти способ объединения описаний таких процессов. Для решения поставленной задачи дополним определение инициатора, добавив к его фундаментальным

свойствам возможность включать в себя параметры. Таким образом, инициатор наряду с фундаментальными свойствами приобретает некоторое “тело” в виде совокупности параметров. Параметры в этой совокупности должны быть упорядочены. Назовем эту совокупность *локальной средой* процесса. Будем в дальнейшем считать, что “тело” инициатора представляет собой ссылку на локальную среду. Таким образом, движение инициатора по треку есть движение ссылки на локальную среду по треку, а сама локальная среда может быть неподвижна. Поскольку процесс и его инициатор связаны между собой взаимно-однозначно, то мы в дальнейшем будем рассматривать выражения “локальная среда процесса” и “локальная среда инициатора” как синонимы.

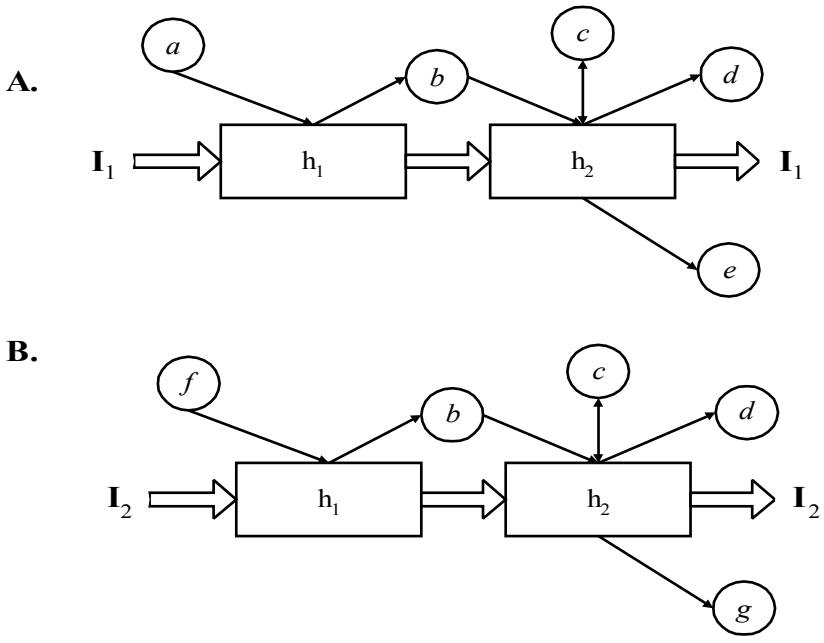


Рисунок 15. Пример подобных описаний процессов

Тогда можно предложить следующую схему свертки описаний двух процессов в одно общее описание (рисунок 16):

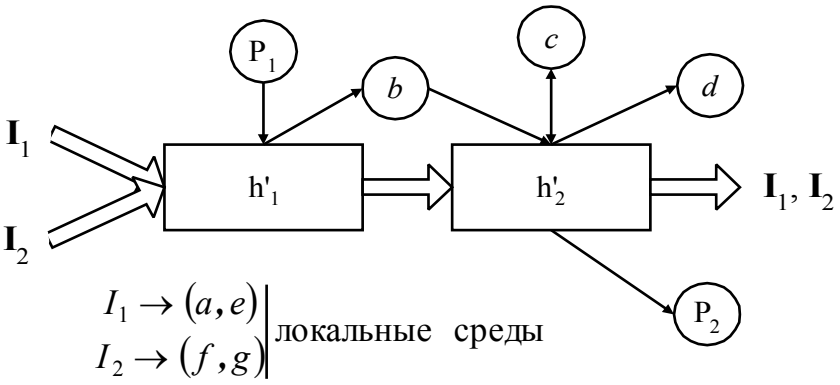


Рисунок 16. Объединенное описание процессов Z_1 и Z_2

Как видно из рисунков, с инициатором $\mathbf{I}_1$ связана локальная среда (a, e), а с инициатором $\mathbf{I}_2$ - локальная среда (f, g). Оператор h_1 модифицирован в оператор h'_1, который связан с параметром b и первым параметром локальной среды инициатора. Оператор h'_2 связан с параметрами b, c, d и вторым параметром локальной среды инициатора. Операторы h'_1 и h'_2 будем называть *объединенными*. Отличие объединенных элементарных операторов от простых элементарных операторов состоит в возможности взаимодействовать не только с явно заданными параметрами, но и с параметрами локальных сред инициаторов. Инициаторы I_1 и I_2 присутствуют в этой схеме *одновременно*.

Очевидно, эти рассуждения могут быть распространены на случай ***n*** параллельно протекающих процессов. Процессы, сгенерированные треком или структурой, использующими объединенные элементарные операторы и локальные среды называются *подобными*.

Таким образом, удалось еще более снизить размерность описания множества процессов, введя *отношение подобия* процессов. Для описания совокупности подобных процессов достаточно иметь *одно* объединенное описание трека или структуры и *множество* одинаково структурированных локальных сред, привязанных к инициаторам.

8. Обобщенные операторы, вложенность, блоки

На треке можно задать некоторое *плотное* разбиение элементарных операторов на подмножества. Это разбиение обычно выполняется с целью получения функционально однородных подмножеств операторов. Совокупность операторов, входящих в одно подмножество, назовем *обобщенным* оператором.

Пример разбиения приведен на рисунке 17. Здесь подмножество операторов h_1, h_2, h_3 объединено в один обобщенный оператор H_1, а h_4, h_5 - в обобщенный оператор H_2. В результате получаем трек обобщенных операторов.

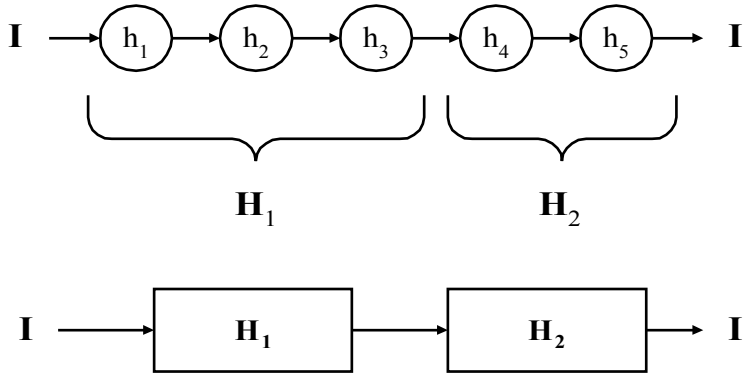


Рисунок 17. К понятию обобщенного оператора

Плотное разбиение может быть выполнено и на треке обобщенных операторов. Рекурсивность определения позволяет вводить понятия обобщенных операторов различного уровня: первого, второго и т.д.

Будем называть трек $\langle\ h_1\ ,\ h_2\ ,\ h_3\ \rangle$ *вложенным* в оператор H_1. Если сцепление инициатора с элементарным оператором вычисляло одну точку процесса, то сцепление инициатора с обобщенным оператором порождает подпроцесс, равный процессу сцепления инициатора со всеми операторами вложенного трека. Это утверждение следует из требования плотности разбиения. Отсюда следует, что *время сцепления инициатора с обобщенным*

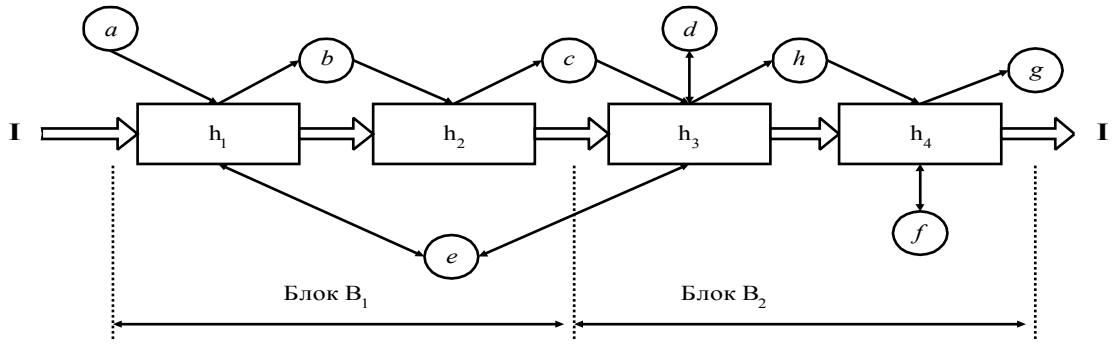
оператором равно времени прохождения инициатора по вложенному треку. Это свойство называют также *временной вложенностью.* Если вложенный трек свернут в структуру, то, в общем случае, будем говорить о *вложенных описаниях*. Таким образом, используя рекурсивность разбиений треков, формируется *многоуровневая система вложенных описаний*, обладающая свойством временной вложенности для всех обобщенных операторов на всех смежных уровнях. Переход к описанию обобщенными блоками соответствует *операции свертки* процессов.

Использование обобщенных операторов позволяет еще более снизить размерность описания процессов, однако необходимо иметь в виду, что для генерации процесса мы должны уметь выполнить *операцию развертки* для каждого обобщенного оператора, определив все вложенные треки. В ходе анализа процессов функционирования системы удобно использовать вложенность процессов описаний обобщенных операторов достаточно высокого уровня. Однако в ходе реализации процесса необходимо иметь возможность последовательной развертки обобщенных операторов верхних уровней в треки операторов нижних уровней вплоть до получения треков элементарных операторов.

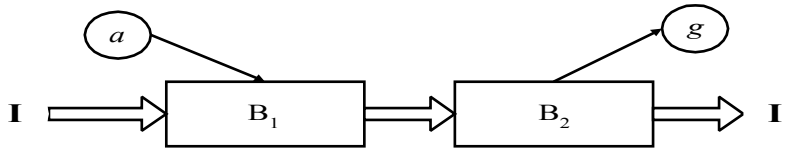
Совокупность обобщенного оператора и связанных с ним параметров образует *блок*. Какие параметры включаются в состав блока, зависит от постановки задачи моделирования и заранее никак не регламентируется.

На рисунке 18 приведен пример построения блока. На рисунке 18.а показана операторно-параметрическая схема трека элементарных операторов. На этой схеме выполнено разбиение на два блока B_1 и B_2. Как видно из рисунка18.в, к блоку B_1 отнесены операторы h_1 и h_2, а также параметры *b, c, e.* К блоку B_2 относятся операторы h_3, h_4 и параметры *d, h, f* (рисунок

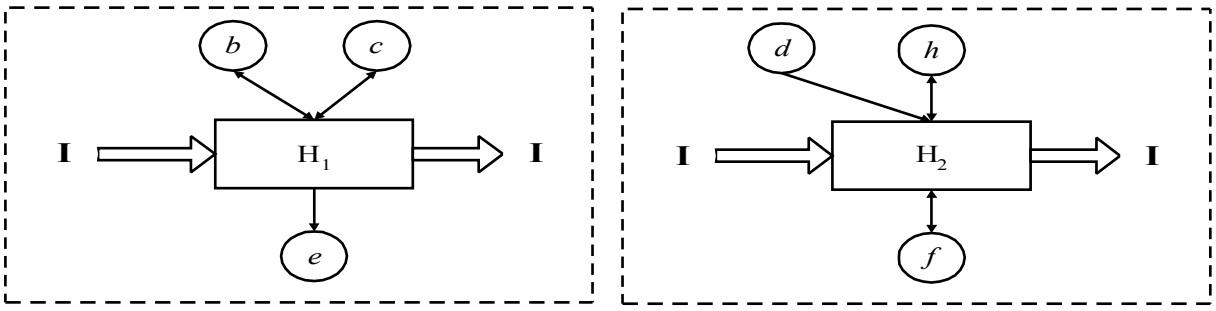
18.г). Параметры *a* и *g* не принадлежат ни одному из блоков. Полученная блочная схема приведена на рисунке 18.б.



а) операторно-параметрическая схема



б) блочная схема



в) г)

Рисунок 18. К понятию блока

Любой параметр по отношению к заданному блоку может быть *внутренним*, если включен в состав блока, либо *внешним*, если не принадлежит блоку. Так, параметр *a* - внешний по отношению к блоку B_1. Внутренние параметры, в свою очередь, могут быть *входными* (параметр *d* блока B_2), *рабочими* (параметры *h, f* блока B_2 и параметр *b* блока B_1) и *выходными* (параметр *e* блока B_1).

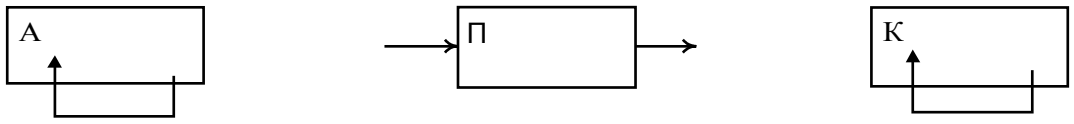
В результате разбиения операторно-параметрической схемы и последующей операции свертки в блоки, получим *блочную схему* (рисунок 18.б).

Таким образом, можно определить *блок общего вида* как *структуру* и связанную с операторами структуры *совокупность параметров*. Процесс в блоке начинает развиваться, когда в блок поступает инициатор. Принципиально будем различать два типа блоков: *агрегат* и *процессо*р.

Агрегат.

На практике часто возникает необходимость описывать процесс функционирования некоторой машины по преобразованию значений параметров в соответствие с заданным циклическим алгоритмом. Такой процесс можно описать с помощью блока общего вида, в котором существует один инициатор, а трек циклически замкнут. Блок, в котором развивается *один единственный циклический процесс*, будем называть *агрегатом*, или *А*-блоком.

Будем в дальнейшем считать, что инициатор находится в агрегате по определению, что избавляет нас от необходимости разработки процедур его доставки в блок. Таким образом, агрегат содержит единственный инициатор и трек элементарных операторов, замкнутый внутри блока. Поступление каких-либо внешних инициаторов в агрегат невозможно. Будем обозначать агрегат на схемах в виде (рисунок 19.а).



а) агрегат б) процессор в) контроллер

Рисунок 19. Обозначения блоков

Обмен между агрегатом и другими блоками возможен исключительно посредством параметров. Поскольку в агрегате развивается единственный процесс, то *не имеет смысла использовать в нем локальную среду и объединенные операторы.*

Таким образом, агрегат представляет собой некоторую структуру, порождающую замкнутый трек, и включает единственный внутренний инициатор, находящийся в блоке по определению.

Процессор.

Блок, предназначенный для генерации процессов, инициаторы которых являются внешними по отношению к блоку, называется *процессором*, или *П*-блоком. Инициаторы, поступившие извне, сцепляются с блоком, порождая процессы, и затем покидают его. Поскольку процессор генерирует множество одновременно протекающих процессов, в нем используются исключительно объединенные элементарные операторы, а инициаторы должны содержать локальные среды. Таким образом, процессор представляет собой описание произвольной структуры, содержащей объединенные операторы. Процессы порождаются в этом блоке лишь при поступлении в него извне инициаторов, содержащих локальные среды. Из вышесказанного следует, что *процессор порождает параллельно протекающие во времени подобные процессы.*

Будем обозначать процессор на схемах в виде (рисунок 19.б).

Контроллер.

Рассмотрим вновь блок типа агрегат. Как было показано выше, он не имеет возможности взаимодействовать с внешними инициаторами. С тем, чтобы снять это ограничение, введем над инициаторами операции пассивизации и активизации.

Операция *пассивизации* переводит инициатор в класс обычных параметров.

Операция *активизации*, наоборот, обычный параметр переводит в класс инициаторов.

Если агрегат содержит операторы, выполняющие указанные операции, то такой агрегат назовем *контроллером*, или *К*-блоком. *Контроллер, таким образом, представляет собой агрегат, выполняющий операции над внешними инициаторами в соответствии с собственным алгоритмом функционирования*. Операции над инициаторами суть операции над процессами. Таким образом, контроллер исполняет роль управляющего звена в некоторой блочной схеме. Будем обозначать контроллер на схемах как (рисунок 19.в).

9. Конфликты на ресурсах

Процессы Z_i в системе Q развиваются *параллельно*. Это значит, что они изменяют значения параметров системы в течение одного и того же интервала времени. Достаточно типичны ситуации, когда по логике функционирования системы накладываются ограничения на изменение некоторых параметров несколькими процессами одновременно в течение заданного либо обусловленного интервала времени.

Совокупность параметров системы, на изменение которых сформулированы некоторые ограничивающие условия, называется *ресурсом* R. Таким образом, $R \subset Q$. Если объект O_k изменяет параметры ресурса R, то $R \subset O_k$.

Захват ресурса R процессом Z означает получение разрешения процессу Z изменять значения параметров $q \in R$.

Конфликт на ресурсе есть возникновение ситуации, когда тому или иному процессу отказано в захвате ресурса до момента выполнения некоторого наперед заданного условия. Из определения ресурса следует, что конфликт на ресурсе возможен лишь для пересекающихся объектов. Таким образом, необходимо добиться *согласования* процессов в этих объектах. Рассмотрим следующие способы разрешения конфликтных ситуаций.

А. **Синхронизация**. Утверждение 4 предлагает наиболее универсальный способ построения согласованных процессов: разнесение во времени их интервалов определения. При этом способе задаются периодически повторяющиеся интервалы времени захвата ресурса для каждого претендующего на него процесса. На рисунке 20 показан пример выделения таких интервалов для случая конфликта трех процессов.

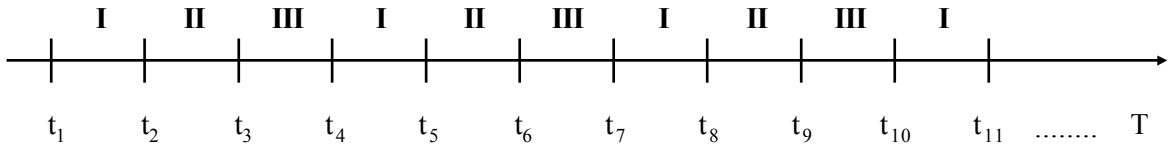


Рисунок 20. Синхронизация процессов

Процесс Z_1 может захватывать ресурс R лишь в интервалах (t_1,t_2), (t_4,t_5), (t_7,t_8) и т.д., процесс Z_2 - в интервалах (t_2,t_3), (t_5,t_6), (t_8,t_9) и т.д., процесс Z_3 - в интервалах (t_3,t_4), (t_6,t_7), (t_9,t_{10}) и т.д. Захват ресурса возможен на период одного интервала. Этот способ широко используется, например, в электронике для синхронизации параллельных процессов, причем задание временных интервалов осуществляется путем определения серий синхросигналов. Операционные системы разделения времени (СРВ) используют этот метод, как базовый.

Б. **Семафоры**. Если условие захвата ресурса не ограничивает время использования этого ресурса захватившим его процессом, то в этом случае удобно использовать семафоры. Понятие семафора впервые было введено в работе [3]. Семафор есть простая логическая переменная, однозначно соответствующая ресурсу. Значение семафора ‘0’ означает, что ресурс может быть захвачен процессом, значение семафора ‘1’ блокирует захват ресурса. На рисунке 21 показан пример использования семафора С при захвате ресурса R двумя процессами Z_1 и Z_2.

Применение семафоров для управления захватом ресурсов широко используется в системах управления, в частности, в операционных системах .

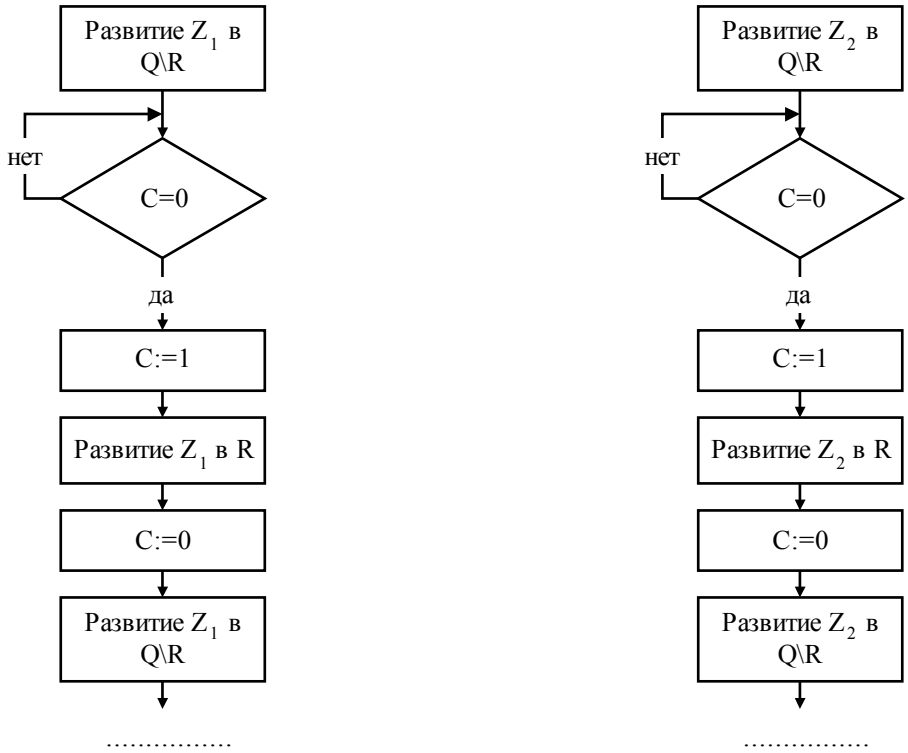


Рисунок 21. Применение семафора

В. **Контроллеры**. Наиболее общий способ управления процессами при захвате ресурсов состоит в создании соответствующих К-блоков. Так, для

примера двух процессов может быть предложена следующая блочная схема (рисунок 22).

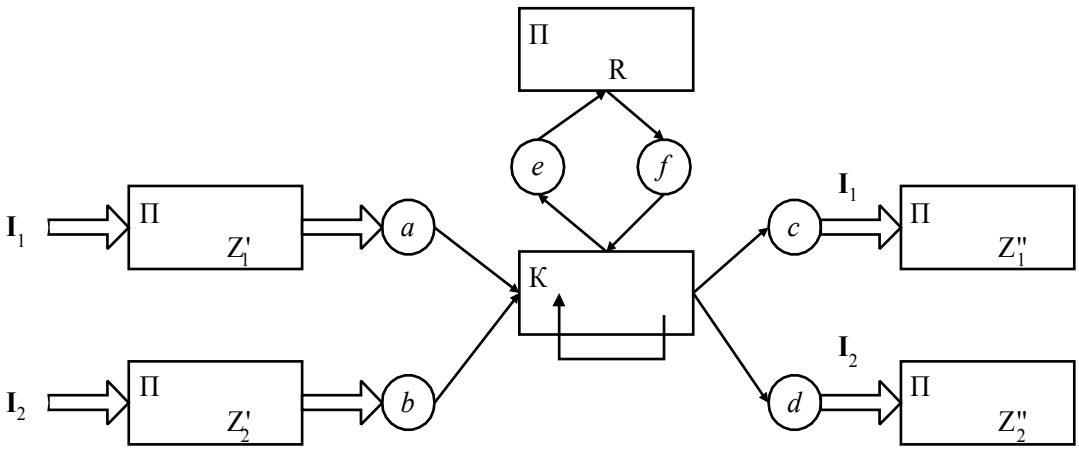


Рисунок 22. Применение *К*-блоков при разрешении конфликтов на R

Здесь первые П-блоки реализуют процессы Z_1’ и Z_2’ в пространстве $Q\backslash R$; затем производится пассивизация инициаторов I_1 и I_2, они переводятся в параметры *a* и *b* соответственно. *К*-блок рассматривает ситуацию с входными параметрами в соответствие с собственным алгоритмом. Приняв решение о захвате ресурса каким-либо процессом, *К*-блок передает *a* или *b* в параметр *e* и активизирует его, отсылая в *П*-блок ресурса *R*. После завершения процесса в *R* выдается сигнал в параметре *f*, в ответ на который *К*-блок передает параметр- инициатор в *c* либо *d* и активизирует его, отсылая на продолжение процесса Z_1’’ либо Z_2’’ в соответствующие *П*-блоки. Использование *К*-блоков является наиболее универсальным способом управления множеством процессов при захвате ресурсов.

10. Схемы описаний функционирования системы

При описании функционирования системы обычно строится блочная схема, задающая логику взаимодействия, пересечения, уничтожения множества параллельно существующих процессов. Рассмотренная в

настоящей главе формализованная модель описания процессов дает возможность с единых позиций провести классификацию существующих подходов в этой области.

Агрегативная схема. Агрегативная схема предполагает использование только *А*-блоков. Как показано выше, в такой схеме взаимодействие может осуществляться лишь через параметры. Пример агрегативных моделей рассмотрен в [2], где показано, что каждый *А*-блок в такой модели может представляться некоторым конечным автоматом.

Если функционирование системы может быть описано совокупностью конечных автоматов, взаимодействующих между собой через множество входных и выходных параметров, то применение агрегативных схем представляется наиболее рациональным. К таким системам можно отнести электронные схемы, блочные схемы функционирования АСУ и т.п.

Процессная схема. Если в основе описания функционирования системы лежит применение *П*-блоков, то такое описание называется *процессным*. Взаимодействие в процессных схемах осуществляется, в основном, через локальные среды процессов. Типичным примером процессного подхода является система имитационного моделирования SIMULA-2, включенная впоследствии в язык SIMULA-67. По существу процесс функционирования системы разбивается на совокупность пересекающихся подобных процессов в объектах. Для каждой такой совокупности формируется свой П-блок.

Процессный подход наиболее эффективен, когда имеем дело с множеством явно выраженных локальных процессов, например, при описании биологических, экономических, социальных и т.п. систем.

Агрегативно-процессные схемы. Наиболее распространен подход при описании функционирования сложной системы с применением всех видов блоков - А, П и К. Примерами могут быть системы моделирования GPSS,

SOL, СЛЕНГ, ДИС и другие. В каждой из них существуют свои ограничения на алгоритмы этих блоков. Так, в языке GPSS существуют операторы-агрегаты (GENERATE, TERMINATE), операторы навигационного типа (TRANSFER), объекты-контроллеры. Большинство остальных операторов имеют тип процессоров. В языке GPSS введены инициаторы в явном виде (TRANSACT), доступные пользователю и способные создавать локальные среды (параметры транзактов). Однако, в языке GPSS пользователь лишен возможности задавать алгоритмы блоков и использует лишь библиотечные конструкции. В языках SOL и СЛЕНГ была сделана попытка устранить эти недостатки.

Потоковая схема. Если в блочной схеме общего вида ограничиваться использованием блоков с простыми алгоритмами, а саму схему изобразить в виде сети А - блоков, где линии связи соответствуют движению инициаторов, то получим *потоковую* схему. Такая схема отражает движение *потоков инициаторов* между блоками. Поскольку в такой схеме нет процессоров, то инициаторы, с одной стороны, являются носителями последовательности выполнения операторов в своих процессах, а с другой стороны, инициируют процессы в агрегатах, с которыми они сцепляются. Учитывая, что алгоритмы блоков достаточно просты, то построение потоковых схем может быть выполнено без особых трудностей. Однако необходимо иметь в виду, что при этом существенно возрастает размерность самого описания, нередко теряется обозримость схемы в целом. Примерами таких схем являются сети Петри, Наура, Е-сети.

Сети массового обслуживания. Сети массового обслуживания являются особым видом потоковых схем, когда все процессы пересечены на ресурсах. Разрешение конфликтных ситуаций в этих схемах выполняется с помощью контроллеров, которые называются системами массового

обслуживания (СМО). *Система массового обслуживания есть объединение К-блока и одного или нескольких А-блоков: К-блок реализует дисциплину обслуживания требований, а А-блоки - процесс изменения параметров ресурса*.

При исследовании систем массового обслуживания аналитическими методами выбираются достаточно простые алгоритмы *К*-блоков, а алгоритмы *А*-блоков сводятся, как правило, к задержке инициатора на некоторое время в ресурсе. Такие системы могут содержать единственный *К*-блок, совмещающий вышеуказанные действия.

Таким образом, формализация функционирования систем в виде сети массового обслуживания возникает в том случае, если в схемах общего вида отсутствуют независимые процессоры, а все процессы пересечены на ресурсах. Поток требований на каждый контроллер есть не что иное, как поток инициаторов процессов, а каждое отдельное требование, будучи инициатором, определяет процесс.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

ОПРЕДЕЛЕНИЯ

1. Система, объект
2. Процесс, подпроцесс
3. Оператор общего вида описания процесса
4. Определение сцепленности объектов
5. Алгоритмическая модель процесса
6. Элементарный оператор
7. Трек
8. Структура
9. Операторно-параметрическая схема
10. Объединенный элементарный оператор
11. Локальная среда процесса
12. Агрегат
13. Процессор
14. Контроллер
15. Ресурс, конфликт на ресурсе

ОБЪЯСНИТЬ РАЗДЕЛ

1. Операция свертки процессов
2. Операция развертки процессов
3. Операция проецирования процесса.
4. Операция сложения процессов
5. Классификация операторов общего вида
6. Отношение сцепленности объектов, отношение зависимости операторов
7. Эквивалентные операторы, структура, виды структур

8. Однородные процессы. Понятие объединенного элементарного оператора. Локальные среды процессов.

9. Ресурсы, конфликты на ресурсах. Разрешение конфликтов с помощью «семафора». Преимущества и недостатки метода. Примеры.

10. Ресурсы, конфликты на ресурсах. Разрешение конфликтов с помощью «временной синхронизации». Преимущества и недостатки метода. Примеры.

11. Ресурсы, конфликты на ресурсах. Разрешение конфликтов с помощью блоков – контроллеров. Примеры.

12. Виды блоков: агрегат, процессор, контроллер.

ЛИТЕРАТУРА

1. Советов Б.Я., Яковлев С.А. Моделирование систем. - М.: Высшая школа, 1999 – 271 с.

2. Бусленко Н.П., Калашников В.В., Коваленко И.Н. Лекции по теории сложных систем. - М.: Сов.Радио, 1973. - 438 с.

3. Дейкстра Э. Взаимодействие последовательных процессов. - М.:Мир, 1972. – 364 с.

4. Рабинович Е.В Теория вычислительных процессов.- Новосибирск: НГТУ, 2007. – 210 с.

5. Черненький В.М. Процессно - ориентированная концепция системного моделирования АСУ: Дисс. док. тех. наук. - М.,2000. – 350 с.