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

УДК 519.876.5

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПОСТРОЕНИЯ ИМИТАЦИОННОГО ПРОЦЕССА

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

Построение имитационной модели АСУ выполняется на основе описания процессов функционирования, таким образом, модель описания параллельных процессов позволяет выполнить исследование в области методов построения имитационных моделей. Имитационный процесс рассматривается, как один последовательный вычислительный процесс, на который отображается система параллельных взаимосвязанных процессов. Алгоритмическая модель описания процесса (АМП) позволила получить универсальную системную модель имитационного процесса, сформировать структуры моделирующих алгоритмов.

Основные положения

Пусть заданы два элементарных оператора h_l и h_k одного процесса Z, причем $h_l \rightarrow h_k$, (h_k сцеплен с h_l.).

Утверждение 1. Если $h_l \rightarrow h_k$, то

а) $t_k \geq t_l$

б) первым должен вычисляться оператор h_l.

Доказательство: Поскольку процесс Z развивается в соответствии с треком элементарных операторов и порядком α на T, то сцепление элементарного оператора в момент времени t_1 с любым оператором, имеющим $t \geq t_l$ невозможно. Но поскольку $h_l \rightarrow h_k$, то следовательно $t_k \geq t_l$. Т.к. $h_l \rightarrow h_k$, то отсюда следует, что пространство состояний h_l является частью аргументов оператора h_k. Таким образом, вычисление h_k невозможно без определения состояния h_k. Что и требовалось доказать.

Рассмотрим влияние отношения сцепления на последовательность выполнения операторов при моделировании. Пусть заданы два процесса Z_1 и Z_2, условно изображенные на рис. 1.

Дискретные состояния процесса Z_1 пронумерованы от 1 до 13, а процесса Z_2 - от 14 до 26. Рассмотрим четыре типовых случая:

а) моделируется один процесс Z_1, причем его операторы не сцеплены между собой.

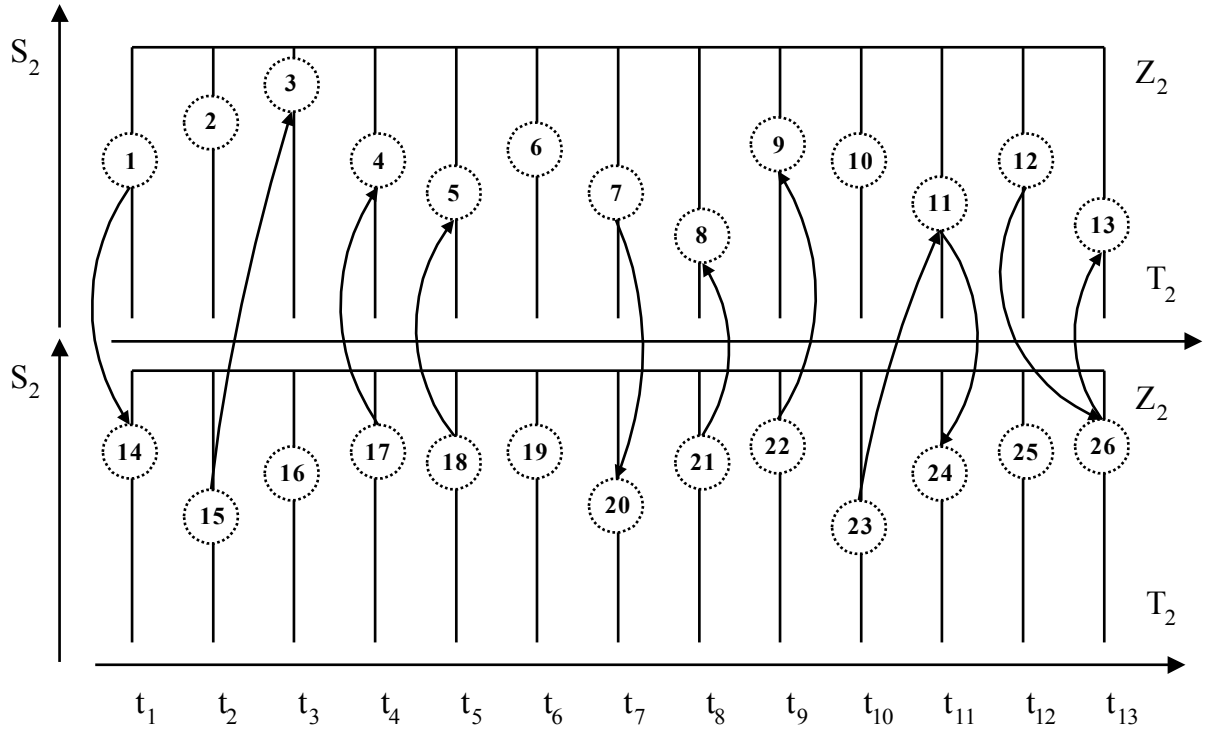


Рисунок 1. Пример сцепления процессов

б) моделируется процесс Z_1 без ограничения на сцепленность элементарных операторов;

в) моделируются два процесса Z_1 и Z_2, при этом эти процессы не сцеплены между собой;

г) моделируются два сцепленных процесса Z_1 и Z_1

На рисунке 1 стрелками указано отношение сцепления.

А. Моделирование процесса Z_1 при несцепленных операторах $h_i(\forall i)$.

Поскольку сцепление $h_i \rightarrow h_{i+1}$ отсутствует для всех i, то последовательность вычислений элементарных операторов не имеет значения и операторы $h_i(\forall i)$. могут вычисляться в любом порядке.

Б. Моделирование процесса Z_1 в общем случае.

Естественно предположить, что последовательные состояния одного процесса сцеплены между собой.

Предположим, что $h_i \rightarrow h_{i+1}$, для всех $i = \overline{1,n}$. Из утверждения 1 следует, что в таком случае $t_{i+1} > t_i$ для всех $i = \overline{1,n}$. Таким образом, последовательность сцепленных операторов строго следует порядку α на T. Из этого же утверждения следует, что последовательность вычислений операторов должна определяться этим же порядком.

В. Моделирование несцепленных между собой процессов Z_1 и Z_2.

Предполагаем, что процессы Z_1 и Z_2 в отдельности представлены общим случаем. Если h_i^1 - операторы процесса Z_1, а h_j^2 - операторы процесса Z_2, то по предположению отсутствует сцепление между h_i^1 и h_j^2 для всех i и j. Так же, как и в случае **А,** здесь последовательность вычислений элементарных операторов из разных процессов не имеет значения. Однако, поскольку все h_i^1 сцеплены между собой, и все h_j^2 также сцеплены между собой, важно, чтобы в этой последовательности выполнялся порядок α_1 для процесса Z_1 и α_2 для процесса Z_2. В частности, возможен вариант вычисления сначала всех операторов процесса Z_1, а затем всех операторов процесса Z_2. Очевидно, что порядок α_1 и α_2 при этом сохраняется.

Г. Моделирование сцепленных между собой процессов Z_1 и Z_2. Пример сцепления показан на рисунке 1.

В этом случае должен быть обязательно выдержан порядок α_1 для Z_1 и α_2 для Z_2, поскольку в общем случае внутри каждого процесса существует сцепленность элементарных операторов. Кроме того, необходимо выполнить условия Утверждения 1 для любой пары сцепленных операторов из разных процессов. Так, для приведенного на рисунке 5 примера в соответствии с условиями Утверждения 1, из операторов h_1 и h_{14} первым должен вычисляться оператор h_1, из h_5 и h_{18} первым - h_{18} и т.д.

Получим возможный порядок вычислений: h_1, h_{14}, (h_2, h_{15}), (h_3, h_{16}), h_{17}, h_4 и т.д.

В скобках указаны пары операторов из разных процессов, для которых можно поменять порядок вычислений.

На основании проведенного анализа можно сделать следующие выводы:

1. Для каждого процесса в ходе вычисления операторов необходимо строго придерживаться порядка α на Т.

2. Выполнение п.1. обеспечивает реализацию сцепленности операторов $h_i \rightarrow h_j$, если $t_i < t_j$, в рамках реализации одного процесса.

3. При выполнении п.1 указание сцепленности операторов с разным значением t_i в рамках одного процесса не добавляет новых ограничений на порядок расчета и может быть опущено.

Действительно, если вычислены элементарные операторы h_{10} и h_{23} в момент времени t_{10}, то указание $h_{23} \rightarrow h_{11}$ не меняет порядок вычислений, поскольку к моменту времени t_{11} будет рассчитан оператор h_{23}. При этом условие сцепленности выполняется автоматически.

4. Для определения порядка расчета операторов, принадлежащих разным процессам, важно знание отношения сцепления для операторов разных процессов с одинаковым значением времени t_i.

Таким образом, можно сформулировать следующий алгоритм определения порядка вычисления элементарных операторов для совокупности параллельных процессов.

Пусть заданы треки процессов Z^i $(i = \overline{1,n})$: $\left\langle \left\{ h_j^i \right\}, \beta^i \right\rangle$ для всех i. Пусть текущее значение времени равно t и все элементарные операторы h^i , у которых $t^i \le t$, вычислены. Для всех h_j^i , имеющих $t^i = t$ и обладающих условным временным оператором $(h_j^i)^t$, определим для каждого очередной момент времени t_{j+1}^i сцепления инициатора по своему треку, задаваемый этим временным оператором. Получим множество $\left\{ t_{j+1}^i \right\}$. Назовем его *активным временным множеством*.

Это множество содержит по одному значению от каждого процесса, остановленного на элементарном операторе, содержащем временное условие продвижения инициатора. Определим очередное значение времени по формуле:

$$t_0 = \boldsymbol{min} t^i_{j+1}, (\forall i) \qquad (1)$$

Последовательное применение формулы 1 строит упорядоченное множество $\langle T^t, \alpha^t \rangle$, где α^t -линейный порядок на T^t. Докажем ряд его свойств.

Для процесса Z^i имеем множество T^i и линейный порядок α^i ($i = \overline{1,n}$). Рассмотрим множество $T = \bigcup T^i (\forall i)$ и порядок α^T на T, полученный транзитивным замыканием порядков α^i.

Утверждение 2. Множества T^t и T равны.

В самом деле, любой элемент $t \in T^t$ был определен по формуле (2) из множества $\{t^i_{j+1}\}$, элементы которого принадлежат T^i ($i = \overline{1,n}$). Поскольку $T = \bigcup T^i (\forall i)$, то они же принадлежат и T. Следовательно, каждый элемент множества T^t принадлежит и множеству T. Аналогично доказывается, что если элемент принадлежит T, то он же принадлежит и T^t. Таким образом, утверждение 2 доказано.

Утверждение 3. На упорядоченном множестве $\langle T^t, \alpha^t \rangle$ сохраняются все порядки α^i ($i = \overline{1,n}$).

Рассмотрим множество T^i с порядком α^i. Возьмем два любых его элемента t^i_k и t^i_l. Пусть в соответствие с α^i $t^i_k < t^i_l$. Тогда, в соответствие с (2) t^i_k окажется минимальным элементом в $\{t^i_{j+1}\}$ раньше, чем t^i_1, и следовательно, войдет во множество T^t раньше, чем t^i_1. Поскольку упорядочение в T^t определяется порядком поступления элементов в T^t, то, следовательно, и в T^t $t^i_k < t^i_1$. Аналогично можно провести доказательство и для остальных процессов. Таким образом, утверждение 3 доказано.

На основании Утверждений 2 и 3 можно сделать заключение о том, что формула (1) позволяет построить новое множество T, включающее все элементы множеств T^i $(\forall i)$, и провести упорядочение его элементов таким образом, что сохраняются все отношения порядка в каждом из них.

Поскольку каждому $t \in T^i$ $(\forall i)$ ставится в соответствие свой элементарный оператор в треке, то определение порядка на T дает возможность однозначно определить порядок на всем множестве элементарных операторов заданных процессов.

Классы одновременных событий

Рассмотрим последовательность выполнения элементарных операторов в системе. Пусть система содержит 9 объектов, в каждом из которых развивается процесс. На рисунке 2 приведен пример организации треков на некотором временном интервале. Пусть в некоторый момент времени система находилась в состоянии, показанном на рисунке 6 слева. Назовем его начальным состоянием. Для каждого процесса Z_i указан текущий элементарный оператор в следующих обозначениях:

$$\langle h_n^{i,c}, h_n^{i,y} \rangle, \tag{2}$$

где i - номер процесса (он же - номер строки);

n - порядковый номер элементарного оператора в своем треке;

c - символ “состояние”;

y=л - символ логического условия продвижения инициатора;

y=t - символ временного условия продвижения инициатора.

Предположим, что к этому моменту времени все элементарные операторы вычислены. Тогда активное временное множество на текущий момент равно $\{t^1_{10}, t^3_{11}, t^6_{21}, t^9_{26},\}$

Определим в соответствии с (1) очередной момент времени, как:

$$t_0 = \boldsymbol{min}\ \{t^1_{10}, t^3_{11}, t^6_{21}, t^9_{26}\} \tag{3}$$

Для примера рис. 2 $t_0 = t^1_{10}$.

Таким образом, из начального состояния система переходит в новое состояние в момент времени t^1_{10}, соответствующее временному условию $h^{1,t}_9$.

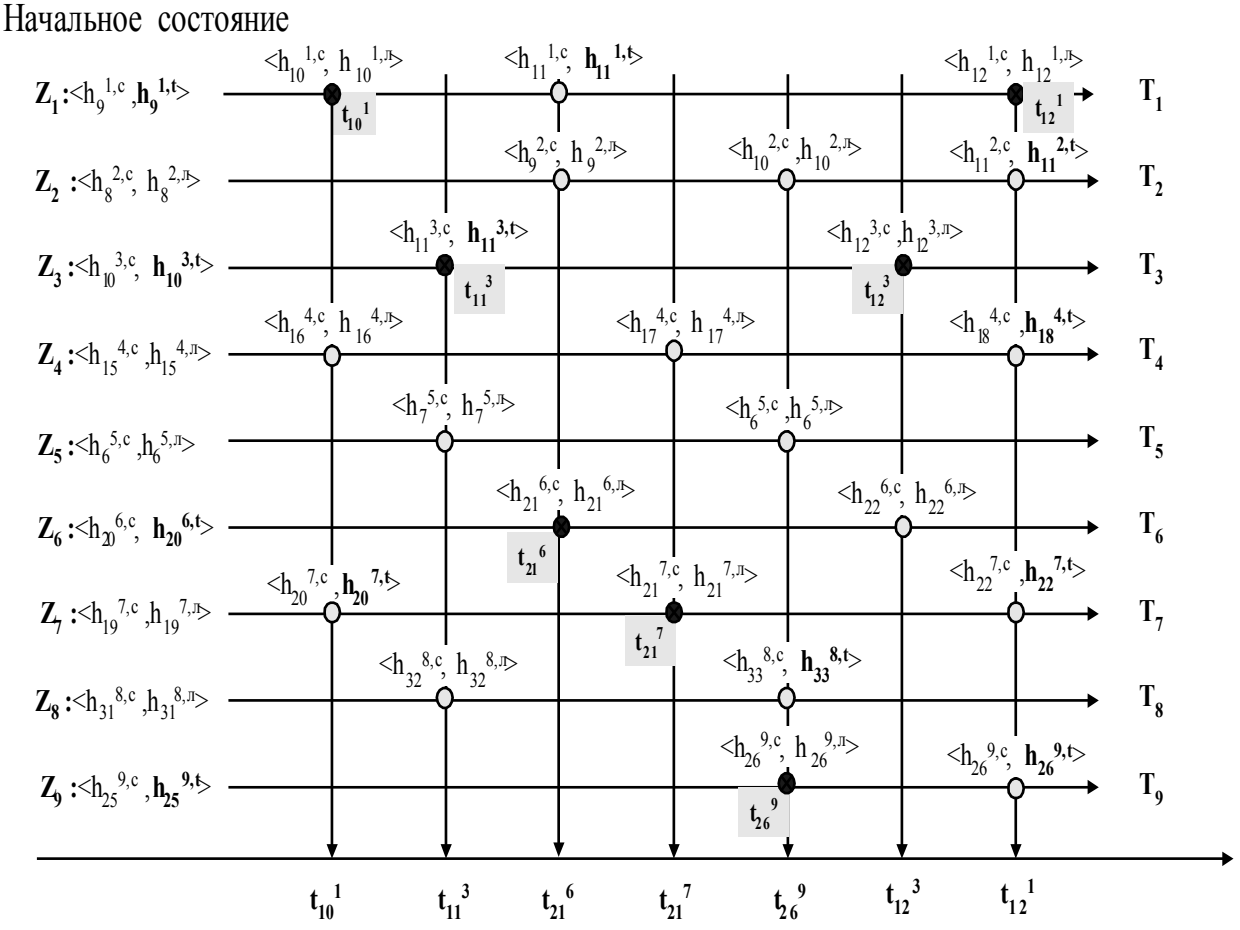


Рисунок 2. Пример реализации процессов через события

В этот момент времени инициатор I_1 процесса Z_1 перемещается в элементарный оператор ($h^{1,c}{}_{10}$, $h^{1,л}{}_{10}$) , вычисляя новое состояние объекта O_1, а значит, и системы в целом. При этом оказывается выполненным условие $h^{4,Л}{}_{15}$ и $h^{7,Л}{}_{19}$.. В треке процесса Z_4 выполняется оператор состояния $h^{4,c}{}_{16}$ и устанавливается логическое условие $h^{4,Л}{}_{16}$, а в треке процесса Z_7 – $h^{7,c}{}_{20}$ и временное условие $h^{7,t}{}_{20}$. На рисунке 2 соответствующие события отмечены светлыми кружками. Событие по временному условию показано на рисунке 2 в виде темного кружка. Так как больше условий не выполняется, то необходимо перейти к новым элементарным операторам в соответствии с треками. В рассматриваемый момент времени следующее активное временное множество имеет вид $\{t_{11}^3, t_{21}^6, t_{21}^7, t_{26}^9\}$. Согласно (1) необходимо перейти к новому моменту времени $t_0 = \boldsymbol{min}\{t_{11}^3, t_{21}^6, t_{21}^7, t_{26}^9\}$. В нашем случае

$t_0 = t_{11}^3$. Первым выполняется оператор $\langle h_{11}^{3,c}, h_{11}^{3,t} \rangle$, который изменяет состояние объекта O_3 и системы в целом, а также формирует новый заказ времени передвижения своего инициатора $\mathbf{I}_3$ в момент времени t_{12}^3. При этом выполняются логические условия для процессов Z_5 и Z_8. Инициаторы $\mathbf{I}_5$ и $\mathbf{I}_8$ вызывают выполнение операторов $\langle h^{5,c}{}_7, h^{5,л}{}_7 \rangle$ и $\langle h^{8,c}{}_{32}, h^{8,л}{}_{32} \rangle$ соответственно. Формируется новое активное временное множество: $\{t_{12}^3, t_{21}^6, t_{21}^7, t_{26}^9\}$, наименьшим в котором является t_{21}^6. С оператора $\langle h_{21}^{6,c}, h_{21}^{6,л} \rangle$ и начинается следующий цикл вычислений, и т.д.

На этом примере хорошо иллюстрируется алгоритм продвижения модельного времени и порядок выполнения элементарных операторов в каждом процессе. Сделаем некоторые обобщения.

Выполнение каждого элементарного оператора назовем *событием* в системе. *Событие активное, если оно следует в треке за элементарным оператором, содержащем* h^t. На рисунке 2 условия, содержащие h^t, выделены жирным шрифтом. К активным событиям относятся выполнение $\langle h_{10}{}^{1,c}, h_{10}{}^{1,л} \rangle$, $\langle h_{11}{}^{3,c}, h_{11}{}^{3,t} \rangle$, $\langle h_{21}{}^{6,c}, h_{21}{}^{6,л} \rangle$ и т.д.

Событие пассивное, если оно следует в треке за элементарным оператором, содержащем $h^л$. На рисунке 2 к пассивным событиям относятся выполнение $\langle h_{11}{}^{1,c}, h_{11}{}^{1,t} \rangle$, $\langle h_9{}^{2,c}, h_9{}^{2,л} \rangle$, $\langle h_{17}{}^{4,c}, h_{17}{}^{4,л} \rangle$ и т.д.

Множество событий, происходящих в один и тот же момент модельного времени, назовем *классом одновременных событий* (КОС). На рисунке 2 класс одновременных событий в момент времени t_{10}^l включает события для 1, 4 и 7-го процессов; в момент времени t_{11}^3 КОС составляют события для 3, 5 и 8-го процессов и т.д.

Рассмотрение КОС примера на рис. 2 показывает, что в каждом КОС содержится активное событие. Введем следующие допущения:

Допущение 1. В каждом КОС содержится одно и только одно активное событие.

Тогда: а) все остальные события в КОС, если они есть, являются пассивными;

б) количество КОС равно мощности объединенного множества времен Т.

Допущение 2. Первым событием в любом КОС является активное событие.

Если это так, то оставшиеся пассивные события для определения последовательности своего выполнения требуют знания отношения сцепления. Представим отношение сцепления в КОС в виде направленного графа. На рисунке 3 приведен пример некоторого КОС с заданным на нем отношением сцепления.

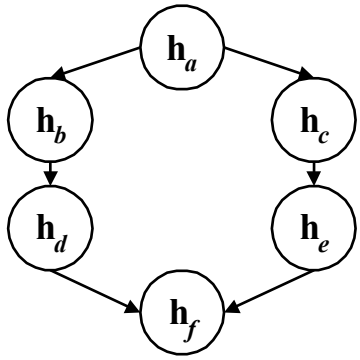


Рисунок 3. Пример КОС

Здесь: h_a - активное событие; h_b, h_c, h_d, h_e, h_f- пассивные события.

Из графа следует, что после h_a необходимо выполнить h_b либо h_c; затем h_d, либо h_e; и наконец h_f.

Задание отношения сцепления для каждого КОС является самостоятельной задачей. Однако можно предложить таким образом определять условие выполнения каждого пассивного события, чтобы оно содержало условие выполнения предыдущего события, т.е. содержало описание отношения сцепления с предыдущим событием.

Так, условие выполнения события h_d должно содержать выражение, проверяющее выполнение события h_b. Если это удается сделать, то алгоритм формирования КОС выглядит следующим образом:

1. Выполняется активное событие.

2. Проверяются условия всех возможных в системе пассивных событий.

3. Если выполняется условие пассивного события, то оно вычисляется. Алгоритм продолжается с п.2.

Важно найти признак, по которому можно определить завершенность КОС.

Теорема А

КОС завершен, если все условия, заданные операторами $h^{л}$ во всех треках, равны 0.

Поскольку КОС содержит одно активное событие и оно выполняется первым, то все остальные события пассивные. Пассивное событие по определению выполняется, если определяющее его условие равно 1. Однако, по предположению, все условия, заданные $h^{л}$ равны 0. Таким образом, выполнение пассивного события невозможно, а единственное активное событие уже выполнено. Поскольку не выполняется ни один элементарный оператор, состояние системы и параметры условных операторов не могут быть изменены. Состояние системы зафиксировано и не будет изменено вплоть до нового КОС. Что и требовалось доказать.

Теорема В

Первым событием в КОС является активное событие.

Доказывая теорему А, мы показали, что когда завершен КОС, то все условия в условных операторах системы равны 0, и состояние системы не может быть изменено ни одним пассивным событием. Значит, следующий КОС может начаться только с активного события. Что и требовалось доказать.

Таким образом, наше допущение 2 о начальной функции активного события в КОС доказано строго в теореме В.

МОДЕЛИРУЮЩИЙ АЛГОРИТМ СКАНИРУЮЩЕГО ТИПА

Моделирующий алгоритм в любом случае, как это следует из выше изложенного, должен включать следующие составные части:

- подпрограммы событий, реализующие элементарные операторы;
- алгоритм формирования модельного времени;
- алгоритм выбора очередного КОС;
- алгоритм генерирования КОС.

Подпрограмма события представляет собой программную реализацию одного элементарного оператора, включающего оператор состояния, оператор условия продвижения инициатора и навигационный оператор. В этих подпрограммах, в общем случае, все параметры являются глобальными. В каждом же конкретном случае часть параметров может быть локализована. Однако все параметры, через которые осуществляется обмен, являются глобальными. Если подпрограмма события реализует объединенный элементарный оператор, то она должна иметь доступ к значению инициатора, определяющего локальную среду данного процесса.

В самом общем виде моделирующий алгоритм представлен на рисунке:

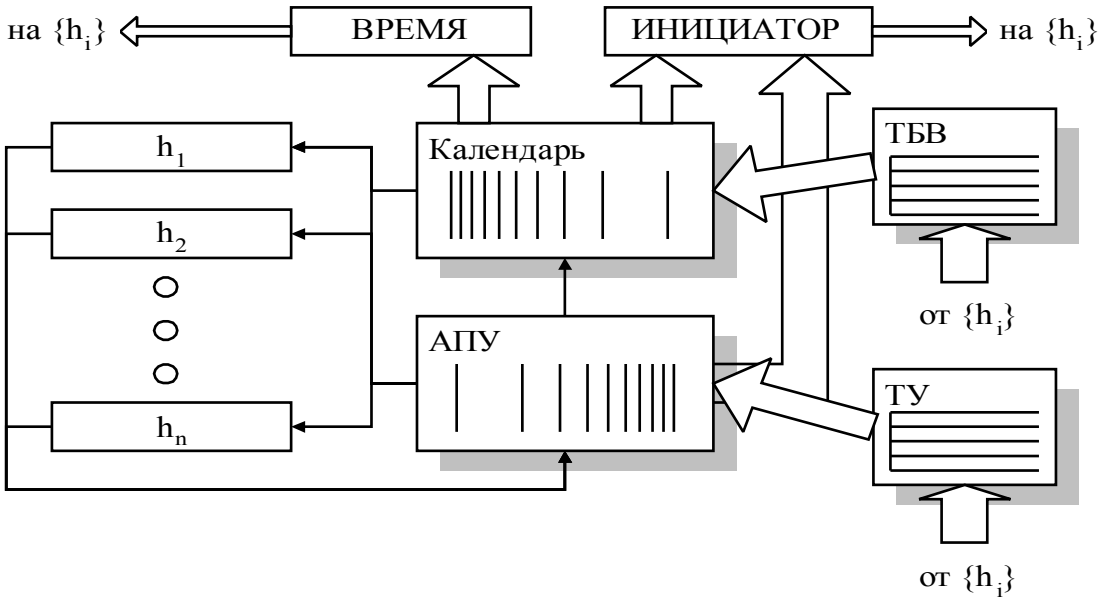


Рисунок 4. Моделирующий алгоритм сканирующего типа

Здесь (h_1, h_2,.., h_n) – неупорядоченная совокупность подпрограмм событий, реализующих элементарные операторы треков всех процессов в системе.

Параметр **ВРЕМЯ** - содержит текущее значение модельного времени;

Параметр **ИНИЦИАТОР** - содержит значение текущего инициатора (ссылку на локальную среду процесса).

КАЛЕНДАРЬ - алгоритм, реализующий монотонно возрастающее продвижение модельного времени по формуле (4) и начало нового КОС в системе.

АПУ - Алгоритм Проверки Условий, обеспечивающий построение КОС для текущего значения модельного времени.

ТБВ - Таблица Будущих Времен, структура которой приведена на рисунке 5. Каждая строка ТБВ соответствует одному процессу и содержит следующие элементы описания будущего *активного* события:

столбец 1 - значение момента времени активизации инициатора, определяемого предшествующим оператором h^t ;

столбец 2 – инициатор процесса;

столбец 3 - адрес подпрограммы активного события в треке данного процесса.

-1-	-2-	-3-
момент активизации	значение инициатора	адрес активной подпрограммы
• • •.	• • •.	• • •.

Рисунок 5. Структура таблицы ТБВ.

Совокупность значений атрибута “момент активизации” таблицы ТБВ составляет в любой момент модельного времени активное временное множество.

ТУ - Таблица Условий, структура которой приведена на рисунке 6. Каждая строка ТУ соответствует одному процессу и содержит следующие элементы описания будущего *пассивного* события:

столбец 1 – логическое условие активизации инициатора, определяемое предшествующим оператором $h^{л}$;

столбец 2 – инициатор процесса;

столбец 3 - адрес подпрограммы пассивного события в треке данного процесса.

-1-	-2-	-3-
условие логическое	значение инициатора	адрес очередной подпрограммы
• • •	• • •.	• • •.

Рисунок 6. Структура таблицы ТУ

КАЛЕНДАРЬ определяет *первое активное событие* в новом КОС в соответствии с (1.). Его алгоритм выглядит следующим образом:

1. Поиск минимального значения в столбце 1 ТБВ. Пусть это значение равно t_k, где k - номер строки ТБВ.

2. **ВРЕМЯ** := t_k

3. **ИНИЦИАТОР** := <значение столбца 2 в ТБВ по строке k>

4. С := <значение столбца 3 в ТБВ по строке k >

5. Затирание k -ой строки ТБВ.

6. Передача управления по адресу, хранящемуся в С.

Из алгоритма видно, что КАЛЕНДАРЬ ведет модельное время и инициирует выполнение активного события - первого события в каждом КОС.

АПУ генерирует пассивные события КОС и его алгоритм имеет следующий вид:

1. Просчет всех логических условий, заданных в столбце 1 ТУ. В зависимости от результата, полученного при вычислении логического условия j-ой строки ТУ, выполняется шаг 2 либо шаг 3.

2. Если логическое условие равно 0 (ложь), то j:=j+1 и повторяется шаг 1.

3. Если логическое условие равно 1 (истина), то происходит разбор j-ой строки:

- **ИНИЦИАТОР** :=<значение столбца 2 в ТУ по j-ой строке>;
- D:=<значение столбца 3 в ТУ по j-ой строке>;
- затирание j-ой строки ТУ;
- передача управления по адресу, хранящемуся в D.

4. Если все логические условия в столбце 1 равны 0 и нет ни одного условия, равного 1, управление передается программе КАЛЕНДАРЬ, так как исчерпаны все события текущего КОС и необходим переход к новому КОС, начинающемуся с активного события (см. теорему А).

Как видно из рисунка 4, подпрограммы событий после своего выполнения передают управление в алгоритм АПУ. Каждая подпрограмма h_i в ходе своего выполнения меняет состояние системы и определяет условие продвижения инициатора своего процесса по треку. Если подпрограмма h_i содержит условие типа h^t, то h^t заполняет строку в ТБВ, определяет момент активизации инициатора. Навигационный оператор $h^н$ в составе h^t определяет адрес следующей по треку подпрограммы событий. В эту же таблицу помещается и значение текущего инициатора (ссылка на локальную среду). Если подпрограмма h_i содержит условие типа $h^л$, то $h^л$ заносит строку в ТУ, помещая в столбец 1 логическое условие, а в остальные - инициатор и адрес следующей по треку подпрограммы событий аналогично вышеописанному.

Таким образом, предложенный моделирующий алгоритм реализует все необходимые действия в соответствии с разработанной в диссертации алгоритмической моделью процесса. При этом задача генерации трека по ходу моделирования возлагается на подпрограммы событий.

Оценим вычислительную эффективность этого алгоритма.

Пусть средняя длина подпрограммы события составляет a команд, для выполнения КАЛЕНДАРЯ необходимо r команд, для АПУ - P команд на просчет одной строки. Пусть ТУ содержит в каждый момент модельного времени в среднем L строк с условиями, а в каждом КОС в среднем содержится l пассивных событий.

Тогда общее количество команд K, затрачиваемое на реализацию одного КОС, в среднем составит:

$$K = r + (l+1)\cdot a + \frac{L}{2}\cdot l \cdot P \qquad (4)$$

Полезными следует считать затраты на выполнение подпрограмм событий. Таким образом, эффективное (полезное) количество команд G равно:

$$G=(l+1)a \qquad (5)$$

Затратность алгоритма оценим, как:

$$q = \frac{K}{G} \qquad 6)$$

Таким образом:

$$q = 1 + \frac{r}{(l+1)a} + \frac{L\cdot l\cdot P}{2(l+1)a} \qquad (7)$$

Как правило, $l>>1$, а значения a и r соизмеримы. Таким образом, вполне можно пренебречь вторым слагаемым. В этих условиях третье слагаемое будет иметь вид $\frac{L\cdot P}{2\cdot a}$. И окончательно: $q \approx 1 + \frac{L\cdot P}{2\cdot a}$.

В случае, когда моделирующий алгоритм содержит небольшое количество подпрограмм событий, и каждая из подпрограмм имеет достаточно большой объем, то $P<<a$ и значения L невелики. В этом случае значение q будет не намного отличаться от 1.

Однако если моделирующий алгоритм содержит большое количество достаточно коротких подпрограмм событий, то $P\approx a$ и значение L велико. В

этом случае $q >> 1$: так, при $L=10$, $q=6$, а при $L=50$ значение q превышает 25. Это объясняется тем, что на каждое событие происходит обращение к АПУ с целью поиска очередного пассивного события в КОС, и большая часть временных ресурсов центрального процессора моделирующей ЭВМ затрачивается на сканирование и просчет условий в ТУ.

Моделирующий алгоритм линейного типа

Высокая затратность сканирующего алгоритма вызвана необходимостью многократного просчета логических условий в ТУ для автоматизации генерирования КОС. Таким образом, чтобы сократить чрезмерные затраты машинного времени на сканирование ТУ, необходимо изменить способ генерирования КОС. В данном разделе предлагается процедуры генерации КОС разместить непосредственно в подпрограммах событий и исключить АПУ из моделирующего алгоритма. Такой модернизированный моделирующий алгоритм назовем *моделирующим алгоритмом линейного типа*, его структурная схема приведена на рисунке 7.

В этом алгоритме КАЛЕНДАРЬ выбирает ближайшее активное событие из ТБВ по тому же алгоритму, что и для моделирующего алгоритма сканирующего типа, и передает управление соответствующей подпрограмме активного события. Далее подпрограмма активного события после своего выполнения передает управление той подпрограмме пассивного события, которое должно выполняться в соответствии с графом КОС. Эта подпрограмма, в свою очередь, передает управление следующей по графу КОС подпрограмме пассивного события и т.д. до тех пор, пока не будут исчерпаны все события текущего КОС. Последняя подпрограмма в текущем КОС передает управление КАЛЕНДАРЮ, что соответствует переходу к новому КОС.

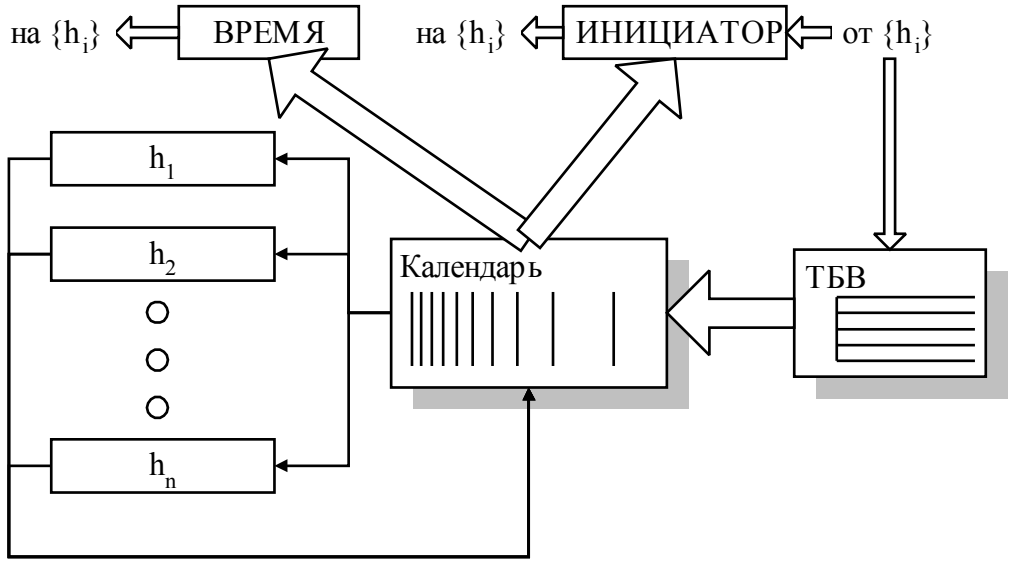


Рисунок 7. Моделирующий алгоритм линейного типа

В качестве примера построения моделирующего алгоритма линейного типа рассмотрим следующую одноканальную систему массового обслуживания:

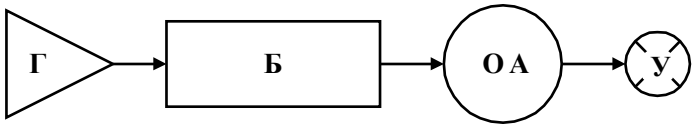


Рисунок 8. Пример моделируемой системы

Здесь:

Г -генератор требований;

Б - очередь требований;

ОА - обслуживающий аппарат;

У - блок уничтожения требований.

При имитации этой системы можно выделить следующие события:

Вых_Г - выход требования из генератора Г;

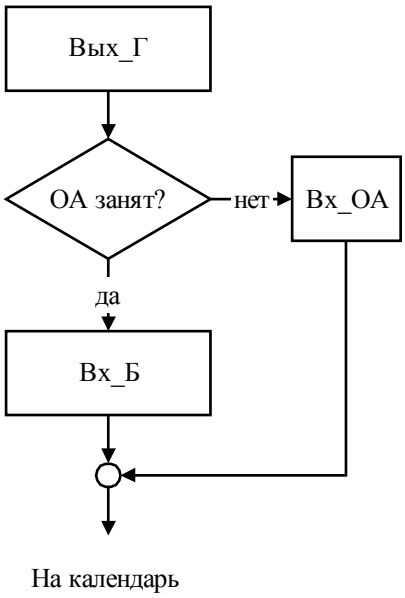
Вх_Б - прием требований в очередь к ОА;

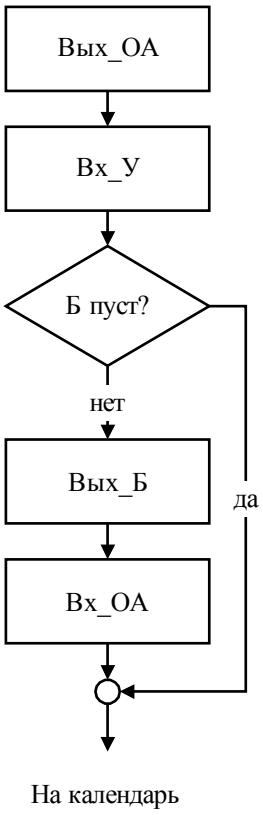
Вых_Б - выдача требования в ОА из Б;

Вх_ОА - прием требования на обработку в ОА;

Вых_ОА - окончание обработки требования в ОА;

Вх_У - прием требования в блок уничтожения У.





Граф КОС1 Граф КОС2

Рисунок 9. Графы КОС

Подпрограммы событий назовем теми же именами. Очевидно, активными могут быть события, связанные с Вых_Г и Вых_ОА. Остальные события - пассивные. Таким образом, в модели возможны два вида КОС: КОС1 и КОС2 (рисунок 9).

На рисунках показаны условия, определяющие генерацию КОС. Причем само условие включается в верхнюю по отношению к нему подпрограмму события. Так, проверка “ОА занят ?” в КОС1 должна быть включена в подпрограмму Вых_Г.

Очевидно, что коэффициент затратности такого алгоритма близок к 1. Однако это достигается путем усложнения подпрограмм событий в силу необходимости задавать в них в явном виде все возможные варианты генерации КОС.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Определение отношения сцепленности элементарных операторов.
2. Понятие квазипараллельного процесса. Правило корректности отображения параллельных процессов на квазипараллельный процесс.
3. Активное временное множество, алгоритм формирования модельного времени.
4. Классификация событий. Активные и пассивные события.
5. Класс одновременных событий. Свойства класса одновременных событий.
6. Моделирующий алгоритм сканирующего типа. Состав компонент, таблицы, структура.
7. Оценка эффективности моделирующего алгоритма сканирующего типа.
8. Моделирующий алгоритм линейного типа. Состав компонент, таблицы, структура.
9. Оценка эффективности моделирующего алгоритма линейного типа.

ЛИТЕРАТУРА

1. Киндлер Е. Языки моделирования.-М.: Энергоатомиздат, 1985.-288 с.

2. Аверилл М. Лоу, В. Дэвид Кельтон. Имитационное моделирование.- СПб.: BHV, 2004.- 848 с.

3. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование.- СПб.: БХВ-Петербург, 2005.- 403с.

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