электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Алгоритмы генерации процесса имитации

77-30569/292147
11, ноябрь 2011
Черненький В. М.
УДК 004.519.876.5

МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Поскольку предлагаемая структура алгоритмов генерации опирается на алгоритмическую модель описания процессов (АМП), то кратко изложим ее основные положения [1]. 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. Назовем этот оператор *элементарным*, поскольку он вычисляет одно значение состояния $s^i \in S$ в один заданный момент времени t_i:

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

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

ω - случайное число.

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

$$h_1^c, h_2^c \ldots, h_i^c, \ldots h_n^c.$$

Введем новый элемент модели - *инициатор*. Он обладает свойствами динамичности и инициативности. Динамичность - инициатор имеет возможность перемещаться от оператора к оператору; будем называть попадание инициатора на оператор *сцеплением инициатора с элементарным оператором*. Инициативность - в момент сцепления инициатора с

оператором происходит *выполнение (инициирование) элементарного оператора*, что соответствует вычислению нового состояния процесса. При этом полагаем, что выполнение элементарного оператора происходит *мгновенно*. Таким образом, описание процесса может быть выполнено путем задания трека и перемещения по нему инициатора ***I***, сцепляющегося с элементарными операторами h_i^c в заданные моменты времени t_i изменения состояния процесса. Алгоритмическая модель описания процесса предполагает, что *моменты сцепления инициатора с элементарными операторами определяют сами элементарные операторы.* С этой целью вводится *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.

Таким образом, окончательно определим элементарный оператор h_i, как двойку:

$$h_i = \left\langle h_i^c, h_i^y \right\rangle.$$

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}. Линейную последовательность элементарных операторов назовем *треком* TR:

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

Таким образом, можно АМП определить как двойку:

$$\text{АМП} = <TR, I>.$$

Имитационный процесс рассматривается, как один последовательный вычислительный процесс, на который отображается система параллельных взаимосвязанных процессов.

Пусть заданы треки процессов Z^i ($i = \overline{1,n}$): $\left\langle \{h_j^i\}, \beta^i \right\rangle$ 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 $\{t_{j+1}^i\}$. 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:

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

Выполнение каждого элементарного оператора назовем *событием* в системе. *Событие активное, если оно следует в треке за элементарным оператором, содержащем* h^t. *Событие пассивное, если оно следует в треке за элементарным оператором, содержащем* h^{π}. Множество событий, происходящих в один и тот же момент модельного времени, назовем *классом одновременных событий* (КОС). 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.

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.

4. КОС завершен, если все условия, заданные операторами $h^{д}$ во всех треках, равны 0. Переходим к п.1.

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. Рассмотрим алгоритм, реализующий в полной мере все вышеуказанные функции. Назовем его моделирующим алгоритмом сканирующего типа (рис. 1).

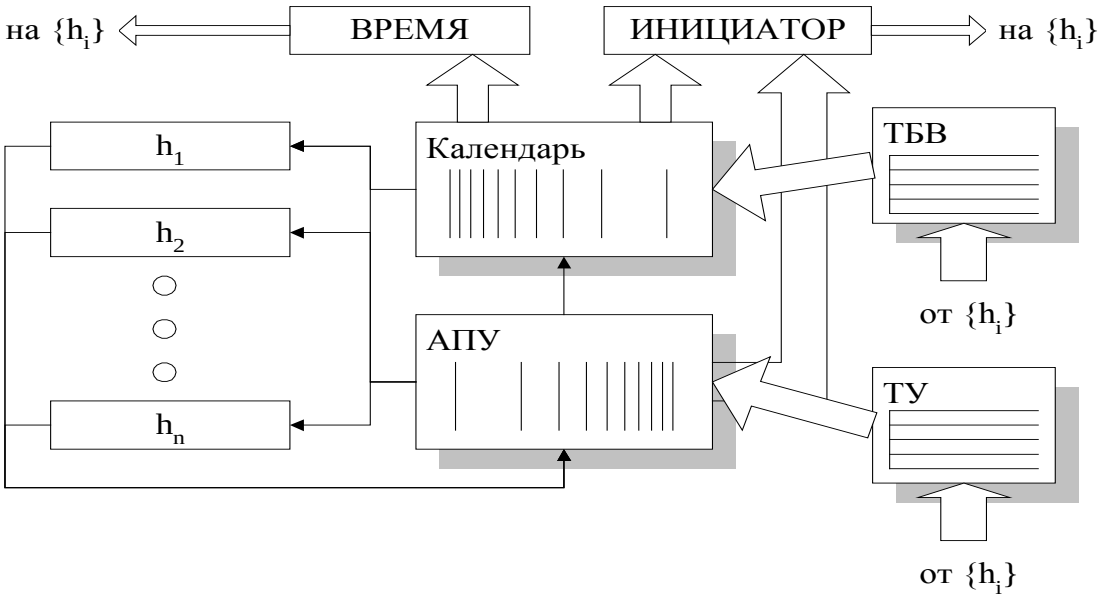


Рисунок 1. Моделирующий алгоритм сканирующего типа

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 (1) 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 рис. 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 2. 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 рис. 3. 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 3. Структура таблицы ТУ

КАЛЕНДАРЬ определяет *первое активное событие* в новом КОС в соответствии с формулой (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.

Как видно из рис. 1, 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. В эту же таблицу помещается и значение текущего инициатора (ссылка на локальную среду). Если подпрограмма h_i содержит условие типа $h^{\prime\prime}$, то $h^{\prime\prime}$ 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 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:

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

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

$$G=(l+1)a \qquad (2)$$

Затратность алгоритма оценим, как:

$$q = \frac{K}{G} \tag{3}$$

Таким образом:

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

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 на много отличаться от 1.

Однако если моделирующий алгоритм содержит большое количество достаточно коротких подпрограмм событий, то $P \approx a$ и значение L может быть большим.. 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 этой связи предлагается процедуры генерации КОС разместить непосредственно в подпрограммах событий и исключить АПУ из моделирующего алгоритма. Такой модернизированный моделирующий алгоритм назовем *моделирующим алгоритмом линейного типа* (рис. 4).

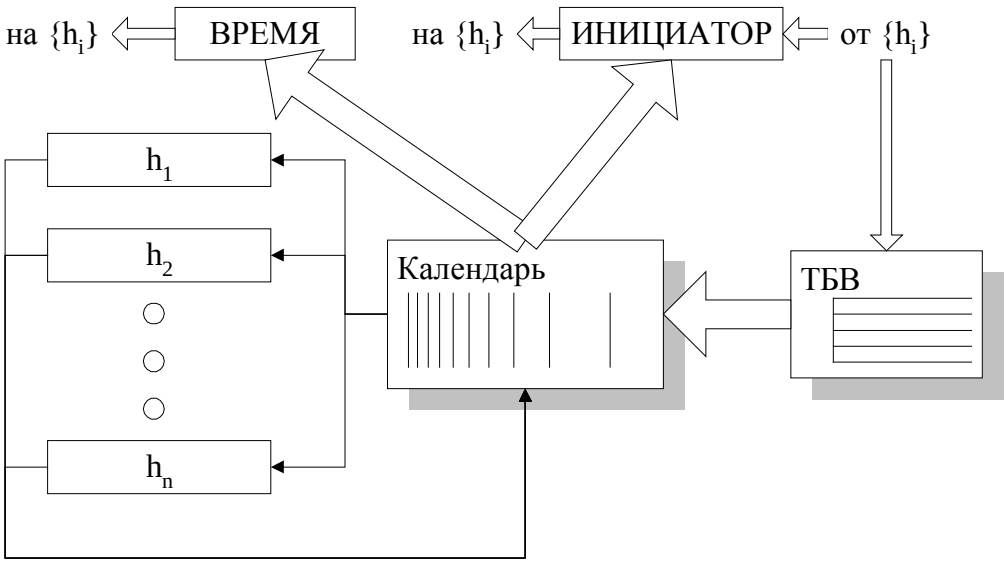


Рисунок 4. Моделирующий алгоритм линейного типа

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.

В качестве примера построения моделирующего алгоритма линейного типа рассмотрим следующую одноканальную систему массового обслуживания (рис. 5):

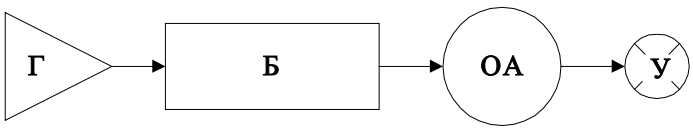


Рисунок 5. Пример моделируемой системы

Здесь:

Г -генератор требований;

Б - очередь требований;

ОА - обслуживающий аппарат;

У - блок уничтожения требований.

При имитации этой системы можно выделить следующие подпрограммы событий:

Вых_Г - выход требования из генератора Г;

Вх_Б - прием требований в очередь к ОА;

Вых_Б - выдача требования в ОА из Б;

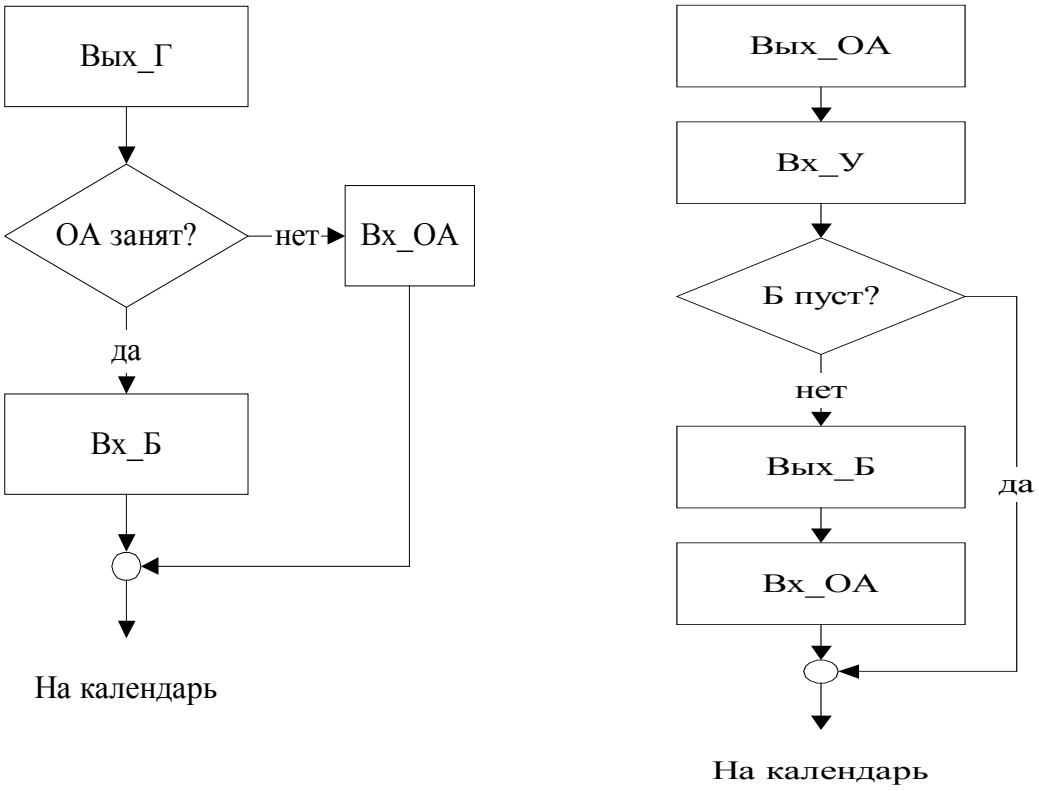
Вх_ОА - прием требования на обработку в ОА;

Вых_ОА -окончание обработки требования в ОА;

Вх_У - прием требования в блок уничтожения У.

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 (рис. 6)

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.



Граф КОС1 Граф КОС2

Рисунок 6. Графы КОС

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.

Таким образом, в статье показано, что в соответствии с концепцией алгоритмической формы описания процессов, можно сформулировать универсальный алгоритм генерации имитационного процесса: моделирующий алгоритм сканирующего типа. К недостаткам алгоритма следует отнести его высокую для ряда систем вычислительную затратность. С тем, чтобы максимально сохранить пользовательские преимущества алгоритма сканирующего типа, но снизить его вычислительную затратность, можно предложить ряд изменений этого алгоритма, прежде всего в области правил генерации КОС. Один вариант такого алгоритма приведен в

настоящей статье под названием алгоритма линейного типа, имеющего малый коэффициент затратности, однако увеличивающий нагрузку на пользователя.

Литература

1. Черненький В.М. Процессно-ориентированная концепция системного моделирования АСУ: диссертация ... доктора технических наук: 05.13.06, М., 2000 - 299 с.ил.

electronic scientific and technical periodical
SCIENCE and EDUCATION
EL № FS 77 - 30569. №0421100025. ISSN 1994-0408

Algorithm of simulation process' generating

77-30569/292147

 # 11, November 2011

Chernen'kii V.M.

Bauman Moscow State Technical University
chernen@bmstu.ru

Rules and algorithms of simulation process generating, based on the description of discrete system's functioning as a set of connected series-parallel processes, represented as algorithmic models are presented in this article. Fundamentals of algorithmic process model are briefly described. Rules providing correct generating of a class of simultaneous events (CSE) were introduced. Content and algorithms of basic components of generating algorithm such as a set of event subprograms, an algorithm of CSE selection, a program “Calendar”, a program which checks the conditions, a structure of controlling tables. Algorithms of two modeling program were included. They are scanning and linear types of program. Their performance was analyzed.

Publications with keywords: modelling, simulation, control algorithms, quasi-parallel process, event, generating of class of simultaneous process
Publications with words: modelling, simulation, control algorithms, quasi-parallel process, event, generating of class of simultaneous process

Reference

Chernen'kii V.M., The process - oriented concept of system modeling ACS (Dr.Sci.Tech. dissertation), Moscow, 2000, 299 p.