Тема 5. МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ

6.1. Моделирование параллельных процессов

Практически любая более или менее сложная система имеет в своем составе компоненты, работающие одновременно, или, как принято говорить на языке техники, параллельно. Параллельно работающие подсистемы могут взаимодействовать самым различным образом, либо вообще работать независимо друг от друга. Способ взаимодействия подсистем определяет вид параллельных процессов, протекающих в системе. В свою очередь, вид моделируемых процессов влияет на выбор метода их имитации.

6.1.1.Виды параллельных процессов в сложных системах

Асинхронный параллельный процесс — такой процесс, состояние которого не зависит от состояния другого параллельного процесса (ПП).

Пример асинхронных ПП из области вычислительной техники: выполнение вычислений процессором и вывод информации на печать.

Синхронный ПП — такой процесс, состояние которого зависит от состояния взаимодействующих с ним ПП.

Пример синхронного ПП: работа торговой организации и доставка товара со склада (нет товара — нет торговли).

Один и тот же процесс может быть синхронным по отношению к одному из активных ПП и асинхронным по отношению к другому. Так, при работе вычислительной сети по технологии «клиент-сервер» каждый из узлов сети синхронизирует свою работу с работой сервера, но не зависит от работы других узлов.

Подчиненный ПП — создается и управляется другим процессом (более высокого уровня). Весьма характерным примером таких процессов является ведение боевых действий подчиненными подразделениями.

Независимый ПП— не является подчиненным ни для одного из процессов. Скажем, после запуска неуправляемой зенитной ракеты ее полет можно рассматривать как независимый процесс, одновременно с которым самолет ведет боевые действия другими средствами.

Способ организации параллельных процессов в системе зависит от физической сущности этой системы.

Остановимся несколько подробнее на особенностях реализации параллельных процессов в вычислительных системах (ВС). Это обусловлено следующей причиной.

Разработка и использование любой ИМ предполагает ее программную реализацию и исследование с применением ВС. Поэтому для реализации моделей, имитирующих параллельные процессы, в некоторых случаях применимы механизмы, используемые при выполнении параллельных вычислений.

Вместе с тем, реализация параллельных процессов в ВС имеет свои особенности:

- на уровне задач вычислительные процессы могут быть истинно параллельными только в многопроцессорных ВС или вычислительных сетях;

• многие ПП используют одни и те же ресурсы, поэтому даже асинхронные ПП в пределах одной ВС вынуждены согласовывать свои действия при обращении к общим ресурсам;

• в ВС дополнительно используется еще два вида ПП: родительский и дочерний ПП; особенность их состоит в том, что процесс-родитель не может быть завершен, пока не завершатся все его дочерние процессы.

В силу перечисленных особенностей для организации взаимодействия параллельных процессов в ВС используются три основных подхода:

• на основе «взаимного исключения»;

• на основе синхронизации посредством сигналов;

• на основе обмена информацией (сообщениями).

«Взаимное исключение» предполагает запрет доступа к общим ресурсам (общим данным) для всех ПП, кроме одного, на время его работы с этими ресурсами (данными).

Синхронизация подразумевает обмен сигналами между двумя или более процессами по установленному протоколу. Такой «сигнал» рассматривается как некоторое событие, вызывающее у получившего его процесса соответствующие действия.

Часто возникает необходимость передавать от одного ПП другому более подробную информацию, чем просто «сигнал-событие». В этом случае процессы согласуют свою работу на основе обмена сообщениями.

Перечисленные механизмы реализуются в ВС на двух уровнях — системном и прикладном.

Механизм взаимодействия между ПП на системном уровне определяется еще на этапе разработки ВС и реализуется в основном средствами операционной системы (частично — с использованием аппаратных средств).

На прикладном уровне взаимодействие между ПП реализуется программистом средствами языка, на котором разрабатывается программное обеспечение.

Наибольшими возможностями в этом отношении обладают так называемые языки реального времени (ЯРВ) и языки моделирования.

Языки реального времени — это языки, предназначенные для создания программного обеспечения, работающего в реальном масштабе времени, например для разработки различных автоматизированных систем управления (предприятием, воздушным движением и т. д.). К ним, в частности, относятся: язык *Ада,* язык *Модула* и практически единственный отечественный язык реального времени — *Эль-76* (использовавшийся в многопроцессорных вычислительных комплексах семейства «Эльбрус»).

6.1.2. Методы описания параллельных процессов в системах и языках моделирования

Языки моделирования по сравнению с языками реального времени требуют от разработчика значительно менее высокого уровня подготовки в области программирования, что обусловлено двумя обстоятельствами:

• во-первых, средства моделирования изначально ориентированы на квазипараллельную обработку параллельных процессов;

• во-вторых, механизмы реализации ПП относятся, как правило, к внутренней организации системы (языка) моделирования и их работа скрыта от программиста.

В практике имитационного моделирования одинаково широко используются как процессно-ориентированные языки (системы) моделирования, например *SIMULA,* так и языки, ориентированные на обработку транзактов (например, язык *GPSS).* В тех и других используются аналогичные методы реализации квазипараллелизма, основанные на ведении списков событий. В процессно-ориентированных системах используются списки событий следования, а в транзактных системах — списки событий изменения состояний.

Современные языки и системы моделирования, ориентированные на использование в среде многозадачных операционных систем типа Windows, частично используют их механизмы управления процессами, что делает их применение еще более эффективным. В пакете MATLAB также имеется собственный язык моделирования, и к нему в полной мере можно отнести сказанное выше. Тем не менее во многих случаях оказывается полезным знание общего механизма реализации ПП в языках моделирования.

Рассмотрим его применительно к моделированию на основе тракзактов.

В этом случае под событием понимается любое перемещение транзакта по системе, а также изменение его состояния (обслуживается, заблокирован и т. д.).

Событие, связанное с данным транзактом, может храниться в одном из следующих списков.

Список текущих событий. В этом списке находятся события, время наступления которых меньше или равно текущему модельному времени. События с «меньшим» временем связаны с перемещением тех транзактов, которые должны были начать двигаться, но были заблокированы.

Список будущих событий. Этот список содержит события, время наступления которых больше текущего модельного времени, то есть события, которые должны произойти в будущем (условия наступления которых уже определены — например, известно, что транзакт будет обслуживаться некоторым устройством 10 единиц времени).

Список прерываний. Данный список содержит события, связанные с возобновлением обработки прерванных транзактов. События из этого списка выбираются в том случае, если сняты условия прерывания.

В списке текущих событий транзакты расположены в порядке убывания приоритета соответствующих событий; при равных приоритетах — в порядке поступления в список.

Каждое событие (транзакт) в списке текущих событий может находиться либо в активном состоянии, либо в состоянии задержки. Если событие активно, то соответствующий транзакт может быть продвинут по системе; если продвижение невозможно (например, из-за занятости устройства), то событие (и транзакт) переводится в состояние задержки.

Как только завершается обработка (продвижение) очередного активного транзакта, просматривается список задержанных транзактов, и ряд из них переводится в активное состояние. Процедура повторяется до тех пор, пока в списке текущих событий не будут обработаны все активные события. После этого просматривается список будущих событий. Модельному времени присваивается

значение, равное времени наступления ближайшего из этих событий. Данное событие заносится в список текущих событий. Затем просматриваются остальные события списка. Те из них, время которых равно текущему модельному времени, также переписываются в список текущих событий. Просмотр заканчивается, когда в списке остаются события, времена которых больше текущего модельного времени.

В качестве иллюстрации к изложенному рассмотрим небольшой пример.

► *Пусть в систему поступают транзакты трех типов, каждый из которых обслуживается отдельным устройством. Известны законы поступления транзактов в систему и длительность их обслуживания. Таким образом, в системе существуют три параллельных независимых процесса (PI, P2, P3).*

Временная диаграмма работы системы при обслуживании одного транзакта каждого типа показана на рис.2.7.

На рисунке события, относящиеся к процессу P1, обозначены как $C1_i$, относящиеся к P2 и к P3 — соответственно как $C2_i$. и $C3_i$. Моменты времени $t_{вх}$ и $t_{вых}$ соответствуют началу и окончанию обслуживания транзакта.

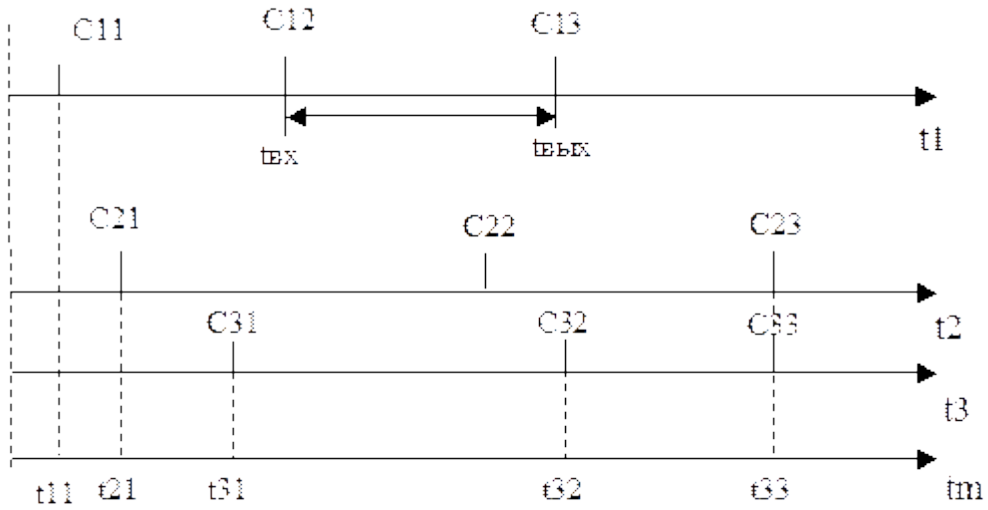

Рис. 5.6

Для каждого процесса строится своя цепь событий, однако списки событий являются общими для всей модели. Формирование списков начинается с заполнения списка будущих событий. Как было отмечено выше, в этот список помещаются события, время наступления которых превышает текущее значение модельного времени. Очевидно, что на момент заполнения списка время наступления прогнозируемых событий должно быть известно. На первом шаге tм=0, и в список будущих событий заносятся события C11, C21, C31. Затем событие с наименьшим временем наступления — C11 — переносится в список текущих событий; если одновременных с ним событий нет, то оно обрабатывается и исключается из списка текущих событий. После этого вновь корректируется список будущих событий и т.д., пока не истечет заданный интервал моделирования.

Динамика изменения списков текущих и будущих событий для рассмотренного примера отражена в приведенной ниже таблице.

t	***Список текущих событий***	***Список будущих событий***
0	*0*	*C11, C21,C31*

t11	*C11*	*C21, C31, C12*
t21	*C21*	*C31, C12, C22*
t31	*C31*	*C12, C22, C32*
t12	*C12*	*C22, C32, C13*
t22	*C22*	*C32, C13, C23*
t32	*C32*	*C13, C23, C33*
t13	*C13*	*C23, C33*
t23	*C23, C33*	

Многие авторы книг по имитационному моделированию считают, что знание механизма ведения списков событий просто необходимо разработчику модели; умение проследить в динамике цепь происходящих в модели событий, во-первых, повышает уверенность создателя модели в том, что она работает правильно и, во вторых, существенно облегчает процесс отладки и модификации модели.

6.1.3. Применение сетевых моделей для описания параллельных процессов

Этапу программной реализации модели (т. е. ее описанию на одном из языков программирования) должен предшествовать так называемый этап алгоритмизации. Другими словами, прежде чем превратить имитационную модель в работающую программу, ее создатель должен воспользоваться каким-то менее формальным и более наглядным средством описания логики работы будущей программы. Это требование не является обязательным, т.к. при наличии достаточного опыта программа не очень сложной модели может быть написана сразу. Однако при моделировании более сложных систем даже опытные разработчики бывают вынуждены немного «притормозить» на этапе алгоритмизации. Для описания логики работы модели могут быть использованы различные средства: либо русский язык (устный или письменный), либо традиционные схемы алгоритмов, либо какие-то другие «подручные» средства. Первые два варианта являются, как правило, наиболее знакомыми и наиболее часто используемыми. Однако такие схемы совершенно не приспособлены для описания параллельных процессов.

Одним из наиболее элегантных и весьма распространенных средств описания параллельных процессов — описание ***сетями Петри.*** Рассмотрим те основные сведения, которые необходимы с точки зрения реализации технологии имитационного моделирования параллельных процессов.

Одно из основных достоинств аппарата сетей Петри заключается в том, что они могут быть представлены как в графической форме (что обеспечивает наглядность), так и в аналитической (что позволяет автоматизировать процесс их анализа).

При графической интерпретации сеть Петри представляет собой граф особого вида, состоящий из вершин двух типов — *позиций* и *переходов,* соединенных ориентированными дугами, причем каждая дуга может связывать лишь разнотипные вершины (позицию с переходом или переход с позицией). Вершины-позиции обозначаются кружками, вершины-переходы — черточками. С содержательной точки зрения, переходы соответствуют событиям, присущим исследуемой системе, а позиции — условиям их возникновения. Таким образом, совокупность переходов, позиций и дуг позволяет описать причинно-следственные связи, присущие системе, но в статике. Чтобы сеть Петри «ожила», вводят еще один вид объектов сети — так называемые *фишки,* или метки позиций. Переход считается активным (событие может произойти), если в каждой его входной позиции есть хотя бы одна фишка. Расположение фишек в позициях сети называется ***разметкой сети*** (пример перемещения фишек по сети приведен на рис.5.6).

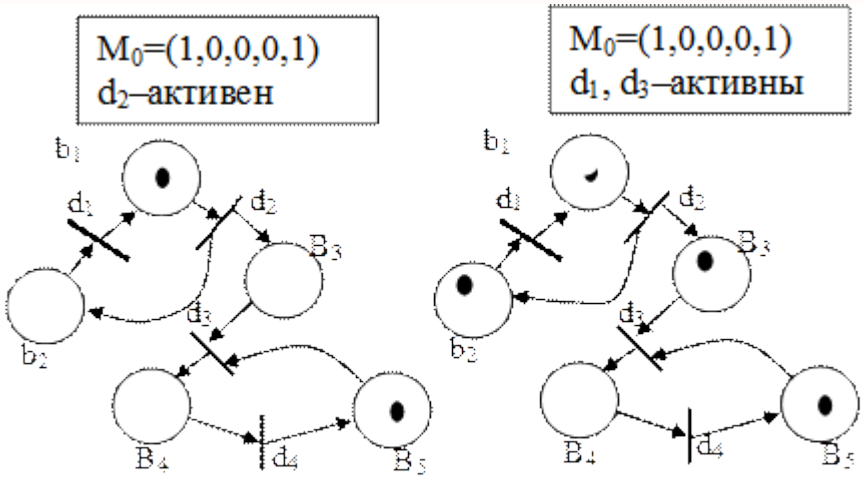

Рис. 5.6

В аналитической форме сеть Петри может быть представлена следующим образом:

$$P=(B,D,I,0,M),$$

где $B = \{b_i\}$ — конечное непустое множество позиций;

$D = \{d_i\}$ — конечное непустое множество переходов;

$I : B \times D \to 0,1$ — входная функция (прямая функция инцидентности), которая для каждого перехода задает множество его входных позиций;

$O : D \times B \to 0,1$ — выходная функция (обратная функция инцидентности), которая для каждого перехода задает множество его входных позиций;

M — функция разметки сети, $M : B \to 0, 1, 2,...$ — ставит каждой позиции сети в соответствие неотрицательное целое число.

С учетом введенных обозначений необходимое условие срабатывания перехода *dj* может быть записано следующим образом:

$$\forall b_i \in I(d_i) \{M(b_i) \geq 1\}$$

(*для всех входных позиций разметка должна быть >1*).

Срабатывание перехода d_j изменяет разметку сети *M(B)* на разметку *M′(B)* по следующему правилу:

$$M'(B)=M(B)-I(d_j)+O(d_j),$$

то есть переход d_j изымает по одной метке из каждой своей входной позиции и добавляет по одной метке в каждую из выходных позиций. Смену разметки обозначают так:

$$d_j$$

$$M_o /— M'$$

Входная и выходная функции сети Петри (***I*** и ***O***) позволяют описать любую сеть с помощью двух матриц размера *т х п* (матриц входных и выходных позиций), имеющих следующую структуру:

	d_1	d_2	...	d_j	...	d_n
b_1	0	1	...	0	...	0
b_2	1	1	...	0	...	1
...	...	...	...	...	...	...
b_j	0	1	...	0	...	1
...	...	...	...	...	...	...
b_m	1	0	...	1	...	0

Основные направления анализа сети Петри следующие:

1. *Проблема достижимости:* в сети Петри с начальной разметкой M_0 требуется определить, достижима ли принципиально некоторая разметка M' из M_0.

С точки зрения исследования моделируемой системы, эта проблема интерпретируется как проблема достижимости (реализуемости) некоторого состояния системы.

2. *Свойство живости.* Под живостью перехода понимают возможность его срабатывания в данной сети при начальной разметке M_0. Анализ модели на свойство живости позволяет выявить невозможные состояния в моделируемой системе (например, неисполняемые ветви в программе).

3. *Безопасность сети.* Безопасной является такая сеть Петри, в которой ни при каких условиях не может появиться более одной метки в каждой из позиций. Для исследуемой системы это означает возможность функционирования ее в стационарном режиме. На основе анализа данного свойства могут быть определены требования к буферным накопителям в системе.

Итак, достоинства сетей Петри заключаются в том, что они:

1) позволяют моделировать ПП всех возможных типов с учетом вероятных конфликтов между ними;

2) обладают наглядностью и обеспечивают возможность автоматизированного анализа;

3) позволяют переходить от одного уровня детализации описания системы к другому (за счет раскрытия/закрытия переходов).

Вместе с тем, сети Петри имеют ряд недостатков, ограничивающих их возможности. Основной из них — время срабатывания перехода считается равным 0, что не позволяет исследовать с помощью сетей Петри временные характеристики моделируемых систем.

В результате развития аппарата сетей Петри был разработан ряд расширений. Одно из наиболее мощных — так называемые .Е-сети (evaluation — «вычисления», «оценка») — «оценочные сети».

В отличие от сетей Петри, в Е-сетях:

1) имеются несколько типов вершин-позиций: простые позиции, позиции-очереди, разрешающие позиции;

2) фишки (метки) могут снабжаться набором признаков (атрибутов);

3) с каждым переходом может быть связана ненулевая задержка и функция преобразования атрибутов фишек;

4) введены дополнительные виды вершин-переходов.

5) в любую позицию может входить не более одной дуги и выходить также не более одной.

В связи с этим любой переход может быть описан тройкой параметров:

$$d_j=(S,t(d_j),\rho(dj)),$$

где S — тип перехода,

$t(d_j)$, — функция задержки,

$\rho(d_j)$ — функция преобразования атрибутов.

Особенности Е-сетей существенно расширяют их возможности для моделирования дискретных систем вообще и параллельных процессов в частности. Технология моделирования систем в виде Е-сетей может быть реализована с помощью инструмента SIMULINK, входящего в состав па