УДК 681.3

Б.Б. НЕСТЕРЕНКО, М.А. НОВОТАРСКИЙ

МОДЕЛИРОВАНИЕ СЛОЖНЫХ ДИСКРЕТНЫХ СИСТЕМ

Abstract: *New version of formal description for complex discrete systems (APRO-nets) is offered. Structural and algorithmic features of APRO-nets are stated. The example of model description for distributed computing system focused on processing of real working load is adduced.*
Key words: *APRO-net, position, simple transition, operational transition, token, action, process.*

Анотація: *Запропонована нова версія формального опису складних дискретних систем – APRO-мережі. Викладено структурні й алгоритмічні особливості APRO-мереж. Наведено приклад опису моделі розподіленої обчислювальної системи, орієнтованої на обробку реального робочого навантаження.*
Ключові слова: *APRO-мережа, позиція, простий перехід, операторний перехід, мітка, активність, процес.*

Аннотация: *Предложена новая версия формального описания сложных дискретных систем – APRO-сети. Изложены структурные и алгоритмические особенности APRO-сетей. Приведен пример описания модели распределенной вычислительной системы, ориентированной на обработку реальной рабочей нагрузки.*
Ключевые слова: *APRO-сеть, позиция, простой переход, операторный переход, метка, активность, процесс.*

1. Введение

За последние десятилетия существенное развитие получили дискретные системы [1, 2], то есть такие системы, функционирование которых может быть представлено совокупностью состояний, модифицируемых путем свершения событий [3]. Постоянное усложнение этих систем ведет к возникновению существенных проблем на этапе тестирования и настройки, что может послужить причиной снижения надежности, производительности или других эксплуатационных показателей. Поэтому весьма актуальными стали вопросы развития формальных средств, позволяющих адекватно описывать и исследовать упомянутые системы в виде моделей. Движущей силой к развитию формальных средств является противоречивость предъявляемых к ним требований. С одной стороны, упомянутые формальные средства должны допускать высокую степень абстрагирования от второстепенных свойств объекта с целью упрощения работы с моделью. С другой стороны, они должны быть достаточно мощными для обеспечения адекватности представления свойств и аспектов системы, существенных на этапах разработки или оценки. Современные формальные средства, кроме описания состояний и архитектуры системы, также должны включать операционную семантику, задающую правила поведения ее компонент. Исходя из указанной противоречивости требований к формальному описанию, высокой степени адекватности можно достичь, только если для каждого конкретного случая создать семантические правила, учитывающие требования, предъявляемые к модели, и особенности моделируемой системы.

Формальные средства для создания моделей последовательных систем хорошо изучены. Например, теория конечных автоматов [4] и дальнейшее ее развитие в виде теории агрегативных систем [5] позволяют исчерпывающим образом описывать детерминированные модели систем, функционирование которых может быть задано в виде единого процесса. Стохастические модели систем подобного типа хорошо представляет теория массового обслуживания [6]. Однако для адекватного представления параллельных или распределенных систем семантические правила формального описания должны учитывать существенно больше информации об организации

вычислений. Полезными могут оказаться данные о том, какие процессы развиваются независимо, когда необходимо учитывать причинно-следственные связи или производить выбор между альтернативными вариантами. В этом случае семантика носит композиционный характер, то есть позволяет построить глобальную модель системы путем установления взаимосвязей между локальными моделями ее компонент.

На сегодня наиболее развитым и хорошо разработанным средством формального описания параллельных и распределенных систем являются сети Петри [7], привлекающие к себе пристальное внимание как теоретиков, так и практических пользователей. За более чем сорокалетнюю историю своего развития сети Петри получили хорошую семантику, достаточно корректно отображающую естественные свойства их параллельности. Однако простота сетей Петри, являющаяся одной из причин их успеха, несет в себе ограничения для адекватного представления сложных систем. Если необходимо более детально описать структуру состояния системы, а также в случае, когда последовательность шагов невозможно свести к простой причинной зависимости или конфликту, сети Петри в большинстве случаев теряют свою адекватность.

Упомянутые ограничения стали одной из причин дальнейшего развития параллельных средств формального описания, расширяющих возможности сетей Петри и получивших название PRO-сетей [8]. Суть данного подхода состоит в отказе от задания жестких правил функционирования переходов за счет введения специальных процедур, управляющих режимами работы переходов и обработкой информации.

Асинхронные PRO-сети или APRO-сети являются новым шагом в развитии PRO-сетей и ориентированы на описание параллельных вычислительных асинхронных структур.

2. Структура APRO-сети

Представим APRO-сеть в виде кортежа:

$$\Phi = (P, T, F, M, V), \tag{1}$$

где $P = \{ p_i \}_{i=1}^{n}$ – конечное множество позиций;

$T = \{ t_j \}_{j=1}^{m}$ – конечное множество переходов;

$F = (P \times T) \cup (T \times P)$ – множество ребер между переходами и позициями;

$M = \left\{ \left(p_k, \{ \mu_l \}_{l=1}^{Max\_p_k} \right) \right\}_{k=1}^{n}$ – конечное множество маркирований;

$V = (\Delta, \Psi, \Lambda)$ – множество глобальных переменных.

Графические элементы APRO-сети представлены в табл. 1. Подобно графическим обозначениям сетей Петри, позиция APRO-сетей обозначается кружком, простой переход – линией, связи – линиями со стрелками, а метка – точкой. Дополнительно введены обозначения, связанные с элементами операторного перехода. Сам операторный переход изображается в виде прямоугольника, а входы и выходы операторного перехода представлены в виде правого и левого полукругов.

Таблица 1. Графические элементы APRO-сети

Обозначение			Графическое изображение	Название
$p \in P$				Позиция
$t \in T$	τ			Простой переход
	θ			Операторный переход
	θ	e_θ		Вход операторного перехода
		x_θ		Выход операторного перехода
$(p,t);(t,p) \in F$				Ребро
$\mu \in M$				Метка

Позиции APRO-сети: $p_i = \{d_i, q_i\}$, где $d_i = \{Id\_p_i, \varphi\_p_i, Cur\_p_i, Max\_p_i, \delta\_p_i\}$ – множество параметров позиции, содержащее такие элементы: Id_p_i – идентификатор позиции, φ_p_i – множество допустимых типов меток, Cur_p_i – текущее количество меток на позиции, Max_p_i – максимально допустимое количество меток, δ_p_i – локальный счетчик времени позиции, q_i – множество меток, размещенных на данной позиции.

Переходы APRO-сети включают два класса переходов: $t = \{\tau, \theta\}$, где τ – класс простых переходов, θ – класс операторных переходов. Простой переход описывает множество элементов: $\tau_j = \{x_j, N_j\}$, где $x_j = \{Id\_\tau_j, \delta\_\tau_j, \Delta\_\tau_j, Level\_\tau_j\}$ – множество параметров перехода, состоящее из таких элементов: Id_τ_j – идентификатор перехода, δ_τ_j – локальный счетчик времени перехода, Δ_τ_j – период простоя, $Level\_\tau_j$ – уровень перехода.

Функциональное ядро перехода: $N_j = \{\rho_j, \pi_j, \gamma_j, \omega_j, A_j, O_j\}$, где ρ_j – процедура активации перехода, π_j – процедура обслуживания перехода, γ_j – процедура деактивации перехода, ω_j – процедура ожидания, A_j – частично упорядоченная последовательность активностей, O_j – частично упорядоченная последовательность выходных меток.

Класс переходов θ обеспечивает композиционные свойства сети и представляет собой множество

$$\theta = (P_\theta, T_\theta, E_\theta, X_\theta, F_\theta), \quad (2)$$

где $P_\theta = \{(p_\theta)_1, \cdots (p_\theta)_a\}$, $a \in N$ – множество позиций;

$T_\theta = \{(\tau_\theta)_1, \cdots (\tau_\theta)_b\}$, $b \in N$, $T_\theta \neq \varnothing$ – непустое множество переходов;

$E_\theta = \{(e_\theta)_1, \cdots (e_\theta)_c\}$, $c \in N$, $E_\theta \neq \varnothing$ – непустое множество входов;

$X_\theta = \{(x_\theta)_1, \cdots (x_\theta)_d\}$, $d \in N$, $X_\theta \neq \varnothing$ – непустое множество выходов;

$F_\theta = (P_\theta \times T_\theta) \cup (T_\theta \times P_\theta) \cup (E_\theta \times T_\theta) \cup (T_\theta \times X_\theta)$ – множество ребер.

Таким образом, операторный переход θ содержит APRO-сеть Φ_θ, определяемую как сеть нижнего уровня по отношению к сети Φ. Переходы сети Φ_θ также могут быть операторными переходами, что позволяет строить вертикально стратифицированные модели с неограниченным количеством уровней.

Во избежание путаницы, элементы операторного перехода будем отмечать индексом с его наименованием. Формальное описание произвольной внутренней позиции $(p_\theta)_i \in \Phi_\theta$ операторного перехода θ соответствует формальному описанию произвольной позиции верхнего уровня $p_i \in \Phi$. Внутренние переходы $(\tau_\theta)_j \in \Phi_\theta$ также имеют идентичное формальное описание с простыми переходами верхнего уровня $\tau_j \in \Phi$.

Входы операторного перехода: $(e_\theta)_j = \{(\eta_\theta)_j, (N_\theta^e)_j, \mu_\theta\}$,

где $(\eta_\theta)_j = \{(Id\_e_\theta)_j, (\delta\_e_\theta)_j, (\Delta\_e_\theta)_j, (Level\_e_\theta)_j\}$ – множество параметров входа перехода θ, включающее такие элементы: $(Id\_e_\theta)_j$ – идентификатор входа, $(\delta\_e_\theta)_j$ – локальный счетчик времени, $(\Delta\_e_\theta)_j$ – период простоя, $(Level\_e_\theta)_j$ – уровень представления e_θ.

Ядро входа: $N_\theta^e = \{(\rho_\theta)_j, (w_\theta)_j\}$, где $(\rho_\theta)_j$ – процедура активации входа, $(w_\theta)_j$ – процедура ожидания, μ_θ – текущая входная метка операторного перехода θ.

Выходы операторного перехода: $(x_\theta)_i = \{(o_\theta)_i, (N_\theta^x)_i, (q_\theta^x)_i\}$, где $(o_\theta)_i = \{(Id\_x_\theta)_i, (\varphi\_x_\theta)_i, (Cur\_x_\theta)_i, (Max\_x_\theta)_i\}$ – множество параметров выхода перехода θ, включающее такие элементы: $(Id\_x_\theta)_i$ – идентификатор выхода, $(\varphi\_x_\theta)_i$ – множество допустимых типов меток, $(Cur\_x_\theta)_i$ – текущее количество меток, $(Max\_x_\theta)_i$ – максимально допустимое количество меток.

Ядро выхода: $(N_\theta)_i = \{(\gamma_\theta)_i\}$, где $(\gamma_\theta)_i$ – процедура деактивации выхода, $(q_\theta^x)_i$ – множество меток на выходе операторного перехода θ.

Ребра сети задают матрицей инцидентности H с элементами

$$H(p_i,t_j)=\begin{cases}-1, (p_i,t_j)\in \mathbf{F},\\ +1, (p_i,t_j)\in \mathbf{F}^{-1},\\ 0, \quad (p_i,t_j)\notin \mathbf{F}, (p_i,t_j)\notin \mathbf{F}^{-1},\end{cases} \quad \begin{matrix}1\le i\le n,\\ 1\le j\le m.\end{matrix}$$

Метки APRO-сети: $\mu_k=\{\lambda_k,\alpha_k\}$, где $\lambda_k=\{Id\_\mu_k,\delta\_\mu_k,\varphi\_\mu_k\}$ – множество параметров метки, включающее такие элементы: Id_μ_k – идентификатор метки, δ_μ_k – время создания метки, φ_μ_k – тип метки, α_k – множество атрибутов метки.

Множество глобальных переменных: $V=(\Delta,\Psi,\Lambda)$, где Δ – подмножество показателей продуктивности, Ψ – подмножество показателей реактивности, Λ – подмножество показателей использования.

3. Алгоритмические аспекты функционирования APRO-сети

Рассмотрим основные аспекты семантики APRO-сети, базирующиеся на процессах, активностях и событиях. Активности APRO-сети выражают совокупность типичных действий перехода на заданном уровне описания. При условии одновременного использования операторных и простых переходов сеть будет содержать активности верхнего уровня и внутренние активности операторных переходов, образуя двухуровневую структуру активностей. Если операторный переход нижнего уровня также содержит операторные переходы, то количество уровней активностей увеличивается на единицу. Таким образом, получаем древовидную структуру активностей. Логическая последовательность активностей объединяется в процесс. Любой из процессов, существующих в сети, может входить в процесс более высокого уровня в качестве субпроцесса.

Будем рассматривать множество активностей A, включающее такие подмножества: $A=\{compute,activate,deactivate,wait\}$. Подмножество $compute$ содержит активности, выполняющие действия, связанные с обработкой информации во время работы перехода. Подмножество $activate$ представляет активности, реализующие действия перехода по его активации, активности подмножества $deactivate$ завершают работу перехода. Активности, заданные подмножеством $wait$, отображают действия перехода в режиме ожидания. Упорядоченная последовательность активностей может образовывать процесс.

Определение 1. Пусть для простого перехода t_j задано множество активностей A_j и три произвольных элемента этого множества: $a_1,a_2,a_2\in A_j$. Множество A_j будем называть частично упорядоченной последовательностью, если для ее элементов задано отношение «$\preceq$» («$a_1\preceq a_2$» – a_1 равняется или предшествует a_2) в соответствии со свойствами:

– если $a_1\preceq a_2$ и $a_2\preceq a_3$ то $a_1\preceq a_3$ (транзитивность);

– если $a_1\preceq a_2$ и $a_2\preceq a_1$ то $a_1=a_2$ (асимметричность);

– $a_1\preceq a_1$ (рефлексивность).

Множеству переходов $T = \{t_1, \cdots, t_j, \cdots, t_m\}$ поставим в соответствие множество процессов $\mathbf{Pr} = \{Pr_1, \cdots, Pr_j, \cdots Pr_m\}$ и зададим подмножества активностей $activate := \{a_j\}_{j=1}^m$, $compute := \{c_j\}_{j=1}^m$, $deactivate := \{d_j\}_{j=1}^m$ и $wait := \{w_j\}_{j=1}^m$. Тогда процесс Pr_j перехода t_j можно представить последовательностью активностей $Pr_j := a_j \preceq c_j \preceq d_j \preceq w_j$. Процесс Pr_j отображает типичный процесс работы перехода, который сначала активируется, выполняет обработку информации, а потом завершается и переходит в режим ожидания.

Рождение и исчезновение активностей происходит путем свершения событий, любая из которых представляет собой мгновенное изменение состояния APRO-сети. Будем рассматривать только события $start$ и $stop$, обеспечивающие рождение и завершение активностей. Тогда процесс Pr_j может быть записан с учетом данных событий:

$$Pr_j := w_j.stop \preceq a_j.start \preceq a_j.stop \preceq c_j.start \preceq c_j.stop \preceq d_j.start \preceq d_j.stop \preceq w_j.start.$$

Совокупный процесс Pr_j всегда однозначно задает текущее состояние соответствующего перехода t_j. Поэтому в качестве общего признака упорядочения последовательности Pr_j выступает время возникновения активностей, входящих в данную последовательность.

На рис. 1 показана структура APRO-сети, состоящей из простого перехода t_j, входных позиций p_a, p_b и выходных позиций p_c, p_d.

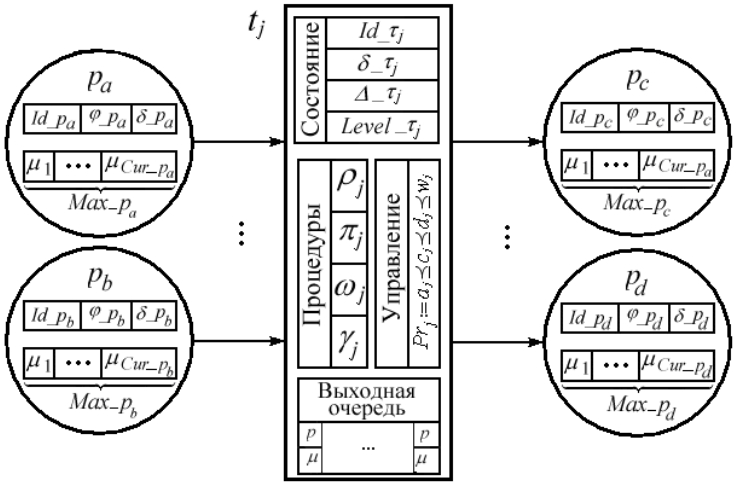

Рис. 1. Структура простого перехода APRO-сети

В соответствии с описанием структуры основными элементами позиций APRO-сети являются параметры состояния и список активных меток. Простой переход содержит параметры состояния, процедуры перехода, управляющий процесс и выходную очередь меток. Последовательное развитие процесса Pr_j происходит за счет того, что свершение текущего события сопровождается созданием нового события с новым временем выполнения. Стартовое событие всегда запускает новую активность посредством выполнения соответствующей процедуры перехода. Структура логических связей процедур перехода показана на рис. 2.

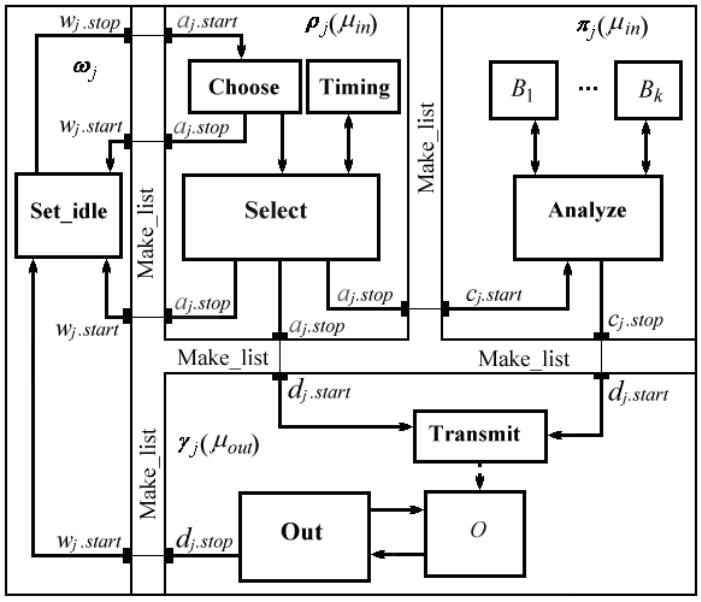

Рис. 2. Структура процедур $\rho_j(\mu_{in})$, $\pi_j(\mu_{in})$, $\gamma_j(\mu_{out})$ и ω_j

Процедура активации $\rho_j(\mu_{in})$ обеспечивает выполнение комплекса операций, соответствующих активности a_j, и состоит из процедур Choose, Select и Timing. Процедура Choose предназначена для выполнения действий, связанных с анализом меток, размещенных на входных позициях перехода t_j. Процедура Select реализует алгоритм проверки метки, выбранной с помощью процедуры Choose. Одно из важных условий проверки заключается в сохранении временной целостности модели путем сравнения локального времени перехода и времени создания метки.

Эту функцию выполняет отдельная процедура Timing. Запуск процедуры $\rho_j(\mu_{in})$ происходит в момент завершения активности w_j и начала активности a_j. Механизм этого изменения состоит в том, что завершающее событие $w_j.stop$ активности w_j порождает стартовое событие $a_j.start$ активности a_j.

Существуют четыре возможных завершения процедуры $\rho_j(\mu_{in})$ в зависимости от результатов анализа входной метки μ_{in}:

1. Если поиск активной метки на входных позициях завершился безрезультатно, то событие завершения активности a_j порождает событие $w_j.start$.

2. Если активная метка μ_{in} найдена процедурой Choose, но значение атрибутов этой метки не разрешает запустить процедуру обработки $\pi_j(\mu_{in})$, то событие завершения активности a_j также порождает событие $w_j.start$.

3. Если активная метка μ_{in} найдена процедурой Choose и имеет корректные параметры, то событие завершения активности a_j порождает событие $c_j.start$.

4. Если активная метка μ_{in} найдена процедурой Choose и является транзитной меткой, то событие завершения активности a_j порождает событие $d_j.start$.

Процедура $\pi_j(\mu_{in})$ реализует совокупность операций, соответствующих активности c_j перехода t_j и состоит из процедуры Analyze и множества процедур $\{B_i(\alpha)|\alpha \in \mu\}_{i=1}^{k}$. Запуск этой процедуры происходит в случае, если метка корректна и может быть обработана в данном переходе, что подтверждается созданием события $c_j.start$. Процедура Analyze обеспечивает оценку параметра φ_μ_{in} текущей метки μ_{in} с целью выбора соответствующей процедуры $B_k(\alpha)\big|_{\alpha\in\mu_{in}}$. Работу процедуры Analyze завершает событие $c_j.stop$, порождающее событие $d_j.start$.

Процедура $\gamma_j(\mu_{out})$ включает операции вывода меток из простого перехода t_j. В ее состав входят процедуры Transmit и Out. Существуют два варианта запуска процедуры $\gamma_j(\mu_{out})$, каждый из которых инициируется соответствующим событием $d_j.start$. Процедура Transmit формирует параметры и атрибуты выходной метки и размещает их в очереди выходных меток. Процедура Out выбирает первую в очереди выходную метку μ_{out} и пытается установить ее на соответствующую выходную позицию. Если метка по тем или иным причинам не установлена на указанной выходной позиции, то она снова возвращается в очередь выходных меток. При условии пустоты очереди выходных меток O_j работа процедуры $\gamma_j(\mu_{out})$ завершается, о чем свидетельствует событие $d_j.stop$, свершение которого порождает событие $w_j.start$.

В соответствии с рис. 2 существуют три возможных причины возникновения события $w_j.start$, запускающего процедуру ожидания ω_j. Основной составляющей данной процедуры является процедура Set_idle, устанавливающая действия перехода после завершения полного цикла обработки информации. Управление частично упорядоченной последовательностью активностей в рамках перехода выполняет процедура Make_list обслуживания последовательности, входящей в состав ядра N_j перехода t_j.

Для формирования следующего цикла активности перехода необходимо выбрать способ продвижения сетевого времени. Этот способ должен быть согласован с обусловленным механизмом имитационного моделирования сети. Поэтому решение проблемы продвижения сетевого времени лежит в плоскости практической реализации алгоритмов функционирования APRO-сети. Формальное описание APRO-сети допускает использование как последовательных, так и параллельных методов продвижения модельного времени.

Наиболее полно разработан метод продвижения сетевого времени с помощью управления частично упорядоченной последовательностью меток, являющийся вариантом подхода, который базируется на продвижении сетевого времени, управляемого списком событий [9]. Этот вариант может быть использован для организации вычислительного процесса в мультипроцессорной вычислительной структуре. Особенность такой структуры состоит в наличии управляющего центра, обеспечивающего поддержку процедур работы с единой, частично упорядоченной, последовательностью меток. Следует отметить, что реализация данного подхода для иерархических сетей вызывает ряд проблем, связанных с осложнениями обслуживания общей последовательности меток для таких сетей. А именно: использование единого списка значительно увеличивает его размер и количество сопроводительной информации. Для решения этой проблемы необходимо обеспечить формирование иерархической последовательности меток, но такой подход усложняет процессы управления сетью. Данный алгоритм продвижения сетевого времени также может вызвать зацикливание в случае, если на сети существуют циклические связи между переходами и время прохождения таких циклов меньше времени срабатывания других переходов сети. Для преодоления упомянутых недостатков в последнее десятилетие активно развиваются подходы к построению локальных последовательностей меток, которые могли бы функционировать без использования глобальных ресурсов при реализации сетевой модели на мультипроцессорной системе [10].

4. Пример описания APRO-сети

Рассмотрим пример композиционного построения APRO-сетевой модели распределенной вычислительной системы с использованием операторных переходов (рис. 3).

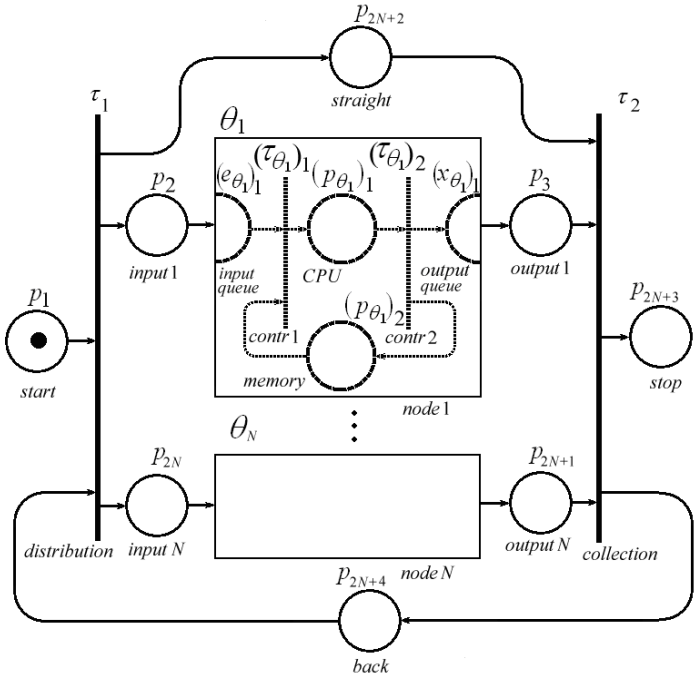

Рис. 3. Модель распределенной вычислительной системы

Объект моделирования представлен простым переходом «*distribution*», выполняющим функции распределения вычислительной нагрузки между N узлами, заданными с помощью

множества операторных переходов $\{node\, i \,|\, 1 \le i \le N\}$. Формирование общего результата из частичных результатов вычислений в узлах выполняет простой переход «*collection*». Переходы «*distribution*» и «*collection*» реализуют обмен данными между узлами путем использования связей через позиции «*straight*» и «*back*». В функции перехода «*collection*» также входит завершение процесса моделирования и установление состояния «*stop*».

Структурно APRO-сеть $\Phi_d = (P, T, F, M, V)$ включает такие элементы:

$$P = \{p_1, p_2, \cdots p_{2N+4}\},$$

$$p_1 = \{\{start, data, 1, N, \delta\_\mu_1\}, \{\mu_1\}\},$$

$$p_2 = \{\{input\,1, data, 0,1,0\}, \varnothing\}, \ \ldots, \ p_{2N} = \{\{input\ N, data, 0,1,0\}, \varnothing\},$$

$$p_3 = \{\{output\,1, data, 0,1,0\}, \varnothing\}, \ldots, p_{2N+1} = \{\{output\ N, data, 0,1,0\}, \varnothing\},$$

$$p_{2N+2} = \{\{straight, data, 0, N, 0\}, \varnothing\}, \ p_{2N+3} = \{\{stop, data, 0, N, 0\}, \varnothing\},$$

$$p_{2N+4} = \{\{back, data, 0, N, 0\}, \varnothing\},$$

$$T = \{\tau_1, \tau_2, \theta_1, \cdots \theta_N\},$$

$$\tau_1 = \{\{distribution, 0,0,0\}, N_1\}, \tau_2 = \{\{collection, 0,0,0\}, N_2\}.$$

Операторные переходы $\theta_i = (P_{\theta_i}, T_{\theta_i}, E_{\theta_i}, X_{\theta_i}, F_{\theta_i})$, $1 \le i \le N$ содержат следующие элементы:

$$P_{\theta_i} = \{(p_{\theta_i})_1, (p_{\theta_i})_2\}, \ (p_{\theta_i})_1 = \{\{CPU, data, 0,1,0\}, \varnothing\}, \ (p_{\theta_i})_2 = \{\{memory, data, 0,1,0\}, \varnothing\},$$

$$T_{\theta_i} = \{(\tau_{\theta_i})_1, (\tau_{\theta_i})_2\}, (\tau_{\theta_i})_1 = \{\{contr\,1, 0,0,1\}, (N_{\theta_i})_1\}, \ (\tau_{\theta_i})_2 = \{\{contr\,2, 0,0,1\}, (N_{\theta_i})_2\},$$

$$E_{\theta_i} = \{(e_{\theta_i})_1\}, \ (e_{\theta_i})_1 = \{\{input\ queue, 0,0,0,10,1\}, (N^e_{\theta_i})_1, \varnothing\},$$

$$X_{\theta_i} = \{(x_{\theta_i})_1\}, \ (x_{\theta_i})_1 = \{\{output\ queue, data, 0,10\}, (N^x_{\theta_i})_1, \varnothing\},$$

$$F_{\theta_i} = \begin{pmatrix} & (e_{\theta_i})_1 & (p_{\theta_i})_1 & (p_{\theta_i})_2 & (x_{\theta_i})_1 \\ (\tau_{\theta_i})_1 & -1 & 1 & -1 & 0 \\ (\tau_{\theta_i})_2 & 0 & -1 & 1 & 1 \end{pmatrix},$$

$$F = \begin{pmatrix} & p_1 & p_2 & p_3 & \cdots & p_{2N} & p_{2N+1} & p_{2N+2} & p_{2N+3} & p_{2N+4} \\ \tau_1 & -1 & 1 & 0 & \cdots & 1 & 0 & 1 & 0 & -1 \\ \tau_2 & 0 & 0 & -1 & \cdots & 0 & -1 & -1 & 1 & 1 \\ \theta_1 & 0 & -1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \theta_N & 0 & 0 & 0 & \cdots & -1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Предложенная APRO-сетевая модель описывает дискретную распределенную систему обработки данных. Особенность данного подхода состоит в возможности имитационного

моделирования систем с реальной рабочей нагрузкой, что обусловлено использованием произвольных структур данных в качестве атрибутов меток и реализацией произвольных алгоритмов обработки этих данных с помощью процедур обработки переходов. APRO-сетевое описание является основой входного языка задания параметров моделей для моделирующего комплекса APRO_Simulator. С помощью формирования массивов статистических данных в соответствии с показателями, объединенными во множество глобальных переменных V, данный комплекс позволяет производить анализ различных характеристик сложных дискретных систем как на этапе их разработки, так и на этапе эксплуатации.

5. Выводы

Стремительное усложнение дискретных систем стимулирует поиск новых подходов к созданию их формальных описаний, лежащих в основе построения различного рода моделей. С одной стороны, такие описания должны обладать высокой степенью абстрагирования с целью упрощения модели, а, с другой стороны, они должны обладать высокой адекватностью описания объектов моделирования для обеспечения корректности полученных результатов. С учетом данных требований разработан инструмент формального описания сложных дискретных систем – APRO-сети.

Научная новизна предлагаемой версии сетевого формального описания сложных дискретных систем состоит в наличии совокупности таких свойств:

– гибкого механизма иерархического представления моделей;

–возможности описания параллельно функционирующих и асинхронно взаимодействующих процессов с использованием современных подходов к продвижению модельного времени;

– возможности построения имитационных моделей с использованием реальной рабочей нагрузки.

Практическая ценность работы обусловлена построением программного комплекса, представляющего собой среду моделирования, в которой входной язык для описания моделей построен с использованием синтаксиса и семантических правил APRO-сетей.

СПИСОК ЛИТЕРАТУРЫ

1. Rosenberg R.M. Analytical Dynamics of Discrete Systems. – New York: Plenum Pub. Corp., 1977. – 424 p.
2. Робертс Ф.С. Дискретные математические модели с приложениями к социальным, биологическим и экологическим задачам. – М.: Наука, 1986. – 496 с.
3. Cassandras C.G., Lafortune S. Introduction to Discrete Event Systems. – New York: Springer, 2006. – 904 p.
4. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979. – 232 с.
5. Бусленко Н.П. Моделирование сложных систем.– М.: Наука, 1978. – 400 с.
6. Гнеденко Б.В., Коваленко И.Н. Введение в теорию массового обслуживания. – М.: Наука, 1987.– 336 с.
7. Питерсон Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 264 с.
8. Нестеренко Б.Б., Новотарский М.А. Мультипроцессорные системы. − К.: Институт математики АН Украины, 1995. − 408 с.
9. Нестеренко Б.Б. Моделювання паралельних процесів: від мереж Петрі до нейронних мереж / Б.Б. Нестеренко, М.А. Новотарський. – К.: 2004. – 66 с. (Препринт / НАН України. Ін-т математики; 2004-2).
10. Fujimoto R.M. Parallel Discrete Event Simulation // Communication of the ACM. – 1990. – Vol. 33, N 10. – P. 30–53.

Стаття надійшла до редакції 11.05.2007