УДК 519.876.5

НЕСТЕРЕНКО Б.Б.,
НОВОТАРСЬКИЙ М.А.

APRO-МЕРЕЖІ ДЛЯ МОДЕЛЮВАННЯ СКЛАДНИХ ДИСКРЕТНИХ СИСТЕМ

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**.**

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 loading is resulted.

Вступ

Сучасні дискретні системи, у відповідності до загальної теорії систем, задають деякою сукупністю станів, модифікація яких можлива шляхом виконання переходів під дією активностей. Оскільки значна частина пристроїв, машин та механізмів, створених людиною, відповідає означенню дискретних систем, розробці інструментів для їх дослідження приділяють значну увагу. Однією з важливих проблем є створення формальних засобів опису дискретних систем з метою побудови моделей, які б адекватно відображали властивості об’єкта моделювання.

Формальні засоби для створення моделей послідовних систем добре вивчені. Наприклад, теорія скінченних автоматів [1] і подальший її розвиток у вигляді теорії агрегативных систем [2] дозволяють вичерпно описати детерміновані моделі систем, функціонування яких зазвичай задають у вигляді єдиного процесу. Стохастичні моделі систем подібного типу добре представляє теорія масового обслуговування [3]. Однак для адекватного представлення паралельних або розподілених систем семантичні правила формального опису повинні враховувати істотно більше інформації про організацію обчислень. Корисними можуть виявитися дані про те, які процеси розвиваються незалежно, якщо необхідно враховувати причинно-наслідкові зв'язки або робити вибір між альтернативними варіантами.

Для опису паралельних систем широкого застосування набули формальні засоби, представлені різними версіями мереж Петрі [4]. Їх успіх базується на тому факті, що опис об’єкта моделювання множиною станів і алгоритмів обробки уніфікує процес переходу від конкретної моделі до абстрактної, тобто такої, яка дозволяє застосувати формальні процедури аналізу. В результаті накопичення досвіду застосування мереж Петрі по-

ряд з безумовними перевагами поступово проявилися також і їх недоліки, основний з яких полягає у використанні фіксованих правил спрацювання переходів. Незважаючи на те, що доведена функціональна повнота цих правил, застосування їх до опису роботи реальних об'єктів призводить до невиправдано громіздких моделей. Цей факт став причиною виникнення як численних модифікацій мереж Петрі, так і нових мереж. Очевидно, що все розмаїття навколишнього світу неможливо вкласти в прокрустове ложе обмеженої множини алгоритмів, що і є внутрішнім стимулом розвитку теорії мережного моделювання.

Згадані обмеження стали однією з причин подальшого розвитку паралельних засобів формального опису, що розширюють можливості мереж Петрі. Такі формальні засоби одержали назву PRO-мережі [5], яка походить від слів «процеси» та «процесори». В основі PRO-мереж лежить відмова від задавання жорстких правил функціонування переходів за рахунок введення спеціальних процедур, що керують режимами роботи переходів і обробкою інформації. Асинхронні PRO-мережі або APRO-мережі є новим кроком розвитку PRO-мереж і орієнтовані на опис паралельних обчислювальних асинхронних структур.

Структура APRO-мережі

Представимо APRO-мережу у вигляді кортежу:

$$\Phi = (P, T, F, M, V), \qquad (1)$$

де $P = \{ p_i \}_{i=1}^{n}$ – скінченна множина позицій,

$T = \{t_j\}_{j=1}^{m}$ – скінченна множина переходів,

$F = (P \times T) \cup (T \times P)$ – множина ребер між переходами й позиціями,

$M = \left\{ \left(p_k, \{\mu_l\}_{l=1}^{Max\_p_k} \right) \right\}_{k=1}^{n}$ – скінченна множина маркувань,

$V = (\Delta, \Psi, \Lambda)$ – множина глобальних змінних.

Подібно графічним позначенням мереж Петрі, позиція APRO-мереж позначається кружком, простий перехід - лінією, зв'язки - лініями зі стрілками, а мітка - крапкою. Додатково введені позначення, пов'язані з елементами операторного переходу. Сам операторний перехід зображується у вигляді прямокутника, а входи й виходи операторного переходу представлені у вигляді правого й лівого півкіл.

Позиції APRO-мережі: $p_i=\{d_i,q_i\}$, де d_i – множина параметрів позиції, q_i – множина міток, розміщених на даній позиції.

Переходи APRO-мережі включають два класи переходів: $t =\{\tau,\theta\}$, де τ – клас простих переходів, θ – клас операторних переходів. Простий перехід описує множина елементів: $\tau_j = \{\chi_j, N_j\}$, де χ_j – множина параме-

трів переходу, $N_j = \{\rho_j, \pi_j, \gamma_j, \omega_j, A_j, O_j\}$ – функціональне ядро переходу, що включає такі елементи: ρ_j – процедура активації переходу, π_j – процедура обслуговування переходу, γ_j – процедура деактивації переходу, ω_j – процедура очікування, A_j – частково впорядкована послідовність активностей, O_j – частково впорядкована послідовність вихідних міток.

Клас переходів θ забезпечує композиційні властивості мережі і представлений множиною:

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

де $P_\theta = \{(p_\theta)_1, \cdots (p_\theta)_a\}$, $a \in N$ – множина позицій;

$T_\theta = \{(\tau_\theta)_1, \cdots (\tau_\theta)_b\}$, $b \in N$, $T_\theta \neq \varnothing$ – непуста множина переходів;

$E_\theta = \{(e_\theta)_1, \cdots (e_\theta)_c\}$, $c \in N$, $E_\theta \neq \varnothing$ – непуста множина входів;

$X_\theta = \{(x_\theta)_1, \cdots (x_\theta)_d\}$, $d \in N$, $X_\theta \neq \varnothing$ – непуста множина виходів;

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

Таким чином, операторний перехід θ містить APRO-мережу Φ_θ, яка є мережею нижнього рівня стосовно мережі Φ. Переходи мережі Φ_θ також можуть бути операторними переходами, що дозволяє будувати вертикально стратифіковані моделі з необмеженою кількістю рівнів.

Щоб уникнути плутанини, елементи операторного переходу будемо позначати індексом з його найменуванням. Формальний опис довільної внутрішньої позиції $(p_\theta)_i \in \Phi_\theta$ операторного переходу θ відповідає формальному опису довільної позиції верхнього рівня $p_i \in \Phi$. Внутрішні переходи $(\tau_\theta)_j \in \Phi_\theta$ також мають ідентичний формальний опис із простими переходами верхнього рівня $\tau_j \in \Phi$.

Входи операторного переходу: $(e_\theta)_j = \{(\eta_\theta)_j, (N_\theta^e)_j, \mu_\theta\}$, де $(\eta_\theta)_j$ – множина параметрів входу переходу θ; $N_\theta^e = \{(\rho_\theta)_j, (w_\theta)_j\}$ – ядро входу, що включає процедуру активації входу $(\rho_\theta)_j$ та процедуру очікування $(w_\theta)_j$; μ_θ – поточна вхідна мітка операторного переходу θ. Виходи операторного переходу: $(x_\theta)_i = \{(o_\theta)_i, (N_\theta^x)_i, (q_\theta^x)_i\}$, де $(o_\theta)_i$ – множина параметрів виходу переходу θ; $(N_\theta)_i = \{(\gamma_\theta)_i\}$ – ядро виходу, яке містить

процедуру деактивації виходу $(\gamma_\theta)_i$; $\left(q_\theta^x\right)_i$ – множина міток на виході операторного переходу θ. Ребра мережі задають матрицею інцидентності H з елементами:

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

Мітки APRO-мережі: $\mu_k=\{\lambda_k,\alpha_k\}$, де λ_k – множина параметрів мітки, α_k – множина атрибутів мітки. Множина глобальних змінних: V=(Δ, Ψ, Λ), де Δ – підмножина показників продуктивності, Ψ – підмножина показників реактивності, Λ – підмножина показників використання.

Алгоритмічні аспекти функціонування APRO-мережі

Розглянемо основні аспекти семантики APRO-мережі, що базуються на процесах, активностях і подіях. Активності APRO-мережі виражають сукупність типових дій переходу на заданому рівні опису. За умови одночасного використання операторних і простих переходів мережа буде містити активності верхнього рівня й внутрішні активності операторних переходів, утворюючи дворівневу структуру активностей. Якщо операторний перехід нижнього рівня також містить операторні переходи, то кількість рівнів активностей збільшується на одиницю. Таким чином, одержуємо деревоподібну структуру активностей. Логічна послідовність активностей утворює процес. Кожний з процесів, що існують у мережі, може входити в процес більш високого рівня як субпроцес.

Будемо розглядати множину активностей A, що включає такі підмножини: $A=\{compute, activate, deactivate, wait\}$. Підмножина *compute* містить активності, що виконують дії, пов'язані з обробкою інформації під час роботи переходу. Підмножина *activate* представляє активності, що реалізують дії переходу, необхідні для його активації, активності підмножини *deactivate* завершують роботу переходу. Активності, задані підмножиною *wait*, відображають дії переходу в режимі очікування. Упорядкована послідовність активностей може утворювати процес.

Множині переходів $T=\{t_1, \Lambda, t_j, \Lambda, t_m\}$ поставимо у відповідність множину процесів Pr=$\{Pr_1, \Lambda, Pr_j, \Lambda, Pr_m\}$ і задамо підмножини активностей:

$activate := \{a_j\}_{j=1}^m$, $compute := \{c_j\}_{j=1}^m$, $deactivate := \{d_j\}_{j=1}^m$,

$wait := \{w_j\}_{j=1}^m$.

Тоді процес Pr_j переходу t_j можна представити послідовністю активностей: $Pr_j := \{a_j, c_j, d_j, w_j\}$. Процес Pr_j відображає типовий процес роботи

переходу, що спочатку активується, виконує обробку інформації, а потім завершується й переходить у режим очікування.

Народження й зникнення активностей відбувається шляхом здійснення подій, кожна з яких є миттєвою зміною стану APRO-мережі. Будемо розглядати тільки події *start* та *stop*, що забезпечують виникнення й завершення активностей. Тоді процес Pr_j може бути записаний з урахуванням даних подій:

$$Pr_j := \{w_j.stop, a_j.start, a_j.stop, c_j.start, c_j.stop, d_j.start, d_j.stop, w_j.start\}$$

Сукупний процес Pr_j завжди однозначно задає поточний стан відповідного переходу t_j. Тому загальною ознакою впорядкування послідовності Pr_j виступає час виникнення активностей, що входять у дану послідовність.

На рис.1 показана структура APRO-мережі, що складається із простого переходу τ_j, вхідних позицій $p_a, \cdots, p_b$ і вихідних позицій $p_c, \cdots, p_n$.

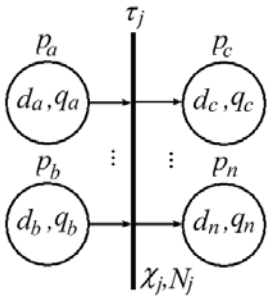

Рис. 1. Структура простого переходу APRO-мережі

Відповідно до опису структури, основними елементами позицій APRO-мережі є параметри стану $d_a, \cdots d_b, d_c, \cdots d_n$ й списки активних міток $q_a, \cdots q_b, q_c, \cdots q_n$. Простий перехід τ_j містить параметри стану χ_j і ядро N_j. Послідовний розвиток процесу Pr_j відбувається за рахунок того, що здійснення поточної події супроводжується створенням нової події з новим часом виконання. Стартова подія завжди запускає нову активність за допомогою виконання відповідної процедури переходу. Структура логічних зв'язків процедур переходу, що входять до складу ядра N_j, показана на рис. 2.

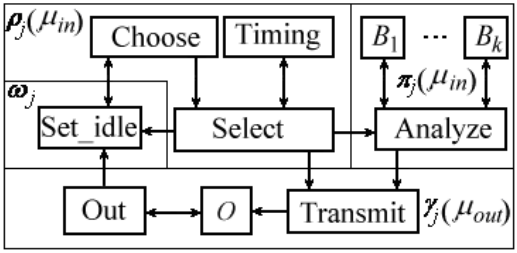

Рис. 2. Структура процедур $\rho_j(\mu_{in})$, $\pi_j(\mu_{in})$, $\gamma_j(\mu_{out})$ *і* ω_j

Процедура активації $\rho_j(\mu_{in})$ забезпечує виконання комплексу операцій, що відповідають активності a_j й складається із процедур Choose, Select і Timing. Процедура Choose призначена для виконання дій, пов'язаних з аналізом міток, розміщених на вхідних позиціях переходу τ_j. Процедура Select реалізує алгоритм перевірки мітки, обраної за допомогою процедури Choose. Одна з важливих умов перевірки полягає в збереженні часової цілісності моделі шляхом порівняння локального часу переходу й часу створення мітки.

Цю функцію виконує окрема процедура Timing. Запуск процедури $\rho_j(\mu_{in})$ відбувається в момент завершення активності w_j і початку активності a_j. Механізм цієї зміни полягає в тому, що завершальна подія $w_j.stop$ активності w_j породжує стартову подію $a_j.start$ активності a_j. Існує чотири можливих завершення процедури $\rho_j(\mu_{in})$ залежно від результатів аналізу вхідної мітки μ_{in}:

1. Якщо пошук активної мітки на вхідних позиціях завершився безрезультатно, то подія завершення активності a_j породжує подію $w_j.start$.

2. Якщо активна мітка μ_{in} знайдена процедурою Choose, але значення атрибутів цієї мітки не дозволяють запустити процедуру обробки $\pi_j(\mu_{in})$, то подія завершення активності a_j також породжує подію $w_j.start$.

3. Якщо активна мітка μ_{in} знайдена процедурою Choose і має коректні параметри, то подія завершення активності a_j породжує подію $c_j.start$.

4. Якщо активна мітка μ_{in}, яка знайдена процедурою Choose, є транзитною міткою, то подія завершення активності a_j породжує подію $d_j.start$.

Процедура $\pi_j(\mu_{in})$ реалізує сукупність операцій, що відповідають активності c_j переходу τ_j й складається із процедури Analyze і множини

процедур $\{B_i(\alpha)|\alpha \in \mu\}_{i=1}^{k}$. Запуск цієї процедури відбувається у випадку, якщо мітка коректна й може бути оброблена в даному переході, що підтверджується створенням події $c_j.start$. Процедура Analyze забезпечує оцінку параметрів поточної мітки μ_{in} з метою вибору відповідної процедури $B_k(\alpha)\big|_{\alpha\in\mu_{in}}$. Робота процедури Analyze завершується подією $c_j.stop$, яка породжує подію $d_j.start$.

Процедура $\gamma_j(\mu_{out})$ включає операції виводу міток із простого переходу τ_j. До складу $\gamma_j(\mu_{out})$ входять процедури Transmit і Out. Існує два варіанти запуску процедури $\gamma_j(\mu_{out})$, кожний з яких ініціюється відповідною подією $d_j.start$. Процедура Transmit формує параметри й атрибути вихідної мітки й розміщує їх у черзі вихідних міток O_j. Процедура Out вибирає першу в черзі вихідну мітку μ_{out} й намагається встановити її на відповідну вихідну позицію. Якщо мітка з тих чи інших причин не встановлена на зазначеній вихідній позиції, то вона знову повертається в чергу вихідних міток. За умови відсутності черги вихідних міток O_j робота процедури $\gamma_j(\mu_{out})$ завершується, про що свідчить подія $d_j.stop$, здійснення якої породжує подію $w_j.start$. Відповідно до рис.2, подія $w_j.start$ завжди запускає процедуру очікування ω_j. Основною складовою даної процедури є процедура Set_idle, що встановлює дії переходу після завершення повного циклу обробки інформації.

Приклад опису APRO-мережі

Розглянемо приклад композиційної побудови APRO-мережної моделі розподіленої обчислювальної системи з використанням операторних переходів (рис. 3).

Об'єкт моделювання включає простий перехід «*distributor*», що виконує функції розподілу обчислювального навантаження між N вузлами, заданими за допомогою множини операторних переходів $\{node\,i\,|\,1 \le i \le N\}$. Формування загального результату із часткових результатів обчислень у вузлах виконує простий перехід «*collector*». Переходи «*distributor*» і «*collector*» реалізують обмін даними між вузлами шляхом використання зв'язків через позиції «*straight*» і «*back*». Перехід «*collector*» також завершує процес моделювання шляхом установки стану «*stop*».

Структурно APRO-мережа $\Phi_d = (P, T, F, M, V)$ включає такі елементи: $P = \{p_1, p_2, \cdots p_{2N+4}\}$,

Позиція $p_1 = \{\{start, data, 1, N, \delta\_\mu_1\}, \{\mu_1\}\}$ містить параметри $d_1 = \{start, data, 1, N, \delta\_\mu_1\}$ і мітки $q_1 = \{\mu_1\}$. Елементи множини параметрів, зокрема, мають такі значення: *start* – назва позиції, *data* – назва типу міток, що можуть розміщуватись на даній позиції, 1– поточна кількість міток на позиції, N – максимальна кількість міток на позиції, δ_μ_1 – значення локального лічильника часу, що дорівнює часу створення поточної мітки. За даною схемою сформовані параметри решти позицій:

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

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

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

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

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

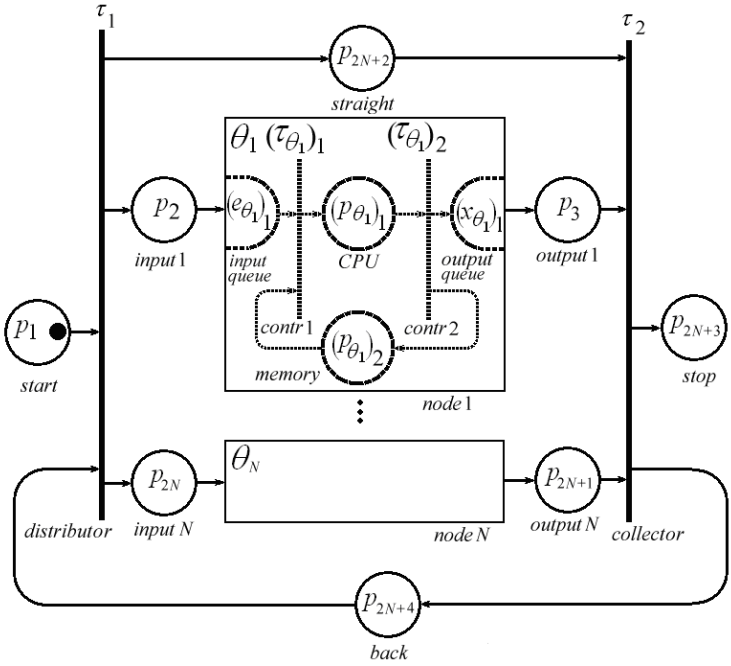

Рис. 3. Модель розподіленої обчислювальної системи

Перехід $\tau_1 = \{\{distributor, 0,0,0\}, N_1\}$ включає параметри $\chi_1 = \{distributor, 0,0,0\}$ і ядро N_1. Елементи множини параметрів мають такі значення: *distributor* – назва переходу, 0 – значення локального лічильника часу переходу, 0 – загальний час простою переходу, 0 – номер рівня переходу. За такими ж правилами сформований перехід

$\tau_2 = \{\{collector, 0,0,0\}, N_2\}$. Операторні переходи $\theta_i = \left(P_{\theta_i}, T_{\theta_i}, E_{\theta_i}, X_{\theta_i}, F_{\theta_i}\right)$, $1 \le i \le N$ містять наступні елементи:

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

$\left(p_{\theta_i}\right)_2 = \{\{memory, data, 0,1,0\}, \varnothing\}$,

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

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

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

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

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

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

Запропонована APRO-мережна модель описує дискретну розподілену систему обробки даних. Особливість даного підходу полягає в можливості імітаційного моделювання систем з реальним робочим навантаженням, що обумовлено використанням довільних структур даних в атрибутах міток і реалізацією довільних алгоритмів обробки цих даних за допомогою процедур обробки переходів. APRO-мережний опис є основою вхідної мови задавання параметрів моделей для моделюючого комплексу APRO_Simulator. За допомогою формування масивів статистичних даних відповідно до показників, об'єднаних в множину глобальних змінних V, даний комплекс дозволяє аналізувати різні характеристики складних дискретних систем як на етапі їх розробки, так і на етапі експлуатації.

Висновки

Стрімке ускладнення дискретних систем стимулює пошук нових підходів до створення їх формальних описів, що лежать в основі побудови різного роду моделей. Такі описи повинні мати високий ступінь абстрагування з метою спрощення моделі і одночасно забезпечувати високу адекватність опису об'єктів моделювання для отримання коректних результатів. З урахуванням даних вимог розроблений інструмент формального опису складних дискретних систем – APRO-мережі.

Наукова новизна запропонованої версії мережного формального опису складних дискретних систем обумовлена сукупністю таких властивостей:

- гнучкого механізму ієрархічного представлення моделей;
- можливості опису паралельно функціонуючих і асинхронно взаємодіючих процесів з використанням сучасних підходів до просування модельного часу;
- можливості побудови імітаційних моделей з використанням реального робочого навантаження.
- Практична цінність роботи полягає у тому, що побудовано програмний комплекс, який є середовищем моделювання з вхідною мовою, створеною на основі синтаксису та семантики APRO-мереж.

Список посилань

1. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979. – 232 с.
2. Бусленко Н.П. Моделирование сложных систем.– М.: Наука, 1978.– 400 с.
3. Гнеденко Б.В., Коваленко И.Н. Введение в теорию массового обслуживания.– М.: Наука, 1987.– 336 с.
4. Питерсон Дж. Теория сетей Петри и моделирование систем.– М.: Мир, 1984.– 264 с.
5. Нестеренко Б.Б., Новотарский М.А. Мультипроцессорные системы.− К.: Институт математики АН Украины, 1995. − 408 с.