УДК 681.3

СИСТЕМИ АЛГОРИТМІЧНИХ АЛГЕБР. ПРИКЛАДНИЙ АСПЕКТ

С.Д. Погорілий, О.О. Камардіна

Київський національний університет ім. Тараса Шевченка
01033, Київ, вул. Володимирська, 64,
тел. : (044) 526 0522, e-mail: sdp@rpd.univ.kiev.ua, olya@tviy.com.ua

Запропоновано підхід до формалізації етапу алгоритмічного проектування мікропроцесорних систем (МПС) на основі математичного апарату модифікованих систем алгоритмічних алгебр В.М. Глушкова. Створено формалізовані специфікації протоколу OSPF та схеми паралельних алгоритмів Флойда - Уоршала і Джонсона. Описано інструментальні засоби підтримки комп'ютерної технології алгоритмічного проектування МПС.

An approach for algorithmic design stage formalization of microprocessor's system (MPS) which is based on use the V.M. Glushkov mathematical apparatus of algorithmic algebras modified systems is proposed. The formalized specifications of the OSPF protocol and Floyd-Warshall's and Jonson's algorithms scheme are created. The support instrumental means of Computer technology algorithmic design of MPS are described.

Вступ

Багатофункціональність та складність сучасних мікропроцесорних систем (МПС) із вбудованими мікропроцесорами вимагають створення нових засобів та комп'ютерних технологій, що дозволяють адекватно описувати та моделювати їх на різних етапах проекту [1 - 3]. Алгоритмічний етап проектування є початковим та найменш дослідженим, але він суттєво впливає на весь подальший процес розробки та визначає її успіх у цілому. Засоби комп'ютерних технологій цього етапу проектування повинні забезпечувати:

- ефективні методи опису алгоритмів функціонування МПС на різних етапах проекту;
- можливість функціональних та еквівалентних формальних трансформацій алгоритмів з метою їх дослідження та аналізу;
- методи опису темпоральних (залежних від часу) та асинхронних фрагментів алгоритмів;
- можливість опису взаємодії алгоритмів різних рівнів керування в МПС: алгоритмічного програмного та апаратного.

Навіть поверхневий аналіз цих складових переконує у складності етапу алгоритмічного проектування, що призводить до значної кількості трансформацій алгоритмів та для ефективної реалізації цього процесу застосування формалізованих методів і специфікацій.

У статті висвітлено досвід застосування для розв'язання вищезазначених задач математичного апарату модифікованих систем алгоритмічних алгебр (САА-М) В.М. Глушкова [4 - 6]. Методологія САА-М належить до засобів програмування, що грунтуються на формалізованих специфікаціях алгоритмів в межах їх трансформаційного синтезу. Проектування алгоритмів за допомогою САА-М-схем дозволяє здійснювати глибокий структурний аналіз алгоритмів та програм (в тому числі паралельних), на основі якого може бути здійснена їх оптимізація за обраними критеріями. Формалізовані засоби аналізу структури алгоритмів та програм у процесі їх проектування в явному вигляді відсутні в сучасних алгоритмічних мовах програмування високого рівня, а їх застосування сприяє відкриттю та обгрунтуванню якісно нових властивостей алгоритмів. Для успішного та ефективного застосування МПС паралельної архітектури користувачам необхідно мати можливість одержувати найдетальніші відомості про різноманітні деталі структури своїх алгоритмів та асоційованих із ними програм. Тому засади формування методів для одержання та використання таких відомостей, пякі пов'язані з дослідженням тонкої інформаційної структури алгоритмів і асоційованих із ними програм, є вельми важливою складовою етапу алгоритмічного проектування МПС.

Орієнтація на застосування математичного апарату САА-М, що є різновидом систем алгоритмічних алгебр (САА) [2,7,8], обгрунтовується такими чинниками:

- апарат САА-М у порівнянні з класичними алгоритмічними системами орієнтований на проектування алгоритмів (та асоційованих з ними програм);
- будь-який алгоритм, записаний в САА-М, являє собою алгебраїчну формулу, що дозволяє застосовувати до цього алгоритму систему достатньо розвинених в САА-М формальних трансформацій з метою оптимізації і подальшого автоматизованого синтезу програмного забезпечення;
- САА-М відповідають концепції структурного програмування;
- наявність досвіду авторів трансформації послідовних схем алгоритмів протоколів маршрутизації [9-11].

У процесі розвитку досліджень особлива увага приділялась аналізу запису алгоритмів за допомогою математичних співвідношень та програм, записаних послідовними мовами. Цей підхід пояснюється тим фактом, що саме в цьому вигляді записано найбільшу кількість світового багажу алгоритмічних знань [12].

1. Формалізація специфікацій протоколу OSPF в термінах САА

Розглянемо застосування математичного апарату САА для розв'язання прикладної задачі маршрутизації в мережах, яка є вельми актуальною, і ефективність функціонування мереж значною мірою визначається методом розв'язання цієї задачі. Дослідження в галузі маршрутизації проводяться вже тривалий час, і деякі їх результати розглянуто в [13,14].

Основним елементом мережі, що розв'язує задачу маршрутизації, є маршрутизатор (Router), який зі структурної точки зору є МПС із фіксованим (тобто записаним у постійний запам'ятовуючий пристрій) програмним забезпеченням.

З використанням математичного апарату САА запишемо алгоритм роботи маршрутизатора мережі, що користується протоколом OSPF версії 2 [15-17] при розрахунку маршрутів в середині автономної системи (AS).

Структури даних алгоритму

Спочатку визначимо структури даних алгоритму, при цьому користуємося даними з [9,11].

start (root) — 32-bit ID вузла-джерела;

LS_infinity — максимальна вага маршруту;

Граф, що є математичною моделлю реальної мережі, представляє тут відповідно розмічений список вузлів-кандидатів, а в базі даних знаходяться стани зв'язків (LSA):

L_tree – список дерева найкоротшого шляху;

L_candidate – список вузлів-кандидатів;

| – вказівник в списку **L_candidate**;

|| – вказівник в списку **L_tree**;

⌉ та **•** – початок і, відповідно, кінець списків;

i_V – поточний вузол, що проглядається;

D – змінна поточної відстані.

САА-схема обчислення таблиці маршрутизації (СОТМ)

Цю схему можна зобразити у вигляді такого неінтерпретованого виразу:

$$\mathbf{COTM=A1*_{\alpha}\{A2\}*A3*A4*A5},$$

де оператори

$\mathbf{A_1}$ – маркування поточної таблиці маршрутизації як недійсної зі зберіганням її копії (тут прийнято наступне правило для позначення операторів та предикатів: перший номер означає позицію в найбільш абстрагованій неінтерпретованій схемі, номера через крапку - позицію в більш детальній схемі нижчого рівня. Наприклад, $\mathbf{A_{2.1}}$ - перший елемент другого оператора абстрагованої схеми,

$\mathbf{\alpha=(}$all areas processed$\mathbf{)}$ – умова того, що всі OSPF-області обраховані;

$\mathbf{A_2}$ – обрахування маршрутів всередині однієї OSPF-області з

$$\mathbf{A_2=A_{2.1}*A_{2.1}},$$

де оператори

$\mathbf{A_{2.1}}$ – розрахунок маршрутів між маршрутизаторами та транзитними мережами;

$\mathbf{A_{2.2}}$ – додавання до таблиці маршрутизації маршрутів до тупикових мереж;

$\mathbf{A_3}$ – розрахунок маршрутів між OSPF-областями;

$\mathbf{A_4}$ – додавання маршрутів через OSPF-області, які можуть передавати інформацію;

$\mathbf{A_5}$ – розрахунок зовнішніх, по відношенню до автономної системи, маршрутів.

Розглянемо докладніше одну з головних частин алгоритму – $\mathbf{A_{2.1}}$.

$\mathbf{A_{2.1}}$ – розрахунок маршрутів між маршрутизаторами та транзитними мережами всередині однієї OSPF-області (наприклад, для OSPF-області А), модифікований алгоритм Дейкстри.

$$\mathbf{A_{2.1}=A_{2.11}*\{A_{2.12}*A_{2.13}\}_{\alpha 2.1}},$$

де оператори

$\mathbf{A_{2.11}}$ – ініціалізація структур данних алгоритму;

$\mathbf{A_{2.12}}$ – розрахунок;

$\mathbf{A_{2.13}=A_{2.13.1}*A_{2.13.2}*A_{2.13.3}*A_{2.13.4}}$– можлива модифікація таблиці маршрутизації

та умова

$$\mathbf{\alpha=(L\_candidate=\varnothing)}.$$

Докладніше розглянемо оператори та умови нижчого рівня:

$$\mathbf{A_{2.11}=A_{2.11.1}*A_{2.11.2}*A_{2.11.3}},$$

де

$A_{2.11.1}$=(L_candidate:=∅);

$A_{2.11.2}$=(A.TransitCapability:=FALSE);

$A_{2.11.3}$=(L_tree:=Vertex(root));

Назвемо вузол, що тільки-но додали до дерева найкоротших шляхів, вузлом **V**. Позначимо вузол призначення з поточної LSA (об'яви про стан зв'язку), що аналізується, вузлом **W**.

$A_{2.12}=A_{2.12.1}*A_{2.12.2}*_{\alpha 2.12}\{A_{2.12.3}\}$;

$A_{2.12.1}$ – пошук в базі даних стану зв'язку області A інформації, асоційованої з ID поточного вузла, запит всіх LSA;

$A_{2.12.2}=\beta_{2.12.3}$(TransitCapability:=true),

де

$\beta_{2.12.3}=(\alpha_{2.12.21}\wedge\alpha_{2.12.22})$

$\alpha_{2.12.21}$=(LSA належить маршрутизатору);

$\alpha_{2.12.22}$=(біт V LSA маршрутизатора встановлено в TRUE);

$\alpha_{2.12}$=(II=•);

$A_{2.12.3}=A_{2.12.31}*\Rightarrow_{lsa}$,

де $\Rightarrow_{lsa}$ – перехід до наступного зв'язку в LSA;

$A_{2.12.31}=\alpha_{2.12.31}(A_{2.12.31.1}*\beta_{2.12.31}(A_{2.12.31.2}))$ – обробка LSA,

де оператори

$A_{2.12.31.1}$ – пошук LSA вузла W в базі даних області A;

$\beta_{2.12.31}=\alpha_{2.12.31.1}\wedge\alpha_{2.12.31.2}\wedge\alpha_{2.12.31.3}\wedge\alpha_{2.12.31.4}$,

де

$\alpha_{2.12.31.1}$=(LSA існує);

$\alpha_{2.12.31.2}$=(LS_age НЕ= MaxAge);

$\alpha_{2.12.31.3}$=(має зворотній зв'язок з вузлом V);

$\alpha_{2.12.31.4}$=((L_tree∩vertex(W))=∅);

$A_{2.12.31.2}=A_{2.12.31.21}*\beta_{2.12.31.21}(A_{2.12.31.22}\vee\alpha_{2.12.31.22}$(NHC(W,V))).

Оператор **NHC(W,V)**, що виконує пошук наступного стрибка (hop) для вузла **W**, описано нижче.

$A_{2.12.31.21}$=(D:=V.cost+LS_cost(V,W));

$\beta_{2.12.31.21}=\alpha_{2.12.31.21.1}\vee\alpha_{2.12.31.21.2}$;

$\alpha_{2.12.31.21.1}$=((L_candidate∩vertex(W))=∅);

$\alpha_{2.12.31.21.2}$=(D<(W.cost in L_candidate));

$A_{2.12.31.22}$=(L_candidate:=L_candidate+vertex(W))*NHC(W,V).

$A_{2.13.1}$ – обираємо вузол, який належить **L_candidate**, що знаходиться ближче всіх до вузла-джерела (якщо є вибір, то згідно протокола, обираємо спочатку вузол, котрий є мережею);

$A_{2.13.2}$=(L_tree:=L_tree∪i_V);

$A_{2.13.3}$=(L_candidate:=L_candidate\i_V);

$A_{2.13.4}$ – Модифікація таблиці маршрутизації: область, що асоціюється, встановлюємо в область A, значення типу шляху встановлюємо в значення ваги нового відкритого шляху).

Розрахунок наступного стрибка (NHC(W,V))

Наведемо також алгоритм розрахунку наступного пересилання (стрибка), що викликається після завершення основного алгоритму. Саме цей алгоритм реально будує таблицю маршрутизації. Єдиним обмеженням є те, що цей алгоритм не обробляє віртуальні зв'язки.

У оператора **NHC(W,V)** – аргументами є:

W – вузол призначення;

V – батьківський вузол.

NHC(W,V)=$\gamma_1(B_1\vee B_2)$,

де предикат

γ_1=(чи є вузол V маршрутизатором?)

та оператори

$B_1=\gamma_{1.1}(B_{1.1}\vee B_{1.2})$;

B_2:=(список W.NH визначається шляхом перевірки LSA вузла W, які мають зворотній зв'язок з вузлом V);

$\gamma_{1.1}$=(чи є вузол V джерелом?);

$$B_{1.1}=\gamma_{1.11}\left(B_{1.11}*\prod\begin{pmatrix} B_{1.12} & B_{1.13} \\ \gamma_{1.12} & \gamma_{1.13}\end{pmatrix}*E\right);$$

$B_{1.2}$=(W.NH:=V.NH) –список наступних стрибків для вузла W наслідується від вузла V;

$\gamma_{1.11}$=(чи є V маршрутизатором, НЕ з'єднаним з джерелом через віртуальний зв'язок?);

$B_{1.11}$=(W.NH.interface:=OSPF-інтерфейс, котрий веде до вузла W**);**

$\gamma_{1.12}$=(чи з'єднаний V через мережу Точка-Багато Точок**);**

$B_{1.12}$=(W.NH.IP визначається з LSA вузла W**);**

$\gamma_{1.13}$=(чи V – напряму підключена до джерела?**)∨(**чи V підключена за допомогою з'єднання Точка-Точка до джерела?**);**

$B_{1.13}$=(W.NH.IP:=" "**).**

У такий спосіб отримано формалізацію компонентів OSPF-протоколу маршрутизації в середині AS, яка відкриває шлях для формальних трансформацій отриманих схем з метою оптимізації частини або всього протоколу. Оптимізацію можна проводити за кількома критеріями, наприклад за часом виконання або об'ємом пам'яті, що використовується системою маршрутизації [10]. Приклади застосування такого підходу детально розглянуто в [7,18]. Використання методики дозволяє не тільки виконувати аналіз формалізованих специфікацій протоколів, що існують, а й фомалізований синтез перспективних протоколів маршрутизації.

2. Створення схем паралельних алгоритмів перспективних протоколів маршрутизації

Задача маршрутизації в Internet розв'язується мільйони разів на добу. Тому час виконання пошуку оптимального маршруту стає критичним параметром алгоритмів маршрутизації і визначає ефективність функціонування мережі в цілому. Це вимагає максимально можливого підвищення продуктивності маршрутизаторів.

Висновки експертів у галузі комп'ютерних технологій збігаються в тому, що ресурс екстенсивного зростання продуктивності МПС внаслідок підвищення складності та тактової частоти мікропроцесорів себе вичерпав. В подальших дослідженнях та розробках акцент перенесено на створення багатоядерних процесорів (свідоцтвом тому є створення таких процесорів фірмами Intel, SUN Microsystems та IBM) і паралельних алгоритмів.

У зв'язку з цим група з розвитку Internet IRTF (Internet Research Task Force), яка координує довгострокові дослідницькі проекти за стеком протоколів TCP/IP, займається створенням нових протоколів та пошуком нових, перспективних методів розв'язання задачі маршрутизації для подолання існуючих недоліків на якісно новому рівні.

Алгоритм Флойда - Уоршала розглядається як альтернативний варіант застосування в перспективних протоколах маршрутизації [19-20]. Це обумовлюється такими властивостями:

- ефективність на щільних графах;
- можливість знаходження найкоротших шляхів для всіх пар вершин орієнтованого графа (це спрощує знаходження альтернативних шляхів);
- пропорційність часу знаходження найкоротших шляхів у графі, що має n вершин, величині n^3;

Існує багато архітектур, що можуть забезпечити підвищення швидкодії алгоритму, і важливим класом серед них є паралельні. В зв'язку цим далі викладено застосування математичного апарату САА-М з метою створення схем паралельних версій алгоритму Флойда - Уоршала [21, 22].

В [19] одержано формалізовану наступну схему:

A0
(n=card(V))*
($D^{(0)}$ = W)
(формування $П^{(0)}$)
***({**
k in V
{
i in V
{ ($d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$**)***
j in V
***(** $(П_{ij}^{(k)} = П_{ij}^{(k-1)}) \vee (П_{ij}^{(k)} = П_{kj}^{(k-1)})$
$d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

(j=наступний вузол)}
(i=наступний вузол)}
(k наступний вузол)})
***(return $D^{(n)}$,$П^{(k)}$),**
де оператори:

- **n=card(V);**
- **$D^{(0)}$ = W;**
- **формування $П^{(0)}$;**

- $d_{ij}^{(k)}$=**min(** $d_{ij}^{(k-1)}$, $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ **);**
- $\Pi_{ij}^{(k)} = \Pi_{ij}^{(k-1)}$;
- $\Pi_{ij}^{(k)} = \Pi_{kj}^{(k-1)}$;
- **(j=наступний вузол);**
- **(i=наступний вузол);**
- **(k=наступний вузол);**
- **return D$^{(n)}$,П$^{(k)}$,**

та предикати:

- **i in V;**
- **j in V;**
- **k in V;**

$d_{ij}^{(k-1)} <= d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$;

V – множина вершин графа;
W – матриця суміжності, що відображає топологію мережі;
D – матриця ваг найкоротших шляхів;

- d_{ij} – елемент матриці D, що знаходиться в і-му стовпчику та j-му рядку;
- D$^{(0)}$ – матриця, всі елементи якої визначаються за правилом:

$d_{ij}^{0} = Вага\ ребра(i, j)\ i \neq j$, якщо у графі існує ребро (дуга) (i,j).

- Вазі ребра d_{ij}^{k} на k-му кроці присвоюється найменше з двох можливих значень:
  - Вага ребра (i,j) на (k-1)-му кроці ($d_{ij}^{(k-1)}$);
  - Вага суми ребер (i,k)+(k,j) на (k-1)-му кроці ($d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$);
- D$^{(n)}$ – результуюча матриця ваг найкоротших шляхів;

Π – матриця передування, що містить найкоротші шляхи;

- π_{ij} – елемент матриці Π, що знаходиться в і-му стовпчику та j-му рядку
- $\pi_{ij}^{(k)}$ – вершина, яка передує вершині j на найкоротшому шляху з вершини і у вершину j з проміжними вершинами з множини {1, 2, …, k}
- $\Pi^{(0)}$ – матриця, всі елементи якої визначаються за правилом:

$$\pi_{ij}^{(0)} \begin{cases} nil,\ якщо\ i = j\ або\ вага\ ребра(i, j)\ дорівнює\ \infty \\ i,\ якщо\ i \neq j\ та\ вага\ ребра(i, j) < \infty \end{cases}.$$

Далі відповідно до [19] застосуємо математичний аппарат САА-М для створення схем паралельних версій алгоритму Флойда-Уоршалла. В результаті одержимо схеми.

A1.1

{[U$_3$] # F # * # {[U$_5$] {[U$_4$] N}} # } *K

де

$d_{ij}^{(k-1)} < d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ = U$_1$;

$d_{ij}^{(k)} := d_{ij}^{(k-1)}$ = A;

$d_{ij}^{(k)} := d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ = B (тоді прийдемо до такої короткої форми: $d_{ij}^{(k)}$ = min($d_{ij}^{(k-1)}$, $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$) = ([U$_1$]A,B));

P$_i$::=buf = F;

$d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ = U$_2$;

$\Pi_{ij}^{(k)} = \Pi_{ij}^{(k-1)}$ = G;

$\Pi_{ij}^{(k)} = \Pi_{kj}^{(k-1)}$ = H;

k in V = U$_3$;
j in V = U$_4$;
i<nrl+nr = U$_5$;
return D$^{(n)}$,П$^{(k)}$ = K;
([U$_1$]A,B)*([U$_2$](G ν H),E)= N;
nrl – номер першого рядка в блоці;
nr – загальна кількість рядків в блоці.

A1.2

$\underline{U_3} \parallel \overline{\underline{U}}_3$ *** # F # * # (**$\underline{U_5} \parallel \overline{\underline{U}}_5$ *** {[U₄] N }) # * {[U₃] # F # }*K**

A1.3

$\underline{U_3} \parallel \overline{\underline{U}}_3$ *** # F # * # (**$\underline{U_5} \parallel \overline{\underline{U}}_5$ *** (({[U₄ v** $\overline{\underline{U}}_1$ **] A}* $\underline{U_4}$ **|| {[U₄ v $\underline{U_1}$] B}*$\underline{U_4}$) || {[U₄ v** $\overline{\underline{U}}_2$ **](G v H) })) # * ({[U₃] F})*K**

У схемах А1.2 та А1.3 застосовано позначення схеми А1.1, а $\underline{U_i}$ означає унарну операцію фільтрації.

Виконана система еквівалентних та функціональних перетворень схеми А0 свідчить про зручність, природність та ефективність застосування математичного апарату САА-М для виконання трансформацій схем алгоритмів МПС, що проектуються. Крім того, вельми важливо відзначити особливість цієї методики: ***САА-М є зручним та потужним засобом фіксації схем паралельних алгоритмів***.

В [23, 24] виконано систему перетворень схем алгоритму Джонсона для розв'язання задачі маршрутизації в розріджених мережах та здобуто низку параметричних регулярних схем паралельного алгоритму.

У процесі проведення досліджень створено формалізовані специфікації вихідних послідовних алгоритмів Флойда - Уоршала та Джонсона, створено їх паралельні версії, а потім шляхом функціональних та еквівалентних трансформацій утворено параметричні регулярні схеми паралельних алгоритмів. З метою експериментальної перевірки властивостей "спектрів" схем виконано їх моделювання на кластері Київського національного університету імені Тараса Шевченказ з використанням різноманітних підходів [20, 24]:

- процесо-орієнтованої парадигми;
- потоко-орієнтованої парадигми;
- парадигми MPI.

В результаті моделювання виконано аналіз швидкодії створених параметричних регулярних схем паралельних алгоритмів та розроблено рекомендації щодо їх застосування.

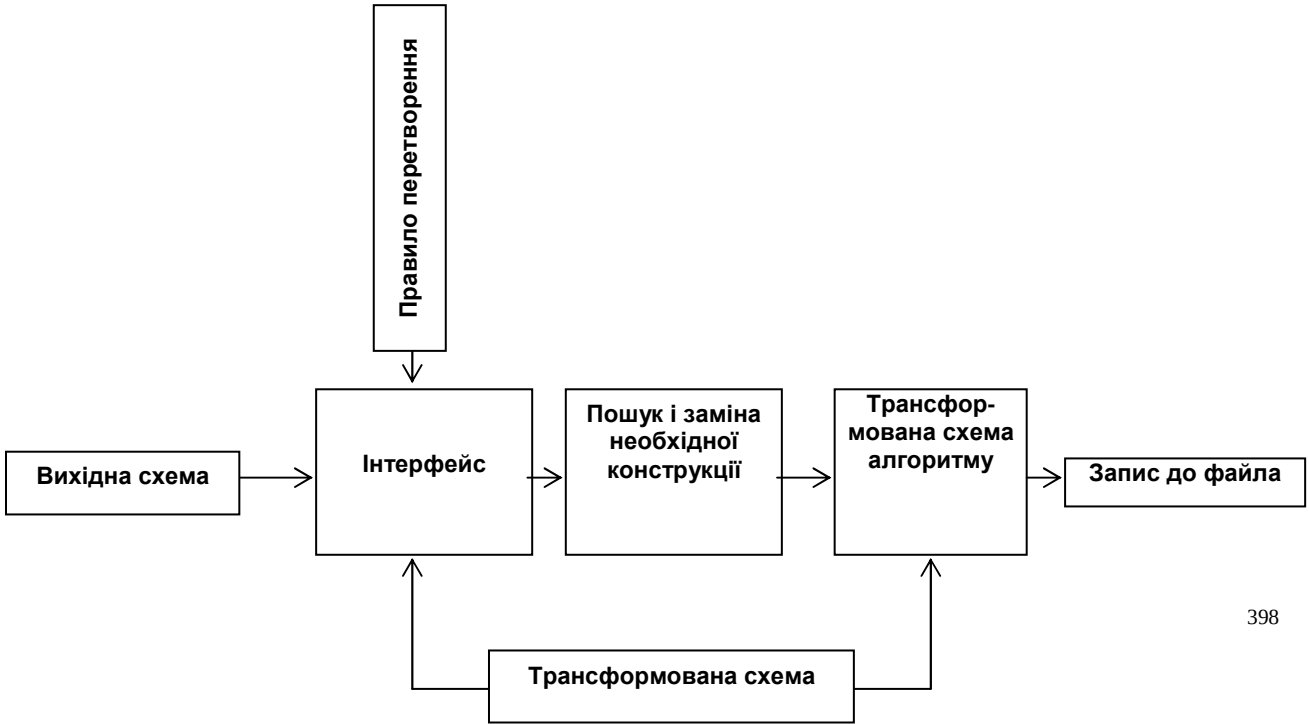
3. Інструментальні засоби комп'ютерних технологій на основі САА-М

Застосування технологій, що грунтуються на математичному апараті САА-М, вимагає наявності інструментальних засобів їх підтримки, які розглянуто нижче.

Система трансформації схем алгоритмів

Проектування алгоритмічного забезпечення МПС передбачає опис формалізованих специфікацій схем алгоритмів та їх подальшу трансформацію. На початку для нескладних алгоритмів ця задача вирішувались досить просто – користувач міг за кілька годин відшукати всі необхідні конструкції і модифікувати та замінити їх еквівалентними. Але для складних алгоритмів МПС, трансформація може займати значно більше часу, а це призведе до збільшення кількості помилок. Тому було запропоновано створити інструментальну систему трансформації схем алгоритмів [25, 26]. Створення такої системи надає можливість швидкої трансформації алгоритмів, прискорює процес дослідження і надає можливість формування протоколу трансформацій схем алгоритмів.

В процесі реалізації інструментальної системи трансформації алгоритмів було сформовано граматику регулярного типу для вихідних схем алгоритмів в САА-М. На рис. 1 наведено технологічний ланцюжок



виконання автоматизованої трансформації схем.

Рис. 1

Після дослідження і вибору необхідних механізмів і засобів створення системи автоматизованої трансформації схем алгоритмів розпочалася робота саме створення системи. Для створення системи, яка б максимально відповідала концепції відкритих систем, було обрану операційну систему UNIX FreeBSD 5.21, до складу якої входить інтерпретатор мови PERL 5.6 [27]. Попередні версії FreeBSD містили реалізацію мови Perl, яка не мала вбудованого механізму динамічних регулярних виразів, через що вони були відкинуті. Система створювалась як консольнt застосування (інтерфейс командного рядка).

Система моделювання та оцінки алгоритмів

З метою оптимізації алгоритму, що досліджується необхідно мати можливість досить просто модифікувати його і оцінювати отриману модифікацію відповідно до обраного критерію [9, 28]. Для автоматизації процесу оцінки алгоритмів було створено **С**истему **М**оделювання та **О**цінки **А**лгоритмів (СМОА) [29].

У більшості алгоритмічних мов програмування високого рівня не існує правил (формул) модифікації записаного алгоритму, тому не можна стверджувати, що дві модифікації алгоритму еквівалентні за певним критерієм. Крім того, ці мови програмування apriori спираються на певну архітектуру ЕОМ, що значною мірою ставить користувача у межі цієї моделі.

Множина операторів та виразів вхідної мови опису алгоритмів СМОА є надмножиною відповідних множин операторів і предикатів САА. Введено явний оператор присвоювання, у зв'язку з чим поняття предикату було розширено до поняття виразу та реалізована можливість оформлення виразу або його частини у формі функції. Реалізовано абстрактний тип даних як список скалярних величин, до елементів якого можна звертатись за допомогою вказівників. Внесено розширення, що не суперечать принципам САА та пов'язані зі зверненням до елементів списку як до масиву за індексами елементів. Множина операторів доповнена операторами введення-виведення інформації.

При проектуванні СМОА ставилась мета створити незалежну від платформи систему, яка б давала змогу обчислити параметри виконання алгоритму на будь-якій апаратній платформі з будь-якою ОС. Для цього необхідно обрати стандартизовану мову програмування, яка реалізована на більшості апаратно-програмних платформ. Виходячи з цього, для реалізації системи СМОА було обрано стандартну мову ANSI Сі, яка повністю задовольняє цим вимогам.

Ствренння програми, налагодження та тестування виконано на платформі HP Apollo 7100 з операційною системою HP-UX 10.20, компілятор Сі – GCC v 2.8.1 виробництва GNU Open Source Software Foundation [30]. Роботу системи протестовано на HP-UX 9.x/10.20, FreeBSD 5.2.x, Linux 6.2.x та з невеликими змінами, пов'язаними з особливостями компілятора системи програмування Borland C++Builder на програмній платформі Windows XP.

Для забезпечення більшої гнучкості системи вона реалізована у вигляді двох взаємопов'язаних компонентів: транслятора САА-схеми у внутрішній проміжний байт-код і процесора байт коду, який власне і виконує алгоритм.

Транслятор САА-схем (рис. 2 а) являє систему, що складається з трьох компонентів: синтаксичного та лексичного аналізаторів та зв'язувача [31-33]. На вхід транслятора подається САА-схема, а на виході отримуємо байт-код.

Вхідна САА-схема записується у файлі опису алгоритму. Синтаксис внутрішньої мови опису алгоритмів детально описано в документації системи та на web-сторінці проекту [12].

Процесор байт-коду (рис 2 б) можна використовувати як у вигляді окремої програми, так і у комплексі з транслятором, без збереження проміжного байт-коду.

Рис. 2, а

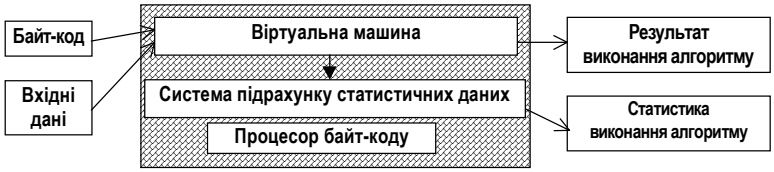


Рис. 2, б

Проміжний байт-код є мобільним між платформами, тобто він не залежить ані від ОС, ані від типу МП, ані від порядку байтів у машинному слові. Іншими словами, байт-код, створений транслятором на одній апаратно-програмній платформі, "зрозуміє" процесор байт-коду на іншій платформі.

Процесор байт-коду, по суті, являє реалізацію вибраної моделі комп'ютера, тобто його можна назвати ***віртуальною машиною***, яка виконує алгоритм, але крім віртуальної машини до складу процесора входить підсистема підрахунку статистичних даних.

У віртуальній машині використано модель комп'ютера, що складається з двох частин, які взаємодіють: процесор і пам'ять довільного доступу, пов'язані шиною, за якою пересилаються дані. Ця модель відповідає моделі машини фон Неймана [28] та відображає реальну архітектуру сучасних комп'ютерів. Варто відзначити, що у створеній системі моделювання алгоритмів підсистема, яка реалізує віртуальну машину, є автономною і дозволяє підключати моделі різноманітних архітектур:

- гарвардської;
- стенфордської;
- берклійської;
- кластерної;
- трансп’ютерної тощо.

Це дозволяє оцінювати ефективність алгоритмів при їх моделюванні на комп’ютерах із різними архітектурами.

Система синтезу програм за схемами їх алгоритмів у візуальних середовищах програмування

Наявність сучасних візуальних середовищ програмування (Delphi, Borland C++ Builder, Visual C++ тощо [34- 35] та розвиток методів формалізованого синтезу програм висувають задачу створення інструментального комплексу автоматизованого синтезу програм за схемами алгоритмів у візуальних середовищах. Для синтезу доцільно використовувати вже налагоджені, досліджені, промодельовані та оптимізовані схеми алгоритмів.

Система складається з таких компонентів [36-38]:

- *інтерфейсна частина* призначена для візуального редагування схем алгоритмів, понять їх реалізації у базі даних (БД). Проведення компіляції розроблених схем та сполучення з візуальними середовищами програмування;
- *компілятор* для трансформації схем алгоритмів, написаних мовою САА/1 [2], у програми заданою цільовою мовою програмування (зараз ANSI C та Паскаль);
- *БД* для зберігання набору понять, їх реалізацій та параметрів;
- *блок сполучення з БД* для забезпечення обміну даними між БД та іншими частинами комплексу.

Створені інструментальні засоби було застосовано для дослідження та формування важливого класу алгоритмів: маршрутизації в мережі Internet. Сформовані підхід та формалізовані специфікації протоколу маршрутизації OSPF, дозволили здійснити низку еквівалентних трансформацій алгоритму Дейкстри, який використано в цьому протоколі. В результаті одержано „спектр” схем алгоритмів, їх було промодельовано за допомогою СМОА, та визначено схеми, які є найбільш оптимальними за критерієм часу виконання [9].

Висновки

1. Обгрунтовано ефективність застосування комп’ютерних технологій, що базуються на математичному апараті САА-М, для формалізованої специфікації схем алгоритмів на етапі алгоритмічного проектування МПС.
2. Наведено приклад формалізації специфікацій протоколу маршрутизації OSPF в термінах САА-М. Показано, що такий підхід надає можливість застосувати потужний математичний апарат для дослідження, трансформації і оптимізації схем алгоритмів і тонкої інформаційної структури асоційованих з ними програм.
3. З використанням математичного апарату САА-М виконано трансформації алгоритмів Флойда - Уоршала та Джонсона, перспективних для майбутніх протоколів маршрутизації, і одержано низку параметричних регулярних схем цих паралельних алгоритмів. Показано доцільність та ефективність САА-М для фіксації схем паралельних алгоритмів.
4. Описано створені інструментальні засоби комп’ютерних технологій формалізованого проектування та трансформації схем алгоритмів на етапі алгоритмічного проектування МПС: систему автоматизованої трансформації параметричних регулярних схем алгоритмів, систему моделювання та оцінки ефективності схем алгоритмів і систему параметризованого синтезу асоційованих зі схемами алгоритмів програм. Створення програмного забезпечення інструментальних засобів виконано за допомогою сучасних засобів проектування програм таких як мови програмування ANSI C, Perl (довільні платформи), Object Pascal (платформа MS Windows, візуальне середовище Delphi).

1. *Погорелый С.Д.* К формализации этапа алгоритмического проектирования микропроцессорных систем. // УсиМ. – 1995.– № 6. – С. 3-8.
2. *Многоуровневое* структурное проектирование программ. Теоретические основы, инструментарий. / Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П. и др. // – М.: Финансы и статистика, 1989. – 208 с.
3. *Цейтлин Г.Е., Ющенко Е.Л.* Многоуровневое структурное проектирование программ (ретроспектива, состояние, перспективы) // Кибернетика. – 1998. - №4. – С. 34 – 41.
4. *Алгеброалгоритмічні* основи програмування. / Дорошенко А.Ю., Фінін Г.С., Цейтлін Г.О. //– К.: Наукова думка, 2004. – 256 с.
5. *Алгоритмічні* алгебри. / Ющенко К.Л., Суржко С.В., Цейтлін Г.О. та ін. // – К.: ІЗМН, 1997. – 480 с.
6. *Цейтлин Г. Е.* Введение в алгортмику. – К.: Сфера, 1998. – 312 с.
7. *Погорілий С.Д.* Програмне конструювання. За редакцією академіка АПН України Третяка О.В. К.: ВПЦ "Київський університет", 2005. – 440 с.
8. *Погорілий С.Д.* Автоматизація наукових досліджень. Основоположні математичні відомості. Програмне забезпечення. За редакцією академіка АПН України Третяка О.В. К.: ВПЦ “Київський Університет”, 2002. - 290с.
9. *Погорілий С.Д., Калита Д.М.* Оптимізація алгоритмів маршрутизації з використанням систем алгоритмічних алгебр // УсиМ. – 2000. – №4. – С. 20-30.
10. *Погорілий С.Д., Калита Д.М.* Про підхід до формалізації протоколів маршрутизації на прикладі протоколу OSPF // Вестник МСУ. – 2000. – №4. – С. 79 - 85.
11. *Погорілий С.Д.* Застосування систем алгоритмічних алгебр для дослідження алгоритмів протоколів маршрутизації в мережі INTERNET. Київський національний університет імені Тараса Шевченка. Наукові записки, том IX. КПВД “Педагогіка”. – 2004. С. 112-118.
12. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – Санкт-Петербург: БХВ-Петербург, 2002. 600 с.
13. *B. Zhang and H.T. Mouftah*. QoS Routing for Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols. IEEE Communications Magazine. October 2005, vol. 43, No. 10. p.p. 110 – 117 (http://www.comsoc.org).
14. *Семёнов Ю.А* Телекоммуникационные технологии (ГНЦ ИТЭФ) (http://citforum.ru/nets/semenov/).
15. *Moy, J.* OSPF Version 2, RFC 2328, Ascend Communications Inc., April 1998 – 211 p.
16. *Moy, J.* OSPF Version 2, RFC 2178, Cascade Communications Corp., July 1997 – 244 p.
17. *Postel, J.* Internet Official Protocol Standards, STD 1, RFC 2200, Internet Architecture Board, June, 1997 – 39 p.
18. *Іванова Д.О., Калита Д.М.* Дослідження, аналіз та трансформація алгоритму Беллмана-Форда. Вісник МСУ МАГІСТЕРІУМ, 9/2004, С. 109 - 118.
19. *Про підхід* до розпаралелювання алгоритму Флойда-Уоршала. / Погорілий С.Д., Камардіна О.О., Бавикін О.І. // Математические машины и системы. –2005. - №3. С. 86-94.
20. *Realization* of Schemes of the Parallel Warshall-Floyd Algorithm. / Kamardina O.O., Maksimov M.V., Pogorilyy S.D. // Proceedings of the Fifth International Young Scientests’ Conference on Applied Physics. June 20 - 22, 2005, Kyiv, Ukraine. Taras Shevchenko national University of Kyiv, Faculty of RadioPhysics. P. 94-95.
21. *Алгоритмы*. Построение и анализ. / *Кормен* Т., Лейзерсон Ч., Ривест Р. // М.:МЦНМО, 2000, С. 510-535.
22. *Седжвик Роберт*. Фундаментальные алгоритмы на С++. Алгоритмы на графах. - СПб: ООО Диасофт ЮП, 2002. - С. 304-310.
23. *Підхід* до підвищення швидкодії алгоритму Джонсона. / Погорілий С.Д., Камардіна О.О., Щербаненко С.В. // УСиМ. – 2005. -№5. - С. 112-120.
24. *Johnson's* algorithm cluster implementation methods. / *Kabachevskiy* V.V., Kamardina O.O., Pogorilyy S.D. // Proceedings of the 1 International Conference “Electronics and Applied Physics”, November, 24-27, Kyiv, Ukraine. Taras Shevchenko national University of Kyiv, Faculty of RadioPhysics, 2005. P.106-107.
25. *Погорілий С.Д., Камардіна О.О.* Дослідження та створення інструментальних засобів автоматизованої трансформації схем алгоритмів. // Проблемы программирования №1-2, 2004г. С. 417-421.
26. *Realization* of equivalent transformations of algorithmic schemes in SAA-M using the Perl language. / Kamardina O.O., Bevz V.V., Pogorilyy S.D. Proceedings of the Fifth International Young Scientests’ Conference on Applied Physics. June 20 - 22, 2005, Kyiv, Ukraine. Taras Shevchenko national University of Kyiv, Faculty of RadioPhysics, P. 108-109.
27. *Холзнер С.* Perl: Специальный справочник. – СПб.: Питер, 2000. – 496 с.
28. *Слободянюк А.И., Погорелый С.Д.* Кроссовые системы отладки программного обеспечения микропроцессора К580ИК80 на малых ЭВМ // УСиМ. – 1982. – №2. – С. 38-41.
29. *Дослідження* та створення інструментальних засобів моделювання та оцінки алгоритмів протоколів маршрутизації: / Погорілий С.Д., Калита Д.М., Мединський О.Г. Матер. 2 междунар. конф. по программированию. УкрПРОГ'2000 // Проблемы программирования. – 2000. – №1-2. – С. 204-208.
30. *GNU* Open Source Software Foundation, http://www.gnu.org.
31. *Проект* AMES - http://users.univ.kiev.ua/~ames.
32. *Грис Д.* Конструирование компиляторов для цифровых вычислительных машин. – М.: Мир, 1975. – 544 с.
33. *Проценко В.С., Чаленко П.И.* Элементы компиляции. – Киев, 1998. – 342 с.
34. *Архангельский А.Я.* Программирование в Delphi 6. – М.: Бином, 2002. – 962 с.
35. *Тейксейра С., Пачеко К.* Delphi 4: Руководство разработчика. – М.: Изд. дом "Вильямс", 1999. 642 с.
36. *Інструментальний* комплекс синтезу програм в середовищі Delphi за методом багаторівневого структурного проектування програм. / Погорілий С.Д., Калита Д.М., Захаров О.І. // Вісн. Київ. ун-ту. – 2000. – Вип. №1. С. 282-289.
37. *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. – М.: Мир, 1978. – В 2-х т.-т.1. - 680 с.; т.2. - 582 с.
38. *Введение* в теорию автоматов, языков и вычислений. / Хопкрофт Д., Мотвани Р., Ульман Д. // М.: Издательский дом "Вильямс", 2002. – 528 с.