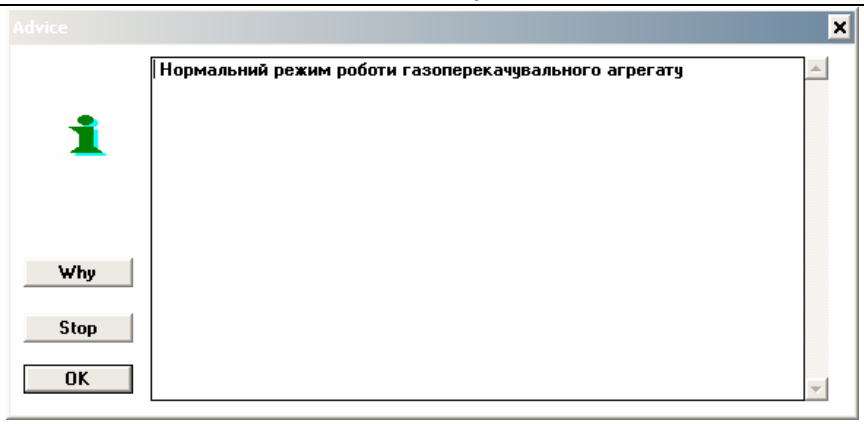

Рис. 6. Оцінка стану ГПА

Висновок

В даній роботі виконаний вибір середовища реалізації експертної системи – діагностування ГПА за параметрами вібрації. Виконано обгрунтування експертної оболонки ESTA. Використання вибраної методики діагностування дефектів з метою досягнення оптимальних умов визначення нормального стану агрегата дозволили зменшити кількість відмов та дефектів ГПА в окремих вузлах.

При розробці ЕС діагностування ГПА передбачена розробка загальної структури, виконана побудова бази знань та побудова внутрішніх правил. Проведена апробація стану ГПА за значеннями амплітуди в спектрі вузла ГПА. Розроблене програмне забезпечення показало свою працездатність в робочому діапазоні технологічних параметрів ГПА. Досліджені значення амплітуди в контрольних точках дозволили виявити ряд дефектів ГПА, що є важливим при експлуатації в нормальному робочому технологічному режимі. Експлуатація експертної системи дозволяє використовувати її в режимі порадника, шляхом виявлення дефекту чи визначення нормального режиму роботи ГПА.

Література

1. Обчислювальна техніка і її застосування. – Москва, 2002. – № 2.
2. Стефанюк В.Л. Експертні системи і їхнє застосування: Курс лекцій.
3. Мызин Н.И., Скварновский А.В., Чудиков Ю.П. Вибрация газоперекачивающих агрегатов. Л.: Недра, 1973. – 144 с.
4. Зарицкий С.П. Диагностика газоперекачивающих агрегатов с газотурбинным приводом. – М.: Недра, 1987. – 198 с.
5. Ширмовська Н.Г. Проектування та реалізація експертних систем для технологічних об’єктів нафтогазового комплексу, № 1 (126), Хмельницький, 2009.

Надійшла 15.12.2009 р.

УДК 004.272, 519.171

А.О. МЕЛЬНИК
Національний університет „Львівська політехніка”
І.Д. ЯКОВЛЄВА
Чернівецький національний університет імені Юрія Федьковича

ПОБУДОВА СТРУКТУРНОЇ МАТРИЦІ ПОТОКОВОГО ГРАФА АЛГОРИТМУ З ЙОГО ОПИСУ НА РІВНІ ТРІАД

В роботі запропоновано підхід до формального опису потокових графів інваріантних до зсуву даних алгоритмів за допомогою структурної матриці. Запропоновано метод заповнення структурної матриці шляхом привласнення її елементам номерів тріад за певним правилом. Кількість вхідних даних визначає ширину структурної матриці, а її висота визначається в процесі аналізу тріад. Це дає можливість не будувати потоковий граф алгоритму, а одразу переходити до його формального опису.

In-process offered approach to the formal specification of potokovikh counts of to the change these invariant algorithms by a structural matrix. The method of filling of structural matrix is offered by an appropriation its elements of numbers of triads by certain rule. The amount of datains determines the width of structural matrix, and its height is determined in the process of analysis of triads. It enables not to build potokoviy count of algorithm, but at once to pass to his formal specification.

Ключові слова: структурна матриця, потоковий граф.

Вступ. Для успішного розв’язання задач на обчислювальних системах паралельної архітектури та

при проектуванні спеціалізованих процесорів доводиться використовувати принципово нові відомості про структуру алгоритмів на рівні зв'язків окремих операцій між собою [1, 2, 3]. Найкращим засобом для пошуку шляхів паралельного виконання алгоритмів та оцінки підходів до їх апаратної реалізації є потокові графи алгоритмів (ПГА) [1]. Але виникає дуже багато питань, що стосуються у першу чергу того, як будувати й вивчати графи алгоритмів, та в якій формі повинні описуватися ПГА. Актуальною метою досліджень є розробка методів формального опису ПГА на основі аналізу тексту програм [1, 2, 3].

В роботі запропоновано підхід до формального опису потокових графів інваріантних до зсуву алгоритмів структурною матрицею з опису алгоритму на рівні тріад. В структурну матрицю, яка описує ПГА, заносяться номери функціональних операторів (ФО) за певним правилом. Це дає можливість не будувати ПГА в процесі аналізу алгоритму, а одразу переходити до його формального опису, а ПГА використовувати тільки для забезпечення наочності графічного подання графів алгоритмів та їх верифікації.

Огляд літератури. Традиційно ПГА описуються наступним чином: спочатку на основі послідовної програми будується граф алгоритму, потім вивчаються його паралельні форми і виконується їх формальний математичний опис [2, 4]. Але такий підхід є досить трудомістким і ймовірність внесення помилок при побудові ПГА збільшується із збільшенням кількості вершин графа.

В роботі [5] запропоновано структурну матрицю, яка на відміну від однойменної матриці [6] зберігає обчислювальні і структурні характеристики ПГА такі як набір ФО, організацію з'єднань між ними, розподіл ФО по ярусах та інші. Ці характеристики об'єднані в множину $S(C,L,K,W)$, де $C=\{n_j\}$ – множина вхідних дуг ПГА, $j=\overline{1,n}$ – номер вхідної дуги ПГА, по яких поступають дані, n – кількість вхідних дуг ПГА; $L=\{l_i\}$, $i=\overline{1,l}$ – номер ярусу ПГА, l – кількість ярусів ПГА; $K=\{f_{ij}\}=\{0..k\}$ - множина функціональних операторів ПГА, $k\in N$, $i=\overline{1,l}$, $j=\overline{1,n}$ – номер функціонального оператора ПГА, $W=\{w_i,w\}$ – множина, яка визначає кількість ФО на кожному *i*-му ярусі і ширину ПГА – *w*. В цій же роботі [5] запропоновано заносити інформацію про ПГА в матрицю, кількість стовпців якої рівна кількості дуг, по яких надходять операнди в найширшому ярусі, кількість рядків рівна загальній кількості ярусів. Дуги в найширшому ярусі нумеруються від 1 до *n*, їх кількість на кожному ярусі є незмінною та дорівнює кількості вхідних та вихідних вершин, з яких (на які) поступають дані. Елементами такої матриці є номера ФО, які присвоєні кожній дузі, по якій на ФО надходять операнди.

Зважаючи на те, що більшість операцій є двомісними, а результат виконання операції часто необхідний для виконання більше як однієї наступної операції, тут використані ФО, які мають дві вхідні і дві вихідні дуги.

Структурна матриця F розмірністю $l\times n$, елементами якої є номери ФО, які присвоєні кожній дузі, по якій на ФО надходять операнди, має наступний вигляд:

$$F=\begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & f_{ij} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ f_{l1} & f_{l2} & \cdots & f_{\ln} \end{pmatrix},$$

де $\forall f_{ij}\in N$, $i=\overline{1,l}$, $j=\overline{1,n}$, l – кількість ярусів (рядків), n – загальна кількість дуг найширшого ярусу, по яких надходять операнди (стовпців).

Заповнюється структурна матриця F за правилом: на перетині *i*-го рядку і *j*-го стовпця ставиться номер відповідного ФО *i*-го ярусу, який виконує операцію над операндом, який поступає *j*-ю дугою ПГА $f_{ij}=k$, де *k* належить множині натуральних чисел, $k\in N$. Нумерація починається від 1 і призначається всім наступним ФО підряд в порядку зростання. Решта елементів такої матриці заповнюються нулями. Може бути два типи ФО: операційні, які передбачають виконання операцій, і перепускні, які передбачають передачу інформації з входу на вихід. Операційні ФО позначаються *k*, а перепускні – 0. Оскільки один ФО виконується над двома даними, які поступають різними дугами, то два елементи структурної матриці F *i*-го рядка приймають одне й те саме значення *k*. Якщо $f_{ij}=0$, то в *i*-му ярусі над даними, які надходять *j*-ю дугою, не виконується жоден операційний ФО, а дані просто передаються на наступний ярус.

Постановка задачі. Оскільки потокові графи алгоритмів э найкращим засобом для пошуку шляхів паралельного виконання алгоритмів та оцінки підходів до їх апаратної реалізації, актуальною э задача отримання інформації про обчислювальні і структурні характеристики ПГА з їх опису в вигляді програми шляхом безпосередньої побудови структурної матриці.

Відомий підхід до побудови структурної матриці потокових графів інваріантних до зсуву алгоритмів.

Представимо ПГА обчислення виразу

$$y=(a+b)\times(c-d)/(e+f), \qquad (1)$$

який зображений на рис. 1, а з допомогою ФО, які мають дві вхідні і дві вихідні дуги (рис. 1, б).

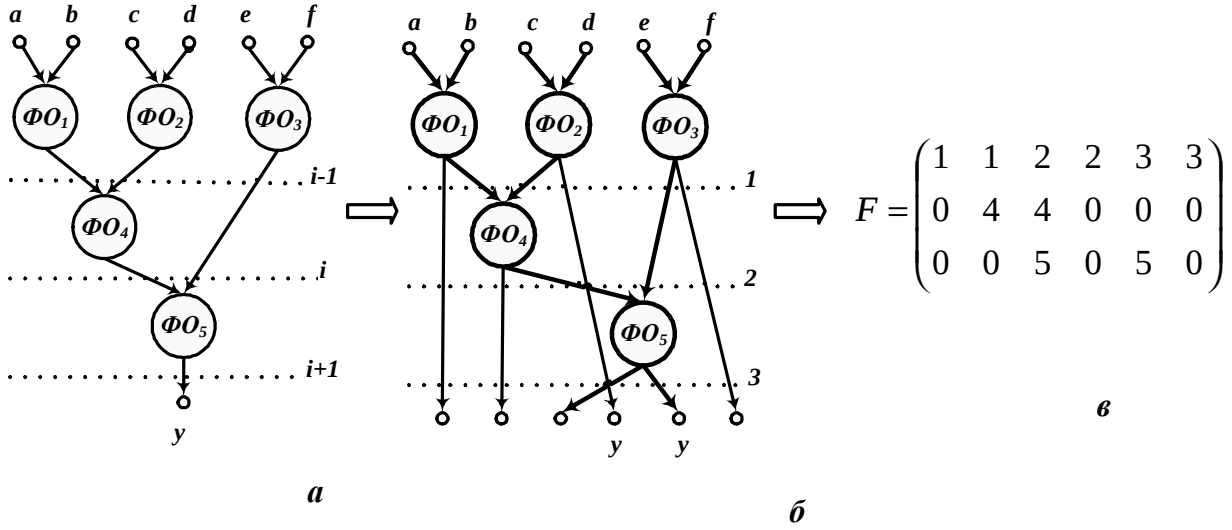

Рис. 1. ПГА обчислення виразу $y=(a+b)\times(c-d)/(e+f)$: а, б – потоковий граф алгоритму; в – структурна матриця; ФО – функціональний оператор

Вважаємо, що всі вхідні дані переміщуються по дугах ПГА одночасно. Якщо на шляху даних, які переміщуються дугами, зустрічається вершина, то дані приймають участь в операції, заданій вершиною ПГА, інакше – просто передаються на наступний ярус. В результаті перетікання даних між вершинами ПГА, і виконання операцій, заданих алгоритмом, на виході ПГА отримується результат $y=(a+b)\times(c-d)/(e+f)$. Результат обчислень міститься тільки на дугах, які виходять із останньої вершини даного ПГА – $ФО_5$. По всіх інших дугах перетікають результати проміжних обчислень

Структурна матриця *F*, яка описує даний ПГА (рис. 1, в), має розмір 6×3, оскільки ПГА має шість вхідних дуг і три яруси. На першому ярусі ПГА знаходяться три вершини $ФО_1$, $ФО_2$, $ФО_3$. Кожна з них виконує операцію над операндами, які надходять 1-ю і 2-ю, 3-ю і 4-ю та 5-ю і 6-ю дугами відповідно. Тому перший рядок структурної матриці *F* буде мати вигляд (1, 1, 2, 2, 3, 3). На наступному ярусі дуги нумеруються в тому ж порядку, а оскільки на другому ярусі тільки одна вершина – $ФО_4$, на яку надходять друга і третя дуги, то другий рядок структурної матриці *F* буде мати вигляд (0, 4, 4, 0, 0, 0). Аналогічно третій рядок структурної матриці – (0, 0, 5, 0, 5, 0). Таким чином отримується структурна матриця *F*, яка зберігає характеристики ПГА обчислення виразу (1) (рис. 1, в).

Пропонований підхід до побудови структурної матриці потокових графів інваріантних до зсуву алгоритмів.

Нехай задана довільна програма. Пронумерувавши всі операції підряд, отримаємо вершини ПГА. Залежність між операціями, для опису зв'язків ПГА, дає представлення цих операцій у вигляді тріад (triple) [7]. Тріади включають три складові: операцію і два операнди. Наприклад, тріади можуть мати вигляд: $<операція>(<операнд1><операнд2>)$. Особливістю тріад є те, що один або обидва операнди можуть бути посиланням на іншу тріаду в тому випадку, коли в якості операнду даної тріади є результат виконання іншої тріади. При записі тріади послідовно нумерують для зручності вказування посилань одних тріад на інші. Наприклад, вираз (1) буде записаний наступним чином:

1. + (a , b)
2. - (c , d)
3. + (e , f)
4. * (^1 , ^2)
5. / (^4 , ^3)

Тут операції позначені відповідним знаком 1..5, а знак ^ означає посилання операнду однієї тріади на результат іншої. Тріади не залежать від архітектури обчислювальної системи, на яку орієнтована результуюча програма. Тому вони є машинно-незалежною формою внутрішнього представлення програми. Таке представлення операцій є зручним для проектування спеціалізованих процесорів різної архітектури.

Розглянемо метод заповнення структурної матриці шляхом занесення до неї номерів тріад. Даний метод для можливості подальшого опрацювання накладає на тріади наступні вимоги: 1) всі вхідні операнди, які в правій частині оператора присвоювання зустрічаються більше одного разу, перейменовуються за допомогою тимчасових змінних, а посилання на одну тріаду не може бути використане більше двох разів (виходячи із прийнятого ФО ПГА [5]); 2) кожному вхідному операнду ставиться у відповідність натуральне число $j=\overline{1,n}$. Тоді n – ширина структурної матриці. Також використовується додаткова проміжна таблиця

тріад, до якої заносяться тріади і номери їх вхідних операндів. Дана таблиця використовується для формування всіх рядків матриці крім першого, оскільки за її допомогою можна відслідкувати залежність за даними.

Алгоритм заповнення структурної матриці ПГА:

Крок 1. Записати операції у вигляді тріад (рис. 2, а) за допомогою тимчасових змінних таким чином, щоб и всі операнди тріад зустрічалися не більше одного разу, а посилання на одну тріаду не було використане більше двох разів.

Крок 2. Кожному вхідному операнду послідовно поставити у відповідність натуральне число $j = \overline{1,n}$ (рис. 2, б).

Крок 3. Здійснити опис динамічної структурної матриці *F* розміром *n* ´ *l*:
а) *n*- кількість стовпців і дорівнює кількості вхідних операндів.
б) $l = 1$, *l* – кількість рядків структурної матриці *F*.

Крок 4. Вибрати із послідовності тріад тріаду, яка не містить в якості операндів посилань на інші тріади. (Операції, задані такими тріадами складають перший ярус ПГА і, відповідно, перший рядок структурної матриці.)

Крок 5. Вибрати із тріади, отриманої на кроці 4, знак операції і під номером тріади занести її в таблицю відповідності номерів ФО виконуваним операціям (рис. 2, в).

Крок 6. В таблицю тріад занести номер даної тріади і номера операнду 1 та операнду 2, яким поставлені у відповідність натуральні числа на кроці 2 (рис. 2, г).

Крок 7. Елементам *l* го рядка структурної матриці *F* із індексами, які дорівнюють номерам операнду 1 та операнду 2 привласнити номер тріади (рис. 2, д).

Крок 8. Вилучити дану тріаду із послідовності тріад.

Крок 9. Якщо в послідовності тріад є тріада, яка не містить в якості операнду посилань на інші тріади, то перейти до кроку 4, інакше – до кроку 10.

Крок 10. Якщо послідовність тріад не порожня, то перейти до кроку 11, інакше – закінчити виконання алгоритму.

Крок 11. Збільшити *l* на одиницю: $l = l + 1$.

Крок 12. Встановити вказівник на першу тріаду з послідовності тріад.

Крок 13. Якщо операнди тріади, на яку вказує вказівник, вхідні дані або тріади, номер яких міститься в таблиці тріад (рис. 2, е), то перейти до кроку 14, інакше перейти до кроку 24.

Крок 14. Вибрати із тріади, отриманої на кроці 13, операцію і під номером тріади занести її в таблицю відповідності номерів ФО виконуваним операціям (рис. 2, ж).

Крок 15. В таблицю тріад занести номер тріади.

Крок 16. Вибрати перший операнд тріади і перейти до кроку 17.

Крок 17. Якщо вибраний операнд є вхідним операндом, то перейти до кроку 18. Інакше – до кроку 19.

Крок 18. В структурну матрицю в рядок *l* в стовпчик з номером, який дорівнює номеру даного вхідного операнда занести номер тріади, а в таблицю тріад номер операнду.

Крок 19. Якщо вибраний операнд є посиланням на тріаду, то перейти до кроку 20, інакше – до кроку 24.

Крок 20. В таблиці тріад по номеру посилання на тріаду вибрати номер одного з двох її операндів і перенести його в поле таблиці тріад, яка аналізується (рис. 2, з).

Крок 21. В структурну матрицю в рядок *l* в номер стовпчика, номер якого дорівнює номеру вхідного операнду тріади, на яку є посилання, записати номер тріади, на яку вказує вказівник (рис. 2, і).

Крок 22. Якщо аналізувався перший операнд, то вибрати другий операнд тріади і перейти до кроку 17. Інакше перейти до кроку 23.

Крок 23. Вилучити дану тріаду із послідовності тріад.

Крок 24. Якщо це не остання тріада послідовності тріад, то зсунути вказівник на наступну тріаду і перейти до кроку 13. Інакше перейти до кроку 10.

Закінчивши виконання алгоритму, отримаємо заповнену структурну матрицю, яка описує ПГА даного алгоритму та таблицю відповідності номерів ФО виконуваним операціям. Наприклад, для фрагменту програми обчислення виразу (1) процес отримання структурної матриці та таблиці відповідності номерів ФО виконуваним операціям зображено на рис. 2.

Для фрагменту програми обчислення виразу (1) структурні матриці, отримані з ПГА (рис. 1, в) та із тріад (рис. 2, н) співпадають. Отже, запропонований метод дає можливість одразу отримувати структурну матрицю із представлення програми у вигляді тріад без побудови ПГА.

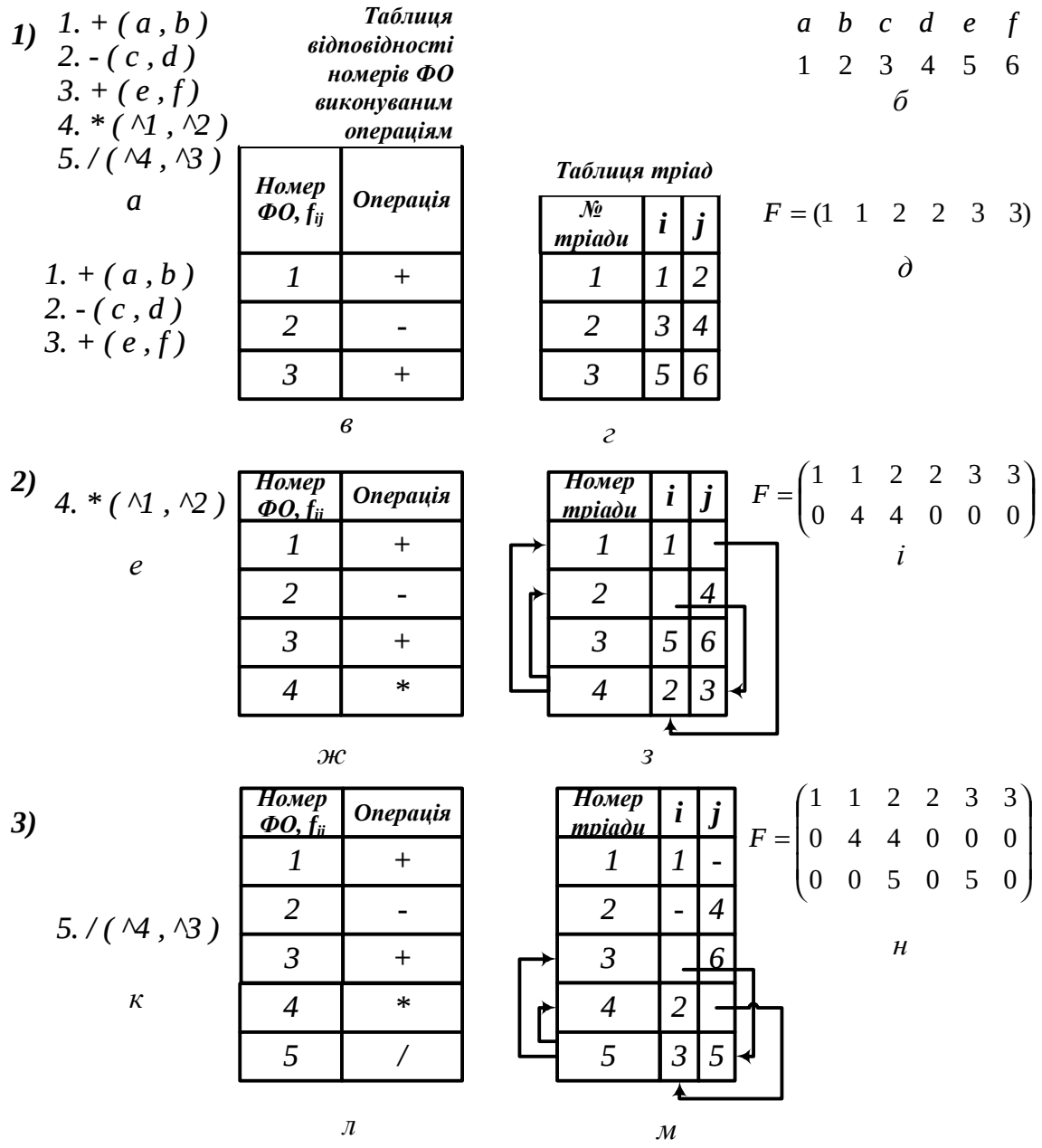

Рис. 2. Отримання структурної матриці та таблиці відповідності номерів ФО виконуваним операціям для виразу $y = (a+b)\times(c-d)/(e+f)$ **представленого тріадами: 1) формування першого рядка структурної матриці: а) тріади; б) відповідність натуральних чисел вхідним операндам; в) таблиця відповідності номерів ФО виконуваним операціям; г) таблиця тріад; д) перший рядок структурної матриці; 2) формування другого рядка структурної матриці: е) тріада, яка містить посилання на інші тріади, що знаходяться в таблиці тріад; ж) таблиця відповідності номерів ФО виконуваним операціям; з) таблиця тріад; і) перші два рядка структурної матриці; 3) формування третього (останнього) рядка структурної матриці: е) тріада, яка містить посилання на інші тріади, що знаходяться в таблиці тріад; ж) таблиця відповідності номерів ФО виконуваним операціям; з) таблиця тріад; і) структурна матриця**

Висновки:

1. Запропоновано підхід до формального опису потокових графів інваріантних до зсуву алгоритмів використовуючи структурну матрицю, який не вимагає побудови ПГА в процесі аналізу алгоритму.

2. Запропоновано метод заповнення структурної матриці шляхом занесення номерів тріад за певним алгоритмом. Кількість вхідних даних, взятих без повторення, визначає ширину структурної матриці, а її висота визначається в процесі виконання алгоритму аналізу тріад.

3. Описані алгоритми виконані як програмний модуль і реалізовані мовою С++.

Література

1. Мельник А.О. Спеціалізовані комп'ютерні системи реального часу. – Львів: НУ „Львівська політехніка", 2002. – 60 с.

2. Воеводин В.В. Вычислительная математика и структура алгоритмов. – М.: Изд-во МГУ, 2006. – 112 с.

3. Анісімов А.В., Кулябко П.П., Терещенко В.М. Паралельні алгоритми в дослідженнях неперервних систем. – К.: РВЦ Київський університет, 1999. – 55 с.

4. Антонов А.С., Воеводин Вл.В. Новый подход к построению методов межпроцедурного анализа программ // Материалы международной конференции УкрПРОГ-98. – К. – 1998. – С. 77-84.

5. Мельник А.О., Яковлєва І.Д. Подання потокового графа алгоритму структурною матрицею // Вісник Хмельницького національного університету. – 2008. – № 4. – С.124-129.

6. Рабкин Е.Л., Фарфоровская Ю.Б. Кафедра высшей математики СПбГУТ им. проф. М.А. Бонч-Бруевича Дискретная математика. Булевы функции и элементы теории графов // http://dvo.sut.ru/libr/himath/w163rabk/

7. Ахо А., Сети Р., Ульман Дж. Д. Компиляторы: принципы, технологии и инструменты. М.: Вильямс, 2001. – 768 с.

Надійшла 8.12.2009 р.

УДК 004.451

С.В. МОСТОВИЙ, О.Л. КОВТУН, І.В. ПРОКОПИШЕН
Хмельницький національний університет

МЕТОД ТА ЗАСОБИ ПРОГНОЗУВАННЯ СТАНУ ВЗАЄМОБЛОКУВАННЯ ПРОЦЕСІВ В ПЕРСОНАЛЬНОМУ КОМП'ЮТЕРІ

Розроблено метод прогнозування стану процесів в персональних комп'ютерах, який дає можливість передбачати настання ситуації взаємоблокування процесів на основі аналізу взаємодії сигнатур процесів.

The method of forecasting of a status of processes in personal computers is developed. It enables to provide approach of deadlock of processes based on analysis of interaction of their signatures.

Ключові слова: взаємоблокування процесів, прогнозування стану процесів.

Вступ

При паралельному виконанні задач можуть виникати такі ситуації, при яких два і більше процеси весь час знаходяться в стані блокування, очікуючи ресурсів системи, які в даний момент утримує інший процес, що, в свою чергу, теж знаходиться в стані блокування. Про такі процеси говорять, що вони знаходяться в стані взаємного блокування, "дедлока" (deadlock) або "клінчу" (clinch). В даний час розглядаються можливі компромісні рішення з погляду накладних витрат на включення засобів боротьби з взаємоблокуваннями і очікуваних від цього вигод. В деяких випадках ціна, яку доводиться платити за те, щоб зробити систему вільною від взаємоблокувань, дуже висока. В системах реального часу виникнення ситуації взаємоблокування може призвести до катастрофічних наслідків.

Постановка задачі

Для вирішення проблеми взаємоблокування відомо багато методів і алгоритмів [1-4]. Вони поділяються на наступні групи:

- Запобігання взаємоблокуванню, що полягає у забезпеченні невиконання однієї з чотирьох умов для взаємоблокування.
- Уникнення взаємоблокування. Полягає у тому, що б не задовільняти запит на ресурс, якщо його виділення може потенційно спричинити взаємоблокування.
- Виявлення взаємоблокувань. Полягає у тому, що б завжди задовільняти запити на ресурси, коли це можливо, і періодично перевіряти систему на наявність взаємоблокувань. Якщо взаємоблокування має місце, то вирішити проблему.
- Ігнорування взаємоблокувань. Має сенс, коли ймовірність виникнення взаємоблокування дуже низька.

Вагома частка взаємоблокувань припадає на взаємоблокування процесів, що виконуються в операційних системах (ОС) для персональних комп'ютерів (ПК). Проте, методи розв’язання задачі взаємоблокування процесів в ОС ПК мають ряд недоліків [4] (блокування роботи ОС, наявність циклів активного очікування, складність програмної реалізації для багатьох процесів, необхідність використання спеціалізованої команди процесора) і є складними для їх реалізації. Тому більшість сучасних ОС не містять засобів для вирішення проблеми взаємоблокування процесів.

Для усунення виявлених недоліків та розв’язання задачі взаємоблокування процесів в [5] запропонована модель прогнозування стану процесів в персональних комп'ютерах, яка базується на використанні сигнатур процесів.

До складу моделі входять наступні структурні частини (рис. 1):

- підсистема виявлення зміни стану процесу – призначена для контролю за поточними параметрами процесів, що вже присутні на ПК в певній ОС, та запуском нових процесів;
- підсистема визначення характеристик ОС даного ПК – призначена для контролю за зміною основних характеристик, що суттєво впливають на наближення процесів до стану взаємоблокування;
- підсистема побудови сигнатури процесу – призначена для побудови сигнатури процесу, що виконує перехід у стан готовності і ще не має сигнатури, та модифікації сигнатури процесів, що змінили