УДК 681.3

МЕТОДИКА АВТОМАТИЗОВАНОЇ ТРАНСФОРМАЦІЇ СХЕМ АЛГОРИТМІВ

І.Ю. Шкуліпа, С.Д. Погорілий

Київський національний університет імені Тараса Шевченка,
01033, Київ 33, вулиця Володимирська, 64,
e-mail: Igor.Shkulipa@gmail.com, SDP@rpd.univ.kiev.ua

Описано методику створення інструментарію автоматизованого перетворення алгоритмів, на основі їх подання у вигляді САА-М-схем. Розглянуто методи застосування еквівалентних перетворень до алгоритмів. Запропоновано методи реалізації автоматизованого перетворювача САА-М-схем алгоритмів. Наведено параметри практичної реалізації системи автоматизованого перетворення алгоритмів.

A method of automated tools for algorithms transformation, based on their representation in the form of SAA-M schemes is covered. The methods to apply equivalent transforms for algorithms are described. Methods to implement an automated SAA-M-schemes transformer are proposed. Implementation of the automated algorithms transformation system is described.

Вступ

Мета процесу оптимізації алгоритмів – одержання на основі базового алгоритму одного або ряду, оптимізованих за певними критеріями.

Основними етапами процесу оптимізації алгоритму є:

- створення базового алгоритму. Основною вимогою до такого алгоритму є його безумовна правильність, - оптимальність алгоритму на даному етапі не є обов'язковим;
- оптимізація алгоритму за обраними критеріями;
- тестування отриманих алгоритмів на певних вихідних даних з метою порівняння їх ефективності з вихідним алгоритмом;
- оцінка ефективності отриманого алгоритма.

Створення базового алгоритма необхідне в тому випадку, коли алгоритм для вирішення завдання ще не створено. При створенні алгоритму виконуються наступні дії: по-перше, перераховуються і формалізуються поняття, що використовуються при вирішенні задачі; по-друге, визначаються оператори та умови, які застосовуються для побудови схеми; по-третє, створюється САА-М – схема рішення задачі з використанням попередньо визначених умов та операторів для побудови базового алгоритма [1].

Система автоматизованого перетворення алгоритмів (САПП) [2] дозволяє значно прискорити процес трансформації алгоритму, уникнути механічних помилок, які обов'язково зустрічаються при перетворенні великих алгоритмів вручну і не вимагає особливих навичок від того, хто її використовує. Система виконує оптимізацію алгоритмів за різними критеріями, дозволяє створювати їх паралельні версії в автоматичному режимі і виконувати генерацію програмного коду відповідно до схеми з використанням різних мов та парадигм паралельного програмування.

Для формалізованого опису алгоритмів на етапі алгоритмічного проектування в роботі пропонується використання математичного апарата модифікованих систем алгоритмічних алгебр (САА-М) В.М. Глушкова. САА-М допускають представлення будь-якого алгоритму в аналітичному вигляді – у вигляді САА-М-схем. У свою чергу САА-М-схеми можуть бути формально модифіковані шляхом застосування до них еквівалентних перетворень, що дає зручну і потужну основу для створення автоматизованого перетворювача алгоритмів.

Застосування цих засобів до алгоритма можливе завдяки тотожним трансформацій, які допомагають спростити схему і дають можливість проводити поглиблену подальшу її трансформацію. Це особливо важливо при створенні нових схем на базі вже існуючих. Поєднання цих засобів – це потужний механізм для встановлення зв'язків між різними алгоритмами, а також для конструювання нових алгоритмів, більш ефективних з тих чи інших критеріїв.

Слід зазначити, що в цьому напряму вже проводився ряд робіт [3, 4]. Всі вони відрізнялися обмеженою функціональністю і низькою кількістю правил перетворення, а також складністю розширення. Особливістю даної роботи є можливість необмеженого накопичення правил перетворення алгоритмів і розширені функціональні можливості для синтезу САА-М-схем і використання різних парадигм паралельного програмування та мов програмування.

Система автоматизованого параметричного перетворення алгоритмів (САПП)

Система САПП – це автоматизований інструментарій, що призначений для проектування та перетворення алгоритмів, з метою отримання нових більш ефективних версій. Основою для перетворення алгоритмів є апарат САА-М В.М. Глушкова [5, 6].

Підсистема перетворення САА-М-схем алгоритмів

Підсистема перетворення схем алгоритмів – це функціональна складова САПП, що призначена для перетворення та оптимізації вхідного алгоритму з метою формування нової або паралельної версії.

Аналітичний вигляд алгоритму базується на його представленні у вигляді формалізованої САА-М-схеми. Еквівалентні перетворення САА-М дають зручну і потужну основу для створення автоматизації процесу оптимізації алгоритму. У термінах САА-М еквівалентні перетворення – це набір теорем і аксіом, які описують правила перетворення частин алгоритмічної схеми в нові операторні конструкції.

Процес перетворення схем алгоритмів відбувається наступним чином:

- на вхід подається файл зі схемою алгоритму або схема алгоритму вставляється у відповідне поле інтерфейсу користувача;
- потім вибираються правила еквівалентних перетворень, які користувач бажає застосувати до заданої схеми;
- запускається робота перетворювача схем алгоритмів, яка включає в себе весь процес перетворення схеми алгоритма. Виконується пошук відповідної частини тексту схеми, заміна її на відповідний еквівалент, і так далі, доти, доки не будуть застосовані всі вибрані перетворення у всіх можливих комбінаціях і не отримано набір нових схем алгоритма;
- після модифікації схеми алгоритму, кожна з отриманих схем, яка відрізняється від інших, зберігається в базі даних системи.

В основі практичної реалізації перетворювача схем алгоритмів лежить інструмент регулярних виразів [7]. Використання регулярних виразів є досить зручним і ефективним методом створення правил еквівалентних перетворень, як для перетворень булевої алгебри, так і для перетворень САА / САА-М. Нижче (рис. 1) схематично зображено процес застосування одного правила перетворення до деякої схеми алгоритму.

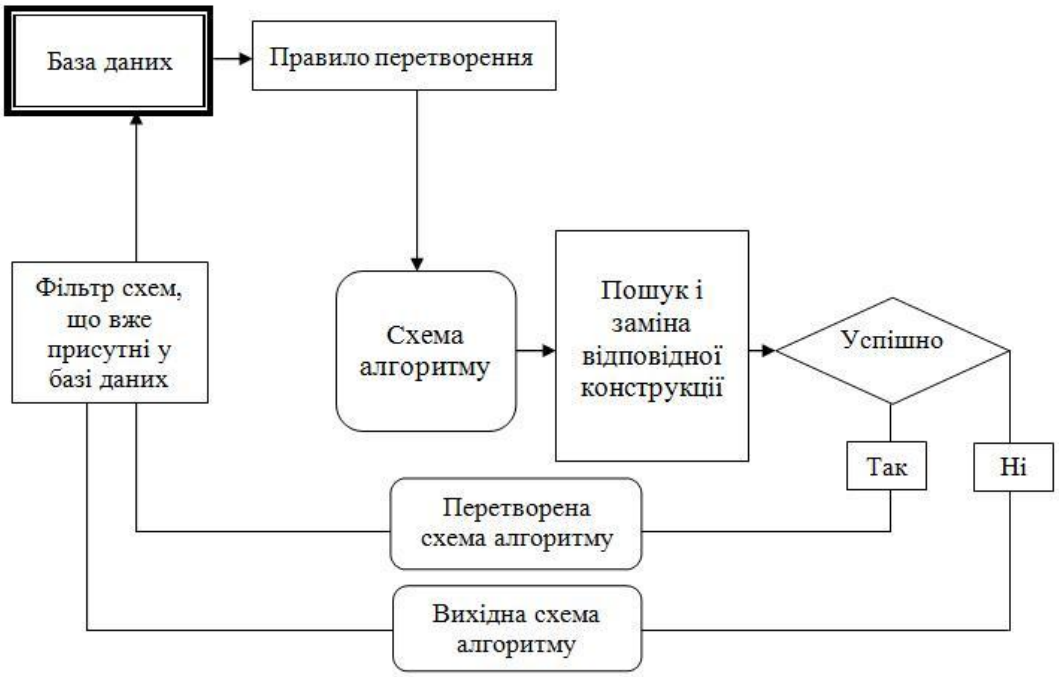

Рис. 1

Рис. 1 ілюструє цикл перетворення однієї схеми алгоритму із застосуванням одного правила перетворення.

Застосування правила еквівалентного перетворення САА-М, фактично, зводиться до трансформації частини аналітичної схеми, що відповідає регулярному виразу для лівої частини перетворення до нової частини, а саме регулярного виразу, що задає праву частину перетворення.

Важливим етапом розбору схеми алгоритмів є синтаксичний аналіз. Першою фазою аналізу схеми є перевірка коректності запису схеми. Основними умовами коректності запису схеми є:

- відсутність у файлі схеми будь-яких символів, що не входять до алфавіту прийнятих умов та позначень;

- кількість дужок лівих певного типу ((,[,{) має дорівнювати кількості дужок правих того ж типу;
- ідентифікатори умов та операторів мають відповідати прийнятим умовним позначенням. Ідентифікатором вважається будь-яка частина схеми, що міститься між двома роздільниками (дужки та позначення операцій над операторами та умовами).

Процес повного циклу роботи всієї підсистеми перетворення САА-М-схем можна проілюструвати на наступному прикладі.

Припустимо, дано алгоритм, представлений у вигляді недетермінованої САА-схеми. Необхідно отримати низку САА-М-схем цього алгоритму, застосовуючи послідовно еквівалентні перетворення САА.

Алгоритм, що застосовується в системі, є ітераційним і схематично являє собою наступне:

- перевіряється коректність запису вхідної схеми, відповідно до вищеописаних правил;
- до вихідної схеми застосовуються вибрані правила перетворень;
- до ряду схем, отриманих після попереднього кроку, застосовуються наступні за списком правила перетворень;
- крок 2 повторюється для всіх схем кроку 1 і так далі для результуючих схем кожного кроку.

Результатом роботи підсистеми перетворення буде ряд схем алгоритму (рис. 2.), які були отримані шляхом послідовного застосування правил еквівалентних перетворень до вихідної схемою алгоритму. Так на першому кроці роботи будуть отримані схеми, які відповідають застосуванню до вихідної схемою кожного з правил перетворення один раз. На наступному етапі отримаємо схеми, що відповідають застосуванню кожного з правил перетворення до схем з попереднього етапу. І так далі до тих пір, доки не будуть послідовно застосовані всі перетворення N разів, де N – загальна кількість вибраних правил перетворення. Результуючий набір схем буде набором схем, отриманих застосуванням всіх правил еквівалентних перетворень у всіх можливих комбінаціях і послідовностях.

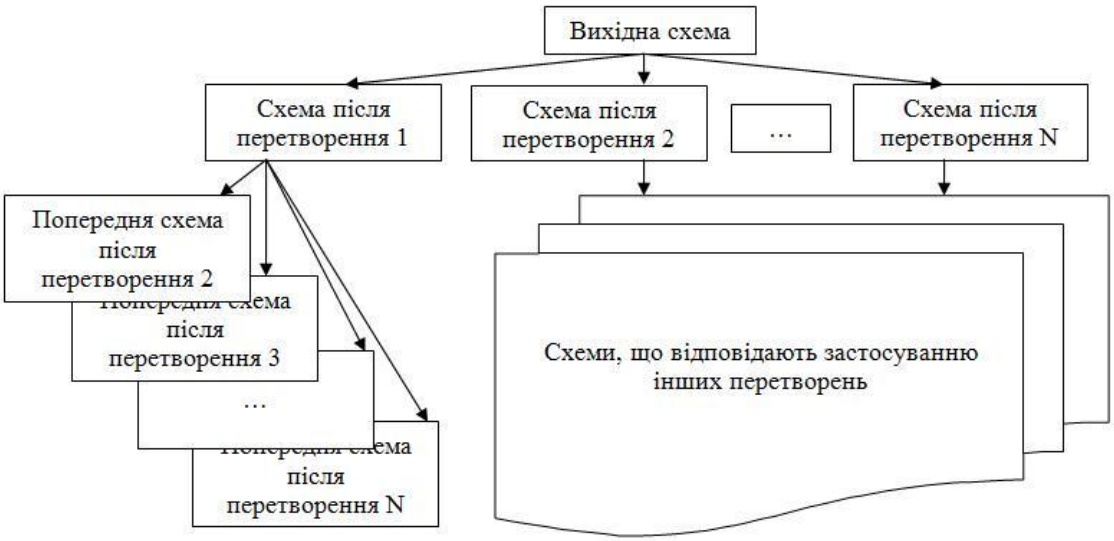

Рис. 2

Таким чином, кількість результуючих схем алгоритмів буде дорівнювати

$S = N^{2-}T + 1$, де $T = 0..N^2$ – кількість еквівалентних схем, яка обумовлюється тим фактом, що не кожне з перетворень може бути застосовано до вхідної схеми алгоритму.

У випадку, коли $T = 0$, тобто всі комбінації перетворень дали новий результат, кількість результуючих схем буде N^2, і відповідно можливостей отримати найбільш оптимальний алгоритм стане більше.

У випадку, коли $T = N^2$, тобто жодне з еквівалентних перетворень не може бути застосоване до вхідної схеми алгоритму, результатом автоматичного розпаралелювання буде вихідна схема алгоритму. В такому випадку передбачена можливість ручного розпаралелювання за даними за допомогою моделювання з використанням апарата мереж Петрі або UML діаграм.

У системі також передбачена можливість цілеспрямованого застосування еквівалентних перетворень. При такому способі трансформації, вибрані перетворення застосовуються до вхідної схеми у кількості та послідовності, що визначається користувачем. Кількість схем при такому перетворенні залежить від заданого користувачем алгоритма застосування перетворень.

Еквівалентні перетворення

У системі використовуються два основних види еквівалентних перетворень: перетворення умов схеми, згідно з правилами булевої алгебри, і перетворення функціональних частин алгоритму, згідно з набору аксіом і

теорем САА-М. Основні перетворення САА-М наведені далі в таблиці. Перетворення розбиті на три основні групи: властивості альтернативи, властивості циклу і загальні властивості.

Таблиця

Властивості альтернативи	
$C * {}_{\alpha}(A, B) = {}_{C \bullet \alpha}(C * A, C * B)$	Внесення оператора до альтернативи (зліва)
${}_{\alpha}(A, B) * C = {}_{\alpha}(A * C, B * C)$	Внесення оператора до альтернативи (справа)
${}_{\alpha}(A, B) = \underline{\alpha} * A \parallel \overline{\underline{\alpha}} * B$	Розпаралелювання альтернативи
Властивості циклу	
${}_{\alpha}\{A * B\} = {}_{\alpha}\{A\} \parallel {}_{\alpha}\{B\}$	Розпаралелювання
$\{A\}_{\alpha} = A * {}_{\alpha}\{A\}$	Перетворення циклів
${}_{\alpha}\{A\} = {}_{\alpha}\{\overline{\underline{\alpha}} * A\}$	Внесення фільтра в тіло циклу
${}_{\alpha}\{A\} = \underline{\alpha} \parallel \overline{\underline{\alpha}} * A * {}_{\alpha}\{A\}$	Розпаралелювання з використанням фільтра
${}_{\alpha}\{A\} = {}_{\alpha}\{A\} * \underline{\alpha}$	Вставка фільтра
${}_{\alpha \vee \beta}\{A\} = {}_{\alpha}\{\overline{\underline{\beta}} * A\} \parallel {}_{\beta}\{\overline{\underline{\alpha}} * A\}$	Розпаралелювання, якщо умова циклу – диз'юнкція
${}_{\alpha}\{\overline{\underline{\beta}} * A\} = {}_{\alpha \vee \beta}\{A\} * \underline{\alpha}$	Перетворення умови циклу
${}_{\alpha \vee \beta}\{B\} * {}_{\alpha}\{A\} = {}_{\alpha}\{\#\underline{\beta} * A \parallel \overline{\underline{\beta}} * B\#\}$	Розпаралелювання циклів зі схожими умовами
${}_{\alpha \wedge \beta}\{A\} = {}_{\alpha}\{A\} * \underline{\beta} \parallel {}_{\beta}\{A\} * \underline{\alpha}$	Розпаралелювання, якщо умова циклу – кон'юнкція
${}_{\alpha \wedge \beta}\{A\} = {}_{\beta}\{A\} * {}_{\alpha}\{A\}$	Розділення циклу
$\overline{\underline{\alpha}} * {}_{\alpha}\{A * \underline{\alpha}\} = \overline{\underline{\alpha}} * A * \underline{\alpha}$	Вилучення циклу із вкладеним фільтром
${}_{\alpha}\{\overline{\underline{\alpha}} * {}_{\bar{\alpha} \wedge \beta}(A, B)\} = {}_{\alpha}\{\overline{\underline{\alpha}} * {}_{\beta}(A, B)\}$	Альтернатива, вкладена у цикл
${}_{\alpha}\{A * {}_{\beta}\{B\} * C\} = \underline{\alpha} \parallel \overline{\underline{\alpha}} * A * {}_{\beta}\{B\} * C * {}_{\alpha}\{A * C\}$	Вкладені цикли
Загальні властивості	
$\underline{\alpha} * A * \overline{\underline{\alpha}} = N$	Фільтри з протилежними умовами
$A * \#B \parallel C\# = \#A * B \parallel A * C\#$	Розподільча властивість паралельного виконання
$\#A \parallel B\# * C = \#A * C \parallel B * C\#$	Розподільча властивість паралельного виконання

Слід зазначити, що кількість перетворень може змінюватися в процесі роботи з системою. Всі регулярні вирази, що описують правила еквівалентних перетворень зберігаються в базі даних системи і можуть бути змінені або додані, використовуючи вбудований редактор правил перетворення. Так само існує можливість додавання нового перетворення, яке було описано, наприклад, за допомогою сторонньої програми роботи з регулярними виразами.

Реалізація

Система САПП реалізована на платформі Microsoft. Net Framework мовою програмування C #. Як система керування базою даних використовується Microsoft Office Access 2007. Платформа Microsoft. Net має вбудовані потужні засоби для обробки регулярних виразів, засоби для побудови дерев і роботи з базами даних.

Система, окрім основних компонент, має так само ряд додаткових функцій, таких як, редактор правил перетворення, редактор схем алгоритмів, редактор співставлень операторів мов програмування і САА, редактор

зв'язків за даними між окремими операторами, редактор набору вбудованих операторів для С-подібних мов програмування.

На даний момент, розробляються два додаткових модулі. А саме модуль генерації САА-схем на основі UML-діаграм, який дозволить інтегрувати апарат САА з сучасною мовою моделювання та візуалізації UML. І модуль перетворення САА-М-схем у мережі Петрі, який дозволить проводити моделювання роботи перетворених САА-М-схем без створення відповідних програм і виконання їх на кластері, що дозволить позбутися необхідності використання часу кластерА виключно для моделювання. Ці модулі так само дозволять розширити функціональність редактора схем алгоритмів, додати можливість візуального відображення САА-М-схем, що дозволить зробити редагування схем більш зручним та наочним.

На рис. 3 наведено знімок екранА, де представлено основне вікно програми з відкритим редактором схем алгоритмів.

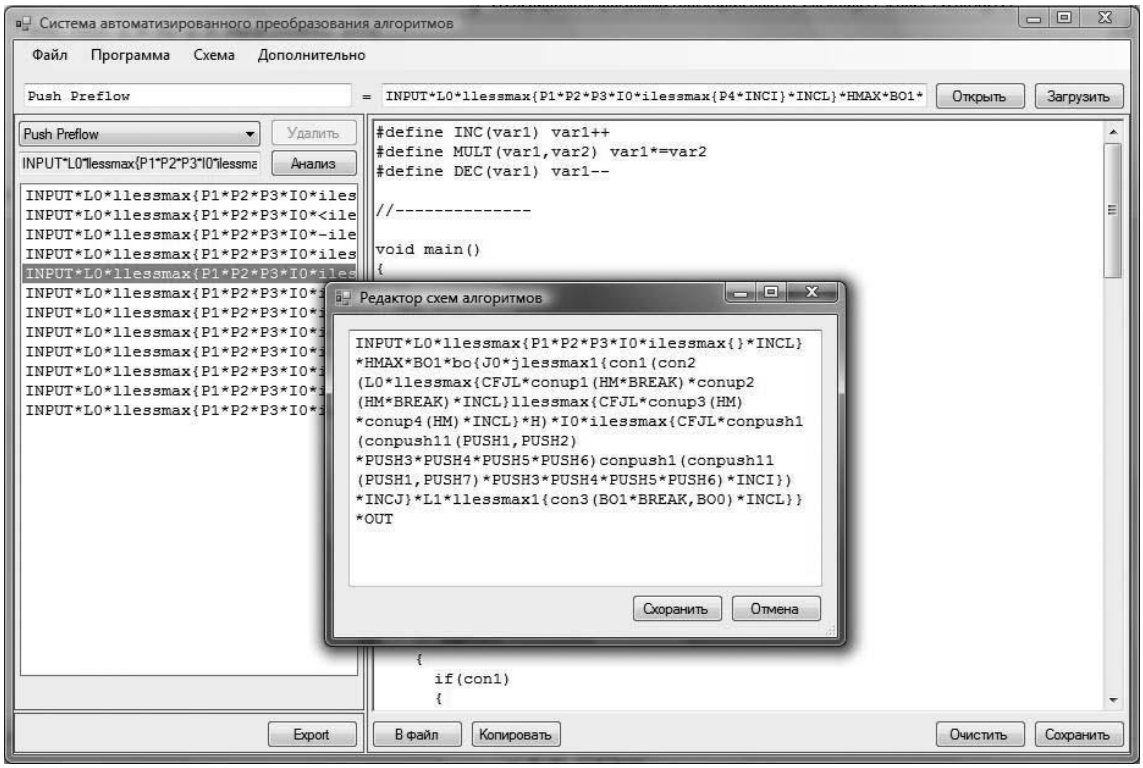

Рис. 3

Приклад роботи підсистеми

Роботу перетворювача схем можна проілюструвати на наступному прикладі. Взято алгоритм проштовхування передпотоку, який використовується для знаходження максимального потоку в мережі. Вихідна схема алгоритму має вигляд:

$$Push\ Preflow=INPUT*L0*_{llessmax}\{P1*P2*P3*I0*_{ilessmax}\{P4*INCI\}*INCL\}*$$
$$HMAX*BO1*_{bo}\{J0*_{jlessmax1}\{con1(_{con2}(L0*_{llessmax}\{CFjl*_{conup1}(HM*BREAK)*_{conup2}(HM*$$
$$BREAK)*INCL\}*_{llessmax}\{CFjl*_{conup3}(HM)*_{conup4}(HM)*INCL\}*H)*I0*_{ilessmax}\{CFji*$$
$$_{conpush1}(_{conpush11}(PUSH1,PUSH2)*PUSH3*PUSH4*PUSH5*PUSH6)*$$
$$_{conpush1}(_{conpush11}(PUSH1,PUSH7)*PUSH3*PUSH4*PUSH5*PUSH6)*INCI\})*INCJ\}*$$
$$L1*_{llessmax1}\{_{con3}(BO1*BREAK,BO0)*INCL\}\}*OUT$$

З використанням автоматизованого перетворювача САА-М-схем алгоритмів, був отриманий ряд нових схем вихідного алгоритму. Далі наведено дві з них, які є найбільш розпаралеленими серед усіх:

$$Push\ Preflow\ 1=\ INPUT*L0*_{llessmax}\{P1*P2*P3*I0*_{ilessmax}\{P4*INCI\}*INCL\}*$$
$$HMAX*BO1*_{bo}\{J0*_{jlessmax1}\{con1(con2(L0*_{llessmax}\{CFJL*$$
$$<\!\!-conup1\text{-}*HM*BREAK||\text{-}!conup1\text{->}$$
$$_{conup2}(HM*BREAK)*INCL\}*_{llessmax}\{CFJL*_{conup3}(HM)*_{conup4}(HM)*INCL\}*H)*I0*$$
$$_{ilessmax}\{CFJL*_{conpush1}(_{conpush11}(PUSH1,PUSH2)*PUSH3*PUSH4*PUSH5*PUSH6)*$$
$$_{conpush1}(_{conpush11}(PUSH1,PUSH7)*PUSH3*PUSH4*PUSH5*PUSH6)*INCI\})*INCJ\}*$$
$$L1*_{llessmax1}\{con3(BO1*BREAK,BO0)*INCL\}\}*OUT$$

$$Push\ Preflow\ 2=INPUT*L0*_{llessmax}\{P1*P2*P3*I0*_{ilessmax}\{P4*INCI\}*INCL\}*$$

$HMAX*BO1*_{bo}\{J0*_{jlessmax1}\{_{con1}(_{con2}(L0*_{llessmax}\{CFJL*$
<-conup1-*HM*BREAK||-!conup1->*
<-conup2-*HM*BREAK||-!conup2->*
$INCL\}_{llessmax}\{CFJL*$
<-conup3-*HM||-!conup3->*
<-conup4-*HM||-!conup4->*
$INCL\}*H)*I0*_{ilessmax}\{CFJL*_{conpush1}($
<-conpush11-*PUSH1||-!conpush11-*PUSH2>*
$PUSH3*PUSH4*PUSH5*PUSH6)_{conpush1}($
<-conpush11-*PUSH1||-!conpush11-*PUSH7>*
$PUSH3*PUSH4*PUSH5*PUSH6)*INCI\})*INCJ\}*L1*_{llessmax1}\{$
<-con3-*BO1*BREAK||-!con3-*BO0>*
$INCL\}\}*OUT$

Моделювання роботи схем алгоритмів здійснюються на кластерах Київського національного університету імені Тараса Шевченка, під управлінням Microsoft Compute Cluster Server та операційної системи Linux [8, 9].

Висновки

Апарат САА В.М. Глушкова виправдав себе, як потужний і зручний засіб для аналітичного представлення алгоритмів та отримання на їх основі нових більш ефективних версій. Використання цього апарата є зручною і потужною основою для розробки автоматизованого інструментарію перетворення алгоритмів.

Запропонована концепція створення автоматизованого перетворювача САА-схем алгоритмів є максимально повною та гнучкою, оскільки розроблялася відповідно до принципу максимальної розширюваності та гнучкості застосування. Можливість додавання та редагування правил перетворення та інтеграція з потужними сучасними засобами розробки й моделювання програмного забезпечення дозволяють знайти найширшу область застосування системи.

1. *Бойко Ю.В., Погорілий С.Д., Шкуліпа І.Ю.* Дослідження паралельних схем алгоритму Прима // Математические машины и системы. – 2007. –№ 2.
2. *Погорелый С.Д., Шкулипа И.Ю.* Концепция создания автоматизированной параметрической системы проектирования параллельных алгоритмов и их программных реализаций // Кибернетика и системный анализ. – 2009. – № 6. –С. 118–124.
3. *Погорілий С.Д., Камардіна О.О.* Дослідження та створення інструментальних засобів автоматизованої трансформації схем алгоритмів, Проблеми програмування. 2004. Спеціальний випуск. Матеріали міжнародної конференції УкрПРОГ . – 2004.
4. *Погорілий С.Д.* Програмне конструювання. Підручник за редакцією академіка АПН України Третяка О.В. ВПЦ Київський університет. 2005 – 438 с.
5. *Е.Л.Ющенко, Г.Е.Цейтлин, В.П.Грицай, Т.К.Терзя*н. Многоуровневое структурное проектирование программ, М.: 1989. – 208 с.
6. *К.Л. Ющенко, С.В. Суржко, Г.О. Цейтлін, А.І. Шевченко*. Алгоритмічні алгебри. – К.: – 1997. – 480 с.
7. *Дж. Хопкрофт, Р. Мотвани, Дж. Ульман* Введение в теорию автоматов, языков и вычислений, 2-е изд. Вильямс. – 2002. – 528 с.
8. *С.Д. Погорілий, Ю.В. Бойко, Д.Б. Грязнов, О.Д. Ломакін, В.А. Мар'яновський.* Концепція створення гнучких гомогенних архітектур кластерних систем 2008. – Проблеми програмування. № 2–3. Спеціальний випуск. Матеріали міжнародної конференції УкрПРОГ 2008.
9. http://cluster.univ.kiev.ua