DOI https://doi.org/10.15407/usim.2020.02.003
УДК 681.3.006

А.М. ПЕТРУШЕНКО, кандидат фіз.-мат. наук, доцент, Київський національний університет імені Тараса Шевченка, 03022, м. Київ, просп. Академіка Глушкова, 4Д, факультет комп'ютерних наук та кібернетики,
anatoly@mytaskhelper.com

ПРИНЦИП МІКРОПРОГРАМНОГО КЕРУВАННЯ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ОПЕРАЦІЙНИХ ПРИСТРОЇВ. II[1]

Розглядається один із можливих підходів до вирішення на базі принципу мікропрограмного керування проблеми комплексної – від постановки задачі до отримання ескіза друкованої плати – автоматизації процесу розробки операційних пристроїв. Підхід демонструється на прикладі синтезу в діалоговій трансформаційній машині – інструментарії алгебро-граматичних методів подання знань у різноманітних предметних областях – керуючого автомата операційного пристрою, що реалізує операцію додавання.

Ключові слова: *принцип мікропрограмного керування, дискретний перетворювач інформації, операційний пристрій, алгебра алгоритмів, алгебро-граматичний метод подання знань, діалогова трансформаційна машина.*

Автоматизація процесу синтезу операційних пристроїв

Мови запису мікроалгоритмів. Здійснення будь-якої операції в ОП починається з підготовчих МО витягування операндів із пам'яті та фіксації їх у певному порядку в регістрах (операційних елементах) ОА. Задля скорочення пояснень, розгляньмо роботу ОП, який виконує окрему операцію, від моменту, коли всі підготовчі операції витягування та фіксації операндів в ОА вже завершено, й КА та ОА встановлено в початкові стани. Для прикладу розгляньмо мікроалгоритм додавання чисел $X = x_0x_1...x_n$ і $Y = y_0y_1...y_n$ у прямих кодах із використанням обернених кодів для віднімання (тут x_0 і y_0 — знаки чисел, x_i і y_i — розряди мантис, $i = 1, ..., n$):

- Якщо знак операнда X є додатнім, тобто $x_0 = 0$, то подати на суматор *SM* прямий код числа X. Якщо ж знак операнда X є від'ємним, тобто $x_0 = 1$, то подати на суматор *SM* обернений код числа X.
- Якщо знак y_0 другого операнда Y і знак заданої операції s_0 збігаються, тобто $y_0=s_0$, то подати на суматор *SM* прямий код числа Y. Якщо ж $y_0 \neq s_0$, то подати на суматор *SM* обернений код числа Y.
- Утворити прямий код результату $R = XsY$, де s — символ операції.

Із наведеного алгоритму випливає, що операційний блок додавання має включати в себе три регістри *RGX*, *RGY*, *RGR*, суматор *SM* і блок місцевого керування. Виходи *RGX* і *RGY* мають бути підключені до входів *SM*, а виходи суматора — до входів регістра *RGR*.

Для запису мікроалгоритмів можна використовувати мову ЛСА, запропоновану А.А. Ляпуновим. Розглянутий мікроалгоритм МАС у термінах ЛСА має вигляд:

[1] Перша частина статті опублікована в журналі "Control Systems and Computers", №1, 2020, С. 3-22.

(МАС) = (НО)(x_0)↑[1](ВК*RGX*)↑[2]↓[1](ВОК*RGX*)↓[2]($\neg y_0 s_0 \vee y_0 \neg s_0$)↑[3](ВК*RGY*)($\neg(\neg y_0 s_0 \vee y_0 \neg s_0)$)↑[4]↓[3](ПОК*RGY*)↓[4](ПК*RGR*)(z_0)↑[5](ВК*RGR*)($\neg z_0$)↑[6]↓[5](ПОК*RGR*)↓[6](КО).

Тут для запису мікроалгоритму використовувалися такі позначення МО: ВК — видача прямого коду з регістра; ВОК — видача оберненого коду з регістра; ПК — прийом прямого коду на регістр; ПОК — прийом оберненого коду на регістр; виконання МАС починається за сигналом початку операції НО, а завершується за сигналом кінця операції КО.

Мова ЛСА — не єдина мова запису мікроалгоритмів. На практиці ширше розповсюдження отримала мова ГСА Л.А. Калужніна. Еквівалентність цих форм подання алгоритмів, їхні особливості, переваги та недоліки описано в [6, 16–24]. Тут лише зазначмо, що класичною працею з теорії ЛСА є праця [22]. Однак, як зазначалося в [24], цю працю написано доволі складною мовою, яка розрахована радше на спеціалістів із математичної логіки, ніж на програмістів. З'ясувалося [20, 24], що коли замість ЛСУ, що мають вид рядків символів, розглядати орієнтовані графи, в яких оператори та ЛУ є вершинами графа, а стрілки зображують передачі управління, то вже завдяки цьому можна зробити більшість конструкцій прозорішими та зрозумілішими і, зокрема, спростити систему перетворень (у [24] кількість відповідних правил скорочено із сімнадцяти до шести).

У цій роботі для запису мікроалгоритмів пропонується використовувати мову САА\Д, яка є природно-лінгвістичною формою подання знань про алгоритми й асоційовані з ними програми й апаратні засоби. Опис алгоритму в мові САА\Д називають САА\Д-схемою.

САА\Д — вхідна мова діалогової трансформаційної машини. Як відомо, в САА операції сигнатури Ω, що набувають значення в F_M, однозначно співвідносяться з головними програмістськими конструкціями — послідовним виконанням операторів (композиція), переходом за умовою (α-диз'юнкція) та циклом (α–ітерація). Тоді операції сигнатури Ω зі значеннями в P_M дають змогу: конструювати різноманітні функції тризначної логіки (булеві операції) та прогнозувати результат обчислення (ліве множення умови на оператор).

У мові САА\Д операціям сигнатури Ω у відповідність поставлено такі конструкції:

- диз'юнкції — α АБО β≡α + β ≡ *DI*(α, β) (тобто у мові САА\Д для диз'юнкції може використовуватися будь-яка із трьох зазначених еквівалентних форм запису);
- кон'юнкції — *CO*(α, β) ≡ α *I* β≡α∗β;
- запереченню — НЕ(α) ≡¬α;
- лівому множенню умови на оператор — ПІСЛЯ *P* УМОВА α;
- композиції відображень — *P**Q*≡*SEQ*(*P*, *Q*)≡ ≡ *P* ПОТІМ *Q*;
- α-диз'юнкції — ЯКЩО α ТО *P* ІНАКШЕ $Q \equiv [\alpha \to P, \neg\alpha \to Q] \equiv (P \leftarrow \alpha \to Q)$;
- α-ітерації — ПОКИ НЕ α ЦИКЛ $P \equiv [\alpha \leftrightarrow P]$.

Для запису абстракцій операторів й умов у мові САА\Д використовуються змістовні ідентифікатори — тексти українською чи будь-якою іншою природною мовою, обрамлені апострофами чи лапками залежно від того, чи є цей змістовний ідентифікатор ідентифікатором умови чи оператора. Абстрактні оператори й умови поділяються на елементарні та складені. Елементарний оператор (умова) — це оператор (умова), що у САА\Д-схемі вважається первинною абстракцією. Складений оператор (умова) будується з елементарних за допомогою керуючих конструкцій мови САА\Д. Мова САА\Д є «відкритою знизу» — вона дає програмістові змогу вводити нові абстракції операторів й умов у межах актуальної предметної області, забезпечує первинність цих абстракцій та отримання похідних абстрактних об'єктів із первинних. Тип змістовного ідентифікатора (елементарний чи складений) задається неявно і визначається з контексту САА\Д-схеми. Мова САА\Д є також «відкритою згори» — вона допускає розширення в ДТМ будь-якими конструкціями, що відповідають операціям модифікованих САА. САА\Д-схеми мають таку структуру:

СХЕМА (назва схеми);
рівність;
...
рівність;
КІНЕЦЬ [СХЕМИ (назва схеми)];

Як бачимо, у САА\Д-схемі між ключовими словами СХЕМА і КІНЕЦЬ розташована послідовність рівностей, відокремлених одна від одної знаком «;». І тут кожна наступна рівність конкретизує (засобами САА\Д) абстракцію певного складеного оператора чи умови, змістовний ідентифікатор якого наявний в одній із попередніх рівностей. При запису рівностей припустимі ланцюжки символів "=", "–", "*", "+" довільної довжини, що забезпечує графічну наочність САА\Д-схем.

Приклад проектування у мові САА\Д мікроалгоритму додавання. Розгляньмо можливий (початковий) порядок дій користувача системи ДТМ (детальніше див., наприклад, [12–15]) при проектуванні в мові САА\Д мікроалгоритму *МАС* додавання чисел у машинах із фіксованою комою. Проектування вестимемо в припущенні, що всі необхідні для розв'язання задачі знання вже містяться в базі знань ДТМ. Зокрема, припускатимемо, що в базі знань містяться такі взаємопов'язані групи понять:

1. «РЕГІСТР <ім'я регістра>», з яким асоційовані, зокрема, операції «ПРИЙМАЄ», «ЗСУНУТИ» тощо. При цьому операція «ПРИЙМАЄ» уточнюється, зокрема, параметрами «ІЗ СУМАТОРА <ім'я суматора>» і, далі, «РЕЗУЛЬТАТ <ім'я результату> ОПЕРАЦІЇ». Операція «ЗСУНУТИ» може уточнюватися, зокрема, параметрами «НА 1 РОЗРЯД ЛІВОРУЧ», «НА 1 РОЗРЯД ПРАВОРУЧ» тощо. Із поняттям «РЕГІСТР <ім'я регістра>» також можуть бути пов'язані умови «ДОРІВНЮЄ 0», «МОЛОДШИЙ РОЗРЯД ДОРІВНЮЄ 1» тощо.

2. «СУМАТОР <ім'я суматора>», зокрема, з операцією «ПРИЙМАЄ» та її уточненнями «З РЕГІСТРА <ім'я регістра>» і, далі, «ПРЯМИЙ КОД» або «ІНВЕРСНИЙ КОД» і, далі, «МАНТИСИ» і, далі, «ПЕРШОГО ОПЕРАНДА <ім'я операнда>» або «ДРУГОГО ОПЕРАНДА <ім'я операнда>» тощо.

3. «ШИНА ДАНИХ <ім'я шини>», зокрема, з операцією «ПРИЙМАЄ» та її уточненнями «З РЕГІСТРА <ім'я регістра>» і, далі, «ПРЯМИЙ КОД» або «ІНВЕРСНИЙ КОД2 і, далі, «МАНТИСИ» або «ПЕРШОГО ОПЕРАНДА <ім'я операнда>» або «ДРУГОГО ОПЕРАНДА <ім'я операнда>» тощо.

4. «ЗНАК <ім'я знака>» і, далі, «ПЕРШОГО ОПЕРАНДА <ім'я операнда>» або «ДРУГОГО ОПЕРАНДА <ім'я операнда>» і, далі, «— ДОДАТНІЙ» або «— ВІД'ЄМНИЙ» тощо.

Нехай у базі знань ДТМ крім наведених взаємопов'язаних груп понять ще містяться необхідні для нашої задачі імена регістрів *RGX*, *RGY*, *RGR*; ім'я суматора *SM*; ім'я шини *BD*; імена *X*,*Y* і *R*, відповідно, першого, другого операндів та результату операції, а також відповідні їм імена знаків x_0, y_0, s_0. Припустімо також, що корстувач налаштував ДТМ на предметну область, яку, для визначеності, назвімо «СИНТЕЗ ОП», вказав, що проектований мікроалгоритм називається «ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ» і відкрив файл з іменем *MAC.SAA*. У цьому разі вміст верхнього інформаційного рядка та робочого поля проектованого виразу в інтелектуальному редакторі – компоненті машини виводу ДТМ — матиме такий вигляд:

СИНТЕЗ ОП. *MAC.SAA*. ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ
СХЕМА *MAC*;
«ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ»===
(#*A*)
КІНЕЦЬ СХЕМИ *MAC*;

де символ #*A* — активний (тобто блимає на екрані дисплея) й управління передається в керуюче вікно. Відповідно до алгоритму подальший порядок дій користувача системи може бути, наприклад, таким. Зробімо в керуючому вікні вибір «Конструкцією САА\Д», а в його підменю — *A***B*. Як наслідок, в робочому полі проектованого виразу (інформаційний рядок опускатимемо) з'явиться:

СХЕМА *MAC*;
«ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ»===
((#*A*) * (#*B*))
КІНЕЦЬ СХЕМИ *MAC*;

де символ #*A* — активний. У вікні керування робимо вибір «Конструкцією САА\Д», а в його підменю — ЯКЩО *V* ТО *A* ІНАКШЕ *B*. Як наслідок робоче поле набуде вигляду:

СХЕМА *MAC*;
«ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ»===
((ЯКЩО #*V* ТО #*A* ІНАКШЕ #*B*) * (#*B*))
КІНЕЦЬ СХЕМИ *MAC*;

де символ #*V* — активний. У цьому стані робимо вибір «УМОВА *V* Є: ЕЛЕМЕНТАРНОЮ». Після цього ДТМ пропонує користувачу два списки: в першому перераховано поняття даної предметної області, у другому — змінні, асоційовані з поняттями з першого списку. Оберімо поняття «ЗНАК» і, далі, змінну — ім'я знака; нехай це буде x_0 (якщо відповідної змінної в базі знань ДТМ немає, то її потрібно на цьому кроці створити — переважно це робить інженер із знань). Після цього послідовно з'являтимуться у відповідних вікнах списки уточнювальних понять. Обираючи з них потрібні, сформуймо за допомогою цих понять речення (елементарну умову) «ЗНАК x_0 ПЕРШОГО ОПЕРАНДА *X* — ДОДАТНІЙ», яке автоматично підставляється у текст САА\Д-схеми. Як наслідок, на екрані дисплея (у робочому полі) з'явиться:

СХЕМА *MAC*;
«ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ»===
((ЯКЩО «ЗНАК x_0 ПЕРШОГО ОПЕРАНДА *X* — ДОДАТНІЙ» ТО #*A* ІНАКШЕ #*B*) * (#*B*))
КІНЕЦЬ СХЕМИ *MAC*;

Активним стає символ #*A*. У цьому стані робимо вибір «ОПЕРАТОР Є: ЕЛЕМЕНТАРНИМ» і формуємо речення предметної області (елементарний оператор) «СУМАТОР *SM* ПРИЙМАЄ З РЕГІСТРА *RGX* ПРЯМИЙ КОД МАНТИСИ ПЕРШОГО ОПЕРАНДА *X*». Це речення автоматично підставляється у текст САА\Д-схеми, активним стає *B* (після конструкції ІНАКШЕ) тощо до завершення проектування мікроалгоритму *MAC* у мові САА\Д (див. далі).

Як уже зазначалося, у загальному випадку САА\Д-схема має багаторівневу структуру. Ознакою того, що проектування САА\Д-схеми певного рівня завершено, є відсутність у проектованому виразі конструкцій виду #символ. Текст САА\Д-схеми автоматично записується в базу знань ДТМ у файл *MAC.SAA*. ДТМ аналізує цю схему на наявність у ній ще не уточнених складених операторів чи умов. Якщо такі операторі чи умови є, то ДТМ «очищує» екран дисплея, змінює вміст верхнього інформаційного рядка та робочого поля проектованого виразу і переходить у стан проектування одного з неуточнених складених операторів чи однієї зі складених умов.

Оскільки в нашому прикладі у тексті САА\Д-схеми вже не залишилося неуточнених складених операторів чи умов, то для ДТМ це є ознакою того, що проектування САА\Д-схеми завершено і ДТМ переходить у стан ПРОЕКТ головного меню. Якщо тепер подивитися на вміст файла *MAC.SAA*, в якому зберігається САА\Д-схема, то виявимо в ньому текст:

СХЕМА *MAC*;
«ДОДАВАННЯ ЧИСЕЛ У ПРЯМИХ КОДАХ»===
((ЯКЩО «ЗНАК x_0 ПЕРШОГО ОПЕРАНДА *X* — ДОДАТНІЙ»
ТО «СУМАТОР *SM* ПРИЙМАЄ З РЕГІСТРА *RGX* ПРЯМИЙ КОД МАНТИСИ ПЕРШОГО ОПЕРАНДА *X*»
ІНАКШЕ «СУМАТОР *SM* ПРИЙМАЄ З РЕГІСТРА *RGX* ІНВЕРСНИЙ КОД МАНТИСИ ПЕРШОГО ОПЕРАНДА *X*») ***
(ЯКЩО «ЗНАК y_0 ДРУГОГО ОПЕРАНДА *Y* — ДОДАТНІЙ»
ТО «СУМАТОР *SM* ПРИЙМАЄ З РЕГІСТРА *RGY* ПРЯМИЙ КОД МАНТИСИ ДРУГОГО ОПЕРАНДА *Y*»
ІНАКШЕ «СУМАТОР *SM* ПРИЙМАЄ З РЕГІСТРА *RGY* ІНВЕРСНИЙ КОД

МАНТИСИ ДРУГОГО ОПЕРАНДА *Y*») ***

«РЕГІСТР *RGR* ПРИЙМАЄ ІЗ СУМАТОРА *SM* РЕЗУЛЬТАТ *R* ОПЕРАЦІЇ») ***

ЯКЩО «ЗНАК z_0 РЕЗУЛЬТАТУ *R* ОПЕРАЦІЇ — ДОДАТНІЙ»

ТО «ШИНА ДАНИХ *BD* ПРИЙМАЄ З РЕГІСТРА *RGR* ПРЯМИЙ КОД РЕЗУЛЬТАТУ *R* ОПЕРАЦІЇ ІНВЕРСНИЙ КОД»

ІНАКШЕ «ШИНА ДАНИХ *BD* ПРИЙМАЄ З РЕГІСТРА *RGR* ІНВЕРСНИЙ КОД РЕЗУЛЬТАТУ *R* ОПЕРАЦІЇ»))

КІНЕЦЬ СХЕМИ *MAC*;

У процесі проектування САА/Д-схеми мікроалгоритму *MAC* ДТМ автоматично будує формулу цієї САА\Д-схеми 0151, вона утворюється із даної САА\Д-схеми через заміну в ній змістовних складених й елементарних операторів та умов відповідними їм кодами:

СХЕМА *MAC*;

*B*0 = ((ЯКЩО *X*1 ТО *Y*1 ІНАКШЕ *Y*2) *

(ЯКЩО *X*2 ТО *Y*3 ІНАКШЕ *Y*4) *

(*Y*5) * (ЯКЩО *X*3 ТО *Y*6 ІНАКШЕ *Y*7))

КІНЕЦЬ СХЕМИ *MAC*;

Легко побачити, що формула САА\Д-схеми є нічим іншим, ніж текстовим аналогом закодованої граф-схеми мікроалгоритму. У деяких випадках формулу САА\Д-схеми зручно інтерпретувати як закодовану граф-схему автомата Мура з позначками станів.

Формулу САА\Д-схеми можна також розглядати як вираз в алгебрі алгоритмів і застосовувати до неї співвідношення, справедливі в цій алгебрі. Для цього формулу потрібно попередньо привести («транслювати») до синтаксично правильного виразу в мові АНАЛІТИК [25], використовуючи таке кодування конструкцій мови САА\Д у мові АНАЛІТИК: диз'юнкції відповідає $\alpha+\beta$, кон'юнкції — $\alpha*\beta$, запереченню — НЕ(α), лівому множенню умови на оператор — ПР(*P*, α), композиції відображень — *P**Q*, α-ітерації — ІТ(α, *P*), α-диз'юнкції — ДИ(α, *P*, *Q*), а потім результат «трансляції» подати на вхід компоненти ДТМ, призначеною для автоматизації аналітичних перетворень [11].

Після завершення проектування САА\Д-схеми її формула надходить на вхід спеціальної утиліти, яка здійснює формування всіх можливих комбінацій умов, що входять у цю САА\Д-схему. Далі ДТМ запитує в користувача:

- перелік операцій, які має реалізовувати ОП;
- інтерфейси ОП;
- тип КА (Мілі чи Мура);
- тип ОА;
- тип елементів пам'яті (*D*-, *T*-, *RS*- чи *JK*-тригери — всі вони задовольняють умові повноти функцій переходів та виходів [1]);
- склад функціонально повної системи логічних елементів. У цій версії ДТМ КА завжди синтезується як автомат Мілі і, якщо необхідно, інтерпретується як автомат Мура, а як функціонально повна система логічних елементів «зашито» булевий базис.

Зазначмо, що крім формули САА\Д-схеми ДТМ у процесі проектування САА\Д-схеми автоматично генерує також фрагмент програми мовою *C*, який відповідає цій схемі. Далі цей фрагмент розширюється стандартним набором декларацій і функцій до програми, яка здійснює:

- позначку станів у формулі САА\Д-схеми;
- кодування вхідних, вихідних символів і символів станів автомата Мілі;
- виконання мікроалгоритму для кожної зі знайдених раніше (спеціальною утилітою) комбінацій умов для пошуку всіх можливих шляхів виконання мікроалгоритму та побудови на базі цієї інформації таблиць переходів і виходів автомата Мілі;
- побудову структурних таблиць переходів і виходів автомата Мілі.

Далі ДТМ будує (в канонічній формі) БФ збудження елементів пам'яті та БФ виходів автомата Мілі; отримані БФ мінімізуються одним із аналітичних методів [14].

Для побудови за отриманими функціями збудження елементів пам'яті та виходів електричної принципової схеми, аналізу питань стійкості її роботи, розробки друкованої плати в ДТМ інтегровано систему *PCAD* [26], яка є набором утиліт, що працюють в інтерактивному режимі й призначені для автоматизації проектування схем радіоелек-

тронної апаратури. Зокрема, утиліта *PCCAPS* є графічним редактором схем радіоелектронної апаратури. ДТМ за названими БФ спочатку генерує командний файл для цієї утиліти, в якому описується процес створення мікропрограмного автомата на логічних елементах. У процесі виконання командного файлу формується принципова схема КА, яка надалі використовується для трасування друкованої плати. Ці дії здійснюються у відповідних утилітах системи *PCPACK, PCAD: PCNODES, PCCARDS, PCPLASE, PCROUTE* тощо. На цих етапах можна задавати різноманітні параметри проектування: крок координатної сітки, відстань між корпусами, ширину друкованих провідників, кількість шарів друкованої плати тощо. Після виконання згаданих етапів отримуємо ескіз друкованої плати.

Результати та висновки

З метою автоматизації процедури синтезу ОП і програм потрібної якості апарат САА був покладений в основу мови САА\Д, яка є природно-лінгвістичною формою подання знань про алгоритми та асоційовані з ними апаратні засоби й програми. Використання мови САА\Д в якості вхідної мови ДТМ дозволяє – за рахунок підключення до неї "цільових" мов і відповідних їм методів проектування – поширити методи синтезу ОП на "твердій" логіці на мову САА\Д і відкриває можливість для здійснення комплексної – від постановки задачі до отримання ескіза друкованої плати – автоматизації проектування ОП, що реалізують МПР виконання заданих операцій. При цьому процес розробки САА\Д-схем в ДТМ базується на дружній взаємодії користувача і ДТМ, яка забезпечується, насамперед, можливістю багаторівневого структурного проектування САА\Д-схем у стилі, близькому до природної мови – ця можливість досягається за рахунок використання у САА\Д-схемах змістовних (складених і елементарних) операторів і умов. Багаторівневість і змістовні ідентифікатори істотно полегшують розуміння алгоритмів, поданих у мові САА\Д.

Крім цього, у даній роботі показана реалізація в ДТМ, по суті, особливого типу поліморфізму, коли в якості параметра САА\Д-схеми виступає предметна область. В результаті, в залежності від вибору предметної області, по одній і тій самій САА\Д-схемі в ДТМ може бути отриманий або ескіз друкованої плати КА деякого ОП, або програма, що асоційована з САА\Д-схемою і моделює роботу даного ОП.

Остання можливість є дуже привабливою для розробників апаратних засобів. Вона дозволяє оцінити характеристики проектованого пристрою за програмою, яка його моделює, що набагато дешевше, ніж виготовлення діючого прототипу. Цей алгоритм (САА\Д-схему) можна багаторазово коректувати і відразу ж бачити зміни характеристик ОП на його програмній моделі. Після того, як буде отримано алгоритм, що задовольняє всім вимогам до пристрою, його транслюють вже не в програму, яка його моделює, а в опис ОП. Таким чином значно скорочуються терміни, підвищується якість і знижується вартість розробки ОП.

Нарешті, САА\Д-схема допускає можливість представлення довільного алгоритму у вигляді САА\Д-формули в деякій алгебрі з метою застосування до цієї формули співвідношень. Як наслідок, пропонований підхід до автоматизації проектування ОП забезпечує можливість їх оптимізації за різними критеріями не лише традиційними методами абстрактної і структурної теорії автоматів, а й методами алгебри алгоритмів.

ЛІТЕРАТУРА

1. *Глушков В.М.* Синтез цифровых автоматов. М.: Физматгиз., 1962. 476 с.
2. *Глушков В.М.* Теория автоматов и вопросы проектирования структур цифровых машин. Кибернетика. 1965. № 1. С. 3–11.
3. *Глушков В.М.* Теория автоматов и формальные преобразования микропрограмм. Кибернетика. 1965. №5. С. 1–10.
4. *Глушков В.М., Летичевский А.А.* Теория дискретных преобразователей. Избранные вопросы алгебры и логики. Новосибирск: Наука, 1973. С. 5–40.
5. *Капитонова Ю.В., Летичевский А.А.* Математическая теория проектирования вычислительных систем. М.: Наука, 1988. 295 с.
6. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. 3-е изд., пераб. и доп. Киев: Наук. думка, 1989. 376 с.
7. *Анисимов А.В.* Рекурсивные преобразователи информации. Киев: Вища шк., 1987. 231 с.
8. *Bauer F.L., Wossner H.* Algorithmische sprache und programmentwicklung. Berlin ets.: Springer Verlag, 1981. 513 p.
9. *Ершов А.П.* Трансформационная машина: тема и вариации Проблемы теоретического и системного программирования. Новосибирск. НГУ, 1979. С. 5–45.
10. *Петрушенко А.Н.* О диалоговых вычислениях в алгоритмических алгебрах. Кибернетика. 1990. №1. С. 13–20.
11. *Петрушенко А.Н.* Об одном подходе к проблеме автоматизации оптимизирующих преобразований алгоритмов и программ. Кибернетика и системный анализ. 1991. №5. С. 127–136.
12. *Петрушенко А.Н.* Об одном подходе к решению проблемы общения человека с вычислительной системой на естественном языке. Проблемы программирования: Сб. науч. трудов. вып. 3. 1998. С. 65–72.
13. *Петрушенко А.Н., Хохлов В.А.* Об использовании естественного языка для представления абстрактных типов данных и полиморфизма. Проблемы программирования. 1999. №1. С. 76–83.
14. *Петрушенко А.Н., Хохлов В.А., Ткачев В.А., Шепетухин Е.С.* Диалоговая трансформационная машина: некоторые функциональные возможности. Проблемы программирования. 2000. № 1–2 (Спец. выпуск). С. 323–334.
15. *Петрушенко А.М., Хохлов В.А.* Концепція діалогових обчислень та деякі проблеми автоматизації програмування. Проблемы программирования. 2004. № 2–3 (Спец. выпуск). С. 37–47.
16. *Самофалов К.Г., Корнейчук В.Н., Тарасенко В.П.* Цифровые ЭВМ. К.: Вища шк., 1989. 423 с.
17. *Самофалов К.Г., Корнейчук В.Н., Тарасенко В.П., Жабин В.Н.* Цифровые ЭВМ. Практикум. К.: Вища шк., 1990. 215с.
18. *Майоров С.А., Новиков Г.И.* Структура электронных вычислительных машин. Л.: Машиностроение, 1979. 384 с
19. *Баранов С.И.* Синтез микропрограммных автоматов. Л.: Энергия, 1979. 232с.
20. *Калужнин Л.А.* Об алгоритмизиции математических задач. Проблемы кибернетики. Вып. 2. М.: Физматгиз, 1959. С. 51–67.
21. *Ляпунов А. А.* О логических схемах программ. Проблемы кибернетики. Вып. 1, М.: Физматгиз, 1958. С. 46–74.
22. *Янов Ю.И.* О логических схемах алгоритмов. Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958. С. 75–127.
23. *Яблонский С. В.* Основные понятия кибернетики. Проблемы кибернетики. Вып. 2. М.: Физматгиз, 1959. С. 7–38.
24. *Ершов А.П.* Операторные алгоритмы. 3 (об операторных схемах Янова). Проблемы кибернетики. Вып. 20. М.: Физматгиз, 1968. С. 181–200.
25. *Системы компьютерной алгебры семейства АНАЛИТИК*. Теория. Реализация. Применение. К., 2010. 762 с.
26. *Разевиг В.Д.* Применение программ PCAD и PSpise для схемотехнического моделирования на ПЭВМ: В 4 выпусках. Вып. 1: Общие сведения. Графический ввод схем. М.: Радио и связь, 1992. 72 с.

Надійшла 16.10.2019

REFERENCES

1. *Glushkov, V.M.,* 1962. Sintez tsifrovykh avtomatov. M.: Fizmatgiz. 476 p. (In Russian).
2. *Glushkov, V.M.*, 1965. “Teoriya avtomatov i voprosy proyektirovaniya struktur tsifrovykh mashin”, Kibernetika, 1, pp. 3–11. (In Russian).
3. *Glushkov, V.M.,* 1965. “Teoriya avtomatov i formalnyye preobrazovaniya mikroprogramm”. Kibernetika, 5, pp. 1–10. (In Russian).
4. *Glushkov, V.M., Letichevskiy, A.A.*, 1973. “Teoriya diskretnykh preobrazovateley”. Izbrannyye voprosy algebry i logiki. Novosibirsk: Nauka, pp. 5–40. (In Russian).
5. *Kapitonova, Yu.V., Letichevskiy, A.A.,* 1988. Matematicheskaya teoriya proyektirovaniya vychislitelnykh sistem. M.: Nauka, 295 p. (In Russian).
6. *Glushkov, V.M., Tseytlin, G.Ye., Yushchenko, Ye.L.*, 1989. Yazyki. Programmirovaniye. 3-ye izd., perab. i dop. Kyiv: Nauk. dumka, 376 p. (In Russian).
7. *Anisimov, A.V.*, 1987. Rekursivnyye preobrazovateli informatsii. Kyiv: Vishcha shk., 231 p. (In Russian).

8. *Bauer, F.L., Wossner, H.*, 1981. Algorithmische sprache und programmentwicklung. Berlin: Springer Verlag, 513 p.
9. *Yershov, A.P.*, 1979. "Transformatsionnaya mashina: tema i variatsii". Problemy teoreticheskogo i sistemnogo programmirovaniya. Novosibirsk, NGU, pp. 5–45. (In Russian).
10. *Petrushenko, A.N.*, 1990. "O dialogovykh vychisleniyakh v algoritmicheskikh algebrakh". Kibernetika, 1, pp. 13–20. (In Russian).
11. *Petrushenko, A.N.,* 1991. "Ob odnom podkhode k probleme avtomatizatsii optimiziruyushchikh preobrazovaniy algoritmov i programm". Kibernetika i sistemnyy analiz, 5, pp. 127–136. (In Russian).
12. *Petrushenko, A.N.*, 1998. "Ob odnom podkhode k resheniyu problemy obshcheniya cheloveka s vychislitelnoy sistemoy na yestestvennom yazyke". Problemy programmirovaniya: Sb. nauch. trudov, 3, pp. 65–72. (In Russian).
13. *Petrushenko, A.N.*, Khokhlov, V.A., 1999. "Ob ispolzovanii yestestvennogo yazyka dlya predstavleniya abstraktnykh tipov dannykh i polimorfizma". Problemy programmirovaniya, 1, pp. 76–83. (In Russian).
14. *Petrushenko, A.N.*, *Khokhlov, V.A., Tkachev, V.A., Shepetukhin, Ye.S.,* 2000. "Dialogovaya transformatsionnaya mashina: nekotoryye funktsionalnyye vozmozhnosti". Problemy programmirovaniya, 1–2 (Spets. vypusk), pp. 323–334. (In Russian).
15. *Petrushenko A.M., Khokhlov V.A.*, 2004. Kontseptsiya dialohovykh obchyslen ta deyaki problemy avtomatyzatsiyi prohramuvannya. Problemy prohrammyrovanyya, 2–3 (Spets. vypusk), pp. 37–47. (In Ukrainian).
16. *Samofalov, K.G., Korneychuk, V.N., Tarasenko, V.P.,* 1989. Tsifrovyye EVM. K.: Vishcha shk., 423 p. (In Russian).
17. *Samofalov, K.G., Korneychuk, V.N., Tarasenko, V.P., Zhabin, V.N.*, 1990. Tsifrovyye EVM. Praktikum. K.: Vishcha shk., 215 p. (In Russian).
18. *Mayorov, A., Novikov, G.I.*, 1979. Struktura elektronnykh vychislitelnykh mashin. L.: Mashinostroyeniye. 384 p. (In Russian).
19. *Baranov, S.*I., 1979. Sintez mikroprogrammnykh avtomatov. L.: Energiya. 232 p. (In Russian).
20. *Kaluzhnin, L.A.,* 1959. "Ob algoritmizitsii matematicheskikh zadach". Problemy kibernetiki., 2, M.: Fizmatgiz, pp. 51–67. (In Russian).
21. *Lyapunov, A.A.*, 1958. "O logicheskikh skhemakh programm". Problemy kibernetiki., 1, M.: Fizmatgiz, pp. 46–74. (In Russian).
22. *Yanov, Yu.I.*, 1958. "O logicheskikh skhemakh algoritmov". Problemy kibernetiki. M.: Fizmatgiz, 1, pp. 75–127. (In Russian).
23. *Yablonski,y S.V.,* 1959. "Osnovnyye ponyatiya kibernetiki". Problemy kibernetiki. M.: Fizmatgiz, 1, pp. 7–38. (In Russian).
24. *Yershov, A.P.*, 1968. "Operatornyye algoritmy". 3 (ob operatornykh skhemakh Yanova). Problemy kibernetiki. M.: Fizmatgiz, 2, pp. 181–200. (In Russian).
25. *Sistemy kompyuterno*y algebry semeystva ANALITIK. Teoriya. Primeneniye. K., 2010. 762 p. (In Russian).
26. *Razevig, V.D.*, 1992. Primeneniye programm PCAD i PSpise dlya skhemotekhnicheskogo modelirovaniya na PEVM: V 4 vypuskakh. 1: Obshchiye svedeniya. Graficheskiy vvod skhem. M.: Radio i svyaz. 72 p. (In Russian).

Received 16.10.2019

*A.M. Petruchenko***,** PhD (Phys.-Math.), Associate Professor,
Taras Shevchenko National University of Kyiv,
03022, Kyiv, Glushkov ave., 4D, Ukraine,
anatoly@mytaskhelper.com

THE PRINCIPLE OF FIRMWARE CONTROL AND DESIGN AUTOMATION OF OPERATING DEVICES. II

Introduction. Promising areas of research that are developing both in Ukraine and abroad include the so-called transformational synthesis methods. According to these methods, a computing system is obtained by phasing the initial description of the system (setting the task) according to the rules, which is knowledge about the problem being solved. In this area of research, two interrelated directions can be distinguished–the theoretical and applied. The theoretical direction requires, in particular, the presentation of various computational models that describe particular parts of computing systems, and the study of basic transformations in these models. The applied direction is associated with the creation of a transformation machine, the commands for which are basic transformations, and the data are expressions in the language over which these transformations are given. The Ukrainian Algebra-Cybernetic School researches the computational model, which is based on the concept of a discrete information converter. Representation of the computing process in this form allowed V.M. Glushkov and his students to create a new theoretical direction in the applied theory of algorithms – the algebra of algorithms. A characteristic feature of this direction is the commona-lity of mathematical models and methods for designing programmes and equipment. The apparatus of algebra of algorithms is the basis of the dialogue transformation machine – the main object of this article.

The purpose is to demonstrate the inextricable link between the fundamental concepts of the general theory of computer systems design and practical methods of designing software and hardware of computer technology, as well as new technological capabilities that arise when using the apparatus of algebra of algorithms in the process of designing programmes and equipment using an interactive transformational machine.

Methods. When implementing the tools (conversational transformation machine) and the synthesis algorithm of operating devices, we used the algebraic-grammatical method of representing knowledge, the method of constructing operating devices based

on the principle of microprogram control, the methods of abstract and structural theory of automata, the methods of algebra of algorithms, etc.

Results. Synthesis methods for operating devices developed for the language of graph diagrams of algorithms and the language of logical diagrams of algorithms are extended to the language CAA \ D – the input language of the dialogue transformation machine. Based on the dialogue transformational machine, a toolkit has been developed that embodies the V.M. Glushkov mathematical model and allows the complex automation of the operating devices: from setting the task to obtaining a sketch of the printed circuit board.

Conclusions. The integral algebraic-grammatical apparatus underlying the dialogue transformational machine combines algebraic, logical, and grammatical formalisms and is focused on the multi-level structural design of classes of algorithms and associated programmes (serial and parallel) and hardware. It is characterized by the analytical style of the specifications of programmes and equipment, focused on their optimizing transformations in order to achieve the necessary quality indicators. At the same time, using the CAA \ Dv language as the input language of the dialog transformational machine allows you to increase the "intelligence" of the computer to a level that provides direct communication with it, in particular, inexperienced in programming users who are specialists in specific areas.

Keywords*: microprogram control principle, discrete information converter, operating device, algorithm algebra, algebra-grammatical method of knowledge representation, dialogue transformation machine.*

*А.Н. Петрушенко***,** кандидат физ.-мат. наук, доцент,
Киевский национальный университет имени Тараса Шевченко,
03022, г. Киев, просп. Академика Глушкова, 4Д,
anatoly@mytaskhelper.com

ПРИНЦИП МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ И АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ ОПЕРАЦИОННЫХ УСТРОЙСТВ. II

Введение. Одной из определяющих тенденций развития науки в целом является ее математизация; не являются исключением и науки, которые исследуют сами компьютеры, принципы их строения и функционирования. Создание компьютеров и универсальных алгоритмических языков приводит к изменению акцентов при оценке значения математизации — ее уровень является теперь не только одним из главных показателей уровня научности, системности представлений о тех или иных предметных областях, но и определяет возможности автоматизации процессов, протекающих в этих областях.

Цель статьи. Компьютеры изменяют не только отношение к математизации, но и ее характер, и ее сущность. В частности, возникает новый метод научных исследований – имитационное моделирование, который соединил в себе черты классических дедуктивных и экспериментальных методов, присущих математике и физике соответственно. Но такой метод проведения эксперимента еще не есть *настоящая* математизация — для ее достижения необходимы алгебры алгоритмов. Именно в существовании такой алгебры и состоит разница между настоящей математизацией и простой формализацией знаний. Разработка такого аппарата связана с работами В.М.Глушкова, который ввел понятие дискретного преобразователя информации и системы алгоритмических (их сначала называли микропрограммные) алгебр. По Глушкову, каждая модель математизации должна состоять: из области объекта, которая должна быть изучена формальными математическими методами; из формального языка, подходящего для более или менее адекватного описания этого объекта; из формальной модели вывода, включающей как общие средства логического вывода, так и средства вывода, специально разработанные для этого языка (алгебру данного языка). Формальный язык вместе с формальной моделью вывода образуют формальную теорию. Чтобы эта теория стала настоящим дедуктивным аппаратом, она должна быть воплощена в инструменте вывода, который составляет четвертую часть модели математизации. В настоящее время известны два типа таких инструментов: человеческий мозг и компьютер. Формальная теория, усвоенная мозгом или компьютером, названа системой познания. Легко видеть, реализация идей В.М.Глушкова в полном объеме сопряжена с корректировкой целей математизации – теперь одной из главных ее целей является повышение *интеллекта* компьютера до уровня, который обеспечивает непосредственное общение с ним несведущих в программировании пользователей – специалистов в конкретных предметных областях. Цель данной статьи — демонстрация одного из возможных подходов к реализации инструментария, воплощающего модель математизации В.М. Глушкова, и его (инструментария) возможностей на примере синтеза операционных устройств.

Методы. При реализации инструментария и алгоритма синтеза операционных устройств использовался алгебро-грамматический метод представления знаний, метод построения операционных устройств на базе принципа микропрограммного управления, методы абстрактной и структурной теории автоматов, методы алгебры алгоритмов и т.д.

Результаты. Разработан инструментарий, воплощающий модель математизации В.М. Глушкова и позволяющий осуществить комплексную автоматизацию проектирования операционных устройств.

Выводы. Необходима работа по приведению инструментария к *товарному* виду.

Ключевые слова: *принцип микропрограммного управления, дискретный преобразователь информации, операционное устройство, алгебра алгоритмов, алгебро-грамматический метод представления знаний, диалоговая трансформационная машина*.