В. Г. Акуловський, кандидат технічних наук, доцент, доцент кафедри інформаційних систем
та технологій Академії митної служби України
В. В. Костенко, старший викладач кафедри
інформаційних систем та технологій Академії
митної служби України

ДЕЯКІ АСПЕКТИ ФОРМАЛІЗАЦІЇ ТА СПЕЦИФІКАЦІЇ ІНФОРМАЦІЙНИХ ЗВ'ЯЗКІВ В АЛГОРИТМАХ

У рамках алгебри алгоритмів формалізовано дані й інформаційні зв'язки між Д-операторами, що утворюють регулярні схеми. Специфіковано інформаційні зв'язки між підсистемами, що утворюють деякі шари алгоритмів. Визначено та знайдено основні властивості даних і утворених ними інформаційних зв'язків.

В рамках алгебры алгоритмов формализованы данные и информационные связи между Д-операторами, образующими регулярные схемы. Специфицированы информационные связи между подсистемами, образующими некоторые слои алгоритмов. Определены и найдены основные свойства данных и образованных ими информационных связей.

Within the framework of algebra of algorithms the given and information connections between the Д-operators forming regular circuits are formalized. Information connections between the subsystems forming some layers of algorithms specified. The basic properties of the data and, formed by them, information connections are determined and found.

Ключові слова. Алгоритм, оператор, множина станів автомата, регулярна схема, композиція, декомпозиція.

Вступ. Роль даних у програмуванні важко переоцінити, що давно усвідомлено програмістським співтовариством (див., наприклад [1]). Це знайшло своє вираження у формалізації даних, як для компіляції програм, так і для розробки алгоритмів. При цьому можливості формалізації даних в алгебрі алгоритмів, що широко використовується при розробці програмних систем, реалізовані далеко не повністю. Тим часом це дало б змогу розраховувати на формалізацію декомпозиції операторів, перетворення алгоритмів, контроль їхньої коректності тощо на досить ранніх стадіях розробки алгоритмів.

Як перший крок на шляху виконання цього завдання обрано задачу специфікації інформаційних зв'язків в алгоритмах, для чого необхідно формалізувати безпосередньо дані, що утворюють такі зв'язки.

Передумовою й основою для виконання даної роботи послужила ідея, висловлена у праці [2], про те, що у відомій моделі ЕОМ В. М. Глушкова [3, 4] як множина станів операційного автомата можуть розглядатися дані, що обробляються алгоритмом.

Постановка завдання. Метою цього дослідження є формалізація даних, Д-операторів, інформаційних зв'язків в алгоритмах і визначення їхніх властивостей у рамках алгебри алгоритмів. У процесі розробки алгоритмів на основі отриманих формалізмів буде здійснено специфікацію даних і утворених ними інформаційних зв'язків. При цьому на процес розробки алгоритму накладається обмеження: всі специфіковані в алгоритмі дані мають бути використані, а всі використовувані дані – специфіковані. Деякі попередні результати, що використані в нашому дослідженні, були отримані в [5, 6].

Результати дослідження.

Формалізація даних і Д-операторів. Виходячи з передумови, сформульованої у вступі, вважатимемо, що інформаційна множина (множина станів операційного автомата) являє собою множину даних Δ.

Щоб визначити поняття даних, розглядатимемо їх як сукупності комірок пам'яті D_i, котрі являють собою множину $\Delta = \bigcup_{i=1}^{k} D_i$ і можуть набувати деяких множин значення D_i^I зі скінченної множини значень $\Psi = \bigcup_{i=1}^{k} D_i^I$. Виходячи із цього, визначимо дані в такий спосіб.

Визначення 1. Даними будемо називати таку множину комірок пам'яті, для яких виконується співвідношення $D_j \sim D_i^I$ ($D_j \subseteq \Delta$, $D_i^I \subseteq \Psi$), тобто таку множину комірок пам'яті D_j, якій в однозначну відповідність поставлено множину значень D_i^I.

Тепер, виходячи із визначення 1, уведемо поняття Д-оператора, що змінює стан операційного автомата.

Визначення 2. Д-оператор $(D)A(D')$ перетворює множину вхідних даних $D \subseteq \Delta$, якій в однозначну відповідність було поставлено множину значень $D^I \subseteq D^I$ ($D \sim D^I$), у множину вихідних даних $D' \subseteq \Delta$, якій у взаємно однозначну відповідність ставиться множина значень $D'^I \subseteq D'^I$ ($D' \sim D'^I$). Множини вхідних D і вихідних D' даних можуть бути порожніми. У цьому випадку визначимо єдину операцію із сигнатури алгебри алгоритмів – композицію, що буде використовуватися надалі.

Визначення 3. Композиція Д-операторів $(D_B)B(D'_B)*(D_C)C(D'_C)$ означає послідовне виконання спочатку Д-оператора $(D_B)B(D'_B)$, , а потім Д-оператора $(D_C)C(D'_C)$. Тобто

$$(D_B)B(D'_B)*(D_C)C(D'_C) = ((D_B)B(D'_B) \cup D_C)C(D'_C \cup (D'_B \setminus D_C)).$$

Оскільки Д-оператори, зв'язані операцією композиції, можуть взаємодіяти один з одним, визначимо

поняття зв'язку між Д-операторами.

Визначення 4. Д-оператори $(D_B)B(D'_B)*(D_C)C(D'_C)$ зв'язані, якщо для них виконується співвідношення: $D'_B \cap D_C \neq \varnothing$.

Таким чином, в організації зв'язку бере участь деяка підмножина вихідних даних Д-оператора $(D_B)B(D'_B)$, що надходить на вхід Д-оператора $(D_C)C(D'_C)$. При цьому будемо говорити, що дані передаються з виходу попереднього Д-оператора на вхід наступного.

Розробка алгоритмів. При розгляді процесу розробки алгоритму будемо виходити з того, що на етапах, які передують етапові розробки алгоритму, розроблено специфікації цього завдання. У результаті маємо у своєму розпорядженні специфікації вихідних ${}^{вих}D$ і результуючих ${}^{рез}D$ даних з переліком функцій, що перетворюють перші в другі.

Використовувані в алгоритмах дані визначимо в такий спосіб.

Визначення 5. Вихідні – це певні ініціалізовані, тобто початкові значення, що мають деякі дані, або дані, що вводяться із зовнішніх пристроїв (клавіатури, магнітних дисків, аналого-цифрових перетворювачів тощо).

Визначення 6. Результуючі – це дані, які є результатом роботи програмної системи, що відображаються (наприклад, на екрані дисплея) і/або зберігаються на зовнішніх пристроях.

Визначення 7. Локальними називатимемо дані, доступні тільки всередині результуючих Д-операторів, що входять у РС (регулярні схеми), поза якими вони не доступні й не специфіковані.

Визначення 8. Проміжні – це дані, відсутні в специфікації завдання, але необхідні для перетворення вихідних даних у результуючі, тому що вони є носіями проміжних результатів, одержуваних на шляху від вихідних до результуючих даних.

Після визначення даних, які обробляються, доречно визначити поняття алгоритму.

Визначення 9. Алгоритмом назвемо перший і єдиний Д-оператор на нульовому рівні декомпозиції, записаний у вигляді $(D_1^0, {}^{вих}D_1^0)A_1^0({}^{рез}D_1^0)$ (верхній індекс – номер рівня декомпозиції, а нижній – порядковий номер Д-оператора в РС), що перетворює за допомогою проміжних даних D_1^0 вихідні дані ${}^{вих}D_1^0$ в результуючі ${}^{рез}D_1^0$.

Оскільки основним підходом до розробки алгоритмів є декомпозиція утворюючих його Д-операторів, введемо, ґрунтуючись на визначенні 3, аксіому.

Аксіома. Будь-який Д-оператор, за винятком елементарних, може бути зображений у вигляді композиції двох інших Д-операторів

$$(D_A)A(D'_A) = (D_B)B(D'_B)*(D_C)C(D'_C),$$

які забезпечують функціональність, адекватну функціональності Д-оператора $(D_A)A(D'_A)$, а дані, специфіковані на входах і виходах Д-операторів у загальному випадку мають такі властивості:

$$D_A \subseteq D_B \cup D_C, \quad D'_A \subseteq D'_B \cup D'_C, \quad D_B \subseteq D_A, \quad D'_C \subseteq D'_A.$$

Д-оператор $(D_A)A(D'_A)$ будемо називати вихідним, а $(D_B)B(D'_B)$ й $(D_C)C(D'_C)$ – похідними.

У результаті декомпозиції Д-операторів одержимо регулярну схему, яку в контексті даної статті визначимо в такий спосіб.

Визначення 10. Регулярною схемою (РС) будемо називати вираз вигляду $(D)A(D') = (D_1)A_1(D'_1)*(D_2)A_2(D'_2)*\ldots*(D_i)A_i(D'_i)*\ldots*(D_n)A_n(D'_n)$ (де нижній індекс – порядковий номер Д-оператора), в якому Д-оператори $A_1, A_2, \ldots, A_i, \ldots, A_n$ виконуються послідовно, і їхня сумарна функціональність адекватна функціональності Д-оператора A. Д-оператор $(D)A(D')$ будемо називати вихідним, а будь-який Д-оператор $(D_i)A_i(D'_i)$ – результуючим. Специфіковані для вихідних і результуючих Д-операторів дані задовольняють такі співвідношення $D = \bigcup_{j=1}^{n} D_j$ і $D' = \bigcup_{j=1}^{n} D'_j$.

Використовуючи визначення 9 і 10, виконаємо перший крок декомпозиції алгоритму й запишемо РС у вигляді:

$$(D_1^0, {}^{вих}D_1^0)A_1^0({}^{рез}D_1^0) = (D_1^1, {}^{вих}D_1^1)A_1^1({}^{рез}D_{1,1}^1 D^{d_1})*(D_2^1, {}^{вих}D_2^1)A_2^1({}^{рез}D_{2,1}^1 D^{d_1})*\ldots$$
$$\ldots*(D_i^1, {}^{вих}D_i^1)A_i^1({}^{рез}D_{i,1}^1 D^{d_1})*\ldots*(D_n^1, {}^{вих}D_n^1)A_n^1({}^{рез}D_n^1). \quad (1)$$

Для вихідних, результуючих і проміжних даних (відповідно до визначення 10) мають місце такі властивості:

$${}^{вих}D_1^0 = \bigcup_{i=1}^{n} {}^{вих}D_i^1, \quad (2)$$

$$D_1^0 = \bigcup_{i=1}^{n} D_i^1, \tag{3}$$

$$D_1^0 = \bigcup_{i=1}^{n} D_i^1 \cup \bigcup_{i=1}^{n-1} D^{d}, \tag{4}$$

тобто всі вихідні, результуючі й проміжні дані, відповідно до обмеження із вступу, розподіляються між Д-операторами, що утворюють РС.

Особливості в розташуванні індексів для всіх проміжних даних D^{d} (приклад для *j*-го результуючого Д-оператора) будуть використані й зрозумілі з подальшого викладу. Зауважимо, що Д-оператори, які утворюють РС, надалі називатимемо підсистемами.

Розгляд другого кроку почнемо з декомпозиції *i*-ї підсистеми з (1), що запишемо у вигляді РС, де на вході вихідної підсистеми специфікуємо локальні дані:

$$(D_i^1, D_i^1, D_i^1) A_i^1(D_i^1, D^{d}) = (D_1^2, D_1^2) A_1^2(D_{1,1}^1, D^{d}) \times (D_2^2, D_2^2) A_2^2(D_{2,2}^1, D^{d}) \times \ldots$$
$$\ldots \times (D_j^2, D_j^2) A_j^2(D_{j,j}^1, D^{d}) \times (D_k^2, D_k^2) A_k^2(D_{k,k}^1, D^{d}). \tag{5}$$

Зазначимо, що локальні (див. визначення 7) на першому рівні декомпозиції дані D_i^1 для другого рівня вже відіграють роль глобальних, тобто для кожного D_j^2 (j набуває значень від 1 до k) виконується $D_j^2 \subseteq D_i^1 \cup D_i^1$ і множина даних D_i^1 у ролі глобальної розподіляється між усіма підсистемами, що входять до РС.

Отримана після специфікації локальних даних РС зберігає властивості 2 і 3, які для даного рівня записуються у вигляді:

$$D_i^1 = \bigcup_{j=1}^{k} D_j^2, \tag{6}$$

$$D_i^1 = \bigcup_{j=1}^{k} D_j^2. \tag{7}$$

Крім цих властивостей, РС (5) у зв'язку з використанням локальних даних має такі властивості:

$$D_i^1 \cup D_i^1 = \bigcup_{j=1}^{k} D_j^2, \tag{8}$$

$$D^{d} = \bigcup_{j=1}^{k} D^{d}, \tag{9}$$

з яких випливає властивість, аналогічна (4)

$$D_i^1 \cup D_i^1 \cup D^{d} = \bigcup_{j=1}^{k} (D_j^2 \cup D^{d}), \tag{10}$$

і, таким чином, обмеження виконується й у цьому випадку.

Застосувавши такий підхід до всіх підсистем першого рівня, одержимо сукупність РС, яку називатимемо шаром алгоритму. При розгляді сукупності результуючих підсистем, що утворюють перший шар алгоритму, будемо використовувати наскрізну нумерацію підсистем усередині шару, і в цьому випадку нижній індекс визначатиме порядковий номер підсистеми в розглянутому шарі.

З огляду на викладене вище, перший шар алгоритму запишемо у вигляді:

$$(D_1^1, D_1^1, D_1^1) A_1^1(D_1^1, D^{d}) = (D_1^2, D_1^2) A_1^2(D_{1,1}^1, D^{d}) *$$
$$*(D_2^2, D_2^2) A_2^2(D_{2,2}^1, D^{d}) * \ldots$$
$$\ldots *(D_j^2, D_j^2) A_j^2(D_{j,j}^1, D^{d}) * \ldots$$
$$\ldots *(D_{k_1}^2, D_{k_1}^2) A_{k_1}^2(D_{k_1,k_1}^1, D^{d}).$$

$$(D_2^1, D_2^1, D_2^1) A_2^1(D_2^1, D_2^1) = (D_{k_1+1}^2, D_{k_1+1}^2) A_{k_1+1}^2(D_{k_1+1,k_1+1}^1, D^{d}) *$$
$$*(D_{k_1+2}^2, D_{k_1+2}^2) A_{k_1+2}^2(D_{k_1+2,k_1+2}^1, D^{d}) * \ldots$$
$$\ldots *(D_{k_1+j}^2, D_{k_1+j}^2) A_{k_1+j}^2(D_{k_1+j,k_1+j}^1, D^{d}) * \ldots$$
$$\ldots *(D_{k_2}^2, D_{k_2}^2) A_{k_2}^2(D_{k_2,k_2}^1, D^{d}). \tag{11}$$

……

$$(D_1^1, {}^{вых}D_1^1, {}^{вн}D_1^1)A_1^1({}^{вых}D_1^1, {}_1D^{\prime 1}) = (D_{k_{0}+1}^2, {}^{вых}D_{k_{0}+1}^2)A_{k_{0}+1}^2({}^{вых}D_{k_{0}+1}^2, {}_{k_{0}+1}D^{\prime 2}) *$$
$$*(D_{k_{0}+2}^2, {}^{вых}D_{k_{0}+2}^2)A_{k_{0}+2}^2({}^{вых}D_{k_{0}+2}^2, {}_{k_{0}+2}D^{\prime 2}) * \ldots$$
$$\ldots *(D_{k_{0}+f}^2, {}^{вых}D_{k_{0}+f}^2)A_{k_{0}+f}^2({}^{вых}D_{k_{0}+f}^2, {}_{k_{0}+f}D^{\prime 2}) * \ldots$$
$$\ldots *(D_{k_1}^2, {}^{вых}D_{k_1}^2)A_{k_1}^2({}^{вых}D_{k_1}^2, {}_{k_1}D^{\prime 2}).$$

……

$$(D_n^1, {}^{вых}D_n^1, {}^{вн}D_n^1)A_n^1({}^{вых}D_{n,n}^1D^{\prime 1}) = (D_{k_{n-1}+1}^2, {}^{вых}D_{k_{n-1}+1}^2)A_{k_{n-1}+1}^2({}^{вых}D_{k_{n-1}+1}^2, {}_{k_{n-1}+1}D^{\prime 2}) *$$
$$*(D_{k_{n-1}+2}^2, {}^{вых}D_{k_{n-1}+2}^2)A_{k_{n-1}+2}^2({}^{вых}D_{k_{n-1}+2}^2, {}_{k_{n-1}+2}D^{\prime 2}) * \ldots$$
$$\ldots *(D_{k_{n-1}+f}^2, {}^{вых}D_{k_{n-1}+f}^2)A_{k_{n-1}+f}^2({}^{вых}D_{k_{n-1}+f}^2, {}_{k_{n-1}+f}D^{\prime 2}) * \ldots$$
$$\ldots *(D_{k_n}^2, {}^{вых}D_{k_n}^2)A_{k_n}^2({}^{вых}D_{k_n}^2),$$

,

де n – кількість вихідних підсистем і, відповідно, кількість РС;

k_n – загальна кількість підсистем, що утворюють шар.

Узагальнивши властивості 6–10 на випадок шару алгоритму, маємо:

$$\bigcup_{i=1}^{n} {}^{вых}D_i^1 = \bigcup_{j=1}^{k_n} {}^{вых}D_j^2,$$

$$\bigcup_{i=1}^{n} {}^{вых}D_i^1 = \bigcup_{j=1}^{k_n} {}^{вых}D_j^2,$$

$$\bigcup_{i=1}^{n}(D_i^1 \cup {}^{вн}D_i^1) = \bigcup_{j=1}^{k_n} D_j^2,$$

$$\bigcup_{i=1}^{n} {}_iD^{\prime 1} = \bigcup_{j=1}^{k_n} {}_jD^{\prime 2}$$

$$\bigcup_{i=1}^{n}(D_i^1 \cup {}^{вн}D_i^1 \cup {}_iD^{\prime 1}) = \bigcup_{j=1}^{k}(D_j^2 \cup {}_jD^{\prime 2})$$

і, виходячи з отриманих властивостей, зробимо висновок про те, що обмеження, наведене у вступі, виконується й для шару алгоритму.

Процес декомпозиції триває до того моменту, коли будуть виділені всі підсистеми, що утворюють проектовану програмну систему. У результаті такої побудови одержимо певну кількість шарів алгоритму, де кожний нижче розташований шар матиме все менш абстрактний (більш деталізований) характер, і, як ми вже зазначали вище, буде досягнута відповідність обмеженню, згаданому у вступі. Тепер перейдемо до специфікації інформаційних зв'язків.

Специфікація інформаційних зв'язків в алгоритмах. З огляду на визначення 4, почнемо з визначення поняття зв'язків у РС.

Визначення 11. Підсистеми $(D_i)A_i(D_i')$ й $(D_j)A_j(D_j')$ (де $i<j$), що входять у РС (1), зв'язані, якщо для них виконується співвідношення $D_i' \cap D_j \neq \varnothing$, тобто множина даних ${}_i\vec{D}_j = D_i' \cap D_j$ зв'язує підсистеми A_i й A_j (про це свідчать використовувані індекси), і ці дані називатимемо єднальними.

З визначення 11 випливає, що дані можуть передаватися від будь-якої підсистеми до однієї або декількох наступних за даною підсистем, тобто передаються ліворуч і праворуч, і саме єднальні дані утворюють інформаційні зв'язки в РС. При цьому в загальному випадку деяка підсистема може бути зв'язана з усіма наступними й попередніми підсистемами. Тобто в будь-якій підсистемі множина вихідних даних містить дані, що поєднують її з усіма наступними підсистемами, а множина вхідних даних містить у собі дані, що поєднують її з усіма попередніми підсистемами.

Такий загальний випадок і будемо розглядати, попередньо доповнивши систему позначень: єднальні дані на вході підсистеми позначимо ${}_*\tilde{D}_j^1$, а на виході – ${}_i\tilde{D}_*^1$, причому лівий нижній індекс – порядковий номер підсистеми в РС називатимемо адресою джерела, а правий нижній індекс – адресою приймача даних. Зауважимо, що для реалізації такої системи позначень в (1) було використано відповідне розташування індексів.

Перепишемо РС (1) з урахуванням уведених позначень:

$$(D_1^0, {}^{вых}D_1^0)A_1^0({}^{вых}D_1^0) = (D_1^1, {}^{вых}D_1^1)A_1^1({}^{вых}D_1^1, {}_1\tilde{D}_1^1, {}_1\tilde{D}_2^1, \ldots, {}_1\tilde{D}_f^1, \ldots, {}_1\tilde{D}_n^1) *$$
$$*(D_2^1, {}^{вых}D_2^1, {}_1\tilde{D}_2^1,)A_2^1({}^{вых}D_2^1, {}_2\tilde{D}_2^1, {}_2\tilde{D}_3^1, \ldots, {}_2\tilde{D}_f^1, \ldots, {}_2\tilde{D}_n^1) * \ldots$$
$$\ldots *(D_f^1, {}^{вых}D_{f,1}^1\tilde{D}_f^1, {}_2\tilde{D}_f^1, \ldots, {}_f\tilde{D}_f^1,$$
$$\ldots, {}_{f-1}\tilde{D}_f^1)A_f^1({}^{вых}D_f^1, {}_f\tilde{D}_{f+1}^1, {}_f\tilde{D}_{f+2}^1, \ldots, {}_f\tilde{D}_p^1, \ldots, {}_f\tilde{D}_n^1) * \ldots$$
$$\ldots *(D_n^1, {}^{вых}D_{n,1}^1\tilde{D}_n^1, {}_2\tilde{D}_n^1, \ldots, {}_f\tilde{D}_n^1, \ldots, {}_{n-1}\tilde{D}_n^1)A_n^1({}^{вых}D_n^1).$$

У побудованій РС не тільки специфіковані дані, але й усі "джерела" та "приймачі" даних поставлено в однозначну відповідність, тобто кожному ${}_i\tilde{D}_p^1$ відповідає ${}_i\tilde{D}_p^1$, і таким чином виконується

$$\bigcup_{i=1}^{n-1}\bigcup_{k=2}^{n} {}_i\vec{D}^1_k = \bigcup_{i=1}^{n-1}\bigcup_{k=2}^{n} {}_i\tilde{D}^1_k.$$

$$D^0_1 = \bigcup_{p=1}^{n} D^1_p \cup \bigcup_{i=1}^{n-1}\bigcup_{k=2}^{n} {}_i\vec{D}^1_k.$$

У зв’язку з наявністю цих властивостей обмеження задовольняються й у цьому випадку.

Оскільки в РС, як правило, не всі підсистеми зв’язані одна з одною, кожна з множин ${}_l\vec{D}^1_p$ може бути порожньою, і, відповідно, порожньою буде множина ${}_l\tilde{D}^1_p$.

Тепер перейдемо до специфікації інформаційних зв’язків у побудованому шарі алгоритму, для чого перепишемо вираз (11) у вигляді, аналогічному (5).

$$(D^1_1, {}^{out}D^1_1, {}^{in}D^1_1)A^1_1({}^{par}D^1_1, {}_1D^{d1}) =$$
$$= (D^2_1, {}^{out}D^2_1)A^2_1({}^{par}D^2_1, {}_1\vec{D}^2_2, {}_1\vec{D}^2_3, \dots, {}_1\vec{D}^2_j, \dots, {}_1\vec{D}^2_{k_1}, \dots, {}_1\vec{D}^2_{k_2}, \dots, {}_1\vec{D}^2_{k_n-1}, {}_1\vec{D}^2_{k_n})*$$
$$*(D^2_2, {}_1\tilde{D}^2_2, {}^{out}D^2_2)A^2_2({}^{par}D^2_2, {}_2\vec{D}^2_3, {}_2\vec{D}^2_4, \dots, {}_2\vec{D}^2_j, \dots, {}_2\vec{D}^2_{k_1}, \dots, {}_2\vec{D}^2_{k_2}, \dots, {}_2\vec{D}^2_{k_n-1}, {}_2\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_j, {}_1\tilde{D}^2_j, \dots, {}_2\tilde{D}^2_j, \dots, {}_{j-1}\tilde{D}^2_j, {}^{out}D^2_j)A^2_j({}^{par}D^2_j, {}_j\vec{D}^2_{j+1}, \dots, {}_j\vec{D}^2_{k_1}, \dots, {}_j\vec{D}^2_{k_2}, \dots, {}_j\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_1}, {}_1\tilde{D}^2_{k_1}, \dots, {}_2\tilde{D}^2_{k_1}, \dots, {}_{k_1-1}\tilde{D}^2_{k_1}, {}^{out}D^2_{k_1})A^2_{k_1}({}^{par}D^2_{k_1}, {}_{k_1}\vec{D}^2_{k_1+1}, \dots, {}_{k_1}\vec{D}^2_{k_2}, \dots, {}_{k_1}\vec{D}^2_{k_n}).$$

……

$$(D^1_2, {}^{out}D^1_2, {}^{in}D^1_2)A^1_2({}^{par}D^1_2, {}_2D^{d1}) =$$
$$= (D^2_{k_1+1}, {}_1\tilde{D}^2_{k_1+1}, {}_2\tilde{D}^2_{k_1+1}, \dots, {}_{k_1}\tilde{D}^2_{k_1+1}, {}^{out}D^2_{k_1+1})A^2_{k_1+1}({}^{par}D^2_{k_1+1}, {}_{k_1+1}\vec{D}^2_{k_1+2}, \dots, {}_{k_1+1}\vec{D}^2_{k_2}, \dots, {}_{k_1+1}\vec{D}^2_{k_n})*$$
$$*(D^2_{k_1+2}, {}_1\tilde{D}^2_{k_1+2}, {}_2\tilde{D}^2_{k_1+2}, \dots, {}_{k_1+1}\tilde{D}^2_{k_1+2}, {}^{out}D^2_{k_1+2})A^2_{k_1+2}({}^{par}D^2_{k_1+2}, {}_{k_1+2}\vec{D}^2_{k_1+3}, \dots, {}_{k_1+2}\vec{D}^2_{k_2}, \dots, {}_{k_1+2}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_1+j}, {}_1\tilde{D}^2_{k_1+j}, {}_2\tilde{D}^2_{k_1+j}, \dots, {}_{k_1+j-1}\tilde{D}^2_{k_1+j}, {}^{out}D^2_{k_1+j})A^2_{k_1+j}({}^{par}D^2_{k_1+j}, {}_{k_1+j}\vec{D}^2_{k_1+j+1}, \dots, {}_{k_1+j}\vec{D}^2_{k_2}, \dots, {}_{k_1+j}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_2}, {}_1\tilde{D}^2_{k_2}, {}_2\tilde{D}^2_{k_2}, \dots, {}_{k_2-1}\tilde{D}^2_{k_2}, {}^{out}D^2_{k_2})A^2_{k_2}({}^{par}D^2_{k_2}, {}_{k_2}\vec{D}^2_{k_2+1}, {}_{k_2}\vec{D}^2_{k_2+2}, \dots, {}_{k_2}\vec{D}^2_{k_2+j}, \dots, {}_{k_2}\vec{D}^2_{k_3}, \dots, {}_{k_2}\vec{D}^2_{k_n}).$$

……

$$(D^1_l, {}^{out}D^1_l, {}^{in}D^1_l)A^1_l({}^{par}D^1_l, {}_lD^{d1}) =$$
$$= (D^2_{k_{l-1}+1}, {}_1\tilde{D}^2_{k_{l-1}+1}, {}_2\tilde{D}^2_{k_{l-1}+1}, \dots, {}_{k_{l-1}}\tilde{D}^2_{k_{l-1}+1}, {}^{out}D^2_{k_{l-1}+1})A^2_{k_{l-1}+1}({}^{par}D^2_{k_{l-1}+1}, {}_{k_{l-1}+1}\vec{D}^2_{k_{l-1}+2}, \dots, {}_{k_{l-1}+1}\vec{D}^2_{k_l}, \dots, {}_{k_{l-1}+1}\vec{D}^2_{k_n})*$$
$$*(D^2_{k_{l-1}+2}, {}_1\tilde{D}^2_{k_{l-1}+2}, {}_2\tilde{D}^2_{k_{l-1}+2}, \dots, {}_{k_{l-1}+1}\tilde{D}^2_{k_{l-1}+2}, {}^{out}D^2_{k_{l-1}+2})A^2_{k_{l-1}+2}({}^{par}D^2_{k_{l-1}+2}, {}_{k_{l-1}+2}\vec{D}^2_{k_{l-1}+3}, \dots, {}_{k_{l-1}+2}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_{l-1}+j}, {}_1\tilde{D}^2_{k_{l-1}+j}, {}_2\tilde{D}^2_{k_{l-1}+j}, \dots, {}_{k_{l-1}+j-1}\tilde{D}^2_{k_{l-1}+j}, {}^{out}D^2_{k_{l-1}+j})A^2_{k_{l-1}+j}({}^{par}D^2_{k_{l-1}+j}, {}_{k_{l-1}+j}\vec{D}^2_{k_{l-1}+j+1}, \dots, {}_{k_{l-1}+j}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^l_{k_l}, {}_1\tilde{D}^2_{k_l}, {}_2\tilde{D}^2_{k_l}, \dots, {}_{k_l-1}\tilde{D}^2_{k_l}, {}^{out}D^l_{k_l})A^l_{k_l}({}^{par}D^l_{k_l}, {}_{k_l}\vec{D}^2_{k_l+1}, \dots, {}_{k_l}\vec{D}^2_{k_l+2}, \dots, {}_{k_l}\vec{D}^2_{k_l+j}, \dots, {}_{k_l}\vec{D}^2_{k_{n-1}}, \dots, {}_{k_l}\vec{D}^2_{k_n}).$$

……

$$(D^{l-1}_n, {}^{out}D^{l-1}_n, {}^{in}D^{l-1}_n)A^{l-1}_n({}^{par}D^{l-1}_n, {}_nD^{d-1}) =$$
$$= (D^2_{k_{n-1}+1}, {}_1\tilde{D}^2_{k_{n-1}+1}, {}_2\tilde{D}^2_{k_{n-1}+1}, \dots, {}_{k_{n-1}}\tilde{D}^2_{k_{n-1}+1}, {}^{out}D^2_{k_{n-1}+1})A^2_{k_{n-1}+1}({}^{par}D^2_{k_{n-1}+1}, {}_{k_{n-1}+1}\vec{D}^2_{k_{n-1}+2}, \dots, {}_{k_{n-1}+1}\vec{D}^2_{k_n})*$$
$$*(D^2_{k_{n-1}+2}, {}_1\tilde{D}^2_{k_{n-1}+2}, {}_2\tilde{D}^2_{k_{n-1}+2}, \dots, {}_{k_{n-1}+1}\tilde{D}^2_{k_{n-1}+2}, {}^{out}D^2_{k_{n-1}+2})A^2_{k_{n-1}+2}({}^{par}D^2_{k_{n-1}+2}, {}_{k_{n-1}+2}\vec{D}^2_{k_{n-1}+3}, \dots, {}_{k_{n-1}+2}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_{n-1}+j}, {}_1\tilde{D}^2_{k_{n-1}+j}, \dots, {}_{k_{n-1}+j-1}\tilde{D}^2_{k_{n-1}+j}, {}^{out}D^2_{k_{n-1}+j})A^2_{k_{n-1}+j}({}^{par}D^2_{k_{n-1}+j}, {}_{k_{n-1}+j}\vec{D}^2_{k_{n-1}+j+1}, \dots, {}_{k_{n-1}+j}\vec{D}^2_{k_n})*\dots$$
$$\dots*(D^2_{k_n}, {}_1\tilde{D}^2_{k_n}, {}_2\tilde{D}^2_{k_n}, \dots, {}_{k_n-1}\tilde{D}^2_{k_n}, {}^{out}D^2_{k_n})A^2_{k_n}({}^{par}D^2_{k_n}).$$

Узагальнимо властивості єднальних даних на випадок шару зі специфікованими інформаційними зв’язками:

$$\bigcup_{l=1}^{k_n-1}\bigcup_{p=2}^{k_n} {}_l\vec{D}^l_p = \bigcup_{l=1}^{k_n-1}\bigcup_{p=2}^{k_n} {}_l\tilde{D}^l_p,$$

$$\bigcup_{l=1}^{n}(D^1_l \cup {}^{in}D^1_l) = \bigcup_{j=1}^{k_n} D^2_j \cup \bigcup_{l=1}^{k_n-1}\bigcup_{p=2}^{k_n} {}_l\tilde{D}^l_p,$$

$$\bigcup_{l=1}^{n} {}_lD^{d1} = \bigcup_{l=1}^{k_n-1}\bigcup_{p=2}^{k_n} {}_l\vec{D}^l_p,$$

$$\bigcup_{l=1}^{n}(D^1_l \cup {}^{in}D^1_l \cup {}_lD^{d1}) = \bigcup_{j=1}^{n}(D^2_j \cup \bigcup_{l=1}^{k_n-1}\bigcup_{p=2}^{k_n} {}_l\vec{D}^l_p).$$

Для локальних даних у першому шарі алгоритму в рамках усіх РС виконується співвідношення:

$$^{в}D_i^1 \subseteq \bigcup_{l=k_{i-1}+1}^{k_i-1} \bigcup_{p=k_{i-1}+1}^{k_i} {}_l\vec{D}_p^i$$

або

$$^{в}D_i^1 \subseteq \bigcup_{l=k_{i-1}+1}^{k_i-1} \bigcup_{p=k_{i-1}+1}^{k_i} {}_l\breve{D}_p^i,$$

тобто локальні дані відіграють роль глобальних тільки в рамках однієї (*i*-ї в цьому випадку) РС.

Отримані властивості дозволяють стверджувати, що обмеження виконується для кожної РС, що входить у даний шар, і для всього шару алгоритму. Зауважимо, що в останньому випадку, як і у всіх попередніх, перша підсистема, для якої немає попередніх, не містить на вході, а остання підсистема, для якої немає наступних підсистем, не містить на виході єднальних даних.

Таким чином, нам вдалося специфікувати зв'язки між підсистемами в рамках першого шару алгоритму і сформулювати властивості даних у цьому шарі. Очевидно, що процес декомпозиції повторюється на кожному кроці розробки, і структура другого й наступного шарів буде повністю повторювати структуру першого шару, зберігаючи при цьому всі вищесформульовані властивості. Розбіжність між шарами алгоритму полягатиме тільки в тому, що кожний наступний шар буде товщим за попередній за рахунок зростання кількості РС. Процес декомпозиції при розглянутому підході триває до того моменту, коли будуть виділені всі підсистеми, що утворюють проектовану програмну систему, і не будуть специфіковані всі зв'язки, існуючі між підсистемами, що утворюють даний шар алгоритму.

При розгляді даного етапу ми брали найзагальніший випадок, коли кожна підсистема, що входить у РС або шар алгоритму, зв'язана з усіма попередніми й усіма наступними за нею. На практиці такі випадки зустрічаються досить рідко, тобто зазвичай не всі підсистеми зв'язані одна з одною, і мають місце не зв'язані (незалежні) з іншими підсистеми. Тому зауважимо, що кожна з множин $^{вх}D_i^{m}$, $^{вих}D_i^{m}$, D_i^{m}, ${}_iD'^{m}$ (у будь-якому сполученні, але не всі відразу) може бути порожньою, і, таким чином, у конкретних розробках припустимі численні окремі випадки підсистем (наприклад: $(D_i^1, {}^{вх}D_i^1, {}^{в}D_i^1)A_i^1({}_iD^{i1})$, $(D_i^1, {}^{в}D_i^1)A_i^1({}^{вих}D_i^1, {}_iD^{i1})$, $(D_i^1, {}^{в}D_i^1)A_i^1({}_iD^{i1})$ і т. д.), що мають наведені у статті властивості.

На основі специфікованих зв'язків уведемо деякі поняття, що випливають із властивостей даних та інформаційних зв'язків.

Визначимо поняття зв'язків між підсистемами в РС (2).

Визначення 12. Множину $S_j^{л} = {}_1\breve{D}_j, {}_2\breve{D}_j, \ldots, {}_i\breve{D}_j, \ldots, {}_{j-1}\breve{D}_j$ назвемо множиною лівих зв'язків *j*-ї підсистеми, а множину $S_j^{пр} = {}_j\vec{D}_{j+1}, {}_j\vec{D}_{j+2}, \ldots, {}_j\vec{D}_p, \ldots, {}_j\vec{D}_n$ – множиною її правих зв'язків. Отже, будемо розрізняти три типи підсистем, які назвемо так:

– зв'язані, у яких $S_j^{л} \neq \varnothing$ і $S_j^{пр} \neq \varnothing$;

– зв'язані ліворуч (праворуч), у яких $S_j^{л} \neq \varnothing$, а $S_j^{пр} = \varnothing$ ($S_j^{пр} \neq \varnothing$, а $S_j^{л} = \varnothing$);

– не зв'язані, у яких $S_j^{л} = \varnothing$ і $S_j^{пр} = \varnothing$.

Множину $S_j^{о} = {}_1\vec{D}_{j+1}, \ldots, {}_1\vec{D}_n, {}_2\vec{D}_{j+1}, \ldots, {}_2\vec{D}_n, \ldots, {}_{j-1}\vec{D}_{j+1}, \ldots, {}_{j-1}\vec{D}_n$ назвемо множиною зв'язків, що обводять *j*-ту підсистему. Кількість обвідних зв'язків A_j підсистеми визначимо як $K_j^{о} = \left|S_j^{о}\right|$:. Кожна з множин ${}_j\vec{D}_l$ (${}_r\vec{D}_s$), що входять у множини $S_j^{л}$, $S_j^{пр}$, $S_j^{о}$, може бути порожньою, і в цьому випадку вона з розгляду виключається. Із цього випливає, що множини $S_j^{л}$, $S_j^{пр}$, $S_j^{о}$ так само можуть бути порожніми.

Тепер уведемо й визначимо поняття довжини інформаційних зв'язків.

Визначення 13. Довжину довільного зв'язку між підсистемами A_i та A_j визначимо для випадку $i < j$ як ${}_id_j = j - i$, а для випадку $j < i$ – як ${}_id_j = i - j$.

Виходячи з визначень 3 і 4, загальну (сумарну) довжину лівих (правих) зв'язків підсистеми A_k запишемо у вигляді $L_k^{л} = \sum_{p=1}^{k-1} {}_pd_k$ ($L_k^{пр} = \sum_{p=k+1}^{n} {}_kd_p$), а загальну довжину зв'язків, що обводять *k*-ту підсистему, – у вигляді $L_k^{о} = \sum_{l=1}^{k-1} \sum_{r=k+1}^{n} {}_ld_r$. У всіх наведених випадках кожен з елементів ${}_id_j$, що входять у наведені вирази, може дорівнювати нулю.

Таким чином, ми визначили поняття, необхідні для специфікації інформаційних зв'язків в алгоритмах, і їхні основні властивості.

Висновки. Запропоновано підхід до формалізації даних і на цій основі специфіковано інформаційні зв'язки в регулярних схемах, що утворюють алгоритм. У процесі формалізації даних та інформаційних зв'язків визначено й отримано властивості як перших, так і других. У результаті на певному етапі розробки алгоритму

виявляється не тільки структуроване уявлення про програмну систему, тобто визначаються склад і функції підсистем, але й взаємозв'язки між усіма підсистемами.

Отримані результати дозволяють у перспективі виконувати завдання, пов'язані з проектуванням алгоритмів, їх перетворенням і контролем коректності (як побудованих, так і перетворених алгоритмів).

Література

1. Турский В. Методология программирования [Текст] / В. Турский. – М. : Мир, 1981. – 264 с.

2. Ющенко Е. Л. Многоуровневое структурное проектирование программ: теоретические основы, инструментарий [Текст] / Е. Л. Ющенко, Г. Е. Цейтлин, В. П. Грицай, Т. К. Терзян. − М. : Финансы и статистика, 1989. – 208 с.

3. Глушков В. М. Теория автоматов и формальные преобразования микропрограмм [Текст] / В. М. Глушков // Кибернетика. – 1965. – № 5. – С. 1−10.

4. Глушков В. М. Алгебра. Языки. Программирование [Текст] / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. − К. : Наукова думка, 1978. − 319 с.

5. Акуловский В. Г. Формализация взаимосвязей операторов и данных в рамках расширенной алгебры алгоритмов [Текст] / В. Г. Акуловский // Кибернетика и системный анализ. − 2008. − № 6. − С. 170−182.

6. Акуловський В. Г. Деякі аспекти формалізації архітектурного етапу розробки алгоритмів [Текст] / В. Г. Акуловський // Проблеми програмування. – 2009. – № 2. – С. 3−11.