УДК 519.683

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
В. В. Поліщук, старший викладач кафедри
інформаційних систем та технологій
Академії митної служби України

ДЕЯКІ ВЛАСТИВОСТІ ДАНИХ У КОМПОЗИЦІЙНИХ СХЕМАХ АЛГОРИТМІВ

Наведено формальний апарат − алгебру алгоритмів, що базується на даних та орієнтована на опис алгоритмів у вигляді регулярних і композиційних схем. Показано можливість специфікації оброблюваних даних у рамках композиційної схеми й отримано властивості цих даних.

Приведен формальный аппарат – алгебра алгоритмов, базирующаяся на данных, ориентированная на описание алгоритмов в виде регулярных и композиционных схем. Показана возможность спецификации обрабатываемых данных в рамках композиционной схемы и получены свойства этих данных.

The formal device – algebra of algorithms which is based on the data, focused on the description of algorithms as regular and composite circuits is resulted. The opportunity of the specification of the processable data is shown within the framework of the composite circuit and the received properties of these data.

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

Вступ. Необхідна умова існування деякої методології програмування – це наявність формального апарату, що забезпечує концептуальну цілісність методології [1]. Таким формальним апаратом є алгебра алгоритмів, запропонована В. М. Глушковим [2], яка в цей час продовжує модифікуватися й розвиватися (наприклад, [3, 4]).

Разом з тим давно відома найважливіша, а в багатьох випадках визначальна роль даних у процесі розробки алгоритмів і програм [1]. Однак можливості алгебри алгоритмів у виконанні завдання формалізації даних використано далеко не повністю. Реалізація невикористаних можливостей підвищить якість розроблюваних алгоритмів, тому що дозволить поєднати два відомих підходи до програмування: від керування й від даних. Крім того, розширити виражальні можливості формального апарату дозволить використання тризначних логічних умов, ефективність яких обґрунтовано в ряді праць [5, 6].

Раніше отримано деякі результати, спрямовані на виконання зазначених завдань [7–9], які в цьому дослідженні розвиватимуться й уточнюватимуться. Зокрема, нижче розглядатимуться питання, пов'язані з визначенням та аналізом властивостей, якими володіють дані в композиційних схемах і за допомогою яких здійснюється формальний запис довільних алгоритмів.

Постановка завдання. Мета статті – розробка формального апарату для опису алгоритмів у вигляді композиційних схем, які враховують склад і структуру даних, що обробляються.

Формальний апарат. Як основу для побудови формального апарату використано систему алгоритмічних алгебр (САА), що базується на моделі ЕОМ Глушкова. Ця модель, як відомо, являє собою два взаємодіючих автомати: керуючий К та операційний О. У подальших міркуваннях будемо виходити з тієї ідеї, яка висловлена в [10], що за різних додаткових припущень функціонування моделі може інтерпретуватися по-різному.

Скориставшись цією ідеєю, вважатимемо, що в модифікованій моделі ЕОМ стан автомата О визначається станом множини даних *D,* носієм якої є цей автомат. При цьому під даними розумітимемо множину змінних $D = \{D_1, D_2, ..., D_i, ..., D_n\}$, кожна з яких іменує деяку сукупність інших простіших змінних, тобто $D_i = \{{}_iD_1, {}_iD_2, ..., {}_iD_j, ..., {}_iD_k\}$, тощо. Таким чином, дані являють собою ієрархію, на нижньому рівні якої розташовано елементарні (атомарні) дані зі своїми ідентифікаторами. Подальші побудови будемо здійснювати, абстрагуючись від тих значень, яких набувають змінні. Тобто розглядатимемо найзагальніші властивості даних, думаючи при цьому, що змінні рівні, якщо вони мають однакову структуру й склад. Запишемо це у вигляді $D_i = D_j$. У контексті цього дослідження терміни "змінні" й "дані" вважатимемо синонімами.

Тепер розглянемо систему алгоритмічних алгебр, що базується на даних (САА\Д).

Нехай $< U, W, \Omega >$ *САА\Д,* де U − множина операторів; W – множина логічних умов, Ω − сигнатура операцій, що складається з логічних операцій Ω_1, які набувають значення на множині W, і операцій Ω_2, що набувають значення на множині операторів U.

На множині W визначено відомі (наприклад, [10]) операції диз'юнкції, кон'юнкції та заперечення.

На множині даних D визначено Д-оператори з U і предикати, які розглянемо нижче.

Вважатимемо, що Д-оператор змінює (перетворює) дані з множини D. Якщо множина $D_j \subseteq D$ є областю визначення, а множина $D'_j \subseteq D$ – областю значення, то деякий

Д-оператор може бути записаний у вигляді $A(D_j) = D'_j$, де $A \in U$, $D_j, D'_j \subseteq D$.

Надалі Д-оператор записуватимемо в більш зручному вигляді $(D)A(D')$, множину даних D називатимемо

вхідними, а множину D' – вихідними даними Д-оператора.

Визначення 1. Д-оператор $(D_A)A(D'_A)$ перетворює множину вхідних даних $D_A \subseteq D$ на множину вихідних даних $D'_A \subseteq D$. При цьому для множин D_A і D'_A припустиме кожне з таких співвідношень: $D_A = D'_A$, $D_A \subset D'_A$, $D_A \supset D'_A$, $D_A \cap D'_A = \varnothing$, $D_A \cap D'_A \neq \varnothing$.

Тепер уведемо поняття невизначеного й тотожного оператора.

Визначення 2. Д-оператор $(D_A)A(D'_A)$ називатимемо невизначеним і позначатимемо *N,* якщо в результаті його виконання обчислювальний процес переходить у невизначений стан.

Слід зазначити, що під невизначеним станом обчислювального процесу розумітимемо втрату керування, порушення послідовності обчислень і зупинку (зациклення) цього процесу.

Визначення 3. Д-оператор $(D_A)A(D'_A)$ називатимемо тотожним і позначатимемо *E,* якщо після його виконання ніякі дані не змінюються і, таким чином, Д-оператор не впливає на стан автомата О.

Далі вважатимемо, що на множині *D* визначено предикати, які утворюють дві множини: множину двозначних – *P*2 і множину тризначних – *P*3 предикатів, таких, що $p_2(D) = \alpha_2 \in E_2 = \{0, 1\}$, де $p_2(D) \in P_2$, $p_3(D) = \alpha_3 \in E_3 = \{0, 1, \mu\}$, де $p_3(D) \in P_3$, а $\alpha_3 = \mu$, коли $\alpha_3 \notin \{0, 1\}$. Операційний автомат, обчислюючи ці предикати, формує на своєму виході елементарні логічні умови, які надходять на вхід керуючого автомата. Зауважимо, що обчислення предикатів не веде до зміни інформаційної множини *D* автомата О.

На множині *U* введемо певні операції, причому так, щоб вони були скрізь визначені.

Операцію *p*-диз'юнкції розглядатимемо у варіантах, що залежать від виду використовуваного предиката.

$$[p_3(\tilde{D})](D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C) = \begin{cases} (D_A)A(D'_A), \text{ якщо } \alpha_3 = 1; \\ (D_B)B(D'_B), \text{ якщо } \alpha_3 = 0; \\ (D_C)C(D'_C), \text{ якщо } \alpha_3 = \mu, \end{cases}$$

де $(D_A)A(D'_A), (D_B)B(D'_B), (D_C)C(D'_C) \in U$, $\tilde{D}, D_A, D'_A, D_B, D'_B, D_C, D'_C \subseteq D$, $p_3(\tilde{D}) \in P_3$, $p_3(\tilde{D}) = \alpha_3$, $\alpha_3 \in W$.

Результат виконання цієї конструкції – виконання одного з трьох можливих операторів, який вибирається відповідно до значення логічної умови α_3, що набуває істинних значень тризначної логіки E_3.

$$[p_2(\tilde{D})](D_A)A(D'_A) \vee (D_B)B(D'_B)) = \begin{cases} (D_A)A(D'_A), \text{ якщо } \alpha_2 = 1; \\ (D_B)B(D'_B), \text{ якщо } \alpha_2 \neq 1. \end{cases}, \text{ де } p_2(\tilde{D}) \in P_2,\ \alpha_2 \in W.$$

Результатом виконання цього варіанта *p*-диз'юнкції є виконання одного з двох можливих операторів, який вибирається відповідно до значення логічної умови α_2 і набуває

істинних значень двозначної логіки E_2. При цьому перший з операторів виконується за

істинного значення логічної умови, а другий – у всіх інших випадках.

Окремим випадком *p*-диз'юнкції є операція *p*-фільтрації (послідовна фільтрація), що вводиться в такий спосіб:

$$[p_2(\tilde{D})](D_A)A(D'_A) \vee E) = [p_2(\tilde{D})](D_A)A(D'_A)) = (D_A)A(D'_A), \text{ якщо } \alpha_2 = 1,$$

де $p_2(\tilde{D}) \in P_2$, $\alpha_2 \in W$.

Результат виконання цієї конструкції – виконання оператора $(D_A)A(D'_A)$ за умови істинності логічної умови α_2.

Наступна операція, яку ми розглянемо, – *p*-ітерація, введемо її в такий спосіб.

$p_{P_3(\tilde{D})}\{(D_A)A(D'_A)\}$ – операція *p*-ітерації складається з обчислення предиката $P_3(\tilde{D})$ й перевірки отриманої логічної умови α_2. Якщо умова α_2 істинна, виконується оператор $(D_A)A(D'_A)$ та знову обчислюється предикат і перевіряється умова α_2. Циклічний процес, що складається з перевірки умови α_2 й виконання оператора, здійснюється доти, доки умова $\alpha_2 = 1$ у противному випадку операція *p*-ітерації завершується.

У всіх розглянутих операціях як предикат можуть виступати сукупності предикатів, у цьому випадку такі, що логічні умови, які ними спродуковані, зв'язуються логічними операціями множини Ω_1.

Операцію композиції (позначається "*") визначимо в такий спосіб.

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

Ця операція має такі властивості:

$$E * (D_A)A(D'_A) = (D_A)A(D'_A) * E = (D_A)A(D'_A);\quad N * (D_A)A(D'_A) = (D_A)A(D'_A) * N = N.$$

де *N* – невизначений, а *E* – тотожний оператори.

Тепер визначимо регулярну схему Д-операторів.

Визначення 4. Подання будь-якого Д-оператора з U через твірні елементи системи $<U, W, \Omega>$ називається регулярною схемою цього Д-оператора (РСД).

У рамках РСД операція композиції має такі властивості:

$$E^* ОП = ОП^* E = ОП;$$

$$N^* ОП = ОП^* N = N,$$

де $ОП$ – довільна операція з Ω_1, N – невизначений, а E – тотожний оператори з U.

Виходячи з визначення 4 і наведених властивостей, стверджуватимемо таке.

Твердження 1. Будь-яка операція множини Ω_2 може розглядатися як деякий Д-оператор.

Доведення. Деякий оператор можна подати у вигляді такої РСД

$$(D_A)A(D'_A) = (D_B)B(D'_B)^* ОП,$$

де ОП довільна операція. Якщо оператор $(D_B)B(D'_B)$ тотожний, то, відповідно до властивості операції композиції, цей вираз матиме такий вигляд: $(D_A)A(D'_A) = E^* ОП = ОП$.

Наслідок. Будь-яку операцію множини Ω_2 можна записувати (оформляти) у вигляді Д-оператора.

Відповідно до твердження 1 і наслідку з нього вищенаведені операції з Ω_2 запишемо в такому вигляді:

– p-диз'юнкція являє собою оператор:

$$[p_1(\tilde{D})]((D_A)A(D'_A) \vee (D_B)B(D'_B) \vee (D_C)C(D'_C) = (D_G)G(D'_G),$$

де $D_G = \tilde{D} \cup D_A \cup D_B \cup D_C$, $D'_G = D'_A \cup D'_B \cup D'_C$, причому $\tilde{D} \neq \varnothing$;

– p-ітерація являє собою оператор:

$$p_2(\tilde{D})\{(D_A)A(D'_A)\} = (D_G)G(D'_G),$$

де $D_G = D_A \cup \tilde{D}$, $D'_G = D'_A$ і $\tilde{D} \neq \varnothing$;

– інші операції записуються аналогічно.

На підставі доведеного твердження введемо поняття композиційної схеми Д-операторів.

Визначення 5. Регулярна схема Д-оператора, в якій використовується єдина операція – композиція, а інші операції множини Ω_2 записано у вигляді Д-операторів, називається композиційною схемою (КС) цього Д-оператора.

Для продовження розгляду наведемо таку аксіому.

Аксіома. Будь-який алгоритм і будь-який Д-оператор, що входить до нього і називається вихідним, може бути поданий КС $(D_A)A(D'_A) = (D_B)B(D'_B)^*(D_C)C(D'_C)$, тобто може бути декомпозований і зображений у вигляді композиції двох інших Д-операторів, що називаються похідними. Функціональність похідних операторів адекватна функціональності вихідного.

Зауважимо, що правдивість наведеної аксіоми може викликати сумніви у випадку елементарних Д-операторів, тобто Д-операторів, що виконують одну елементарну операцію над елементарними даними. Однак очевидно, що в такому випадку декомпозиція може бути здійснена за допомогою найпростіших штучних прийомів, якщо така операція необхідна.

Результати дослідження

Властивості даних у композиційній схемі. Щоб визначити властивості даних у КС, звернемо увагу на той факт, що похідні Д-оператори разом виконують одне спільне для них завдання, адекватне завданню, що виконується вихідним Д-оператором.

Перелічимо ті аспекти декомпозиції, які випливають зі згаданого факту:

– по-перше, похідні Д-оператори використовують (можуть використовувати) загальні вхідні дані D_A;

– по-друге, вихідні дані вихідного Д-оператора D'_A є результатом послідовного виконання результуючих Д-операторів, тобто вони обидва продукують (можуть продукувати) ці дані. При цьому другий Д-оператор продовжує обробку даних (або деякої їхньої підмножини), почату першим Д-оператором;

– по-третє, для обробки даних у переважній більшості алгоритмів використовуються допоміжні дані (далі – проміжні), які, не будучи ні вхідними, ні вихідними, служать для перетворення перших на другі.

Для того щоб урахувати зазначені аспекти декомпозиції, наведемо визначення структури вхідних і вихідних даних, яке було введено в [9].

Визначення 6. Вхідні й вихідні дані Д-оператора $(D)A(D')$ утворено такими підмножинами:

– $D = \hat{D} \cup \tilde{D}$, такими що $\hat{D} \cap \tilde{D} = \varnothing$;

– $D' = \hat{D}' \cup \tilde{D}'$, таких що $\hat{D}' \cap \tilde{D}' = \varnothing$;

які назвемо: $\hat{D}$ – вихідні, $\hat{D}$, $\hat{D}'$ – прохідні, $\tilde{D}$ – похідні. При цьому $D \cap D' = \hat{D} = \hat{D}'$, тобто [...] й $\tilde{D}' \cap D = \varnothing$. Кожна з підмножин, що утворюють множини D і D' (як і самі ці множини), може бути

порожньою.

З визначення випливають такі наведені нижче властивості даних у Д-операторах.

Вихідні дані не з'являються на виході Д-оператора й, таким чином, не змінюються, похідні дані не є вхідними, а продукуються Д-оператором і входять до складу вихідних, а множини прохідних даних [ ] і [ ] являють собою одну множину, по-різному позначену на вході й виході Д-оператора. Тобто ці дані, наявні на вході, обробляються
Д-оператором, після чого входять до складу вихідних.

Відповідно до аксіоми й визначеної вище структури даних на вході й виході
Д-оператора запишемо його КС у вигляді

[ ] (1)

і вважатимемо, що дані надходять на вхід або вихід деякого оператора й передаються з виходу одного похідного оператора на вхід іншого.

З огляду на перший з названих аспектів декомпозиції будемо думати, що для вихідних і прохідних даних виконуються такі співвідношення:

$$\hat{D}_A = \hat{D}_B \cup \hat{D}_C, \text{ де } \hat{D}_B \cap \hat{D}_C = \hat{D}_{BC}; \quad (2)$$

$$\hat{D}_A = \hat{D}_B \cup \hat{D}_C, \text{ де } \hat{D}_B \cap \hat{D}_C = \hat{D}_{BC}. \quad (3)$$

Отже, вхідні дані вихідного Д-оператора можуть бути довільно розподілені між похідними Д-операторами. Причому будь-яка підмножина цих даних (або сполучення цих підмножин) може бути порожньою.

Розгляд другого аспекту декомпозиції Д-оператора почнемо з властивостей прохідних даних, що випливають з визначення 6:

$$\hat{D}'_A = \hat{D}'_B \cup \hat{D}'_C. \quad (4)$$

Прохідні дані, якщо вони наявні у виразі (1), тобто $\hat{D}_A = \hat{D}'_A \neq \varnothing$, й, відповідно (4), $\hat{D}_B \neq \varnothing$ і/або $\hat{D}_C \neq \varnothing$, мають такі властивості.

1. Якщо $\hat{D}_B = \varnothing$ або $\hat{D}_C = \varnothing$, то, відповідно, $\hat{D}_A = \hat{D}_C = \hat{D}'_C = \hat{D}'_A$ або $\hat{D}_A = \hat{D}_B = \hat{D}'_B = \hat{D}'_A$, тобто всі прохідні дані обробляються одним з похідних Д-операторів $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$ або $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, у той час як другий Д-оператор прохідних даних не містить.

2. Якщо $\hat{D}_B \neq \varnothing$ і $\hat{D}_C \neq \varnothing$, то мають місце три випадки:

2.1. Якщо $\hat{D}_A = \hat{D}_B = \hat{D}_C$ ($\hat{D}_B = \hat{D}_C = \hat{D}_{BC}$), то $\hat{D}_A = \hat{D}_B = \hat{D}'_B = \hat{D}_C = \hat{D}'_C = \hat{D}'_A$, тобто всі прохідні дані спочатку обробляються Д-оператором $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, а потім Д-оператором $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$.

2.2. Якщо $\hat{D}_A = \hat{D}_B$, але $\hat{D}_B \neq \hat{D}_C$ або $\hat{D}_A = \hat{D}_C$, але $\hat{D}_C \neq \hat{D}_B$, то $\hat{D}_C \subset \hat{D}_B$ ($\hat{D}_C = \hat{D}_{BC}$) або $\hat{D}_B \subset \hat{D}_C$($\hat{D}_B = \hat{D}_{BC}$), звідки $\hat{D}_A = \hat{D}_B = \hat{D}'_B = \hat{D}'_A$ або $\hat{D}_A = \hat{D}_C = \hat{D}'_C = \hat{D}'_A$. Тобто в першому випадку всі прохідні дані обробляються Д-оператором $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, а обробка частини даних $\hat{D}_C$($\hat{D}_{BC}$) продовжується Д-оператором $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$. У другому – всі прохідні дані обробляються Д-оператором $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, але частина даних $\hat{D}_B$($\hat{D}_{BC}$) попередньо обробляється Д-оператором $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$.

2.3. Якщо $\hat{D}_A \neq \hat{D}_B$ і $\hat{D}_A \neq \hat{D}_C$, то мають місце такі 2 випадки:

2.3.1. Якщо $\hat{D}_B \cap \hat{D}_C = \varnothing$ ($\hat{D}_{BC} = \varnothing$), то $\hat{D}_A = \hat{D}_B \cup \hat{D}_C = \hat{D}'_B \cup \hat{D}'_C = \hat{D}'_A$, тобто кожний з Д-операторів обробляє свою частину прохідних даних і дані $\hat{D}'_B$ – на вихід вихідного
Д-оператора, оминаючи Д-оператор $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, а дані $\hat{D}_C$ потрапляють на вхід
Д-оператора $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, минаючи Д-оператор $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$.

2.3.2. Якщо $\hat{D}_B \cap \hat{D}_C = \hat{D}_{BC} \neq \varnothing$, де $\hat{D}_B / \hat{D}_{BC} = {}_A\hat{D}_B$ і $\hat{D}_C / \hat{D}_{BC} = {}_A\hat{D}_C$, то [ ] $= {}_A\hat{D}'_B \cup \hat{D}'_C = \hat{D}'_A$, де $\hat{D}_B = {}_A\hat{D}_B \cup \hat{D}_{BC}$, $\hat{D}'_C = {}_A\hat{D}'_C \cup \hat{D}'_{BC}$. Тобто множина даних $\hat{D}_{BC}$ послідовно обробляється спочатку Д-оператором $(\tilde{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, а потім Д-операто-
ром $(\tilde{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, на виході якого воно позначається $\hat{D}'_{BC}$. Множина даних ${}_A\hat{D}_B$ обробляється Д-

оператором $(\hat{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, після чого множина ${}_A\hat{D}'_B$ потрапляє на вихід вихідного Д-оператора, оминаючи оператор $(\hat{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$. Множина даних ${}_A\hat{D}_C$ обробляється Д-оператором $(\hat{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, причому множина ${}_A\hat{D}_C$ потрапляє на вхід цього Д-оператора, оминаючи Д-оператор $(\hat{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$, а на виході позначається ${}_A\hat{D}'_C$.

Виходячи з розглянутих властивостей прохідних даних і співвідношень (3), (4), запишемо такі співвідношення, що визначають у загальному випадку склад цих даних на вході й виході похідних Д-операторів:

– на вході Д-оператора $(\hat{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$

$$\hat{D}_B = {}_A\hat{D}_B \cup \hat{D}_{BC}; \tag{5}$$

– на його виході

$$\hat{D}'_B = {}_A\hat{D}'_B \cup \hat{D}_{BC}, \tag{6}$$

де ${}_A\hat{D}'_B \subseteq \hat{D}'_A$, $\hat{D}_{BC} \subseteq \hat{D}_C$;

– на вході Д-оператора

$$(\hat{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C) - \hat{D}_C = {}_A\hat{D}_C \cup \hat{D}_{BC}; \tag{7}$$

– на його виході

$$\hat{D}'_C = {}_A\hat{D}'_C \cup \hat{D}'_{BC}, \tag{8}$$

де $\hat{D}'_C \subseteq \hat{D}'_A$ й $\hat{D}'_{BC} \subseteq \hat{D}'_A$.

Зі співвідношень (5)–(8) випливає, що прохідні дані на вході й виході вихідного Д-оператора являють собою об'єднання таких підмножин:

$$\hat{D}_A = {}_A\hat{D}_B \cup \hat{D}_{BC} \cup {}_A\hat{D}_C; \tag{9}$$

$$\hat{D}'_A = {}_A\hat{D}'_B \cup \hat{D}'_{BC} \cup {}_A\hat{D}'_C. \tag{10}$$

Далі розглянемо похідні дані, для яких виконується таке співвідношення

$$\tilde{D}'_B \cup \tilde{D}'_C = \tilde{D}'_A. \tag{11}$$

Таким чином, з вищенаведених властивостей, визначення 6 і співвідношень (2) і (9) випливає, що вхідні дані вихідного оператора в загальному випадку описуються виразом:

$$D_A = \hat{D}_A \cup \hat{D}_A = (\hat{D}_B \cup \hat{D}_C) \cup ({}_A\hat{D}_B \cup \hat{D}_{BC} \cup {}_A\hat{D}_C), \tag{12}$$

а зі співвідношень (3) і (10) – що вихідні дані вихідного оператора в загальному випадку описуються виразом:

$$D'_A = \tilde{D}'_A \cup \hat{D}'_A = (\tilde{D}'_B \cup \tilde{D}'_C) \cup ({}_A\hat{D}'_B \cup {}_A\hat{D}'_C \cup \hat{D}'_{BC}). \tag{13}$$

Тепер звернемося до третього аспекту декомпозиції та доведемо, що його реалізація в рамках уведених понять неможлива.

Твердження 2. Проміжні дані, якщо вони не є вхідними чи вихідними для вихідного оператора $(\hat{D}_A, \hat{D}_A)A(\hat{D}'_A \tilde{D}'_A)$, не можуть бути отримані з уведених типів даних.

Доведення. Зі співвідношень (12), (13) видно, що всі дані на входах і виходах похідних операторів входять до складу відповідно вхідних і вихідних даних вихідного оператора. Отже, твердження доведене.

Наслідок. Для реалізації третього аспекту декомпозиції слід доповнити склад використовуваних типів даних.

Виходячи з доведеного твердження 2 і наслідку з нього, введемо проміжні дані в такий спосіб.

Визначення 7. Множина даних D_{BC}, що має такі властивості: $D'_B \supseteq D_{BC} \subseteq D_C$, $D_{BC} \cap \tilde{D}'_B = \varnothing$, $D_{BC} \cap \hat{D}'_B = \varnothing$, $D_{BC} \cap \hat{D}_C = \varnothing$, $D_{BC} \cap \hat{D}_C = \varnothing$ і зберігає властивості, сформульовані у визначенні 6, називатимемо проміжними даними.

Доведемо, що побудований тип даних відповідає вимогам, висунутим до проміжних даних, випередивши доведення лемою.

Лема. Проміжні дані множини D_{BC} не перетинаються із жодною з підмножин, що утворюють множину прохідних даних, і тому на вході й виході вихідного Д-оператора відсутні.

Доведення. Оскільки, за визначенням 7, $D_{BC} \cap \hat{D}'_B = \varnothing$, то, відповідно до (6), $D_{BC} \cap_A \hat{D}'_B = \varnothing$, $D_{BC} \cap \hat{D}_{BC} = \varnothing$. За визначенням 6, $\hat{D}'_B = \hat{D}_B$, тому, відповідно до (5), $D_{BC} \cap_A \hat{D}_B = \varnothing$. Оскільки, за визначенням 7, $D_{BC} \cap \hat{D}_C = \varnothing$, то, відповідно до (7), $D_{BC} \cap_A \hat{D}_C = \varnothing$. Відповідно до визначення 6, $\hat{D}_C = \hat{D}'_C$, тому, відповідно до (8), $D_{BC} \cap_A \hat{D}'_C = \varnothing$ і $D_{BC} \cap \hat{D}'_{BC} = \varnothing$. З вищевикладеного і співвідношень (9) і (10) видно, що множина D_{BC} не перетинається із жодною з підмножин, що утворюють множини прохідних даних $\hat{D}_A$ і $\hat{D}'_A$ на вході й виході вихідного Д-оператора, отже, й лему доведено.

Теорема. Дані з множини D_{BC} у композиційній схемі проміжні, оскільки зв'язують похідні Д-оператори, а на вході й виході вихідного Д-оператора відсутні через те, що не перетинаються із жодною з підмножин, які утворюють вхідні й вихідні дані вихідного
Д-оператора.

Доведення. Множина даних D_{BC} надходить з виходу Д-оператора $(\hat{D}_B, \hat{D}_B)B(\hat{D}'_B, \tilde{D}'_B)$ на вхід оператора $(\hat{D}_C, \hat{D}_C)C(\hat{D}'_C, \tilde{D}'_C)$, тобто вони єднальні, тому що, відповідно до визначення (7), $D'_B \supset D_{BC} \subset D_C$. Згідно з визначенням (7), $D_{BC} \cap \tilde{D}'_B = \varnothing$ і $D_{BC} \cap \tilde{D}_C = \varnothing$. А оскільки, відповідно до визначення (6), $D_{BC} \cap D_B = \varnothing$ і $D_{BC} \cap D'_C = \varnothing$, то $D_{BC} \cap \tilde{D}_B = \varnothing$ і [illegible]. З вищевикладеного, доведеної леми і співвідношень (12) і (13) видно, що множина прохідних даних D_{BC} не перетинається із жодною з підмножин, що утворюють множини прохідних даних D_A і D'_A на вході й виході вихідного Д-оператора, і теорема, таким чином, доведена.

Наслідок. Проміжні дані D_{BC} локальні для Д-оператора $(\hat{D}_A, \hat{D}_A)A(\hat{D}'_A \tilde{D}'_A)$, тому що вони не специфікуються на його вході й виході, отже, "не видимі" поза цим *Д*-оператором.

Тепер повернемося до співвідношення (2) і, позначивши $_A\hat{D}_B = \hat{D}_B \setminus \hat{D}_{BC}$ й $_A\hat{D}_C = \hat{D}_C \setminus \hat{D}_{BC}$, множину вихідних даних запишемо у вигляді [illegible].

Виходячи з визначених і доведених властивостей даних у КС, вона може бути записана у вигляді

[illegible]

де специфіковано всі введені типи даних, які мають ці властивості. При цьому множина проміжних даних D_{BC}, використовувана для перетворення вхідних даних вихідного
Д-оператора на вихідні, зв'язує похідні Д-оператори для виконання рішення цього завдання і є локальною в рамках вихідного Д-оператора.

Висновки. У цій статті наведено формальний апарат САА/Д, орієнтований на опис алгоритмів, у вигляді РСД і КС. Причому використовувані в рамках РСД тризначні предикати забезпечують компактність запису деяких фрагментів алгоритмів.

Формалізація даних у рамках КС і отримані властивості цих даних дозволяють виконувати завдання декомпозиції Д-операторів, що утворюють алгоритм, який погоджено з деталізацією даних, специфікованих на входах і виходах цих Д-операторів. У процесі декомпозиції враховується специфіка виконання цієї операції. Отримані властивості даних дозволяють коректно виконувати такі перетворення.

Рекурсивне застосування аксіоми дає змогу одержувати досить детальні композиційні схеми, тобто КС, що містять у своєму складі більше двох Д-операторів.

Уведені прохідні дані є засобом інформаційного зв'язку між операторами, що входять у КС, дозволяють виконувати аналіз таких взаємозв'язків і на підставі такого аналізу оптимізувати перетворення алгоритмів. Деякі результати під час виконання цих завдань отримано у працях [8] і [11].

Напрямки подальшого розвитку отриманих результатів – формалізація процесу деталізації даних, тобто побудова алгебри структур даних, а також пошук методів і засобів, спрямованих на перетворення КС і РСД.

Література

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

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

3. Алгеброалгоритмические модели и методы параллельного программирования [Текст] / Ф. И. Андон, А. Е. Дорошенко, Г. Е. Цейтлин, Е. А. Яценко. – К. : Академперіодика, 2007. – 634 с.

4. Акуловский В. Г. Расширенная алгебра алгоритмов [Текст] / В. Г. Акуловский // Проблеми програмування. – 2007. – № 3. – С. 3–15.

5. Брусенцов Н. П. Трехзначная диалектическая логика [Текст] / Н. П. Брусенцов // Программные системы и инструменты. Тематический сборник № 2. – М. : Факультет ВМиК МГУ, 2001. − С. 36−44.

6. Брусенцов Н. П. Троичное конструктное кодирование булевых выражений [Текст] / Н. П. Брусенцов, Ю. С. Владимирова // Программные системы и инструменты. Тематический сборник № 3. − М. : Факультет ВМиК МГУ, 2002. – С. 6–10.

7. 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.

8. Акуловский В. Г. Некоторые аспекты преобразования алгоритмов на основе формализации информационных связей [Текст] / В. Г. Акуловский // Кибернетика и системный анализ. – 2009. – № 6. – С. 50–54.

9. Акуловский В. Г. Некоторые аспекты формализации данных и декомпозиция Д-операторов алгебры алгоритмов [Текст] / В. Г. Акуловский // Проблеми програмування. – 2009. − № 4. − С. 3–10.

10. 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.

11. Акуловський В. Г. Деякі аспекти формалізації та специфікації інформаційних зв'язків в алгоритмах [Текст] / В. Г. Акуловський, В. В. Костенко // Вісник Академії митної служби України. – 2009. – № 2. – С. 105–114.