УДК 519.9

В.В. Зубенко

ОПЕРАЦІЙНА МОДЕЛЬ КОМУНІКАТИВНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Розглядається операційна модель комунікативних інформаційних систем (КІС). Введено поняття абстактного алгоритму (А-алгоритму) як загальної моделі неавтоматних алгоритмічних систем. Показано, що А-алгоритмам притаманні усі властивості класичних алгоритмів, а для класу, так званих, табличних А-алгоритмів виконуються теореми про регулярний аналіз та синтез.

Вступ

В роботах [1, 2] введено поняття комунікативної платформи інформатики. Проголосивши предметом комунікативної інформатики конструктивні моделі комунікативних процесів та систем (так звані, комунікативні інформаційні системи та процеси), в [3] побудовано композиційну модель таких систем. Ця модель сугубо алгебраїчна, що описує функціональну семантику останніх – структуру інформаційних об'єктів та інформаційних полів (даних), а також функцій і співвідношень, визначених на даних. Тепер будемо розглядати та уточнювати операційну (алгоритмічну) семантику комунікативних інформаційних систем.

1. Комунікативні інформаційні системи та їхні моделі

Нагадаємо деякі поняття, пов'язані з комунікативними інформаційними системами та їхніми моделями. Необхідні деталі можна знайти в [2]. Як зазначено вище, КІС це конструктивна модель комунікативних систем (КС) – центрального поняття комунікативної інформатики. Складовими КС є :

1) предметна область (ПрО) з певною сукупністю інформаційних об'єктів та співвідношень між ними;

2) суб'єкт-ініціатор (далі *ініціатор*), який формує і передає суб'єкту-виконавцю певну вхідну та приймає від нього певну вихідну інформацію;

3) суб'єкт-виконавець (далі *виконавець*), який приймає вхідну інформацію, аналізує її, обробляє та повертає як вихідну ініціатору.

Головне призначення КС полягає у здійсненні *комунікативних процесів*, кожний з яких розгортається в певному часовому просторі та в результаті встановлює зв'язок між певними об'єктами предметної області системи.

За своєю роллю у процесі ці об'єкти розподіляються на вхідні та вихідні. З перших розпочинається процес, останніми закінчується. Комунікативний процес складається з етапів, які разом утворюють його життєвий цикл. Останній розпочинається ініціатором, який формує й передає виконавцю за допомогою засобів зв'язку повідомлення, що містить *запит* на обробку певних вхідних інформаційних об'єктів (не змішувати з запитами в інформаційно-пошуковими системах; тут термін "запит" трактується як загальне поняття з більш широким змістом). Запит окрім останніх містить інформацію про мету обробки, яка може бути сформульована неявно – як вимоги до очікуваних вихідних об'єктів або явно – як детальний опис їх отримання за вхідними об'єктами. В обох випадках мета декларує певне співвідношення між об'єктами і сама подається у вигляді спеціального інформаційного об'єкта – програми. Підкреслимо, що програма не обов'язково є імперативною. Вихідні об'єкти є результатом виконання обробником певної *внутрішньої процедури*, яка реалізує запит.

Зазвичай виконавець має власні інформаційні об'єкти – *внутрішні* (сукупності яких утворюють його стани), на відміну від *зовнішніх* – тих, що фігурують в предметній області. Тому сам запит містить не вхідні об'єкти виконавця, а тільки їхні

прообрази, які після кодування утворюють початковий стан виконавця. Це стосується й вихідних внутрішніх об'єктів поданих у заключному стані процесу обробки. Вони потребують декодування.

Комунікативна платформа накладає певні обмеження на інформаційні системи як модель КС, а саме: вони мають бути обов'язково дескриптивними і не просто дескриптивними – а конструктивними. Дескриптивність КІС означає, що до складу її елементів входять дескриптивні системи (ДС), в яких подаються (описуються) усі її інформаційні об'єкти: вхідні і вихідні дані, запити та внутрішні процедури. Мова йде не про одну, а, як правило, про декілька ДС, тому що інформаційні об'єкти різних суб'єктів системи потребують різних дескриптологічних засобів. Як мінімум – мову специфікації запитів для ініціатора і внутрішню мову для виконавця. Остання описує внутрішні інформаційні об'єкти та внутрішні процедури, які задають правила для реалізації кожної з програм мови специфікацій. Внутрішні мови на відміну від мов специфікацій обов'язково є імперативними.

Конструктивність КІС означає, що її мова (мови) специфікацій та внутрішня мова є фінітними, тобто усі об'єкти в них – скінченно подані, а внутрішня мова ще і обов'язково інтенсіональною. Мови специфікацій та внутрішня мова конструктивних КС самі називаються *конструктивними,* внутрішні мови – *алгоритмічними*, а внутрішні процедури – *алгоритмами.* Наголосимо, що для конструктивності довільної мови специфікацій недостатньо тільки її фінітності – необхідна також наявність алгоритмічної мови, яка задає реалізацію усіх її запитів.

В композиційній моделі КІС ініціатор та виконавець подаються як спеціальні відповідно вхідна й вихідна формалізовані композиційні $\Omega-$системи, пов'язані між собою функціями кодування та декодування об'єктів.

2. Операційна модель КІС

Мета операційної моделі КІС – конкретизувати семантику її алгоритмічної мови. Це вимагає уточнення таких загальних понять як обчислювальна процедура, обчислення та алгоритм. Ці поняття відносяться до фундаментальних понять математики (зокрема, обчислювальної її гілки) та математичної логіки. В процесі їхньої експлікації ми хочемо відійти від прийнятої у теорії алгоритмів практики визначати якийсь конкретний клас алгоритмів (машин Тюрінга, алгоритмів Маркова тощо) і, посилаючись на його універсальність та тезу Чорча, пов'язувати з ним загальне поняття алгоритму та обчислення. Такий підхід, оснований на моделюванні, достатній, можливо, для потреб математичної логіки та конструктивної математики, є занадто спеціалізованим і недостатнім для розбудови основ інформатики та програмування та й, на наш погляд, і для самої математики в цілому. Адже конкретні приклади, які б вони цікаві не були, не можуть замінити загального поняття.

Щоб уточнити загальне математичне поняття обчислювальної процедури, спробуємо з'ясувати його складові. Якщо звернутися до окремо взятого обчислення, то побачимо не одноразовий акт, а серію їх. Розпочавшись у певний фіксований момент часу з заданою початковою інформацією на вході, обчислення розгортається у часі і супроводжується виконанням певних цілеспрямованих елементарних дій з визначенням результату. Тому будь-яке обчислення передбачає, насамперед, наявність певного *обчислювального простору* Π, який складається з лінійно-впорядкованого часового простору Γ і сукупності станів S. Часовий простір додатково задовольняє умові градації: для будь-якого моменту часу τ існують його безпосередньо попередній та наступний моменти – відповідно верхня границя його нижнього конусу

$$^{-}\tau = sup\{x : x \le \tau\}$$

та нижня границя верхнього конусу

$$\tau^{+} = inf\{x : \tau \le x\}.$$

Вважається, що у часовому просторі немає найменшого та найбільшого моментів часу.

Стани представляють у просторі інформаційну складову обчислень. Упо-

рядковані пари $(t,s)\in\Gamma\times S$ називаються *точками* або *конфігураціями* обчислювального простору Π. Необхідні також сукупність $\Delta=\{f_i:S\to S\,|\,i\geq 0\}$ *елементарних операцій* на станах та *функція керування* або *переходів* простору, яка регулює порядок застосування операцій в обчисленнях. Вона пов'язує з кожною конфігурацією обчислення одне або кілька можливих елементарних перетворень його поточного стану. Покладемо її як таку, що має вигляд

$$\delta:\Gamma\times S\to\Delta.$$

У загальному випадку функція керування може бути більш складною як, наприклад, у моделях з глобальним та локальним часовими просторами [4], де вона може задавати й перетворення локального часового компонента простору. Якщо локальний час збігається з основним, то це дає можливість повертати обчислення назад і продовжувати його іншим чином. Функція керування може також розпаралелювати процеси обчислень. Функція керування може бути багатозначною, але тоді мати скінченний індекс, тобто для кожної пари її аргументів має існувати тільки скінченна кількість варіантів значень.

Обчислювальною процедурою у просторі Π із сукупністю елементарних перетворень Δ і функцією керування δ називається трійка $P=\langle\Pi,\Delta,\delta\rangle$.

Кожна обчислювальна процедура породжує певну сукупність обчислювальних процесів у просторі Π. *Обчислення* $p:\Gamma_\tau\to S$ за процедурою P із початком у момент часу τ і початковим станом s_0 визначається рекурентно: $p\tau=s_0$ і для всіх $\varsigma>\tau$ $p\varsigma=f_\varsigma(p\varsigma^-)$, де $f_\varsigma\in\delta(\varsigma,p\varsigma^-)$. Функція керування за поточним моментом часу ς і станом $p\varsigma^-$ у попередній момент часу визначає сукупність елементарних операцій, які можуть бути застосовані для отримання стану процесу в момент ς. Будемо говорити, що обчислення p у момент часу ς *переходить* від попереднього стану до наступного стану $p\varsigma$. Варіантів таких переходів може бути декілька, оскільки функція переходу багатозначна. Насправді їх може бути: а) два і більше (у випадку *недетермінованих процедур*); б) рівно один (у випадку *детермінованих* процедур); в) жодного (обчислення зупиняється).

Зазвичай інтерес становлять не всі можливі обчислення в процедурі, а насамперед ті, що розпочинаються в певний фіксований момент часу з певних виділених вхідних станів і закінчуються певними виділеними вихідними станами. Тому введемо поняття *ініціальної* обчислювальної процедури $P(S_0,S_{\text{fin}},\tau_0)$ над простором Π як четвірки $(P,S_0,S_{\text{fin}},\tau_0)$, де P – певна процедура над простором Π, $S_0\subseteq S$ та $S_{\text{fin}}\subseteq S$ – виділені підмножини відповідно *вхідних* і *вихідних* станів, $\tau_0\in\Gamma$ – початковий момент часу. Далі під процедурою будемо розуміти саме ініціальний її варіант. *Результативним* назвемо обчислення p у процедурі P, що розпочинається в момент часу τ_0 із певного початкового стану $p\tau_0=s_0\in S_0$, закінчується у певний момент часу у вихідному стані, а всі його проміжні стани – не вихідні. Для результативного обчислення p останній його стан $p\tau$ називається *заключним* і є *результатом*. Результат позначається $P^*(s_0)$ і є єдиним, якщо функція керування однозначна. У деяких випадках за результат обчислення береться не сам заключний стан, а значення на ньому певного *результуючого* часткового відображення $Val:S_{\text{fin}}\to B$. В деяких випадках вважають заключними і ті скінченні обчислення, на останніх станах яких не визначена функція керування.

Якщо в результативному обчисленні зняти умову, щоб усі проміжні стани були не вихідними, то таке обчислення називатимемо *гіперрезультативним*. Його вихідні стани можуть зустрічатися й серед проміжних, а результат позначається $P^{**}(s_0)$.

Коли для певного обчислення значення результуючого відображення *Val* на заключному стані не визначене, то воно

вважається *безрезультатним*. Безрезультатними є зазвичай і нескінченні обчислення. Але у деяких випадках можна і тут передбачити механізм для встановлення результату. Адже саме результат є квінтесенцією будь-якого обчислення як процесу. Одним із таких механізмів може бути, наприклад, встановлення на станах певного часткового порядку, і за результат нескінченного обчислення – взяття найменшої верхньої границі його станів. Цей шлях може бути цікавим як доповнення апроксимаційної платформи Д. Скотта [5], яка сама по собі є чисто екстенсіональною і непроцедурною.

Скажемо також про варіанти закінчення обчислень з використанням часових критеріїв. Обчислення можуть закінчуватись (примусово) за досягненням певного моменту часу або тільки після нього чи вичерпання його ліміту тощо.

З кожною процедурою $P(S_0, S_{fin}, \tau_0)$ пов'язані дві відповідності на вхідних станах $P^*: S_0 \to S_{fin}$ та $P^{**}: S_0 \to S_{fin}$, значення яких на початковому стані $s_0 \in S_0$ складають результати всіх результативних і гіперрезультативних обчислень із початком у момент τ_0 на s_0.

Про відповідності P^* та P^{**} говорять, що їх *обчислює* процедура P. Функції P та Q називаються *еквівалентними* ($P \equiv Q$), якщо відповідності P^* та Q^*, які вони обчислюють, збігаються. Стан функції $P(S_0, S_{fin}, \tau_0)$ назвемо *припустимим*, якщо він зустрічається серед станів деякого її результативного обчислення. Покладемо $S_{acc} \subseteq S$ – підмножину всіх припустимих станів процедури P. Для відповідностей, які обчислює процедура стани за межами підмножини S_{acc} зайві. Нехай $P'\left(S_0', S'_{fin}, \tau_0\right)$ – ініціальна функція над простором Π' із множиною станів S_{acc} і обмеженими на Π' елементарними операціями, функцією переходу та вхідними й вихідними станами процедури P. Очевидно, що функції P та P' еквівалентні. Функція, усі стани якої припустимі, називається *зведеною*.

Багато прикладів процедур і саме процедур-неалгоритмів можна знайти вже на поверхні, в класичних задачах алгебри й геометрії (неалгоритмічність процедур тут пов'язана з неконструктивністю їхніх станів, що складаються з актуально нескінченних дійсних чисел та фігур на площині). Так, задачі з планіметрії на побудову за допомогою циркуля й лінійки є фактично задачами на пошук тих чи інших процедур. Станами в них є сукупності фігур на площині, які можуть бути побудовані за допомогою циркуля й лінійки та операцій виділення довільної точки площини, точок перетину фігур, операцій іменування точок і фігур, а також умовних варіантів подібних операцій. Часовий простір – дискретний і збігається з натуральними числами. Приклади таких класичних задач є задача на побудову середини відрізка на площині (з відповідною процедурою розв'язку) або задача пошуку найбільшої спільної міри двох відрізків (алгоритм Евкліда) тощо. Підкреслимо, що ці приклади не штучні, а широко відомі.

Цікава ситуація виникає у випадку, коли результат розв'язку обчислювальної задачі не один, а їх декілька. Якщо розв'язок такої задачі забезпечує недетермінована процедура, то його завжди можна детермінізувати, якщо перейти до гіперобчислень. Наприклад, нехай для розв'язку задачі виконавець використовує дошку й крейду і необхідно побудувати два різні розв'язки якоїсь задачі на побудову. Тоді після побудови першого дошка витирається і процедура повертається в один із попередніх станів, але не в попередню конфігурацію – момент часу інший. Завдяки цьому процес побудови другого розв'язку може бути просто зсунутим в часі і продовжений детерміновано.

Процедури, керування діями в яких реально залежить від часової компоненти, будемо називати *темпоральними* на відміну від *автоматних*, коли таке керування не залежить від часу. Класичними прикладами автоматних процедур є скінченні й магазинні автомати, машини Тьюрінга, деякі ігрові числення тощо. Але в останніх

часові фактори можуть відігравати і важливу роль. В таких випадках вони набувають ознак темпоральних процедур. Так, у багатьох іграх важливою є черговість ходу. У грі в шахи (шашки), наприклад, у всі непарні моменти часу ходять білі, а в парні – чорні. Однак тут є й інші більш принципові обставини, пов'язані з часом – рокіровка короля залежить від того, чи рухався він до цього моменту. У деяких моделях на стратегію вибору чергового ходу може впливати обмеженість часу відведеного на гру (гра в цейтноті).

З найбільш відомих алгоритмічних моделей темпоральними за природою є дискретні перетворювачі В.М. Глушкова, але вже в моделі з локальним часом, яку ми тут не розглядаємо.

Щоб отримати загальне поняття алгоритму, нам залишилося конструктуризувати обчислювальні процедури, тобто вибрати для них алгоритмічну мову. Для цього спочатку розглянемо абстрактну модель виконавців КС у вигляді, так званих, обчислювальних $\Omega-$систем.

Сигнатурою називається шестірка

$$\Omega=(\Omega_0,\Omega_1,\Omega_2,\Omega_3,\Omega_4,\Omega_5),$$

де Ω_0, $\Omega_1=\{t_i: i\geq 0\}$, $\Omega_2=\{s_i: i\geq 0\}$, $\Omega_3=\{f_i: i\geq 0\}$, $\Omega_4=\{p_j: j\geq 0\}$, $\Omega_5=\{\delta_k: k\geq 0\}$ – відповідно сукупності часових констант та змінних, предметних змінних, унарних функціональних символів, предикатних та керувальних символів.

Для інтерпретації сигнатури Ω в певному обчислювальному просторі $\Pi=(\Gamma,S)$ необхідно вибрати в ньому конкретні початкові моменти часу для часових констант, часовим та предметним змінним поставити у відповідність їхній типи – сукупності Γ та S, функціональним, предикатним та керувальним символам – відповідно елементарні операції, предикати на станах та функції керування простору. При цьому предикатним символам p_0 та $p_1=p^*$ відповідають сукупності вхідних та вихідних станів. Таке співставлення I сигнатурним символам їхніх значень називається *інтерпретацією* сигнатури Ω в просторі Π. Значення сигнатурного символа c при інтерпретації I домовимося позначати c^I. Інтерпретація I може бути частковою. Наприклад, якщо часові константи та предикатні символи p_0 та p^* непроінтерпретовані, то сигнатура називається *неінеціалізованою*. В інших випадках можуть залишатись непроінтерпретованими деякі змінні, а також частина імен функцій та предикатів.

Проінтерпретовані сигнатури називаються *обчислювальними* $\Omega-$*системами* і позначаються $C=(\Pi,\Omega,I)$ або Ω_Π^I. Кожному проінтерпретованому керувальному символу δ_k^I такої системи C відповідає певна ініціальна обчислювальна процедура у просторі Π.

Як зазначалося вище, комунікативна платформа накладає певні обмеження на КС та їхні моделі, а саме – вони мають бути конструктивними. Це стосується і обчислювальних $\Omega-$систем як моделей виконавців. За означенням, має бути фінітна інтенсіональна ДС L_Ω для подання усіх елементів обчислювальних $\Omega-$систем: моментів часу, станів, елементарних операції та функцій керування. Алфавіт мови L_Ω обов'язково включає сигнатуру Ω, а головними її об'єктами є *схеми алгоритмів*, які подають функції керування процедур. Інтенсіонал конкретної схеми алгоритму фінітно описує значення функції керування для кожної припустимої конфігурації її процедури.

Двійка $A=(C,L_\Omega)$ називається *формалізованою обчислювальною* $\Omega-$*системою,* а проінтерпретовані схеми алгоритмів системи A – ***абстрактними алгоритмами.*** Скорочено перші будемо називати також *алгоритмічними* $\Omega-$*системами*, другі – A*-алгоритмами* чи просто *алгоритмічними системами* та відповідно *алгоритмами,* якщо відомо яка система A мається на увазі.

Як і процедури, абстрактні алгоритми можуть бути детермінованими й недетермінованими. Відповідності, що обчислюються ними називаються *алгоритмічно обчислювальними*.

Наведемо основні властивості

A-алгоритмів, які безпосередньо випливають з їхнього означення.

Масовість. A-алгоритм може бути застосований до будь-якого вхідного стану.

Темпоральність. Обчислення за A-алгоритмом відбуваються в глобальному часовому просторі, елементи якого можуть впливати на вибір перетворень поточного стану.

Елементарність. У кожний момент часу в обчисленні виконується одна елементарна операція з фіксованої сукупності таких операцій.

Визначеність. Порядок застосування елементарних операцій в обчисленні не довільний, а визначається функцією переходу за поточною конфігурацією обчислення.

Результативність. У кожному A-алгоритмі є механізм завершення обчислень.

Фінітність. Скінченність подання A-алгоритму.

Релятивність. Відносність поняття A-алгоритму. Конструктивність A-алгоритму прямо залежить від його алгоритмічної мови. Алгоритм в одній алгоритмічній системі не обов'язково буде алгоритмом в іншій. Інший аспект релятивності A-алгоритмів пов'язаний з частковістю інтерпретацій в алгоритмічних системах, а саме, у деяких випадках символи елементарних операції (усі чи їхня частина) можуть бути не проінтерпретованими. Такі символи операцій називаються *оракулами*, а відповідні алгоритмічні системи – *системами з оракулами*. Оракули можна трактуватися по різному, але в основному – як операції, дія яких невідома в межах алгоритмічної системи. Це може бути, наприклад, тимчасово чи навпаки – дія відома, але тільки поза межами системи або взагалі не відома.

Як бачимо, A-алгоритмам притаманні всі основні властивості класичних числових і словарних алгоритмів. Однак з'явилися й нові специфічні риси – темпоральність та релятивність.

Алгоритмічні Ω-системи можуть виступати моделями виконавців – вихідними системами в КІС. При цьому внутрішніми процедурами виступають A-алгоритми, станами – інформаційні поля, а внутрішньою мовою – алгоритмічна мова. Алгоритмічні вихідні системи називають ще *операційними*, щоб підкреслити інтенсіональний характер внутрішніх процедур як алгоритмів у противагу екстенсіональній природі операторів оновлення в композиційній моделі вихідних систем.

Четвірка $\boldsymbol{S} = (\boldsymbol{S}_{in}, \boldsymbol{S}_{out}, c, d)$, де $\boldsymbol{S}_{in}$ та $\boldsymbol{S}_{out}$ – деякі вхідна та алгоритмічна вихідна системи, а c та d – відповідно функції кодування та декодування, називається **алгоритмічною (операційною)** моделлю комунікативних інформаційних систем**.**

У зв'язку з релятивністю виникає задача порівняння різних класів A-алгоритмів за потужністю. Будемо казати, що A-алгоритм A_1 з множиною станів S_1 є моделлю A-алгоритму A_2 з множиною станів S_2 відносно функції кодування $\varphi : S_2 \to S_1$, якщо для кожного із вхідних станів s A-алгоритму A_2 виконується $A_2^*(s) = \varphi^{-1}(A_1^*(\varphi s))$.

Нехай K_1 та K_2 – довільні класи A-алгоритмів над обчислюваними просторами $\Pi_1 = \langle \Gamma_1, S_1 \rangle$ та $\Pi_2 = \langle \Gamma_2, S_2 \rangle$, відповідно. Зафіксуємо певну функцію кодування $\varphi : S_2 \to S_1$. Кажуть, що клас A-алгоритмів K_1 *моделює* клас A-алгоритмів K_2, якщо для кожного A-алгоритму з K_2 у класі K_1 існує його модель. Якщо класи A-алгоритмів моделюють один одного відносно певних фіксованих функцій кодування, то вони називаються *еквівалентними* відносно цих функцій.

Фіксуючи конкретні обчислювальні простори Π та відповідні алгоритмічні мови, можна отримати весь спектр класичних алгоритмічних та ігрових систем (машини Тьюрінга, алгоритми Маркова й Колмогорова – Успенського, дискретні перетворювачі, трансдюсери, різні класи схем програм, формальні граматики й числення

Поста, ігрові числення – шахи тощо).

Усі перелічені класи класичних алгоритмів неігрових числень є автоматними, еквівалентними відносно достатньо простих функцій кодування і за Чорчем – *універсальними* (не стосується ігрових числень).

Для ілюстрації деяких можливостей операційної моделі наведемо один важливий клас абстрактних алгоритмів – табличні алгоритми [4].

Нехай часовий простір Γ ізоморфний адитивній групі Z цілих чисел і P – довільна процедура у просторі

$$\Pi = \langle Z, S \rangle .$$

Насправді, табличні алгоритми можуть бути визначені у будь-якому факторизованому часовому просторі. Факторизуємо функцію переходу δ за часовим аргументом і станами. Зафіксуємо деяку часову константу $k \geq 0$ і певне відношення еквівалентності π на станах із S. Будемо казати, що функція переходу δ періодична за модулем відношення π із періодом k (k*-періодична за модулем* π), якщо для будь-яких еквівалентних станів p та s і довільного моменту часу $t \in Z$ виконується $\delta(t, p) = \delta(t \pm k, s)$. Функція переходу δ просто *періодична за модулем* π, якщо вона k-періодична за модулем π для деякого k. Якщо еквівалентність π має скінченний індекс $|\pi|$ і розв'язні класи, то процедура з періодичною функцією переходу є абстрактним алгоритмом, який називається *табличним.*

Табличні алгоритми можна задавати двовимірними таблицями розміром $k \times |\pi|$: перший вимір відповідає числам від 0 до $k-1$, другий – класам еквівалентності π, а в клітинах таблиці розміщуються значення функції переходу.

Традиційні алгоритмічні системи є табличними як абстрактні алгоритми, а оскільки вони автоматні, то й 1 – періодичні (скінченні автомати, МП-автомати тощо). Наприклад, машина Тьюрінга є табличним 1 – періодичним алгоритмом: два її стани еквівалентні, якщо їхні внутрішні стани та стани робочих комірок збігаються.

В роботі [4] показано, що для табличних алгоритмів мають місце теореми про регулярний аналіз та синтез.

Висновки

В роботі здійснена спроба уточнити загальне поняття обчислювальної процедури та абстрактного алгоритму. На їхній основі запропонована операційна модель КІС. Наведено поняття табличного алгоритму як загальної моделі для усіх класичних алгоритмічних систем.

1. *Зубенко В.В.* Програмування: Навчальний посібник.– К.: ВПЦ Київський університет, 2011. – 625 с.
2. *Зубенко В.В.* Про комунікативну інформатику // Вісник КНУ ім. Тараса Шевченка, сер. фіз.-мат. наук, вип. 4. – 2012.
3. *Зубенко В.В.* Композиційна модель комунікативних інформаційних систем // Вісник КНУ ім. Тараса Шевченка, сер. Кібернетика, вип. 13. – 2013.
4. *Зубенко В.В.* Темпоральні процедури та алгоритми // Проблеми програмування. – 2006. – № 2-3. – С. 53–59.
5. *Скотт Д.* Набросок математической теории вычислений // Кибернетический сборник. – М.: Мир, 1977. – Вып. 24. – С. 107–121.

Одержано 24.05.2013

Про автора:

Зубенко Віталій Володимирович
доцент, кандидат фізико-математичних наук, доцент.

Місце роботи автора:

Київський національний університет
ім. Тараса Шевченка
01601, Київ, вул. Володимирська, 64/13.
Тел.: (044) 259 0519.
E-mail: vvz@unicyb.kiev.ua