МОДЕЛЬ ЭКВИВАЛЕНТНЫХ ПРЕОБРАЗОВАНИЙ АЛГОРИТМОВ В САМООРГАНИЗУЮЩЕЙСЯ ИНФОРМАЦИОННОЙ СИСТЕМЕ

EQUIVALENT ALGORITHM TRANSFORMATIONS MODEL IN A SELF-ORGANIZING INFORMATION SYSTEM

A. Shalaev

Annotation

The paper discusses equivalent and homomorphic transformations of algorithms that provide generation of new algorithms with the required properties. The article proposes to replace the original algorithms (or certain parts of them) with algorithms that fit the given specific task conditions (con–text) more accurately in order to improve the overall quality and efficiency. The paper defines the conditions of semantic equivalence of algorithms and rules for their replacement.

Keywords: algorithm, equivalence of algorithms, homomorphism of algorithms, algorithm scheme, self–organizing information system.

Шалаев Александр Александрович
Аспирант,
Пензенский государственный университет

Аннотация

Рассматриваются эквивалентные и гомоморфные преобразования алгоритмов, обеспечивающие формирование алгоритмов с требуемыми свойствами. Предлагается для повышения качества и эффективности алгоритмов использовать замену алгоритмов или их частей ал–горитмами, более точно соответствующими конкретным условиям (контексту) решения задачи. Определены условия семантической эквивалентности алгоритмов и правила их замены.

Ключевые слова:

Алгоритм, эквивалентность алгоритмов, гомоморфизм алгоритмов, схема алгоритма, самоорганизующаяся информационная система.

Способность к самомодификации [1], обеспечивающая изменение алгоритмов решения задач и способов обработки данных, предоставляет самоорганизующейся информационной системе (СИС) очень мощный механизм адаптации к изменениям внешней среды (решаемым задачам и запросам пользователей) и внутренней организации системы. К самомодификации СИС относятся процессы формирования алгоритмов с требуемыми свойствами, замена текущего алгоритма алгоритмом, более соответствующим конкретным условиям (контексту) решения задачи, и замена в текущем алгоритме некоторых частей новыми частями, обеспечивающими решение подзадач с требуемым качеством и эффективностью. Так как механизм самомодификации СИС должен обеспечивать решение одних и тех же задач в разных условиях, то в качестве формальной основы самомодификации целесообразно использовать эквивалентные преобразования алгоритмов.

Эквивалентность алгоритмов определяется следующим образом [2]:

алгоритмы p_1 $(y_1 = p_1(x_1))$ и p_2 $(y_2 = p_2(x_2))$ являются эквивалентными $(p_1 \equiv p_2)$,

если они для всех $x_1 \in X_1$ и $x_2 \in X_2$ при $x_1 = x_2$ выдают результат $y_1 = y_2$ или $|y_1 - y_2| < \varepsilon$, где ε – допустимые отклонения результатов выполнения алгоритмов, обусловленные точностью вычислений или разностью подходов к решению задачи (например, точные или итерационные методы). Следовательно, у эквивалентных алгоритмов совпадают области определения и области значений, т.е. $X_1 = X_2 = X$ и $Y_1 = Y_2 = Y$. Однако эти алгоритмы могут иметь другие разные характеристики (например, выполняться за разное время, требовать разные ресурсы для реализации и т.д.).

С другой стороны, эквивалентность алгоритмов p_1 и p_2 можно рассматривать как равенство отношений

$$\{<x, y_1>\} = \{<x, y_2>\},$$

реализуемых отображениями

$$p_1: X \to Y_1 \text{ и } p_2: X \to Y_2.$$

Поэтому взаимозаменяемость эквивалентных алгоритмов соответствует двум взаимообратным гомоморфизмам $h: p_1 \to p_2$ и $h^{-1}: p_2 \to p_1$.

Эквивалентное преобразование алгоритма заключается в замене алгоритма p_1 на эквивалентный ему алгоритм p_2 или в замене в алгоритме p_1 компонентов p_{1i} на эквивалентные им компоненты p'_{1i} и наоборот.

Допустимыми преобразованиями алгоритмов в рам–

ках модели эквивалентных преобразований являются замена алгоритмов и их частей (функционально законченных последовательностей операторов) на алгоритмы известные системе или на алгоритмы, реализующие частные способы решения задач, формируемые на основе общих методов решения задач, известных системе.

Можно рассматривать два вида эквивалентных преобразований алгоритмов:

- использование разных методов решения задачи для повышения точности решения задачи, получения решения за меньшее число шагов (меньшее время), при других значениях аргументов и т.д., что вызывает необходимость замены одного алгоритма другим. В этом случае эквивалентность алгоритмов определяется эквивалентностью методов решения задач;
- эквивалентные преобразования алгоритмов при реализации одного и того же метода решения задачи, использующие различные структуры и способы упорядочивания данных для представления обрабатываемых объектов, а также различные последовательности действий (операторов).

Вследствие этого СИС могут быть известны достаточно эффективные способы эквивалентных преобразований алгоритмов, которые она успешно сможет использовать в своем функционировании.

Хотя алгоритм – это целостный объект, однако, в подавляющем большинстве случаев, он является сложным (составным, представляющим собой последовательность операторов). Наличие последовательности операторов позволяет структурировать алгоритм путем разбиения его на функционально законченные подпоследовательности, решающие определенные подзадачи, совокупность которых позволяет решить задачу в целом.

Для формализации эквивалентных преобразований алгоритмов необходимо определить понятие схемы алгоритма [3–5].

Схема алгоритма – это математическая модель алгоритма, в которой с достаточной степенью детализации отражаются строение алгоритма (совокупности последовательностей операторов) и взаимодействие его компонентов. В схеме алгоритма конкретные последовательности операторов, операции и функции заменяются абстрактными функциональными и предикатными символами.

Схему алгоритма целесообразно характеризовать базисом и структурой схемы [4]. Базис фиксирует символы, из которых строятся схемы, указывает их роль (составной оператор, простой оператор, входные и выходные значения, функциональные символы и др.), задает вид выражений и операторов схем.

Полный базис схем содержит четыре непересекающихся, счетных множества символов:

1) –

$$X = \{x_1, x_2, \dots, x_{n_x}, y_1, y_2, \dots, y_{n_y}, a_1, a_2, \dots, a_{n_a}\}$$

– множество входных и выходных значений, а также числовых и символьных констант ;

2) –

$$F = \{g_1, g_2, \dots, g_{n_g}, p_1, p_2, \dots, p_{n_p}, \ f_1, f_2, \dots, f_{n_f}, q_1, q_2, \dots, q_{n_q}\}$$

– множество функциональных символов g_i, представляющих управляющие операторы, арифметические и логические операции, множество символов f_k, представляющих составные и простые операторы, множество символов p_j, представляющих функциональное соответствие между входными и выходными значениями абстрактных задач, и множество символов q_e, представляющих отношения (отображения, функции) между входными и выходными значениями конкретных задач;

3) – $C = \{c_1, c_2, \dots, c_{n_c}\}$

– множество символов, представляющих логические условия;

4) – {*task, method* и т.д.}
– множество специальных сим волов,
где n_i – количество символов вида i.

Структура схемы алгоритма будет представляться последовательностью символов базиса, а составные операторы – отдельными (обособленными) схемами алгоритмов.

Тогда схема алгоритма $p(x)$ будет являться совокупностью структуры алгоритма и обособленных схем алгоритмов составных операторов вида:

$$\begin{aligned} &task(idz, x, y, q, c, (\mu(x, y) \mid (task1(x, y1, q_1), task_2(x_2, y_2, q_2), \dots, taskm(xm, y, qm)));\\ &[method(id\mu, x', y', f, p, c');\\ &p(p_1(x', y''), p_2(x'', y'''), \dots, p_n(x''', y'));] \end{aligned} \qquad (1)$$

где

task(...) – структура алгоритма, представленная описанием задачи и методом решения или последовательностью ее подзадач;

method(...) – метод решения задачи;

p_i – составной оператор, задающий алгоритм решения задачи или подзадачи;

id_z, id_μ – идентификаторы задачи и метода;

x, y, x', y' – входные и выходные значения задачи и метода, удовлетворяющие условиям $x \subseteq x'$, $y \subseteq y'$;

c – условия (контекст) решения задачи;

μ – метод решения задачи с идентификатором $id\mu$, позволяющий вычислять решение задачи;

c' – характеристики решения задачи, обеспечиваемые методом μ (например, точность, время, необходимый объем памяти и др. характеристики решения задачи);

q, f – правила соответствия входов и выходов в задаче и методе.

При этом схема алгоритма простой задачи задается способом решения μ, а сложная задача – последовательностью подзадач более низкого уровня *taski* (метод будет указан у всех простых подзадач).

Между компонентами q, f и p установлены следующие отношения:

$$q \equiv f \cong p,$$

где

$f \cong p$ – отношения системного изоморфизма и системного полиморфизма между абстрактной задачей и алгоритмом ее решения, определяющие степень их необходимого подобия и допустимого разнообразия [6].

Пример 1: формальное описание программной реализации решения квадратного уравнения

$ax^2 + bx + c = 0$ для общего случая.

$$\text{task}(1, x = (a,b,c,), y = (x_1, x_2), q = (ax_1^2 + bx_1 + c = 0,\ ax_2^2 + bx_2 + c = 0), c = (ax^2 + bx + c = 0), \mu_1);$$

$$\text{method}(1, x' = (a,b,c,), y' = (x_1, x_2), f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0), p, c' = (ax^2 + bx + c = 0));$$

```
p = (var discriminant = b*b-4*a*c;
if(discriminant < 0) Console. WriteLine
("Нет действительных корней!");
else if(discriminant > 0)
}
else
{
var sqrt=System.Scrt(discriminant);
x1 = (–b + sqrt) / (2 * a);
x2 = (–b – sqrt) / (2 * a);
Console.WriteLine ("Два действительных
корня : x1 = {0 : f3} || x2 = {1 : f3}", x1, x2);
}
else
{
x1 = x2 = (–b + System.Match.
Scrt(discriminant)) / (2 * a);
Console.WriteLine ("Два одинаковых
корня : {0 : f3}", x1);
})
```

В модели (1) сложная задача в общем случае представлена последовательностью подзадач, однако она может иметь и более сложную структуру, сформированную с использованием операторов управления, соответствующих правилам используемого языка программирования:

1) **условный оператор**

$g_1(c, task_1, task_2)$ if (c) {$task_1$} else {$task_2$}

где

c – условие перехода к задачам:

c = true $task_1$, c = false $task_2$.

2) **оператор выбора**

$g_2(c, a1{:}task_1, a_2{:}task_2, ..., an{:}task_n[, default: task_{n+1}])$ switch (c) {case a_1: $task_1$; break; case a_2: $task_2$; break; ...; case a_n: $task_n$; break; [default: $task_{n+1}$]}

где

c – параметр, по значениям a_i которого выбирается соответствующая задача;

[...] – часть по умолчанию не обязательна и может быть опущена.

3) **цикл с параметром**

$g_3(c \equiv i =$, task(x_0, y_0)) ≡ for ($i = a_1$; $i\ \theta\ a_n$; i += s) {task(x_i, y_i)}

где

c – условие перебора значений от a_1 до a_2 с шагом s (при $s > 0$ операция θ есть "<=", а при $s < 0$ операция есть ">=").

4) **цикл с предусловием**

$g_4(c, task)$ while (c) {task; $p(_c)$;}

где

c – условие повторения выполнения задачи task;

$p(c)$ – оператор модификации условия в процессе исполнения цикла.

5) **цикл с постусловием**

$g_5(c, task)$ do {$task; p(c)$;} while(c);

где

c – условие повторения выполнения задачи task;

$p(c)$ – оператор модификации условия в процессе исполнения цикла.

Основываясь на модели (1) можно разработать модель эквивалентных преобразований алгоритмов, базирующуюся на семантической эквивалентности $q \equiv f$,

выражаемую условием:

$((x_1 = x) \quad (x_1 \in task_1(x_1, y_1, q_1))) \& ((y = y_m) \& (y_m\ task_m(x_m, y_m, q_m)))$

$((q \equiv q_1 \cdot q2 \cdot ... \cdot q_m) \& q_1 \in task_1(x_1, y_1, q_1) \& q_2 \in task_2(x_2, y_2, q_2) \& ... \&$

$q_m \quad task_m(x_m, y_m, q_m))$.

Тогда модель эквивалентных преобразований (взаимозаменяемость) алгоритмов p и p' можно задать в виде:

$$task(x, y, q) \equiv task'(x', y', q') \Leftrightarrow (x = x') \& (y = y') \& (q \equiv q'). \quad (2)$$

Реализация модели (2) возможна следующими способами:

1) Выбор метода решения: если в простой задаче не указан требуемый метод решения, то конкретный метод решения может быть выбран из возможных методов решения данной задачи в зависимости от контекста ее решения.

2) Замена задачи task на task', где task' решает эквивалентную задачу.

3) Замена в сложной задаче task подзадачи taski на подзадачу taski' в соответствии со способом 1.

Таким образом, эквивалентные преобразования алгоритмов могут иметь место и осуществляться путем замены задач или составляющих их подзадач, однако существенные изменения характеристик алгоритмов при данном подходе будут сильно ограничены вследствие равенства входных и выходных значений и очень близких способов решения задачи. Поэтому для обеспечения более широких изменений характеристик алгоритмов решения задачи целесообразно рассматривать только направленную замену одного алгоритма другим (например, только гомоморфизм $h: p \quad p'$).

Гомоморфизм алгоритмов

– определяется следующим образом:

алгоритм p_1 $(y_1 = p_1(x_1))$ является гомоморфным алгоритму p_2 $(y_2 = p_2(x_2))$, если для всех $x_1 \in X_1$

при $x_1 = x_2$ и $x_2 \quad X_2$ выдается результат $y_1 = y_2$ или $|y_1 - y_2| < \varepsilon$.

Следовательно, у гомоморфных алгоритмов области определения и области значений соотносятся как $X_1 \subset X_2$ и $Y_1 \subseteq Y_2$. Поэтому гомоморфные алгоритмы можно рассматривать как эквивалентные только в области определения X_1.

Гомоморфное преобразование алгоритма

–заключается в замене алгоритма p_1 на гомоморфный ему алгоритм p_2 в пределах области определения X_1 или в замене в алгоритме p_1 компонентов p_{1i} на гомоморфные им компоненты p_{1i} в пределах области определения $X_{1i} \subset X'_{1i}$.

При использовании гомоморфизма $h: p \to p'$ возможны три случая:

а) Замена алгоритма p на алгоритм p', решающего эквивалентную или более общую задачу. В этом случае правило семантической эквивалентности будет иметь вид:

$((x \subseteq x') \& (y \subseteq y') \& (q \equiv f)) \mid (((x \subseteq x'_1) \& (x'_1 \in task'_1(x'_1, y'_1, q'_1))) \& ((y \in y'_m) \& (y'_m \in task'_m(x'_m, y'_m, q'_m))) \& ((q \equiv q'_1 \quad q'_2 \quad ... \quad q'_m) \quad q_1 \quad task'_1(x'_1, y'_1, q'_1) \& q'_2 \in task'_2(x'_2, y'_2, q'_2) \& ... \& q'_m \in task'_m(x'_m, y'_m, q'_m)))$.

При этом модель эквивалентных преобразований алгоритмов будет иметь вид:

$$task(x, y, q) \to task'(x', y', q') \Leftrightarrow (x \subseteq x') \& (y \subseteq y') \& (|y - y| < \varepsilon) \& (q \equiv q'). \quad (3)$$

Реализация модели (3) возможна следующими способами:

1) Выбор метода решения: если в простой задаче не указан требуемый метод решения, то конкретный метод решения может быть выбран из возможных методов решения данной или более общей задачи в зависимости от контекста решения задачи.

2) Замена задачи task на task', где task' решает эквивалентную или более общую задачу.

3) Замена в сложной задаче task подзадачи taski на подзадачу taski' в соответствии со способом 1.

4) Замена в сложной задаче task подзадачи taski, вычисляющей сложную функцию, на циклическое выполнение подзадачи taski', рекуррентно вычисляющую эту функцию в соответствующих точках.

Пример 2:

– формальное описание замены алгоритма p, решающего квадратное уравнение

$ax_2 + bx + c = 0$ при $b = 2k$, на алгоритм p', решающий уравнение при любых значениях b.

1. Исходный алгоритм

task(5, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + 2kx_1 + c = 0, ax_2^2 + 2kx_2 + c = 0)$, $c = (ax^2 + bx + c = 0, b = 2k)$), μ_2);
method(2, $x' = (a, b, c)$, $y' = (x_1, x_2)$, $f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, p_2, $c' = (ax^2 + bx + c = 0$, b % 2 == 0));
p_2 = (var k = b / 2;
var discriminant = k * k - a * c;
if (discriminant < 0) Console.WriteLine("Нет действительных корней!");
else if (discriminant > 0)
{
var sqrt = System.Math.Sqrt(discriminant);
x1 = (-k + sqrt) / a;
x2 = (-k - sqrt) / a;
Console.WriteLine("Два действительных корня: x1 = {0:f3} || x2 = {1:f3}", x1, x2);
}
else if (discriminant == 0)
{
x1 = x2 = -k / a;
Console.WriteLine("Два одинаковых корня: {0:f3}", x1);
})

2. Новый (более общий) алгоритм

task(5, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, $c = (ax^2 + bx + c = 0, b = 2k)$, μ_1);
method(1, $x' = (a, b, c)$, $y' = (x_1, x_2)$, $f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, p, $c' = (ax^2 + bx + c = 0)$);
p = (var discriminant = b * b - 4 * a * c;
if (discriminant < 0) Console.WriteLine("Нет действительных корней!");
else if (discriminant > 0)
{
var sqrt = System.Math.Sqrt(discriminant);
x1 = (-b + sqrt) / (2 * a);
x2 = (-b - sqrt) / (2 * a);
Console.WriteLine("Два действительных корня: x1 = {0:f3} || x2 = {1:f3}", x1, x2);
}
else
{
x1 = x2 = (-b + System.Math.Sqrt(discriminant)) / (2 * a);
Console.WriteLine("Два одинаковых корня: {0:f3}", x1);
})

б) Замена алгоритма p на совокупность алгоритмов *p', p", ..., p'"* решающий эквивалентную или частные задачи. В этом случае правило семантической эквивалентности будет иметь вид:

$$(x = x' \cup x'' \cup \ldots \cup x''') \;\&\; (x' \in \text{task}'(x', y', q')) \;\&\; (x'' \in \text{task}''(x'', y'', q'')) \;\&\; \ldots \;\&$$
$$(x''' \in \text{task}'''(x''', y''', q''')) \;\&\; (y = y' \cup y'' \cup \ldots \cup y''') \;\&\; (y' \in \text{task}'(x', y', q'))) \;\&$$

В этом случае модель эквивалентных преобразований алгоритмов будет иметь вид:

$$\text{task}(x, y, q) \rightarrow \text{task}\left(x, y, g_2\left(\text{c, a}' : \text{task}', \text{a}'' : \text{task}'', \ldots[, \text{default} : \text{task}''']\right)\right) \Leftrightarrow$$
$$(x = x' \cup x'' \cup \ldots \cup x''') \;\&\; (y = y' \cup y'' \cup \ldots \cup y''') \;\&\; (q \equiv q' \equiv q'' \equiv \ldots \equiv q'''). \quad (4)$$

Реализация модели (4) возможна следующими способами:

1) Выбор метода решения: если в простой задаче не указан требуемый метод решения, то конкретный метод решения может быть выбран из возможных методов решения данной задачи в зависимости от контекста решения задачи.

2) Замена задачи task на task', где task' решает эквивалентную задачу.

3) Замена в сложной задаче task подзадачи task_i на подзадачу task_i' в соответствии со способом 1.

4) Декомпозиция в сложной задаче task подзадачи taski на систему подзадач task_i'... taski''' с использованием управляющих операторов g_1 или g_2.

Пример 3:

– формальное описание совокупности алгоритмов, решающих частные задачи и замещающих алгоритм *p* из примера 1.

task(1, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, $c = (ax^2 + bx + c = 0)$, $(g_1$(c = (b % 2 == 0), task(2), @task(3)));

task(2, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + 2kx_1 + c = 0, ax_2^2 + 2kx_2 + c = 0)$, $c = (ax^2 + bx + c = 0, b = 2k))$, μ_2);

method(2, $x' = (a, b, c)$, $y' = (x_1, x_2)$, $f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, p_2, $c' = (ax^2 + bx + c = 0$, b % 2 == 0));

p_2 = (var k = b / 2;
var discriminant = k * k - a * c;
if (discriminant < 0) Console.WriteLine("Нет действительных корней!");
else if (discriminant > 0)
{
var sqrt = System.Math.Sqrt(discriminant);
x1 = (-k + sqrt) / a;
x2 = (-k - sqrt) / a;
Console.WriteLine("Два действительных корня: x1 = {0:f3} || x2 = {1:f3}", x1, x2);
}
else if (discriminant == 0)
{
x1 = x2 = -k / a;
Console.WriteLine("Два одинаковых корня: {0:f3}", x1);
})

task(3, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, $c = (ax^2 + bx + c = 0)$, μ_1);

method(1, $x' = (a, b, c)$, $y' = (x_1, x_2)$, $f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, p, $c' = (ax^2 + bx + c = 0)$);

```
p =  ( var discriminant = b * b - 4 * a * c;
     if (discriminant < 0) Console.WriteLine("Нет действительных корней!");
     else if (discriminant > 0)
     {
        var sqrt = System.Math.Sqrt(discriminant);
        x1 = (-b + sqrt) / (2 * a);
        x2 = (-b - sqrt) / (2 * a);
        Console.WriteLine("Два действительных корня: x1 = {0:f3} || x2 = {1:f3}", x1, x2);
     }
     else
     {
        x1 = x2 = (-b + System.Math.Sqrt(discriminant)) / (2 * a);
        Console.WriteLine("Два одинаковых корня: {0:f3}", x1);
     })
```

В соответствии с примером 3 задачи task(1) и task(2) входят в один программный модуль, а задача task(3) должна быть реализована самостоятельным программным модулем.

в) Замена алгоритма p на алгоритм p', решающий частную задачу для активного подмножества значений $x' \subset x$. В этом случае правило семантической эквивалентности будет иметь вид:

$$((x' \subset x)\ \&\ (y' \subset y)\ \&\ (q \equiv f)) \mid (((x'_1 \subset x)\ \&\ (x'_1 \in \text{task}'_1(x'_1, y'_1, q'_1)))\ \&\ ((y'_m \subset y)\ \&\ (y'_m \in \text{task}'_m(x'_m, y'_m, q'_m)))\ \&\ ((q \equiv q'_1 \bullet q'_2 \bullet \ldots \bullet q'_m)\ \&\ q'_1 \in \text{task}'_1(x'_1, y'_1, q'_1)\ \&\ q'_2 \in \text{task}'_2(x'_2, y'_2, q'_2)\ \&\ \ldots\ \&\ q'_m \in \text{task}'_m(x'_m, y'_m, q'_m))).$$

При этом модель эквивалентных преобразований алгоритмов будет иметь вид:

$$\text{task}(x, y, q) \rightarrow \text{task}'(x', y', q') \Leftrightarrow (x' \subset x)\ \&\ (y' \subset y)\ \&\ (q \equiv q'). \qquad (5)$$

Реализация модели (5) возможна следующими способами:

1) Выбор метода решения: если в простой задаче не указан требуемый метод решения, то конкретный метод решения может быть выбран из возможных методов решения данной задачи в зависимости от контекста решения задачи.

2) Замена задачи task на task', где task' решает эквивалентную задачу.

3) Замена в сложной задаче task подзадачи taski на подзадачу taski' в соответствии со способом 1.

Использование трех моделей (3–5) эквивалентных преобразований (модель (2) является частным случаем модели (3)) предоставляет системе мощные средства преобразования алгоритмов, осуществляемых путем замены задач или составляющих их подзадач. В случае важности отдельной части задачи (некоторой последовательности исполняемых операторов) она всегда может быть представлена самостоятельной подзадачей, что не противоречит модели эквивалентных преобразований и обеспечивает требуемую точность преобразования алгоритмов (чувствительность алгоритмов к преобразованиям).

Пример 4:

формальное описание замещающего алгоритма p', решающего частную задачу для активного подмножества значений b ($b = 2k$), для алгоритма p из примера 1.

task(1, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, $c = (ax^2 + bx + c = 0)$, (g_1(c = (b % 2 == 0), task(2), p_1));

task(2, $x = (a, b, c)$, $y = (x_1, x_2)$, $q = (ax_1^2 + 2kx_1 + c = 0, ax_2^2 + 2kx_2 + c = 0)$, $c = (ax^2 + bx + c = 0)$), μ_2);

method(2, $x' = (a, b, c)$, $y' = (x_1, x_2)$, $f = (ax_1^2 + bx_1 + c = 0, ax_2^2 + bx_2 + c = 0)$, p_2, $c' = (ax^2 + bx + c = 0, b \% 2 == 0)$);

p_1 = Console.WriteLine("задача неразрешима при данном входе");

p_2 = (var k = b / 2;

```
var discriminant = k * k - a * c;
if (discriminant < 0) Console.WriteLine("Нет действительных корней!");
else if (discriminant > 0)
{
   var sqrt = System.Math.Sqrt(discriminant);
   x1 = (-k + sqrt) / a;
   x2 = (-k - sqrt) / a;
   Console.WriteLine("Два действительных корня: x1 = {0:f3} || x2 = {1:f3}", x1, x2);
}
else if (discriminant == 0)
{
   x1 = x2 = -k / a;
   Console.WriteLine("Два одинаковых корня: {0:f3}", x1);
})
```

Таким образом, разработанная модель эквивалентных преобразований алгоритмов позволяет осуществлять все допустимые преобразования алгоритмов на основе алгоритмов, известных системе, или разрабатываемых частных способов на основе общих методов решения задач. Причем модели эквивалентных преобразований (3, 4) преимущественно направлены на повышение корректности и надежности решения задач, а модель (5) – на повышение эффективности решения задач. Допустимо использование комбинации моделей эквивалентных преобразований алгоритмов. Например, в модели (4) в качестве вариантов могут использоваться модели (3) и (5).

ЛИТЕРАТУРА

1. Дрождин В. В. Организация и функционирование самоорганизующейся информационной системы // Новые ин–формационные технологии и системы: тр. X Междунар. научно–техн. конф. Пенза: Изд–во ПГУ, 2012. С. 231–233.

2. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987. 288 с.

3. Касьянов В.Н. Оптимизирующие преобразования программ. М.: Наука, 1988. 336 с.

4. Калайда В.Т. Теория вычислительных процессов. Томск: Изд–во ТУСУР, 2007. 131 с.

5. Крупноблочная схема программ. URL: http://pco.iis.nsk.su/grapp2/html/ReqtypetNamekrupnoblo3na%5E_shema_programm.htm (дата обращения: 14.07.2015).

6. Урманцев Ю.А. Эволюционика, или общая теория развития систем природы, общества и мышления. М.: Либро–ком, 2009. 240 с.

© А.А. Шалаев, (shell@live.ru), Журнал «Современная наука: актуальные проблемы теории и практики»,