УДК 004.021

Анализ методов преобразования алгоритмов

В. К. Ремизов, А. В. Григорьев
ФГБОУ ВО «Донецкий национальный технический университет», г. Донецк
e-mail: vsevolod.remizov@gmail.com, grigorievalvl@gmail.com

Аннотация
В статье рассмотрены методы преобразования алгоритмов. Описан алгоритм преобразования программ к виду, имеющему ограниченную когнитивную сложность. Определены место и роль алгоритма в рамках существующих подходов. Предложенный алгоритм обеспечивает решение задачи ограничения когнитивной сложности программ, но имеет недостаток, который требует его доработки. Перспективным направлением исследования является доработка и реализация описанного алгоритма преобразования программы к виду, имеющему ограниченную когнитивную сложность.

Введение

При написании программ часто возникает необходимость преобразования алгоритма с целью снижения его вычислительной (временной) или когнитивной (восприятие структурной сложности людьми) сложности. Для этого фрагменты алгоритма заменяются эквивалентными, то есть такими алгоритмами, которые имеют одну и ту же область определения и реализуемые функции, но разную систему правил [1]. Этой необходимостью и обусловлена актуальность данной работы.

Цель предлагаемой статьи – провести анализ основных методов преобразования алгоритмов для последующей реализации преобразования программ к виду, имеющему ограниченную когнитивную сложность.

Задача ограничения когнитивной сложности заключается в снижении сложности структуры алгоритма с целью упрощения понимания этого алгоритма людьми с разным уровнем подготовки. Она включает в себя:

- определение языка для описания моделей;
- построение модели предметной области;
- построение меры когнитивной сложности;
- упрощение построенной модели, ориентируясь на уровень ее когнитивной сложности [2-6].

Для достижение поставленной цели необходимо решить следующие задачи:

– провести анализ основных методов преобразования алгоритмов;

– провести анализ алгоритма преобразования программ к виду, имеющему ограниченную когнитивную сложность;

– определить место и роль алгоритма в рамках существующих подходов;

– определить перспективные направления работы.

Методы преобразования алгоритмов

На данный момент существуют следующие, наиболее характерные среди различных типов, методы преобразования алгоритмов:

– метод, использующий графовую модель алгоритма;

– метод, использующий свойства множеств, предикатов и операций над ними;

– метод, использующий предикативные грамматики;

– метод, использующий разрезание гиперграфа;

– и другие.

Рассмотрим данные методы и опишем их достоинства и недостатки с точки зрения когнитивной сложности.

Метод, использующий графовую модель алгоритма

В [7] рассмотрена задача эквивалентного преобразования алгоритма с предикатами простого типа на парных комбинациях, что позволило автору создать новую классификацию методов преобразования. В качестве модели алгоритма в данном методе используется графовая модель, представленная на рис. 1. В ней вершинам графа соответствуют операторы обработки данных, ветвления и слияния потоков управления. Выполнение оператора ветвления означает, что множество операторов разделяется на две части: одна с предикатом условия «истина», а другая с предикатом «ложь». А выполнение оператора слияния означает, что далее операторы объединяемых множеств выполняются одинаково [7].

Предлагаются следующие преобразования структуры алгоритма, которые выбираются на основании анализа предикатов [7]:

– инверсия условий передачи управления;

– изменение последовательности слияния потоков управления;

– вынесение начала ветвления или условия выхода из цикла из конструкции ветвления;

– разделение потока управления в точке слияния, за которой следует оператор изменения данных;

– разделение потока управления в точке слияния, за которой следует оператор ветвления;

– изменение последовательности ветвления потоков управления;

– вынесение оператора изменения данных из ветвления или изменение данных до выхода из цикла;

– внесение оператора изменения данных в ветвление или проверка условия выхода из цикла до изменения данных.

– Достоинства и недостатки данного метода представлены в табл. 1.

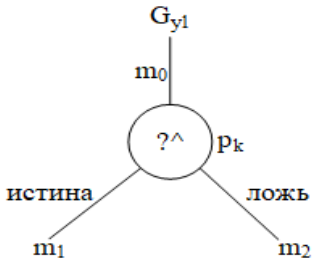

Рисунок 1 – Представление условия передачи управления в графовой модели

Таблица 1 – Достоинства и недостатки метода 1

Достоинства	Недостатки
Предложены всевозможные преобразования алгоритмов, в том числе с циклами	Не доказано, что предложенных преобразований достаточно для любого взятого алгоритма
Предложены механизмы, позволяющие понять эквивалентные ли алгоритмы с разной формой записи	Преобразования не оптимальны, увеличивают структурную сложность

Метод, использующий свойства множеств, предикатов и операций над ними

В [8] рассмотрены эквивалентные преобразования алгоритма, включающего множества и предикаты, с целью снижения его вычислительной сложности. Для этого в алгоритме находятся соответствующие операторы и заменяются на более эффективные.

Выделяются следующие преобразования, использующие свойства множеств и операций над ними [8]:

– удаление элемента множества замещением;

– замена выражения алгебры подмножеств логически эквивалентным и требующим меньшего числа операций;

– выбор порядка выполнения операции пересечения более чем двух множеств;

– использование свойства дистрибутивности операций над множествами.

Преобразования, использующие свойства предикатов и операций над ними [8]:

– задание предикатами связей между множествами;

– определение результата операции над характеристическими множествами предикатов как характеристического множества результата операции над ними;

– использование операции композиции над двухместными предикатами;

– использование свойств логических операций над предикатами.

Достоинства и недостатки данного метода представлены в табл. 2.

Таблица 2 – Достоинства и недостатки метода 2

Достоинства	Недостатки
Представлена классификация способов снижения вычислительной сложности алгоритма, использующих свойства множеств, предикатов и операций над ними	Нет четкого алгоритма преобразования
	Не затрагивается структурная сложность

Метод, использующий предикативные грамматики

В [9] алгоритм представляется в виде структурного графа (см. рис. 2), полученного при помощи предикативной грамматики.

Структурные предикативные грамматики используются для определения и анализа семантической структуры в виде графа, в основе которого лежит семантическое дерево программы. В таких грамматиках для описания структуры программы используются языки первого порядка, в которых объектами являются термы, образованные специальными функциями-конструкторами.

Структурные предикативные грамматики обеспечивают построение конечного ориентированного упорядоченного графа, который называется структурным [9]. После построения структурного графа на его основе строится граф зависимостей по данным (рис. 3). Граф зависимостей по данным отображает зависимости между операторами алгоритма и затем используется для оптимизации программы, например, путем удаления операторов, которые присваивают значение не использующейся переменной [9]. Структурно предикативные грамматики используются как для снижения вычислительной, так и когнитивной сложности. Достоинства и недостатки данного метода представлены в табл. 3.

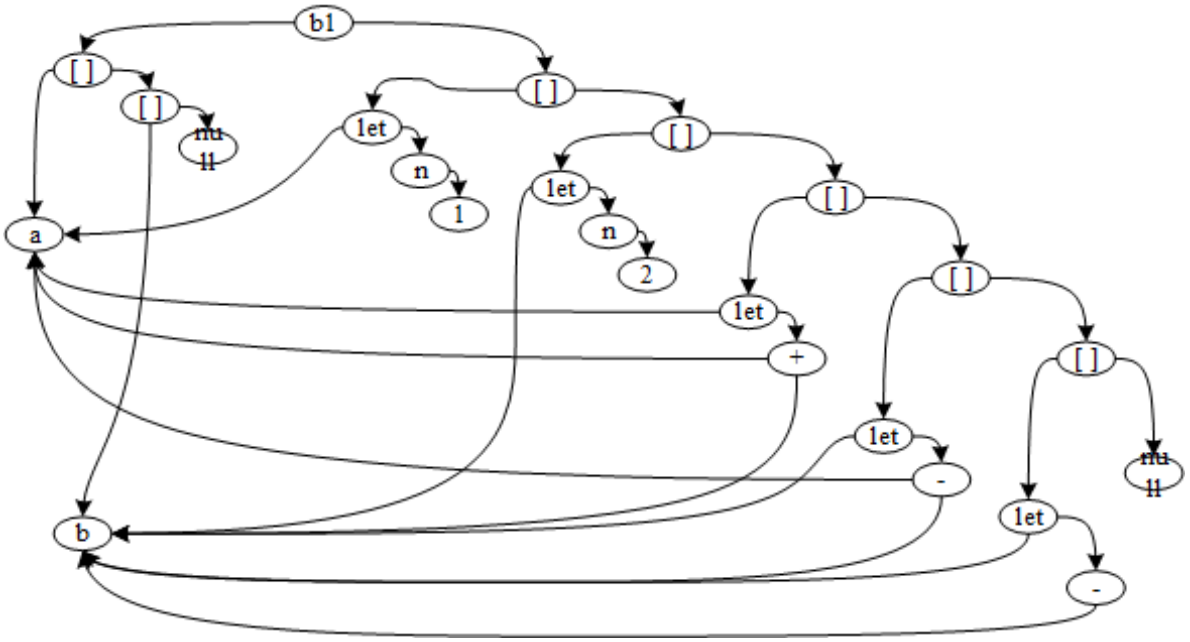

Рисунок 2 – Представление алгоритма в виде структурного графа

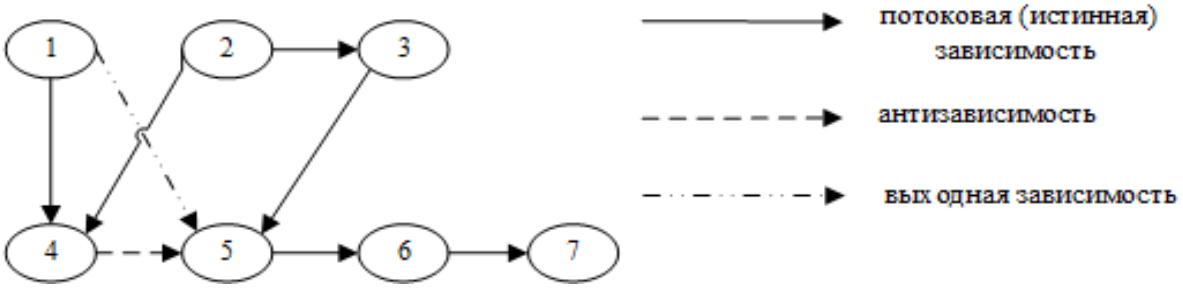

Рисунок 3 – Граф зависимостей по данным

Таблица 3 – Достоинства и недостатки метода 3

Достоинства	Недостатки
Представлен алгоритм унификации.	Нет четкого алгоритма преобразования
Представлен алгоритм построения графа зависимостей по данным	Решается частная задача

Метод, использующий разрезание гиперграфа

В [10] представлены способы преобразования алгоритмов при помощи разрезания гиперграфа, с целью снижения вычислительной сложности алгоритма. Под гиперграфом понимается граф, в котором ребра могут соединять любое множество вершин (см. рис. 4).

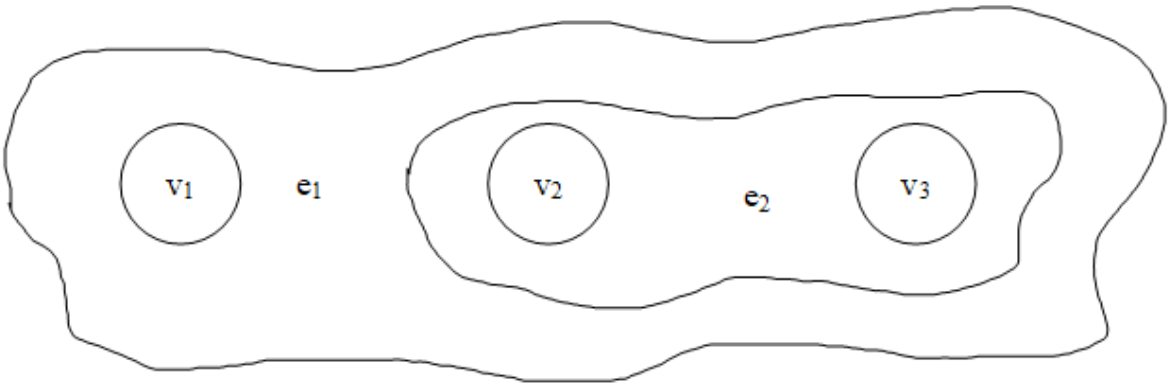

Рисунок 4 – Гиперграф

Предложенный метод представляет собой итерационный алгоритм парного замещения – делается парная перестановка вершин из двух разрезов, а затем выполняется оценка при помощи критерия оптимальности – минимума ребер, попадающих в разрез.

В результате в разрез попадают или из разреза уходят только ребра, связанные с этими вершинами. Достоинства и недостатки данного метода представлены в табл. 4.

Таблица 4 – Достоинства и недостатки метода 4

Достоинства	Недостатки
Представлен способ снижения вычислительной сложности алгоритма при помощи разрезания гиперграфа	Трудоемкость процесса разрезания из-за парных перестановок

Выводы по анализу методов

Ограничение когнитивной сложности является более сложной задачей, чем те задачи, которые были описаны в данном разделе. Это потребовало разработки нового алгоритма, который развил используемые в предыдущих методах идеи.

Алгоритм преобразования программ к виду, имеющему ограниченную когнитивную сложность.

Рассмотрим алгоритм преобразования программ к виду, имеющему ограниченную когнитивную сложность, определим его достоинства и недостатки и наметим перспективы дальнейшего развития.

В [2] описан способ преобразования алгоритма, который снижает его структурную сложность, не затрагивая вычислительную сложность. При этом сам алгоритм не меняется, а меняется лишь форма его подачи.

Данный метод использует структурные предикативные грамматики, и-или дерево и граф связей для представления алгоритма в виде состава блоков и связей между ними.

Затем полученная структура изменяется путем перемещения части подблоков прототипа в новые подблоки. Т.е. часть алгоритма, связанная по смыслу, объединяется в подмодуль, который имеет ограниченную когнитивную сложность, и затем заменяется вызовом этого подмодуля.

Рассмотрим алгоритм работы метода. Данный алгоритм построен на базе описанных ранее методов, с учетом их недостатков. Исходный прототип ***Р***, состоящий из множества подблоков ***Б***, проверяется на допустимую когнитивную сложность.

Если сложность выше заданной, приступаем к преобразованиям. Для этого формируем пустой список ***Sп***, в который затем будем вносить извлекаемые подблоки. На базе этого списка создаем новый «пустой» блок-аккумулятор ***П***, имеющий пустой список свойств, составляющих его внешнюю и внутреннюю границу, и вносим его в ***Б***. Формируем список запрещенных блоков ***Sz*** и вносим в него внутреннюю границу блока ***Б1*** и блока-аккумулятора ***П*** (см. рис. 5).

Далее ищем набор подблоков ***{Бs}***, не относящихся к запрещенным блокам и близких к находящимся в списке ***Sп***.

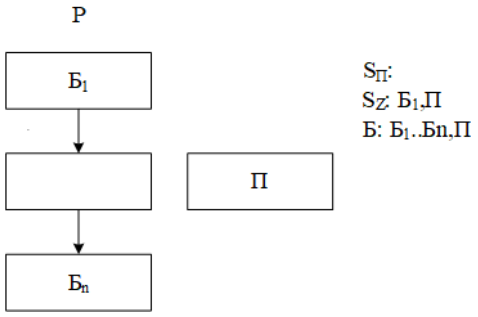

Рисунок 5 – Алгоритм работы метода. Часть 1

Для этого вводится критерий близости, который определяется составом и весом эквивалентных отношений. Состав таких отношений может характеризоваться количеством: связей по свойствам, имен в цепочке идентификации типа блока, наименования типов свойств, значений свойств, заданных в порядке взаимного включения типов свойств.

Если список ***Sп*** еще пустой, а найденных подблоков несколько, тогда определяем блок ***Бs***, удаление которого снизит когнитивную сложность больше остальных (см. рис. 6).

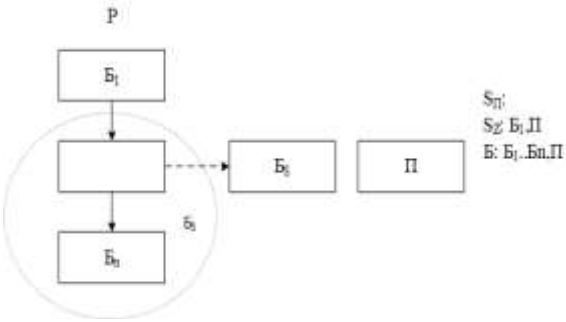

Рисунок 6 – Алгоритм работы метода. Часть 2

Найденный блок вносим в состав списка ***Sп***, формируем новый блок-аккумулятор ***П*** со списком свойств, составляющих его границу, и вносим его в ***Б***. Формируем связи блока ***Бs*** с подблоками блока ***П***. Границы блока ***П*** пополняем свойствами, посредством которых блок ***Бs*** связан с внешней средой. Удаляем из исходного прототипа ***Р*** блок ***Бs*** и включаем связи с новым блоком ***П*** (см. рис. 7).

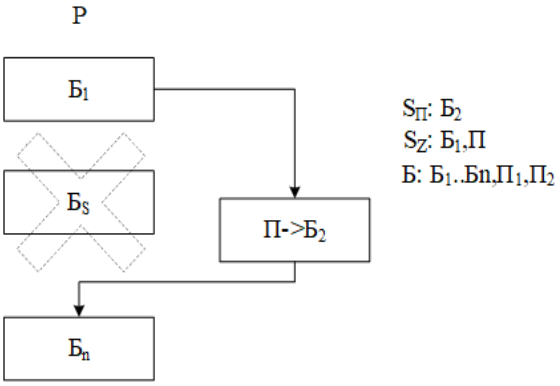

Рисунок 7 – Алгоритм работы метода. Часть 3

Если ***Sz=P***, то новый блок ***П*** – искомый, а значит алгоритм заканчивается. В противном случае оцениваем когнитивную сложность блока ***П***, если она не превышает заданную, то выбираем следующий вторичный подблок (см. рис. 8).

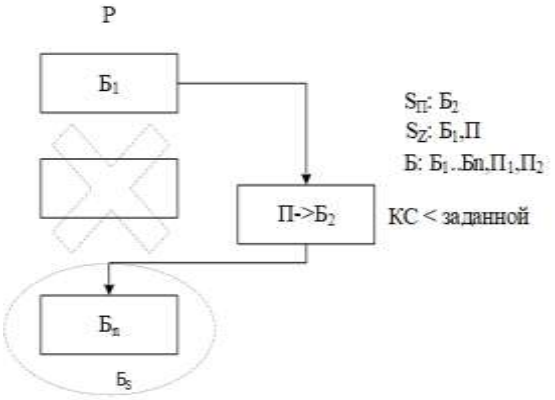

Рисунок 8 – Алгоритм работы метода. Часть 4

Если же подблок ***П*** превысил необходимый уровень когнитивной сложности, то возвращаемся к предыдущему варианту ***Sn***, ***P*** и ***П***, вносим блок ***Бs*** в список запрещенных блоков ***Sz*** и переходим к выбору следующего вторичного блока.

Финальным этапом алгоритма является создание нового «пустого» блока ***П***, имеющего пустой список подблоков и свойств, составляющих его границы, и внесение его в ***Б*** (см. рис. 9-10).

Достоинства и недостатки данного метода представлены в табл. 5.

Место алгоритма показано на рис. 11.

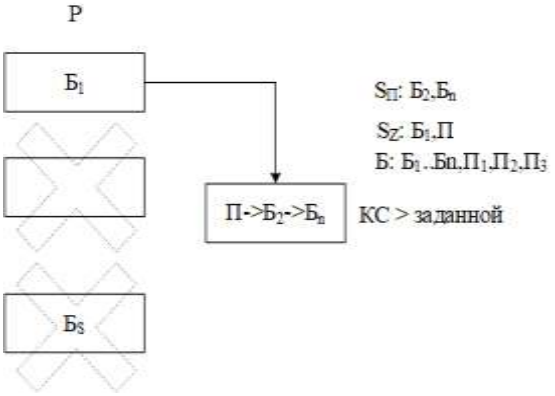

Рисунок 9 – Алгоритм работы метода. Часть 5

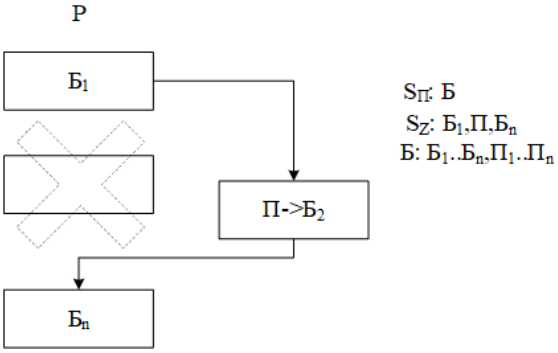

Рисунок 10 – Алгоритм работы метода. Часть 6

Таблица 5 – Достоинства и недостатки метода 5

Достоинства	Недостатки
Представлен агрегатный подход преобразования алгоритма к виду, имеющему ограниченную когнитивную сложность	Не гарантируется, что вынесенный подблок будет иметь необходимый уровень когнитивной сложности
Не используются парные перестановки, т.к. необходимый блок выбирается сразу благодаря введенному критерию близости	
Все части алгоритма, включая циклы, представляются в виде состава блоков, имеющих свои входы и выходы	

Описанный алгоритм использует фрагменты описанных ранее методов (графовое представление алгоритма, свойства предикатов, структурные предикативные грамматики, граф зависимостей по данным и разрезание гиперграфа).

Рисунок 11 – Место алгоритма в рамках существующих подходов

Однако, он разрабатывался с учетом устранения их недостатков. В результате:

- понижается структурная сложность;
- решается более сложная задача;
- преобразования более оптимальные, это достигается за счет ухода от парных перестановок.

Выводы

В статье проведен анализ основных методов преобразования алгоритмов, а также анализ алгоритма преобразования программ к виду, имеющему ограниченную когнитивную сложность. Определены место и роль алгоритма в рамках существующих подходов. Описанный алгоритм обеспечивает решение задачи ограничения когнитивной сложности программ, но имеет недостаток, который требует его доработки.

Перспективным направлением исследований является доработка и реализация описанного алгоритма преобразования программы к виду, имеющему ограниченную когнитивную сложность, так как реализация данного метода может существенно упростить программный код и сделать его более понятным для восприятия.

Литература

1. Алферова, З. В. Теория алгоритмов: учебное пособие по специальности "Организация механизированной обработки экономической информации" / З. В. Алферова. – М.: Статистика, 1973. – 164 с.

2. Григорьев, А. В. Ограничение когнитивной сложности моделей / А. В. Григорьев // Прогрессивные технологии и системы машиностроения: Международный сб. научных трудов. - Донецк: ДонГТУ, 2000. - Вып. 10. - С. 49-58.

3. Григорьев, А. В. Методика тестирования для определения когнитивной сложности моделей различных предметных областей / А. В. Григорьев // Научные труды Донецкого государственного технического университета. Серия: Информатика, кибернетика и вычислительная техника. - Донецк: ДонГТУ, 1999. – Вып. 6 (ИКВТ-99). - С. 246-251.

4. Григорьев, А. В. Оценка когнитивной сложности моделей / А. В. Григорьев // Научные труды Донецкого государственного технического университета. Серия: Информатика, кибернетика и вычислительная техника. - Донецк: ДонГТУ, 1999. – Вып. 6 (ИКВТ-99). – С. 252-259.

5. Григорьев, А. В. Адаптивная система ограничений на сложность при синтезе новых решений в интеллектуальных САПР / А. В. Григорьев // Искусственный интеллект. – Донецк, 2001. – N 2. – С. 152–167.

6. Григорьев, А. В. Комплекс средств и методов работы с формальными грамматиками в семиотической концептуальной модели предметной области интеллектуальных САПР / А. В. Григорьев // Информатика и кибернетика. – Донецк: ДонНТУ, 2017. - №1(7). – С. 46-72.

7. Иванова, Г. С. Эквивалентные преобразования структур алгоритмов [Электронный ресурс] / Г. С. Иванова // Машиностроение и компьютерные технологии, 2009. - №11. - URL: https://cyberleninka.ru/article/n/ekvivalentnye-preobrazovaniya-struktur-algoritmov (дата обращения: 19.10.2023).

8. Овчинников, В. А. Оптимизирующие преобразования алгоритмов, использующие свойства множеств, предикатов и операций над ними [Электронный ресурс] / В. А. Овчинников, Г. С. Иванова // Вестник МГТУ им. Н.Э. Баумана. Серия «Приборостроение», 2013. - №4 (93). - URL: https://cyberleninka.ru/article/n/optimiziruyuschie-preobrazovaniya-algoritmov-ispolzuyuschie-svoystva-mnozhestv-predikatov-i-operatsiy-nad-nimi (дата обращения: 19.10.2023).

9. Крицкий, С. П. Реализация оптимизирующих преобразований программ с помощью структурных предикативных грамматик [Электронный ресурс] / С. П. Крицкий, Б. Ю. Тапкинов // Известия вузов. Северо-Кавказский регион. Серия: Естественные науки, 2006. - № S1. - URL: https://cyberleninka.ru/article/n/realizatsiya-optimiziruyuschih-preobrazovaniy-programm-s-pomoschyu-strukturnyh-predikativnyh-grammatik (дата обращения: 19.10.2023).

10. Овчинников, В. А. Способы снижения вычислительной сложности алгоритмов, вытекающие из принципа формирования решений [Электронный ресурс] / В. А. Овчинников // Инженерный журнал: наука и инновации, 2013. - Вып. 11. - URL: http://engjournal.ru/catalog/it/hidden/1046.html (дата обращения 09.11.2023).

Ремизов В.К., Григорьев А.В. Анализ методов преобразования алгоритмов. *В статье рассмотрены методы преобразования алгоритмов. Описан алгоритм преобразования программ к виду, имеющему ограниченную когнитивную сложность. Определены место и роль алгоритма в рамках существующих подходов. Предложенный алгоритм обеспечивает решение задачи ограничения когнитивной сложности программ, но имеет недостаток, который требует его доработки. Перспективным направлением исследования является доработка и реализация описанного алгоритма преобразования программы к виду, имеющему ограниченную когнитивную сложность.*

Ключевые слова: *преобразование алгоритма, вычислительная сложность, когнитивная сложность, граф, множество, предикативная грамматика*

Remizov. V.K., Grigoriev A.V. Analysis of algorithm conversion methods. *The article discusses the methods of converting algorithms. An algorithm for converting programs to a form with limited cognitive complexity is described. The place and role of the algorithm within the framework of existing approaches are determined. The proposed algorithm provides a solution to the problem of limiting the cognitive complexity of programs, but has a drawback that requires its improvement. A promising area of research is the refinement and implementation of the described algorithm for converting a program to a form with limited cognitive complexity.*

Key words: *algorithm conversion, computational complexity, cognitive complexity, graph, sets, predicative grammar*

Статья поступила в редакцию 08.12.2023
Рекомендована к публикации профессором Зори С. А.