ИНФОРМАТИКА,
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
И УПРАВЛЕНИЕ

INFORMATICS,
COMPPUTER ENGINEERING
AND CONTROL

УДК 004.4'42

DOI: 10.17212/1814-1196-2020-4-37-46

Оптимизация графов потока управления в промежуточных представлениях языка функционально-поточкового параллельного программирования^{*}

В.С. ВАСИЛЬЕВ^{1,а}, А.И. ЛЕГАЛОВ^{2,б}

¹ 660041, РФ, г. Красноярск, пр. Свободный, 82, Институт космических и информационных технологий, Сибирский федеральный университет

² 101000, РФ, г. Москва, ул. Мясницкая, 20, факультет компьютерных наук, Национальный исследовательский университет «Высшая школа экономики»

^а vsvasilev@sfu-kras.ru ^б alegalov@hse.ru

Функционально-поточковые языки программирования предназначены для разработки архитектурно-независимых параллельных программ и поддерживают управление вычислениями по готовности данных. В связи с тем, что в настоящее время преобладают параллельные вычислительные системы, а их программирование на императивных языках сопряжено с проблемами переносимости, разработка инструментальных средств архитектурно-независимого параллельного программирования является актуальной задачей. Формируемая на функционально-поточковых языках программа задает граф потока данных. В ходе ее трансляции формируются промежуточные представления в виде информационного графа и накладываемого на него управляющего графа. В процессе выполнения программы по дугам управляющего графа передаются сигналы готовности данных. Явное выделение управляющего графа позволяет не только изменять стратегии управления вычислениями и обеспечивать адаптацию программы под особенности архитектуры, но и применять специфические методы оптимизации управляющих зависимостей.

В работе предлагаются методы трансформации, обеспечивающие оптимизацию управляющего графа. При генерации управляющего графа по информационному в него вносятся избыточные дуги, удаление которых не влияет на результат работы программы, однако приводит к более эффективному ее выполнению. Показано, что в функционально-поточковых программах, помимо зависимостей по управлению, свойственных для других языков программирования, возникают дополнительные, связанные с особенностями реализации отложенных или условных вычислений, описываемых задержанными списками. Приведено формальное описание избыточных зависимостей разного вида, а также эффективный алгоритм их выявления.

Разработанный подход может применяться для таких языков функционально-поточкового программирования, как Пифагор и Smile.

^{*} Статья получена 06 августа 2020 г.

Ключевые слова: оптимизация кода, граф потока управления, анализ управляющих зависимостей, функционально-потокное программирование, управление вычислениями, параллельное программирование, анализ программ, задержанные вычисления

ВВЕДЕНИЕ

Разработка параллельных программ является трудоемким процессом, что обусловливается многообразием параллельных вычислительных архитектур и подходов, отличающихся механизмами синхронизации и обмена сообщениями, топологией, быстродействием и способами организации отдельных вычислительных узлов системы, каналов передачи данных между ними, необходимостью в каждой ситуации преодолевать ресурсные ограничения и конфликты специфическими методами. Для преодоления ряда проблем параллельного программирования был предложен архитектурно-независимый подход на основе функционально-потокной парадигмы параллельного программирования [1]. Основной идеей подхода является использование языка, ориентированного на описание программ для неограниченных вычислительных ресурсов с управлением по готовности данных [2].

Несмотря на то что концепция неограниченного параллелизма в настоящее время практически не используется в процессе разработки реальных параллельных программ, она может оказаться перспективной для формирования систем программирования, обеспечивающих последующую трансформацию в ресурсно-ограниченные и архитектурно-зависимые параллельные программы [3]. Поэтому актуальной является задача исследования вариантов использования различных аспектов преобразования архитектурно-независимых параллельных программ, в частности, связанных с трансформацией методов управления вычислениями, обеспечивающих в конечном счете возможность привязки к реальным архитектурам. В качестве примера продвижения в этом направлении следует отметить работы по ресурсно-независимому параллельному программированию высокопроизводительных систем на кристаллах – язык COLAMO [4, 5].

В настоящее время ведутся работы над введением в функционально-потокные языки программирования статической типизации [6], выполняются попытки синтеза СБИС [7, 8]. Существующие инструментальные средства позволяют производить отладку, верификацию [9, 10] и оптимизацию [11] программ. Однако до сих пор методы оптимизации применялись лишь для информационных зависимостей.

Представленный в работе [12] подход базируется на построении управляющего графа (УГ) программы по реверсивному информационному графу (РИГ). Это позволяет как формировать разнообразные стратегии управления вычислениями, так и трансформировать уже сформированные управляющие графы с сохранением уровня параллелизма [13]. Изменение стратегии управления обеспечивает переносимость программы на различные вычислительные архитектуры. Возможны подходы к трансформации УГ, связанные с изменением параллелизма вплоть до организации последовательных вычислений при неизменном информационном графе.

Одним из широко используемых способов представления программы внутри оптимизатора являются различные формы графа потока управления [14, 15], аналогичные УГ функционально-потокных языков программи-

рования. Особенностью функционально-поточковых языков является то, что в них УГ явно подается на интерпретатор и используется для организации процесса управления вычислениями – описанные в статье методы оптимизации применимы только к таким языкам.

1. ПОСТАНОВКА ЗАДАЧИ

Функционально-поточковая модель вычислений предполагает, что во время выполнения программы по дугам УГ передаются сигналы готовности [2], а узлы графа, соответствующие операциям программы, собирают передаваемые сигналы и после готовности своих аргументов инициализируют начало вычислений. Механизм обработки поступающих сигналов определяется управляющим автоматом. УГ, построенный по информационному, нередко содержит избыточные зависимости – управляющие дуги, не влияющие на порядок вычисления. Удаление избыточных дуг сокращает число сигналов, передаваемых во время выполнения программы, а следовательно, является оптимизирующим преобразованием.

В работе рассматриваются вопросы, связанные с трансформацией УГ, при которой сохраняется максимальный параллелизм. Задачей является разработка методов оптимизации управляющих графов функционально-поточковых языков программирования.

2. ОПТИМИЗАЦИЯ ПРЯМЫХ УПРАВЛЯЮЩИХ ЗАВИСИМОСТЕЙ

Если между вершинами УГ существует путь, то будем говорить, что имеет место *прямая* управляющая зависимость.

Пример, демонстрирующий наличие избыточных управляющих дуг, представлен на рис. 1. Стрелками обозначены зависимости по управлению. Возле узлов в квадратных скобках указан список зависимостей. Так, например, оператор Y не зависит от других узлов, а узел X зависит от операторов Y и Z . На рис. 1, *a* приведен граф до оптимизации, на рис. 1, *б* – после нее. Видно, что, несмотря на удаление части дуг, зависимости не изменились.

Рис. 1. Оптимизация управляющих зависимостей

Fig. 1. Optimization of control dependencies

Дуга $A \rightarrow B$ является избыточной, если существует дуга $C \rightarrow B$, а узел C имеет зависимость от A .

Необходимо учитывать, что по дугам УГ распространяются сигналы управления, которые обрабатываются узлами. Удаление дуги приводит к тому, что сигнал не будет передан, а значит, и обработан. В этой связи возможны следующие варианты выполнения эквивалентной трансформации.

1. Добавление константных управляющих сигналов без изменения узлов УГ (рис. 1, в). Такой список сигналов формируется до начала выполнения программы, поэтому с их помощью можно более эффективно спланировать порядок вычислений. С другой стороны, интерпретатор может сохранять образы функций после обработки константных сигналов, и использование его при последующих вызовах может быть особенно эффективным для многократно вызываемых и рекурсивных функций.

2. Изменение узлов УГ (если это возможно), заключающееся в замене исходного управляющего автомата на такой автомат, который считает часть сигналов управления принятыми.

Оптимизация может быть выполнена по следующему алгоритму.

1. Выполнить обход управляющего графа сверху вниз (от констант к оператору `return`), при этом для каждого узла $Node_i$ рассчитать зависимости – множество $Deps_i$ – следующим образом.

1.1. $Deps_i = \emptyset$

1.2. Для каждой дуги j , входящей в узел $Node_i$, выполнить:

1.2.1. если удаление дуги j не изменит зависимости узла i , т. е. $Deps_i \subset Deps_j$, то дуга j избыточная, выполняется ее удаление;

1.2.2. иначе множество зависимостей узла j добавляется к множеству $Deps_i$: $Deps_i = Deps_i \cup Deps_j$.

2. Конец (оптимизация выполнена).

В качестве примера рассмотрим функцию вращения трехмерной фигуры вокруг горизонтальной оси:

```

1. figure_rotate << funcdef X {
2.   Figure << X:1;
3.   Alpha_rad << X:2;
4.
5.   PointsCount << Figure;|;
6.   Angles << (Alpha_rad, PointsCount):dup;
7.
8.   return << (Figure, Angles):#:|:x_rotate;
9. }
```

Оператор формирования списка (*Figure, Angles*) имеет лишнюю зависимость от *Figure*, так как вычисление *Angles* также зависит по управлению от *Figure*. Описанная ситуация хорошо видна на фрагменте соответствующего управляющего графа: дуга между узлами 1 и 6 является лишней (рис. 2).

Рис. 2. Фрагменты реверсивного информационного (а) и управляющего графов (б)

Fig. 2. Fragments of reversing control (a) and information (b) graphs

3. ОПТИМИЗАЦИЯ УПРАВЛЯЮЩИХ ЗАВИСИМОСТЕЙ ЗАДЕРЖАННЫХ СПИСКОВ

Дополнительный вид управляющих зависимостей возникает при использовании задержанных списков. Код, вложенный в задержанный список, не получит управление до момента раскрытия списка. Следовательно, если узел, выполняющий раскрытие, имеет некоторые зависимости, то эти зависимости имеют и все узлы, вложенные в список.

На рис. 3, а группировка в задержанный список показана пунктирной рамкой. В задержанный список вложены узлы X и Y, при этом Y имеет прямую управляющую зависимость от узла C. Задержанная константа Z задает этот список, ее использование не приводит к вычислению узлов списка до тех пор, пока к ней не будет применена операция раскрытия (узел E – интерпретация оператора «точка»). Управляющие зависимости операции раскрытия задержанного списка будем называть *задержанными управляющими зависимостями*. Узлы задержанного списка (X и Y) начнут выполняться лишь после того, как будет выполнена операция раскрытия, следовательно, они косвенно имеют все управляющие зависимости (от узлов C и G).

Рис. 3. Управляющие зависимости задержанных списков до (а) и после (б) оптимизации

Fig. 3. Control dependencies of delayed lists before (a) and after (b) optimization

С учетом задержанных зависимостей дуга C–Y является избыточной. Оптимизация выполняется по такому же алгоритму, как для прямых управляющих зависимостей, однако список зависимостей узлов, вложенных в задержанный список, дополняется зависимостями, которые имеет узел раскрытия списка. Результат оптимизации приведен на рис. 3, б.

В качестве примера рассмотрим функцию вычисления среднего арифметического списка и ее информационный и управляющий графы (рис. 4).

```

1. direct_average << funcdef X {
2.   Len << X:;
3.   [((Len,0):[=,!=]):?]^(
4.     "empty list",
5.     {(X,+):foldt, Len):/}
6.   ):. >> return;
7. }

```

Вычисление Len используется внутри задержанного списка, однако раскрытие списка не произойдет до того, как он будет подан на операцию интерпретации, которая также зависит от Len. Узел 10 зависит по управлению от узла 1 (соответствующего вычислению Len), однако от узла 1 также зависит узел 14 (соответствует операции интерпретации, на которую подается задержанный список). Задержанный список гарантированно не будет раскрыт до вычисления узла 14, а узел 1 будет вычислен ранее. Следовательно, дуга 1→10 является избыточной.

Рис. 4. Фрагменты информационного (а) и управляющего (б) графов функции

Fig. 4. Fragments of the information graph (a) and control (b) graphs of the function

ЗАКЛЮЧЕНИЕ

Рассмотренные подходы позволяют трансформировать граф, используемый для управления вычислениями в событийной машине, интерпретирующей функционально-поточные параллельные программы. При этом уменьшение числа управляющих связей в целом не уменьшает исходного параллелизма, обеспечивая повышение скорости вычислений за счет уменьшения управляющих дуг и замены их управляющими сигналами, непосредственно поступающими на вход соответствующих автоматов.

Замена управляющих дуг сигналами позволяет более эффективно планировать порядок вычислений. Кроме того, интерпретатор при загрузке функции в память может обрабатывать список сигналов и выполнять частичные вычисления. Таким образом, обработка списка сигналов, в отличие от сигналов готовности, передаваемых по дугам графа, выполнится лишь единожды, наиболее сильный эффект такая оптимизация окажет для рекурсивных функций.

Оптимизирующие трансформации управляющих графов встроены в существующую среду оптимизации функционально-поточных параллельных

программ [12], которая до сих пор поддерживала только методы оптимизации информационных графов. Описанные подходы применимы к программам на языках программирования Пифагор и Smile, так как они используют одинаковый формат представления управляющих графов.

СПИСОК ЛИТЕРАТУРЫ

1. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. – 2005. – Т. 10, № 1. – С. 71–89.
2. Легалов А.И., Савченко Г.В., Васильев В.С. Событийная модель вычислений, поддерживающая выполнение функционально-поточковых параллельных программ // Системы. Методы. Технологии. – 2012. – № 1 (13). – С. 113–119.
3. Динамически изменяющийся параллелизм с асинхронно-последовательными потоками данных / А.И. Легалов, И.В. Матковский, М.С. Ушакова, Д.С. Романова // Моделирование и анализ информационных систем. – 2020. – Т. 27, № 2. – С. 164–179. – DOI: 10.18255/1818-1015-2020-2-164-179.
4. Дордопуло А.И., Левин И.И. Ресурснезависимое программирование гибридных реконфигурируемых вычислительных систем // Суперкомпьютерные дни в России: труды международной конференции, 25–26 сентября 2017 г., г. Москва. – М.: Изд-во МГУ, 2017. – С. 714–723.
5. Левин И.И., Дордопуло А.И., Гудков В.А. Программирование реконфигурируемых вычислительных узлов на языке COLAMO. – Таганрог: ЮФУ, 2011. – 114 р.
6. Легалов А.И., Легалов И.А., Матковский И.В. Добавление статической типизации в язык функционально-поточкового параллельного программирования // Электронные библиотеки. – Т. 23, № 4. – С. 788–807. – DOI: 10.26907/1562-5419-2020-23-4-788-807.
7. Непомнящий О.В., Рыженко И.Н., Легалов А.И. Метод архитектурно-независимого высокоуровневого синтеза СБИС // Известия ЮФУ. Технические науки. – 2018. – № 8 (202). – С. 38–47.
8. Непомнящий О.В., Рыженко И.Н., Легалов А.И. Методы, алгоритмы и программные инструменты архитектурно независимого высокоуровневого синтеза однокристальных цифровых систем // Суперкомпьютерные технологии (СКТ-2018). – Ростов н/Д., 2018. – Т. 1. – С. 104–109.
9. Удалова Ю.В. Верификация функционально-поточковых параллельных программ с помощью интервальных формул // Образовательные ресурсы и технологии. – 2016. – № 2 (14). – С. 259–262.
10. Ушакова М.С., Легалов А.И. Верификация программ со взаимной рекурсией на языке Пифагор // Моделирование и анализ информационных систем. – 2018. – Т. 25, № 4. – С. 358–381.
11. Васильев В.С. Оптимизация программ функционально-поточкового языка Пифагор // Перспективы развития информационных технологий. – 2014. – № 20. – С. 7–14.
12. A toolkit for the development of data-driven functional parallel programmes / A.I. Legalov, V.S. Vasilyev, I.V. Matkovskii, M.S. Ushakova // Parallel Computational Technologies, PCT 2018 / ed. by L. Sokolinsky, M. Zymbler. – Cham: Springer, 2018. – DOI: 10.1007/978-3-319-99673-8_2.
13. Легалов А.И. Об управлении вычислениями в параллельных системах и языках программирования // Научный вестник НГТУ. – 2004. – № 3 (18). – С. 63–72.
14. Рыбаков А.А. Анализ алгоритмов оптимизации расположения в памяти линейных участков программы // Известия высших учебных заведений. Электроника. – 2013. – № 1. – С. 47–53.
15. Битнер В.А., Заборовский Н.В. Построение универсального линеаризованного графа потока управления для использования в статическом анализе кода алгоритмов // Моделирование и анализ информационных систем. – 2013. – Т. 20, № 2. – С. 166–177. – DOI: 10.18255/1818-1015-2013-2-166-177.

Васильев Владимир Сергеевич, старший преподаватель кафедры вычислительной техники Сибирского федерального университета. Основное направление научных исследований – функциональное программирование, параллельное программирование, проектирование программного обеспечения. Имеет более 20 печатных работ и учебных пособий. E-mail: vsvasilev@sfu-kras.ru

Легалов Александр Иванович, доктор технических наук, профессор департамента программной инженерии факультета компьютерных наук Высшей школы экономики. Основное направление научных исследований – технологии программирования, параллельное программирование, методы конструирования программных объектов, архитек-

турно-независимое параллельное программирование, эволюционная разработка программного обеспечения. Имеет более 200 печатных работ и учебных пособий. E-mail: alegalov@hse.ru

Vasilev Vladimir S., senior lecturer, department of computing engineering, SFU. His research interests are currently focused on functional programming, parallel programming, and program design. He has more than 20 publications and teaching manuals. E-mail: vsvasilev@sfu-kras.ru

Legalov Alexander I., D.Sc. (Eng.), professor, department of software engineering, HSE. His research interests are currently focused on programming technologies, parallel programming, software designing methods, architecture-independent parallel programming, and evolutionary software development. He has more than 200 publications and teaching manuals. E-mail: alegalov@hse.ru

DOI: 10.17212/1814-1196-2020-4-37-46

Control flow graphs optimization in intermediate representations of the functional dataflow parallel programming language*

V.S. VASILEV^{1,a}, A.I. LEGALOV^{2,b}

¹ *Siberian Federal University, 79 Svobodny Prospekt, Krasnoyarsk, 660041, Russian Federation*

² *Higher School of Economics, National Research University, 20 Myasnitskaya Street, Moscow, 101000; Russian Federation*

^a *vsvasilev@sfu-kras.ru* ^b *alegalov@hse.ru*

Abstract

Functional dataflow programming languages are intended for the development of architecture-independent parallel programs and support the control of computations based on data availability. Due to the fact that at present parallel computing systems are very widespread, and their programming in imperative languages is associated with portability problems, the development of architecturally independent parallel programming tools is an urgent task. When such a program is translated, intermediate representations are formed as the information graph and the corresponding control graph. During program execution, data readiness signals are transmitted along the arcs of the control graph. An explicit selection of the control graph allows us not only to change the computational control strategies and ensure the adaptation of the program to the architecture features, but also to apply specific methods for optimizing control dependencies.

The paper proposes transformation methods that provide optimization of the control graph. When generating a control graph from an informational one, redundant arcs are introduced into it, the removal of which does not affect the result of the program, but leads to its more efficient execution. It is shown that in dataflow programs, in addition to control dependencies inherent in other programming languages, additional ones associated with the implementation features of deferred or conditional computations described by delayed lists arise. A formal description of redundant dependencies of various types is given, as well as an effective algorithm for their identification.

The developed approach can be applied to such dataflow programming languages as PIFAGOR and Smile.

Keywords: dataflow code optimization, control flow graph, control dependency analysis, dataflow programming, computation control, parallel programming, program analysis, delayed computation

REFERENCES

1. Legalov A.I. Funktsional'nyi yazyk dlya sozdaniya arkhitekturno-nezavisimyykh parallel'nykh program [Functional language for creating of architectural independent parallel programs]. *Vychislitel'nye tekhnologii = Computational Technologies*, 2005, vol. 10, no. 1, pp. 71–89.
2. Legalov A.I., Savchenko G.V., Vasilev V.S. Sobytiinaya model' vychislenii, podderzhivayushchaya vypolnenie funktsional'no-potokovykh parallel'nykh program [Computation event model backing the execution of functional data flow concurrent programs]. *Sistemy. Metody. Tekhnologii = Systems. Methods. Technologies*, 2012, no. 1 (13), pp. 113–119.
3. Legalov A.I., Matkovskii I.V., Ushakova M.S., Romanova D.S. Dinamicheski izmenyayushchiysya parallelizm s asinkhronno-posledovatel'nyimi potokami dannykh [Dynamically changing

* Received 06 August 2020.

parallelism with the asynchronous sequential data flows]. *Modelirovanie i analiz informatsionnykh sistem = Modeling and Analysis of Information Systems*, 2020, vol. 27, no. 2, pp. 164–179. DOI: 10.18255/1818-1015-2020-2-164-179.

4. Dordopulo A.I., Levin I.I. [Resource-independent programming of hybrid reconfigurable computer systems]. *Superkomp'yuternye dni v Rossii: trudy mezhdunarodnoi konferentsii*, 25–26 sentyabrya 2017 g., g. Moskva [Russian supercomputing days: Proceedings of the international conference (September 25–26, 2017, Moscow, Russia)]. Moscow, 2017, pp. 714–723. (In Russian).

5. Levin I.I., Dordopulo A.I., Gudkov V.A. *Programmirovaniye rekonfiguriruemyykh vychislitel'nykh uzlov na yazyke COLAMO* [Programming reconfigurable computing nodes in the COLAMO language]. Taganrog, Southern Federal University Publ., 2011. 114 p.

6. Legalov A.I., Legalov I.A., Matkovskiy I.V. Dobavlenie staticheskoi tipizatsii v yazyk funktsional'no-potokovogo parallel'nogo programmirovaniya [Evolution of Visualization Methods for Research Publication Collections]. *Elektronnye biblioteki = Russian Digital Libraries Journal*, vol. 23, no. 4, pp. 788–807. DOI: 10.26907/1562-5419-2020-23-4-788-807.

7. Nepomnyaschy O.V., Ryzhenko I.N., Legalov A.I. Metod arkhitekturno-nezavisimogo vysokourovnevnogo sinteza SBIS [The method of architecturally independent high-level synthesis of VLSI]. *Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskie nauki = Izvestiya Southern Federal University. Engineering sciences*, 2018, no. 8 (202), pp. 38–47.

8. Nepomnyaschy O.V., Ryzhenko I.N., Legalov A.I. [Methods, algorithms and software tools for architecture-independent high-level synthesis of single-chip digital systems]. *Superkomp'yuternye tekhnologii (SKT-2018)* [Supercomputer technologies (SCT-2018)]. Rostov-on-Don, 2018, pp. 104–109. (In Russian).

9. Udalova Yu.V. Verifikatsiya funktsional'no-potokovykh parallel'nykh programm s pomoshch'yu interval'nykh formul [Verification of parallel functional dataflow programs using interval formulas]. *Obrazovatel'nye resursy i tekhnologii = Educational Resources and Technologies*, 2016, no. 2 (14), pp. 259–262.

10. Ushakova M.S., Legalov A.I. Verifikatsiya programm so vzaimnoi rekursiei na yazyke Pifagor [Verification of programs with mutual recursion in the pifagor language]. *Modelirovanie i analiz informatsionnykh sistem = Modeling and Analysis of Information System*, 2018, vol. 25, no. 4, pp. 358–381.

11. Vasil'ev V.S. Optimizatsiya programm funktsional'no-potokovogo yazyka Pifagor [Optimization of programs of the functional-streaming language Pythagoras]. *Perspektivy razvitiya informatsionnykh tekhnologii = Information technology development prospects*, 2014, no. 20, pp. 7–14.

12. Legalov A.I., Vasilyev V.S., Matkovskii I.V., Ushakova M.S. A toolkit for the development of data-driven functional parallel programmes. *Parallel Computational Technologies, PCT 2018*. Ed. by L. Sokolinsky, M. Zymbler. Cham, Springer, 2018. DOI: 10.1007/978-3-319-99673-8_2.

13. Legalov A.I. Ob upravlenii vychisleniyami v parallel'nykh sistemakh i yazykakh programmirovaniya [About computation control in parallel system and programming languages]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta = Science bulletin of the Novosibirsk state technical university*, 2004, no. 3 (18), pp. 57–66.

14. Rybakov A.A. Analiz algoritmov optimizatsii raspolozheniya v pamyati lineynykh uchastkov programmy [Analysis of optimization algorithms for memory location of program linear sections]. *Izvestiya vysshikh uchebnykh zavedenii. Elektronika = Proceedings of Universities. Electronics*, 2013, no. 1, pp. 47–53.

15. Bitner V.A., Zaborovsky N.V. Postroenie universal'nogo linearizovannogo grafa potoka upravleniya dlya ispol'zovaniya v staticheskom analize koda algoritmov [The construction of an universal linearized control flow graph for static code analysis of algorithms]. *Modelirovanie i analiz informatsionnykh sistem = Modeling and Analysis of Information System*, 2013, vol. 20, no. 2, pp. 166–177. DOI: 10.18255/1818-1015-2013-2-166-177.

Для цитирования:

Васильев В.С., Легалов А.И. Оптимизация графов потока управления в промежуточных представлениях языка функционально-потокowego параллельного программирования // Научный вестник НГТУ. – 2020. – № 4 (80). – С. 37–46. – DOI: 10.17212/1814-1196-2020-4-37-46.

For citation:

Vasilev V.S., Legalov A.I. Optimizatsiya grafov potoka upravleniya v promezhutochnykh predstavleniyakh yazyka funktsional'no-potokovogo parallel'nogo programmirovaniya [Control flow graphs optimization in intermediate representations of the functional dataflow parallel programming language]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta = Science bulletin of the Novosibirsk state technical university*, 2020, no. 4 (80), pp. 37–46. DOI: 10.17212/1814-1196-2020-4-37-46.

ISSN 1814-1196, <http://journals.nstu.ru/vestnik>
Science Bulletin of the NSTU
Vol. 80, No 4, 2020, pp. 37–46