# ОТЗЫВ

официального оппонента Зуева Евгения Александровича
на диссертацию Головешкина Алексея Валерьевича
на тему «Устойчивая алгоритмическая привязка к коду программы»,
представленную на соискание учёной степени кандидата технических наук
по специальности 2.3.5 — математическое и программное обеспечение
вычислительных систем, комплексов и компьютерных сетей.

Диссертация Головешкина А.В. посвящена решению задачи устойчивой алгоритмической привязки к коду программы. Также решается вспомогательная задача легковесного парсинга программ.

**Целью** диссертационного исследования является разработка моделей и алгоритмов, позволяющих сконструировать описание участка кода программы по абстрактному синтаксическому дереву («привязаться» к коду), сохранить это описание и впоследствии алгоритмически найти по нему данный участок в отредактированной программе («перепривязаться»). Требуемая «устойчивость» привязки заключается в том, что искомый код должен быть найден всегда, за исключением случаев, когда он был удалён в ходе редактирования. На базе указанных моделей и алгоритмов реализуется панель разметки кода, интегрируемая в различные среды разработки.

**Актуальность** работы заключается в том, что изучение кода с целью поиска участков, имеющих отношение к решаемой задаче (к прорезающей функциональности, над которой ведётся работа), занимает у программиста в несколько раз больше времени, чем собственно написание кода. Собранную информацию важно не потерять, чтобы не заниматься повторными поисками. Ситуации, когда данная информация теряется, возникают часто: программист не работает над задачей непрерывно, переключается на другие задачи и другие виды деятельности, теряя «рабочее состояние» текущей задачи. Устойчивая привязка к коду позволяет сохранить информацию о расположении участков кода, нужных программисту, и впоследствии автоматически найти эти участки. В долгосрочной перспективе это решает проблему отсутствующей или устаревшей документации: с участками кода, к которым произведена привязка, можно связать пометки и дополнительную информацию, построив тем самым разметку кода.

Наиболее важными **результатами**, полученными в работе, на мой взгляд, являются следующие:

1) предложенный метод легковесного парсинга на основе упрощённых LL(1) и LR(1) грамматик, использующий модифицированные алгоритмы LL(1) и LR(1) синтаксического анализа;
2) реализованный генератор легковесных парсеров LanD со встроенным языком описания легковесных грамматик;
3) разработанный метод устойчивой алгоритмической привязки к меняющемуся коду на основе контекстов, использующий предложенные автором модели и алгоритмы;
4) реализованная на основе этих алгоритмов практически применимая панель разметки кода, интегрируемая в различные среды разработки;
5) предложенный метод учёта многострочных фрагментов кода в абстрактном синтаксическом дереве программы.

Результаты 1, 3 и 5 образуют **научную новизну** диссертационного исследования.

**Практическая значимость** полученных Головешкиным А.В. результатов заключается в практической применимости разработанного им генератора легковесных парсеров и панели разметки кода. Генератор легковесных парсеров LanD и одноимённый язык описания грамматик позволяют создавать простые и короткие грамматики промышленных языков программирования (автор приводит примеры грамматик для языков C#, Java и входного языка генератора анализаторов yacc). Генерируемые парсеры могут быть использованы в задачах обратной инженерии программ, при разборе кода с ошибками и т. д. Панель разметки кода может применяться в задачах разработки и сопровождения программ, служить для ускорения навигации по коду и сохранения знания о предназначении его участков с последующим обменом этим знанием внутри коллектива разработчиков. Также разметка может быть использована при обучении программированию.

**Достоверность** полученных результатов обеспечивается использованием известных и проверенных положений и методов теории алгоритмов, теории формальных языков и грамматик, математической строгостью изложения. Для проверки результатов проводятся вычислительные эксперименты, практические результаты подтверждены внедрениями. Результаты исследования апробированы на 8 всероссийских и международных конференциях, на 3 научных семинарах, опубликованы в 10 печатных работах, из которых 1 статья проиндексирована в

международной базе данных Scopus, 4 статьи опубликованы в журналах из перечня ВАК. Имеются 3 свидетельства о государственной регистрации программы для ЭВМ.

Диссертация **состоит из** введения, пяти глав, заключения и списка литературы. Общий объём диссертации составляет 170 страниц текста с 21 рисунком и 12 таблицами. Библиография содержит 162 наименования.

Во **введении** даются определения основных понятий, вводимых в диссертации: «привязка», «перепривязка», «разметка». Обосновывается актуальность диссертационного исследования, анализируется степень разработанности темы исследования, формулируются цели и задачи работы, научная новизна и практическая значимость; приводятся данные об апробации работы и о публикациях, в которых изложены результаты диссертации; формулируются положения, выносимые на защиту.

В **первой главе** проводится обзор литературы по теме исследования. Рассматриваются работы, посвящённые проблематике прорезающих функциональностей как таковых, вопросам изучения кода, поиска информации о назначении участков кода, сохранения собранной информации. Приводятся сведения об инструментальных средствах, предложенных научным сообществом для упрощения исследования кода в процессе разработки программы. Проводится обзор исследований, посвящённых проблемам, смежным с проблемой устойчивой привязки к коду, а также работ, посвящённых легковесному парсингу программ. Для каждой из рассматриваемых групп работ автор формулирует связь с решаемой в диссертационном исследовании задачей, тем самым подчёркивая важность проведённого диссертационного исследования.

Во **второй главе** рассматривается задача легковесного парсинга кода для построения абстрактного синтаксического дерева, достаточного для использования в задаче привязки к коду. Для случаев LL(1) и LR(1) вводятся формальные определения легковесной грамматики со специальным терминальным символом Any и упрощённой грамматики — легковесной грамматики с символом Any, строящейся по некоторой «полной» грамматике. Символ Any используется вместо явного описания структуры областей программы, несущественных для целей привязки. Автор отмечает принципиальные отличия вводимого им подхода от ближайших аналогов, по-своему вводящих концепцию символа Any. Для работы с упрощёнными грамматиками модифицируются алгоритмы LL(1) и LR(1) синтаксического анализа. Доказывается, что парсер, сгенерированный по

упрощённой LL(1) грамматике, разбирает все правильные программы из «полного» языка, и что любая легковесная LL(1) или LR(1) грамматика является упрощённой относительно некоторого множества «полных» грамматик. Формулируется практический способ разработки легковесной грамматики: грамматика пишется с нуля и итеративно уточняется с тем, чтобы в итоге она стала упрощённой для грамматики, порождающей, как минимум, правильные программы на целевом языке. В разделе экспериментов демонстрируется разработка упрощённой LL(1) грамматики для языка C# на разработанном автором языке описания легковесных грамматик LanD.

В **третьей главе** вводятся элементы «синтаксического сахара» для языка LanD, позволяющие писать более короткие и понятные грамматики. Для случаев LL(1) и LR(1) предлагаются алгоритмы восстановления от ошибки, основанные на обработке Any. Анализируется влияние восстановления от ошибок на сложность алгоритмов парсинга, делается вывод о том, что в среднем сложность можно по-прежнему считать линейной. В разделе экспериментов демонстрируется успешное применение легковесных LL(1) и LR(1) парсеров языков C# и Java для разбора крупных промышленных проектов.

В **четвёртой главе** описываются модели и алгоритмы, разработанные для устойчивой привязки к синтаксическим элементам программы — участкам кода, соответствующим некоторым нетерминальным символам легковесной грамматики. Вводится шестиэлементная модель для описания таких участков (точек привязки), состоящая из типа (нетерминального символа, соответствующего участку) и пяти структур, называемых «контекстами». Структура каждого из контекстов описывается подробно. Вводится алгоритм переприязки, проводящий сравнение ранее сохранённых описаний точек привязки одного типа, расположенных в одном файле, с описаниями, которые можно построить для элементов того же типа, присутствующих в отредактированной версии файла. Подробно рассматриваются алгоритмы поконтекстных сравнений, алгоритм-отсечение, обрабатывающий простые случаи поиска, эвристики для настройки весовых коэффициентов, отражающих важность каждого из контекстов в конкретной ситуации. Анализируется сложность алгоритма переприязки, приводятся результаты замеров производительности. Экспериментально оценивается устойчивость производимой привязки, а также проводится сравнение с ближайшим аналогом.

В **пятой главе** устойчивая привязка к синтаксическим элементам расширяется на «произвольные» однострочные и многострочные участки кода.

Привязка к строке происходит за счёт привязки к объемлющему синтаксическому элементу и расширения описания этого элемента дополнительным «контекстом строки». Для контекстов строки приводится дополнительный алгоритм сравнения. Привязка к многострочным фрагментам сводится к привязке к синтаксической сущности путём обрамления фрагмента псевдокомментариями (или другими лексическими элементами, пропускаемыми в ходе синтаксического анализа) и встраивания соответствующего ему узла в синтаксическое дерево программы. При этом встраивать разрешается только фрагмент, удовлетворяющий определённым условиям, означающим его согласованность с синтаксической структурой программы. Приводится алгоритм встраивания узла для такого фрагмента, анализируется сложность алгоритма, доказывается корректность дерева, получившегося после встраивания. В разделе экспериментов оценивается устойчивость привязки к однострочным фрагментам.

В **заключении** перечисляются основные результаты, полученные в диссертации, и обозначаются возможные направления для дальнейших исследований.

Считаю, что текст диссертации хорошо структурирован, научные положения, логика проведения исследования и выводы обосновываются автором; состоятельность теоретических результатов, предложенных в каждой главе, подтверждаются экспериментами на больших объемах промышленного кода. Формулируются рекомендации по практическому применению разработанных алгоритмов.

Необходимо сделать несколько **замечаний**.

1. Следует признать, что наибольший интерес с точки зрения оппонента представляет часть работы, посвященная «легковесным» грамматикам и, соответственно, феномену легковесного синтаксического разбора. В работе содержится всестороннее рассмотрение данного понятия, в том числе, в исторической перспективе. При этом, складывается ощущение, что в большинстве практических случаев построение легковесной грамматики из «полной» возможно эмпирическим путем и не требует создания специальных формализмов и программных инструментов для этих целей. Кроме того, при наличии общедоступных парсеров полных грамматик того или иного языка создание специальных «урезанных» грамматик и основанных на них распознавателях имеет преимущественно теоретический интерес и с практической точки зрения представляется сомнительным. Аргумент автора о возможном отсутствии полной

(нормативной) грамматики для некоторого языка программирования вряд ли может считаться убедительным.

2. Ключевое понятие всей диссертационной работы – *привязка* – можно неформально определить как смысловое описание некоторого фрагмента исходного кода в виде набора структур специального вида (*контекстов*). Совокупность фрагментов кода, относящимися к прорезающим функциональностям, и соответствующих привязок, образует *разметку* кода.

В диссертационной работе большое внимание уделяется формальному определению введенных понятий и теоретическому обоснованию алгоритмов, с ними связанных. При этом, практически отсутствуют как практические примеры семантических привязок и разметки в целом, так и описание логики их формирования и использования разработчиками. Некоторые (схематические и явно недостаточные) примеры приводятся в главе 5, которая трактует расширение понятия привязки, однако и в этой главе ощущается явный перекос внимания в сторону теоретических рассмотрений в ущерб содержательному объяснению их целей и смысла.

3. Хотя диссертационная работа содержит детальные описания и обоснования алгоритмов первичной и повторной привязок фрагментов исходного кода – в работе практически отсутствует информация технологического характера: как разработанные и реализованные алгоритмы используются или могут использоваться в практической деятельности по разработке и сопровождению программ. Упомянутая в разд. 4.5 «панель привязки» – компонент, встроенный в IDE Visual Studio – описана весьма лаконично и не позволяет судить ни о ее назначении, ни о реальной применимости в процессе разработки. Этот недостаток, в точки зрения оппонента, особенно огорчителен, так как, судя по информации из Введения, инструменты, созданные автором на основе результатов работы, реально используются в ряде организаций. К этому можно добавить, что ряд публикаций автора, в частности, его магистерская диссертация содержат более конкретную информацию о реальных инструментах и аспектах их использования.

Указанные замечания не снижают значимости полученных результатов и не влияют на положительную оценку диссертационного исследования. Диссертационная работа Головешкина А.В. «Устойчивая алгоритмическая привязка к коду программы» является законченной научно-квалификационной работой, вносящей научный вклад в области теории формальных языков и анализа эволюции программного обеспечения, а также решающей актуальную прикладную

задачу разметки прорезающих функциональностей, представленных в коде программы.

Тема диссертации и её содержание **соответствуют паспорту специальности** 2.3.5 — математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей. Автореферат **правильно и полно** отражает содержание работы.

По моему мнению, диссертация удовлетворяет всем критериям, установленным Положением о порядке присуждения учёных степеней, а её автор, Головешкин Алексей Валерьевич, безусловно заслуживает присуждения учёной степени кандидата технических наук по специальности 2.3.5 — математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей.

Руководитель лаборатории операционных систем, языков программирования и компиляторов Автономной некоммерческой организации высшего образования «Университет Иннополис», кандидат физико-математических наук (05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей)

29.11.2022г.

Зуев Евгений Александрович