"УТВЕРЖДАЮ"

Директор ФИЦ ИУ РАН

академик РАН И.А. Соколов

24» апреля 2017 г.

Отзыв ведущей организации
на диссертацию Кошелева Владимира Константиновича
«Межпроцедурный статический анализ для поиска ошибок в исходном коде программ на языке C#»,
представленной на соискание ученой степени кандидата физико-математических наук по специальности 05.13.11 (математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей)

Наряду с тестированием и динамическим анализом, статический анализ исходного кода широко используется для поиска дефектов в программах. Многие современные стандарты сертификации требуют, чтобы программное обеспечение было проверено на наличие дефектов, в том числе при помощи инструментов статического анализа. От статических анализаторов для применения в промышленном цикле разработки требуются: высокая скорость анализа, высокое качество анализа. Для соблюдения этих требований статические анализаторы вынужденно используют нестрогий анализ, позволяющий достичь компромисса между временем работы, количеством пропусков дефектов и процентом ложных срабатываний. При этом для поиска сложных межпроцедурных дефектов необходимо использовать анализ программ, учитывающий зависимости по данным (чувствительность к потоку), условия переходов (чувствительность к путям), контексты вызовов (чувствительность к контексту) и работу с динамической памятью. В развитии методов межпроцедурного анализа в применении к задачам поиска потенциальных ошибок заключается ***актуальность*** проведенного исследования.

В течение последних десяти лет язык программирования C# стабильно входит в шестерку самых распространенных языков программирования. В настоящий момент чувствительный к контексту, потоку и путям анализ с целью выявления дефектов для языка программирования C# осуществляют лишь закрытые коммерческие инструменты, для которых отсутствует детальное описание используемых подходов и алгоритмов. Существующие открытые решения для поиска дефектов ограничиваются обходом абстрактного синтаксического дерева или простым внутрипроцедурным анализом, не проводя детального анализа потоков управления и данных.

Большинство систем для автоматического поиск ошибок с открытым исходным кодом для языка C# не проводят сложного анализа, учитывающего зависимости по управлению, условия перехода и поведение вызываемых методов. Среди инструментов для анализа программ на языке C/C++ можно отметить clang static analyzer, также использующий символьное выполнение. Основные отличия по сравнению с clang static analyzer:

- упрощённая модель памяти за счет особенностей языка C#;
- объедение символьных состояний, существенно увеличивающее производительность анализа;
- алгоритмы оптимизации размера формул;
- алгоритм работы с чистыми методами;
- критерий выдачи предупреждений;
- граф вызовов состоит из методов, а не из модулей трансляции;
- поддержка исключений.

Коммерческие инструменты статического анализа, как правило, не имеют бесплатной версии для сравнения. Используемые методы анализа для коммерческих инструментов также неизвестны. Более того, даже когда результаты доступны, сравнение с ними запрещается лицензионным соглашением.

Общепризнанным способом организации масштабируемого межпроцедурного анализа является использование резюме функций, сжато описывающие их поведение. Конкретная форма и алгоритмы построения резюме целиком зависят от использующихся методов внутрипроцедурного анализа. Методов статического анализа, использующихся в открытых инструментах, на поверку оказывается недостаточно для удовлетворения указанных выше

требований. В свою очередь, для коммерческих инструментов анализа отсутствует описание использованных методов.

Метод символьного выполнения, являясь чувствительным к потоку и путям, успешно применяется при анализе программ для поиска ошибочных входных данных, используя для этого решатели задачи выполнимости формул. В работе предлагается воспользоваться методом символьного выполнения для проведения внутрипроцедурного анализа. Метод символьного выполнения является ресурсоёмким, поэтому для достижения высокой скорости анализа в задачи проеведенного исследования входили:

– алгоритм объединения состояний символьного выполнения;

– алгоритм упрощения формул, основанный на свойствах доминаторов;

– подход для поддержки циклов с фиксированным числом итераций;

– метод моделирования вызовов с использованием резюме;

– подход для уменьшения числа запросов к решателю за счет доказательства свойств программы методами статического анализа.

Использование этих подходов позволяет достичь требуемой скорости при сохранении высокого качества анализа.

Методы статического анализа применяют различные критерии для выдачи предупреждений о наличии дефектов в зависимости от целей анализатора и использующихся алгоритмов анализа. Для успешного применения символьного выполнения для поиска дефектов необходимо разработать метод определения критерия выдачи предупреждений, учитывающий особенности символьного выполнения и позволяющий соблюсти баланс между числом пропусков ошибок и количеством ложных предупреждений.

Идеальный статический анализатор должен хорошо масштабироваться, обнаруживать все ошибки времени выполнения, не иметь ложных срабатываний и пропусков ошибок и применяться ко всем программам без ограничений. Ключевая сложность разработки статического анализа заключается в поиске компромисса между масштабируемостью, классом обнаруживаемых ошибок, процентом ложных срабатываний и пропуском ошибок и дополнительными ограничениями, накладываемыми на анализируемую программу.

Для достижения поставленных целей были ***сформулированы и решены следующие задачи***:

1. Разработка внутрипроцедурных чувствительных к потоку и путям алгоритмов анализа, позволяющих на основе вычисленной ими информации выполнять поиск широкого класса дефектов; доказательство корректности алгоритмов.

2. Разработка межпроцедурных чувствительных к потоку, контексту и путям алгоритмов анализа, включающих алгоритм построения резюме метода по результатам внутрипроцедурного анализа и алгоритм применения резюме в точках вызова.

3. Разработка метода определения критерия выдачи предупреждения о наличии дефекта с учетом особенностей проводимого анализа, подходящего для широкого класса дефектов. Разработка конкретных детекторов для приведенного критерия выдачи предупреждения.

4. Оценка на практике характеристик предложенных методов и алгоритмов на соответствие заявленным требованиям.

Формулы представляются в виде ациклического графа, где выражения являются вершинами, а рёбра соответствую операндам операций. Таким образом, добавления нового выражения, при условии, что его операнды уже были добавлены, добавит лишь одну вершину. Формулы задают точное множество возможных значений переменных и памяти в рамках рассматриваемых упрощающих предположений. Сами по себе упрощающие предположения могут как расширять, так и сужать множество значений. Например, развертка циклов сужает множество возможных значений, а предположение об отсутствии псеводнимов (алиасов) может как сужать, так и расширять множество возможных значений. В работе оценка не производится. Циклы с фиксированным числом итераций достаточно редки, но всё же бывают. Например, они возникают при обработке битов числа, инициализации коллекций заданного размера или обработки данных заданного формата. Я обнаружил около сотни примеров циклов с фиксированным числом итераций среди анализируемых проектов.

В работе рассматриваются лишь дефекты типа доступ к null и утечка ресурсов. С помощью предложенного метода может быть реализован поиска некоторых других дефектов,

например некорректное преобразование типа, деление на 0. Однако анализа типов дефектов, которые могут быть найдены с помощью данного подхода не производится.

В работе не производится детального анализа причин ложных срабатываний. Отмечается то факт, что для каждого упрощающего предположения, расширяющего множество возможных значений переменных и памяти, можно подобрать пример, что данное предположение приведёт к ложному срабатыванию. Аналогично, для предположений, сужающих множество допустимых значений, можно подобрать пример, на котором данное предположение приведёт к пропуску дефекта.

Для упрощения повествования типы входных переменных не рассматриваются. Внутреннее представление организовано таким образом, что при условии корректности сходной программы, типы локальных переменных будут выведены автоматически, исходя из типов выражений в правой части.

Рассмотренные методы реализованы в инструменте статического анализа SharpChecker, в котором реализованны предлагаемые в данной работе алгоритмы. Проводится оценка производительности и анализ результатов инструмента SharpChecker. Среди алгоритмов поиска ошибок, реализованных в инструменте SharpChecker, есть как использующие исключительно синтаксический анализ, так и анализ потоков данных, а также наиболее мощный, описанный в данной работе, чувствительный к контексту и путям выполнения межпроцедурный анализ. Приводится выполненная оценка эффективности оптимизации размера формул Приводится выполненная оценка эффективности оптимизации размера формул при помощи алгоритмов.

Инструмент SharpChecker был применен к набору проектов с открытым исходным кодом. Размеры проектов варьировались от пятидесяти тысяч строк кода до полутора миллионов.

К ***основным результатам*** диссертационной работы можно отнести следующее:

1. Разработан алгоритм внутрипроцедурного анализа, обладающий чувствительностью к потоку и путям, при этом не теряющий информацию в точках объединения.

2. Разработан алгоритм межпроцедурного анализа, чувствительного к контексту, потоку и путям, обобщающий результаты внутрипроцедурного анализа в виде резюме метода и применяющий резюме при обработке вызова данного метода.

3. Разработан критерий выдачи предупреждений, учитывающий чувствительность к путям при поиске дефектов, использование которого позволяет достичь приемлемого уровня ложных предупреждений. Для данного критерия разработаны детекторы, позволяющие обнаружить ошибки вида .доступ по нулевому указателю. и .утечка ресурсов..

4. Доказана корректность предложенного алгоритма внутрипроцедурного анализа и соответствие разработанных детекторов выбранному критерию выдачи предупреждений.

5. Разработан инструмент статического анализа, реализующий разработанные алгоритмы для анализа программ на языке C#. Для данного инструмента проведена экспериментальная оценка его характеристик на соответствие заявленным требованиям. Предложенные алгоритмы и методы позволяют проводить анализ проектов, состоящих из более миллиона строк кода, в отведённое время. При этом качество результатов анализа соответствует заявленным требованиям (более 50% истинных срабатываний для реализованных детекторов).

По диссертации могут быть сделаны следующие ***замечания***:

1. В работе упоминается, что используемые упрощающие предположения приводят как к ложным срабатываниям, так и к пропускам ошибок, однако детального анализа влияния упрощающих предположений не производится.
2. В четвертой главе описывается алгоритм нестрого построения резюме по результатам межпроцедурного анализа, однако отсутствует обоснование невозможности построения строго резюме.

Перечисленные замечания не влияют на положительную оценку диссертации и не ставят под сомнение полученные в ней результаты, которые представляют несомненный интерес, в научном и в практическом плане. Результаты диссертации своевременно опубликованы. Автореферат диссертации полно и правильно отражает ее содержание.

Диссертация была обсуждена на семинаре Отдела систем математического обеспечения Вычислительного центра им. А.А. Дородницына Федерального

исследовательского центра «Информатика и управление» Российской академии наук «__17__» апреля 2017 года.

Диссертация В.К. Кошелева является законченной научно-исследовательской работой и соответствует требованиям ВАК, предъявляемым к диссертациям на соискание ученой степени кандидата физико-математических наук по специальности 05.13.11 –математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей, соответствует п. 1. «Языки программирования, а также программные средства их реализации в вычислительных машинах (ВМ), вычислительных комплексах (ВК) и компьютерных сетях (КС)», п. 6. «Методы оценки показателей качества программных изделий, а также средств автоматизации контроля и приемки программ, ППП, ПК и СП», п.7. «Методы повышения надежности функционирования программ» паспорта специальности. Ее автор, Кошелев Владимир Константинович, заслуживает присуждения ему ученой степени кандидата физико-математических наук по указанной специальности.

Зав. отделом Систем Математического
Обеспечения Вычислительного
Центра ФИЦ ИУ РАН
Д.ф.-м.н. / В.А.Серебряков
тел.: +7 499 135 52 80

Специальность с расшифровкой: 05.13.11 "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей"

Адрес ведущей организации:

Москва, ул. Вавилова, 44 к.2, тел. 7-499-1356260, http://www.ipiran.ru,

e-mail: ipiran@ipiran.ru