СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

– КА - конечный автомат
– ООП - объектно-ориентированное программирование
– JVM - Java Virtual Machine
– MBT - Model Based Testing
– PBT - Property Based Testing
– ПО - программное обеспечение
– СА - статический анализ
– ДА - динамический анализ

ТЕРМИНОЛОГИЯ

– SMT-решатель

– условия корректности

ВВЕДЕНИЕ

С развитием области анализа программ при решении большого количества задач в этом направлении разработчики все чаще стали сталкиваться с необходимостью использования формальных спецификаций. Формальная спецификация - описание поведения программы на специльном или ее исходном языке, включающее в себя такие детали как пред и пост условия вызовов, состояния и переходы между ними, что и делает спецификации идеальным кандидатом для применения в области анализа ПО. Например, спецификации не заменимы в символьном исполнении. Там они используются для аппроксимации поведения внешних библиотек или сложных частей программы, тем самым ускоряя анализ или вовсе делая его возможным в определенных местах. Реализацию данного подхода можно увидеть в USVM[ССЫЛКА] и UtBot[ССЫЛКА]. Еще один пример использовация формальных спецификаций - Taint анализ, когда в них отмечаются потенциальные места ввода и утечки информации. Наличие подобных данных позволяет анализаторам сфокусироваться на проверке размеченных мест, представляющих возможные ошибки, тем самым сильно повышая эффективность[ЕСТЬ ЛИ ССЫЛКА НА ПРОЕКТ?]. Кроме этого, спецификации могут использоваться в MBT и PBT в качестве тестового оракула, отвечая на вопросы о корректности состояния ПО и переходов между ними, а также о том, удовлтеворяют ли входные и выходные данные для различых методов соответствующим предикатам.

При всем этом формальные спецификации в общем случае не поставляются с программными библиотеками или другим ПО, что заставляет задуматься о способах их создания. Полностью ручное составление спецификаций

это довольно монотонная работа, при этом требующая от человека высокой квалификации в области разработки и анализа программ. Однако составление спецификаций можно частично автоматизировать.

Часть формальной спецификации можно найти в исходном коде библиотек или ПО. Если приводить в пример языки в парадигме ООП, то это классы и интерфейсы программы, их поля и сигнатуры методов. Другую же часть спецификации, описывающую поведение программы, а именно состояния и переходы между ними, можно попытаться извлечь из реальных примеров использования.

Именно автоматизации извлечения поведенческих моделей библиотек посвящена данная работа. В рамках ее выполнения будет реализована утилита, позволяющая автоматически получать общедоступные Java проекты и с помощью методов статического и динамического анализа извлекать из них сценарии работы определенной библиотеки с целью последующего восстановления поведенческой модели в виде КА.

В первом разделе представлен обзор существующих решений, связанных с задачами поиска проектов, извлечением из них трасс вызовов и восстановлением модели. Также в данном разделе будет уделено внимание предшествующей работе по данной теме. На основе первого раздела сделан выбор в пользу определенных подходов и решений, используемых в реализации инструмента.

Во втором разделе формулируются требования к создаваемому инструменту и описываются пути решения каждой из задач, стоящих на пути реализации.

Третий раздел…

Четвертый раздел….

Пятый раздел….

В заключении…

1 АНАЛИЗ РЕШЕНИЙ В ОБЛАСТИ ВОССТАНОВЛЕНИЯ ПОВЕДЕНЧЕСКИХ МОДЕЛЕЙ

Первый абзац

1.1 Критерии обзора

1.2 Сравнение методов восстановления КА

1.3 Результаты сравнения

2 ПУТИ РЕШЕНИЯ

3 ПРОЕКТИРОВАНИЕ

4 РЕАЛИЗАЦИЯ

5 ТЕСТИРОВАНИЕ

ЗАКЛЮЧЕНИЕ