Целью работы является разработка метода статического поиска гонок в программах на языке Си. Решаемые задачи представлены на слайде.

Под ***состоянием гонки при доступе к разделяемой области памяти*** будем понимать ситуацию, когда несколько потоков одновременно совершают доступ к общей области памяти, и хотя бы один из них выполняет операцию записи в неё. На слайде показан пример, в котором возможно возникновение гонок при одновременном доступе к глобальной переменной count из разных потоков, т.к. доступ к ней не защищен никакими средствами взаимоисключения.

Существует 3 основные группы методов поиска гонок в программах:

* **Формальная верификация**, основная идея которой заключается в установлении соответствия между программой и требованиями к программе, описывающими цель разработки.
* **Динамические методы**, основанные на изучении потока событий, генерируемых программой во время выполнения. Недостаток - состояние гонки может быть зафиксировано, только если оно возникло в проверяемом варианте исполнения программы. Достоинство – если состояние гонки было зафиксировано, то оно, скорее всего, является таковым.
* **Статические методы**, основанные на анализе исходного кода программы. Достоинство - теоретическая возможность анализа всех возможных путей выполнения программы. Недостаток – много ошибок 2-го рода. Основные методы: **аннотирование кода**, которое выполняется за счёт добавления в исходный код программы специальных конструкций, содержащих информацию об объектах взаимоисключения, и **анализ потока выполнения**, осуществляемый на основе анализа путей выполнения программы. Разработанный статический метод поиска гонок основан на анализе потока выполнения.

Основная идея статического поиска гонок заключается в том, чтобы удостовериться, что для каждой общей области памяти существует, по крайней мере, одна блокировка, которая захватывается во всех потоках при доступе к этой области памяти. На вход методу даётся ГПУ функций программы, на выходе получается множество мест, в которых возможно возникновение гонок.

Разработанный метод состоит из 3 этапов:

* Построение путей выполнения функций
* Построение таблиц защищённого доступа для потоков
* Определение мест возможного возникновения гонок

Построение таблиц защищённого доступа для потоков состоит из 5 этапов. Сначала выполняется символьное исполнение каждого из путей для определения изменения состояний перекрестных ссылок во время выполнения каждого пути. Затем на основе этого выполняется вычисление относительных блокировок для каждой инструкции на каждом из путей. После анализа всех путей выполнения какой-либо функции для неё строится обобщение относительного множества блокировок, которое используется при анализе других функций, в которых производится её вызов. На основе путей, размеченных значениями перекрёстных ссылок и относительными множествами блокировок, строятся таблицы защищенного доступа для всех функций. После чего ищутся все места создания потоков. Для каждого из них на основе имеющихся таблиц защищенного доступа для функций и известных состояний перекрестных ссылок на момент создания производится конкретизация уже имеющихся таблиц для функций с учётом передаваемых при инициализации потоков значений.

Ограничения разработанного метода представлены на слайде.

Построение путей выполнения функций выполняется с ограничением количества раз, которое каждый базовый блок ГПУ может встречаться в пути. Алгоритм, используемой для этого функции walk, представлен на слайде. Для определения всех путей функции необходимо выполнить вызов функции walk с путём p, равным пустому списку, и блоком v, равным начальному базовому блоку.

Определение состояний перекрестных ссылок выполняется на основе символьного исполнения каждого из путей. Анализируемые ситуации приведены на слайде.

Относительное множество блокировок – это пара, состоящая из множества захваченных блокировок и множества освобожденных блокировок. Вначале анализа каждого пути относительное множество блокировок полагается пустым. Изменение множества блокировок производится только в момент вызова какой-либо функции. Для получения измененного множества блокировок используется функция lock\_update. Первым аргументом является текущее множество блокировок, которое получено к моменту вызова, вторым – обобщение относительно множества блокировок для вызываемой функции, в котором формальные параметры заменяются передаваемыми в функцию аргументами. После получения результирующих относительных множеств блокировок для каждого из путей для функции строится обобщение. Для этого применяется функция lock\_summary.

Под защищенным доступом понимается тройка, состоящая из области, к которой производится доступ, относительного множества блокировок и типа доступа. Вначале анализа путей таблица для функции полагается пустой. Затем в процессе анализа каждого из путей в неё добавляются записи, соответствующие доступам к глобальным переменным и областям, переданным в функцию через указатели. Также в неё добавляются записи из таблиц для вызываемых функций. Алгоритм добавления таких записей представлен на слайде. … + про конкретизацию для потоков. (еще не придумал как кратко описать)