Слайд 1. Тема работы.

Уважаемые члены государственной экзаменационной комиссии, вашему вниманию представляется работа на тему «Генерация тестов для синтаксического анализа методами суперкомпиляции».

В данной работе рассматривается техника генерации наборов тестов для проверки корректности работы предсказывающего синтаксического анализатора LL(1)-грамматик.

Слайд 2. Постановка задачи.

Основная задача формулируется следующим образом:

<Текст со слайда>

Слайд 3. Методы тестирования парсеров.

Под конечным набором тестов понимается набор из позитивных и негативных тестов. **Позитивные тесты** – тесты, проверяющие в совокупности максимум корректных путей в программе. В контексте синтаксического анализатора позитивный тест – это предложение языка. **Негативные тесты** – тесты, проверяющие работу программы при некорректных входных данных. В терминах синтаксического анализа это означает наличие некоторой синтаксической ошибки.

Методы генерации позитивных тестов можно разделить на стохастические, строящие случайные цепочки языка, и основанные на критериях покрытия. Основополагающими критериями являются критерий Пардома и более сильный критерий покрытия всех пар.

Методы генерации негативных тестов в основном основывается на методах мутаций позитивных тестов или входной грамматики.

Слайд 4. Предсказывающий анализатор.

Логика работы анализатора напрямую зависит от заполнения таблицы предсказывающего анализа. Если дополнить такую таблицу определенными правилами восстановления при ошибках, а также расширить грамматику путем введения фиктивных нетерминалов для каждого терминала, то становится возможным построение графа состояний распознавателя, из каждой вершины которого исходят дуги по всем возможным терминалам.

Слайд 5. Граф состояний анализатора.

Такой граф описывает общий процесс разбора входных цепочек. Вершина графа – состояние – есть содержимое магазина распознавателя, ребро – переход со снятием символа входного потока. Однако в общем случае такой граф является бесконечным (например, для скобочных структур).

Слайд 6. Суперкомпиляция. Этапы. Граф конфигураций.

Решить проблему свертки бесконечного графа помогают техники суперкомпиляции.

**Суперкомпиляция** – процесс анализа и преобразования программ, основанный на схеме метасистемного перехода. Под метасистемным переходом подразумевается переход от частного к общему, от некоторой программы к метапрограмме, следящей за выполнением исходной и строящей граф суперкомпиляции.

В процессе суперкомпиляции выделяют стадии построения дерева конфигураций, его свертки в конечный граф конфигураций с помощью выделения обобщения и вложения, а также построение на основе графа остаточной программы. Таким образом, свернутый граф состояний распознавателя можно трактовать как граф конфигураций.

Слайд 7. Вложение и обобщение.

Основными методами при свертке бесконечного графа являются вложение и обобщение.

Вложение возникает, когда в процессе построения графа конфигураций непустой префикс стека текущей вершины совпадает со стеком некоторой вершины-предка. В таком случае текущая вершина заменяется на, так называемую, let-вершину, обозначающую факт свертки в графе.

Обобщение, основанное на отношении Турчина для стеков, возникает, когда стек текущей вершины в некотором смысле повторяет развитие стека вершины-предка, то есть имеет место цикл или рекурсия. В таком случае текущий подграф отбрасывается, и вместо него создается let-вершина, из которой части стека развиваются независимо.

Слайд 8. Критерий полноты тестирования для графа.

Обходя граф конфигураций с помощью модифицированного поиска в ширину, помечая в процессе построения тестов ребра графа, можно получить позитивный и негативный наборы тестов.

Слайд 9. Реализация. Компоненты.

Стадия анализа входной грамматики позаимствована из курсового проекта студента кафедры ИУ9 Михаила Макарова. Принимая на вход грамматику в РНБФ, и определяя, является ли она LL(1), в случае положительного ответа анализатор строит таблицу предсказывающего разбора и некоторые дополнительные структуры, подающиеся на вход построителю графа. В свою очередь в результате построения граф подается на вход генераторам позитивных и негативных тестов.

Слайд 10. Тестирование.

Тестирование разработанного генератора тестов осуществлялось путем подачи ему на вход различного вида LL(1)-грамматик. Также был проведен тест самоприменимости для собственной грамматики.

В указанном примере показан процесс построения позитивных тестов для анализатора грамматики арифметических выражений. Жирным цветом обозначены пройденные в результате построения теста корректные ребра.

Построение негативных тестов происходит по аналогичной схеме, за исключением того, что требуется посетить все ошибочные ребра и не завершать построение в нефинальных вершинах.

Слайд 11. Заключение.

<Текст на слайде>

Слайд 12. Благодарности.

В завершение, выражаю благодарность руководителю моей работы Александру Владимировичу Коновалову, а также научному сотруднику Института программных систем РАН Антонине Николаевне Непейвода за помощь при написании данной работы.