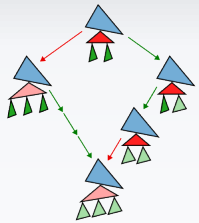


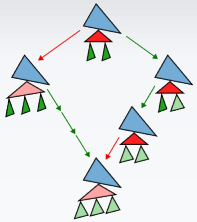
Лабораторная работа 3: Построение PDA по CFG

Вариант определяется последней цифрой в зачётке и вашим предыдущим вариантом.

(Номера 0,1,3,4,9) PDA на основе LL(k) ($1 \leqslant k \leqslant 3$)

(Номера 2,5,6,7,8) PDA на основе LR(0)

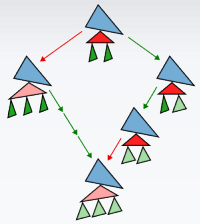
(МАТ в ЛР2) Фазз-модуль для PDA



Техническое задание

- Со стороны генераторов PDA — построить PDA по данной грамматике. Реализовать парсинг по PDA (причём полученный PDA может и не быть детерминированным).

- Со стороны фаззера — удалить цепные правила, сгенерировать наборы строк, принадлежащих и не принадлежащих языку входной грамматики в ХНФ.

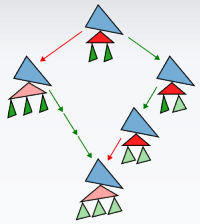


Входные данные

- КС-грамматика с правилами в следующем синтаксисе:

$$
\begin{array}{rcl}
[\mathtt{rule}] & ::= & [\mathtt{blank}]^*[\mathtt{NT}][\mathtt{blank}]^*->[\mathtt{blank}]^* \\
 & & ([\mathtt{NT}][\mathtt{blank}]^*|[\mathtt{T}][\mathtt{blank}]^*)^+[\mathtt{EOL}]^+ \\
[\mathtt{T}] & ::= & [a-z] \\
[\mathtt{NT}] & ::= & [A-Z][0-9]?|[[A-z]^+([0-9])^*]
\end{array}
$$

- Вариант LL(k) должен также запрашивать в качестве параметра запуска значение k (при нулевом сообщать об ошибке).
- Варианты, генерирующие PDA, должны сохранять его правила перехода в файле, который можно передать в режим парсера вместе со строкой, которую необходимо разобрать.
- Фазз-модуль должен генерировать наборы строк по запросу: число тестов с разметкой (положительные или отрицательные) и иметь возможность настройки.



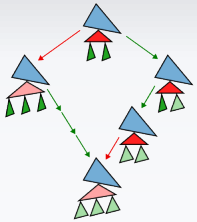
Генерация тестов

Тесты генерируются случайным блужданием по матрице биграмм терминалов. Матрица строится на основе классических множеств First, Follow для нетерминалов, а также инверсных к ним множеств Last и $\mathrm{Precede}$.

Биграмма (γ_1, γ_2) добавляется в матрицу, если выполняется хотя бы одно из следующих условий

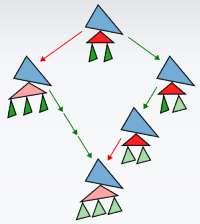
- $\gamma_1\gamma_2$ встречается в правой части правила грамматики;
- $\exists A_1(\gamma_1 \in \mathrm{Last}(A_1) \& \gamma_2 \in \mathrm{Follow}(A_1))$;
- $\exists A_2(\gamma_1 \in \mathrm{Precede}(A_2) \& \gamma_2 \in \mathrm{First}(A_1))$;
- $\exists A_1, A_2(\gamma_1 \in \mathrm{Last}(A_1) \& \gamma_2 \in \mathrm{First}(A_1) \& A_2 \in \mathrm{Follow}(A_1))$.

Терминалы из $\mathrm{First}(S)$ считаются стартовыми, терминалы из $\mathrm{Last}(S)$ считаются финальными. С маленькой вероятностью генератор может выбрать очередной терминал, не согласованный с матрицей биграмм.



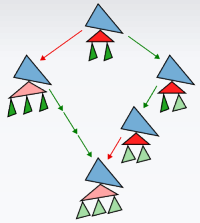
Разметка тестов

После генерации тест нужно классифицировать, т.е. определить, принадлежит ли данная строка языку грамматики. Для этого можно использовать любой алгоритм парсинга строки по произвольной грамматике, включая обычный алгоритм поиска в глубину с возвратами.



Построение и валидация PDA по CFG

- PDA строится на базе алгоритма разбора грамматики соответствующего класса (LL(k)-грамматик — таблицы, LR(0)-грамматик — позиционного автомата), но с учётом возможных конфликтов. В случае конфликтов во входных грамматиках в построенном PDA может возникать недетерминизм.
- Если грамматика не содержит конфликтов, то соответствующий PDA должен получиться детерминированным. Например, если в варианте 1 дано k=2 и LL(2)-грамматика, то результат работы алгоритма должен быть детерминированным (в смысле определения DPDA). Проверять детерминизм необходимо автоматически, так же, как и требуемое свойство грамматики.



Фазз-тестирование

Варианты, строящие PDA, должны также иметь режим тестирования, в котором в главной итерации у фазз-модуля запрашивается определённое количество тестов и проверяются результаты их разбора в PDA на предмет совпадения с данными от фазз-модуля.
Дополнительно тестирующий режим может просто запросить файл с размеченными тестами (без интерактива с фазз-модулем) и также проверить их на совпадение с разметкой. Формат размеченных тестов:

$$([\text{строка}][\texttt{blank}]^{+}[0|1][\texttt{blank}]^{*}[\text{EOL}]^{+})^{*}$$

Здесь 0 — свидетельство о том, что тест не распознаётся грамматикой, 1 — что распознаётся.