



Дополнительное задание 1

Полностью обесценится 6 декабря. Стоимость 3 балла.

- Написать реализацию Follow_1 и First_1 множеств для входной грамматики.
- На её основе построить реализацию нахождения First_k множества.

Самые быстрые реализации First_k (по результатам сдачи основного задания Л.Р.) получают ещё 1–3 балла.



Дополнительное задание 2

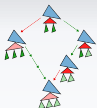
Полностью обесценится 10 декабря. Стоимость 3 балла.

- Установить какой-нибудь генератор парсеров для вашего любимого языка (но не YACC и не Bison).
- Разобраться, какие ограничения этот генератор накладывает на входную грамматику, и написать об этом краткий отчёт.
- Написать в языке этого генератора грамматику для языка академической регулярки.



Лабораторная номер 4

- ❶ ($\equiv 0 \pmod 3$) По данной контекстно-свободной грамматике построить для неё SLR-автомат с переменным lookahead-ом.
- ❷ ($\equiv 1 \pmod 3$) Определение аппроксимации синхронизирующих токенов.
- ❸ ($\equiv 2 \pmod 3$) По данной контекстно-свободной грамматике и слову из её языка построить её LL(1)-эквивалент и аппроксимации AST.



SLR-автомат

- ❶ Число N , ограничивающее lookahead, читается из того же источника, что и грамматика. Полагаем $k = 1$.
- ❷ Строим LR(0)-автомат (с эндмаркером!). Ищем конфликты.
- ❸ Конфликтов нет или $k > N$ — завершаем работу.
 - ❶ Есть конфликт вида $A_1 \rightarrow \gamma_1 \bullet$, $A_2 \rightarrow \gamma_2 \bullet$. Строим $F_1 = \text{Follow}_k(A_1)$, $F_2 = \text{Follow}_k(A_2)$.
 - ❷ Есть конфликт вида $A_1 \rightarrow \gamma_1 \bullet$, $A_2 \rightarrow \gamma_2 \bullet \gamma_3$. Строим $F_1 = \text{Follow}_k(A_1)$, $F_2 = \text{First}_k(\text{First}_k(\gamma_3) ++ \text{Follow}_k(A_2))$.
- ❹ Если $F_1 \cap F_2 = \emptyset$, то конфликт разрешён. Отбрасываем конфликт.
- ❺ Если $F_1 \neq F_2$, но $F_1 \cap F_2 \neq \emptyset$, то увеличиваем k на 1 и перестраиваем lookahead-множества во всех неразрешённых (или подвешенных) конфликтах (переходим на шаг 3).
- ❻ Если $F_1 = F_2$ — подвешиваем конфликт (переходим к другому).
- ❼ Если все конфликты остались только подвешенными (т.е. при актуальном lookahead у всех $F_1 = F_2$), то k уже не увеличиваем.
- ❽ Результат: LR(0)-автомат, у которого конфликты помечены как подвешенные при некотором k , либо как разрешённые (при минимально найденном k), либо как не разрешённые при $k = N$.



Дополнительные задания

- Вместо LR(0)-автомата строить LR(1)-автомат (т.е. в каждой ситуации $A_i \rightarrow \gamma_1 \bullet A_j \gamma_2$ строить свой собственный lookahead по γ_2 при переходе внутрь разбора A_j). На LR(1)-автоматы можно посмотреть здесь: [Grammophone](#) (+3 балла).
- В случае наличия подвешенных либо неразрешённых конфликтов пытаться сделать присоединение правого контекста (не более N раз по совокупности; после такого присоединения придётся перестраивать автомат и опять полагать $k = 1$) (+4 балла)



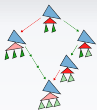
Определение кандидатов на синхронизирующие токены

- 1 Выявить нетерминалы, определяющие регулярные языки (через замыкания праволинейности и леволинейности). Объединить эти нетерминалы в множество M — кандидатов. Также добавить в это множество все терминальные символы (определяющие язык только из себя). Выбросить из M нетерминалы, язык которых содержит ε . Вывести множество кандидатов на синхронизирующие токены.
- 2 Для каждого $A_i \in M$ построить множество $NTPrecede(A_i)$ его precede-нетерминалов (т.е. нетерминалов, которые могут предшествовать ему в сентенциальной форме).
- 3 Вывести множество пар $\langle A_i, A_j \rangle$ таких, что $A_j \in NTPrecede(A_i)$ и $\mathcal{L}(A_i) \cap \mathcal{L}(A_j) = \emptyset$.



Дополнительные баллы

- (+2 балла) В подзадаче выявления регулярных нетерминалов подключить мюресы с контекста (проверочным будет в том числе тест Жука). Если на вашем языке мюресов не было, то +4 балла.
- (+4 балла) Вместо простого условия $\mathcal{L}(A_i) \cap \mathcal{L}(A_j) = \emptyset$ потребовать сразу два условия:
 $\mathcal{L}(A_i) \cap \mathcal{L}_{\text{suffix}}(A_j) = \emptyset$ и
 $\text{First}_k(A_i) \cap \text{Last}_k(\text{Precede}_k(A_j) \cup \text{Last}_k(A_j)) = \emptyset$,
где k — число нетерминалов в грамматике для A_j .
 $\mathcal{L}_{\text{suffix}}(A_i)$ — язык суффиксов слов из $\mathcal{L}(A_i)$. Здесь First, Last, Precede понимаются уже в контексте терминальных языков, а не сентенциальных форм.



LL-преобразования и аппроксимация AST

Скажем, что грамматика G хорошая, если:

- все её правила имеют длину правой части, не превышающую 3 (после расщепления по альтернативам);
- ϵ -правил нет;
- если правая часть правила состоит только из нетерминалов, то хотя бы один из них имеет конечный язык.

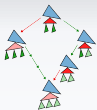
На вход подаётся грамматика G и строка T из $\mathcal{L}(G)$.

В этом варианте предполагается, что на вход подаются только хорошие грамматики. Если это не так, следует сообщить об этом и не выполнять дальнейшие действия. Если $T \notin \mathcal{L}(G)$, об этом также следует сообщить и не выполнять дальнейшие действия.



Общая схема

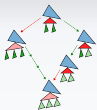
- Построить аппроксимацию AST для произвольного левостороннего разбора T по исходной грамматике G (этих разборов может быть несколько, сгодится любой).
- Произвести устранение левой рекурсии и присоединение левого контекста в G , построить G' .
- Проверить, что G' — LL(1). Независимо от результата, перейти к следующему шагу.
- Проверить, что G' — хорошая. Если это не так, следующий шаг не делать. Иначе построить аппроксимацию AST для разбора T по грамматике G' .



Построение аппроксимации AST

Алгоритм принимает на вход дерево разбора. Итогом работы этого алгоритма является дерево, в котором все узлы помечены терминальными символами. Алгоритм работает от корня вниз.

- Если потомки узла были построены по правилу $A \rightarrow T$ (где T — единственный нетерминал), то сразу же заменяем метку узла (с A) на результат разбора его потомка.
- Если потомки узла были построены по правилу $A \rightarrow \Phi$, где в Φ все нетерминалы имеют конечный язык, тогда просто заменяем метку узла A на терминальную строку, в которую он был разобран.



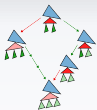
Построение аппроксимации AST

- Если потомки узла были построены по правилу $A \rightarrow T_1 T_2$ (где T_1 имеет конечный язык, а T_2 нет), то заменяем метку узла A на результат разбора T_1 , и рекурсивно продолжаем преобразование в потомке T_2 . Аналогично, если $A \rightarrow T_2 T_1$.
- Если потомки узла были построены по правилу $A \rightarrow T_1 T_2 T_3$, где один из T_i , скажем, T_1 имеет конечный язык, а два нет, то заменяем метку узла A на результат разбора T_1 , и рекурсивно продолжаем преобразование в потомках T_2 и T_3 .
- Если применялось правило $A \rightarrow T_1 T_2 T_3$, где T_2 имеет бесконечный язык, а T_1 и T_3 конечный, тогда заменяем метку A на разбор T_1 , полагая ему два потомка: T_1 и T_3 , в которых продолжаем преобразование.
- В противном случае, если два из трёх нетерминалов в правиле имеют конечные языки, то они стоят рядом, и меткой вместо A полагаем конкатенацию их разбора (оставляя только одного потомка).



Дополнительные баллы

- (+3 балла) Ослабить условие «хорошей» грамматики: разрешить продукции вида $A_i \rightarrow BC$, если у B и C оба языка бесконечные, но за один шаг переписывания хотя бы один из них обязательно порождает терминал или нетерминал с конечным языком. Добавить соответствующее правило перестройки для AST.
- (+4 балла) Добавить лексер: выявить в исходной грамматике множество регулярных нетерминалов (по право- или левوليнейности) $M = \{R_1, \dots, R_k\}$, поставить в соответствие каждому такому нетерминалу константный токен r_i и в дереве разбора для T заменить поддеревья разбора R_i на константы r_i . При этом изменяется понятия «хорошей» грамматики. Грамматика «хорошая с лексером», если в каждой правой части есть хотя бы один нетерминал с регулярным языком.



Синтаксис грамматики для всех вариантов

Чёрным обозначены элементы метаязыка, красным — элементы языка входных данных. Расстановка пробелов произвольна, могут встречаться табуляции, новая строка может начинаться с \n или с \r\n. Начальный нетерминал — [S].

Синтаксис входных данных КС-грамматики (появилась возможность включать альтернативы в правила правой части):

$\langle \text{grammar} \rangle$::=	$\langle \text{rule} \rangle^+$
$\langle \text{rule} \rangle$::=	$\langle \text{nterm} \rangle \text{-} \rightarrow (\langle \text{term} \rangle \mid \langle \text{nterm} \rangle)^* ((\langle \text{term} \rangle \mid \langle \text{nterm} \rangle)^+)^*$
$\langle \text{term} \rangle$::=	$[A-z] \mid [0-9]$
$\langle \text{nterm} \rangle$::=	$[[A-z]^+ [0-9]^*]$



Слайд для красоты

- (Первый вариант) × 2 Подготовка документации по методам фреймворка.
- (Второй вариант) (на троих!) Автоматическая проверка адекватности документации по связыванию местоименных конструкций.
- (Третий вариант) Авторендер диаграммы по псевдокомментариям.
- (Четвертый вариант) Подгонка размера тайпсета по сообщениям об ошибках.



Уже не предупреждение,
а прямая угроза

Если кто-то оставит строковое
представление грамматик на
бэкенде, тот записывается в ряды
латентных рефальщиков и делает
5 лабораторную только на Рефале.