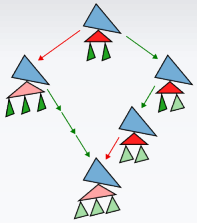
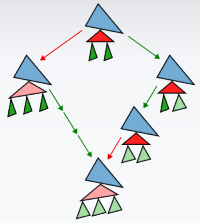


Дополнительное задание 1

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.

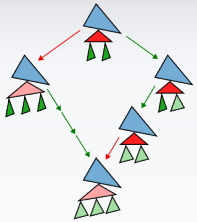
Самые быстрые реализации First_k (по результатам сдачи основного задания Л.Р.) получат ещё 1–3 балла.



Дополнительное задание 2

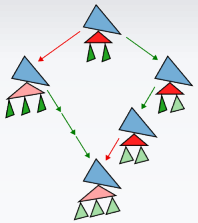
Полностью обесценится 10 декабря. Стоимость 3 балла.

- 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.



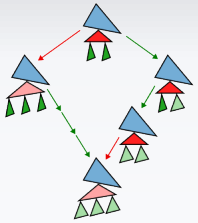
Лабораторная номер 4

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



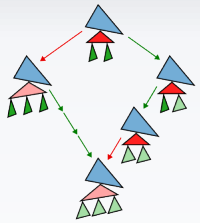
SLR-автомат

1. Число N, ограничивающее lookahead, читается из того же источника, что и грамматика. Полагаем $k = 1$.
2. Строим LR(0)-автомат (с эндмаркером!). Ищем конфликты.
3. Конфликтов нет или $k > N$ — завершаем работу.
    3. Есть конфликт вида $A_1 \to \gamma_1\bullet$, $A_2 \to \gamma_2\bullet$. Строим $F_1 = \mathsf{Follow}_k(A_1)$, $F_2 = \mathsf{Follow}_k(A_2)$.
    3. Есть конфликт вида $A_1 \to \gamma_1\bullet$, $A_2 \to \gamma_2 \bullet \gamma_3$. Строим $F_1 = \mathsf{Follow}_k(A_1)$, $F_2 = \mathsf{First}_k(\mathsf{First}_k(\gamma_3) +\!+ \mathsf{Follow}_k(A_2))$.
4. Если $F_1 \cap F_2 = \varnothing$, то конфликт разрешён. Отбрасываем конфликт.
5. Если $F_1 \neq F_2$, но $F_1 \cap F_2 \neq \varnothing$, то увеличиваем k на 1 и перестраиваем lookahead-множества во всех неразрешённых (или подвешенных) конфликтах (переходим на шаг 3).
6. Если $F_1 = F_2$ — подвешиваем конфликт (переходим к другому).
7. Если все конфликты остались только подвешенными (т.е. при актуальном lookahead у всех $F_1 = F_2$), то k уже не увеличиваем.
8. 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 $k = N$.



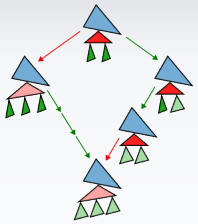
Дополнительные задания

- Вместо LR(0)-автомата строить LR(1)-автомат (т.е. в каждой ситуации $A_i \to \gamma_1 \bullet A_j\gamma_2$ 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)



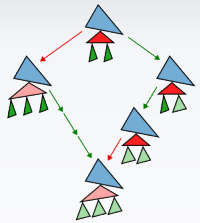
Определение кандидатов на синхронизирующие токены

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 $\mathsf{NTPrecede}(A_i)$ 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 $A_j \in \mathsf{NTPrecede}(A_i)$ и $\mathscr{L}(A_i) \cap \mathscr{L}(A_j) = \varnothing$.



Дополнительные баллы

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

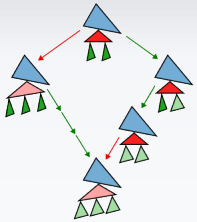


LL-преобразования и аппроксимация AST

Скажем, что грамматика G хорошая, если:

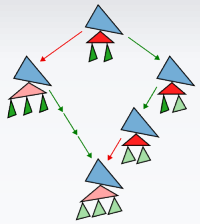
- все её правила имеют длину правой части, не превышающую 3 (после расщепления по альтернативам);
- ε-правил нет;
- если правая часть правила состоит только из нетерминалов, то хотя бы один из них имеет конечный язык.

На вход подаётся грамматика G и строка T из $\mathscr{L}(G)$.
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 $T \notin \mathscr{L}(G)$, об этом также следует сообщить и не выполнять дальнейшие действия.



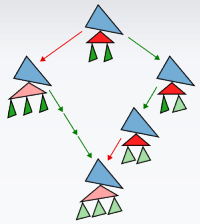
Общая схема

- 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'.



Нетерминалы с константными языками

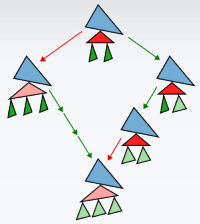
- В итоговой грамматике могут появиться правила длины больше 3, из-за конкатенации нескольких нетерминалов с константными языками (или констант). Например, так:
  $[\mathrm{S}] \to \mathtt{ab}[\mathrm{S}][\mathrm{C}]\mathtt{d}$
  $[\mathrm{C}] \to \mathtt{cda} \mid e$
- Такие конкатенации нужно объявить правыми частями новых нетерминалов. Для их именования можно использовать символы, не входящие в алфавит исходных нетерминалов (это позволит избежать конфликтов имён). В примере выше, например, так:
  $[\mathrm{S}] \to [\mathtt{a} +\!\!+ \mathtt{b}][\mathrm{S}][\mathrm{C} +\!\!+ \mathtt{d}]$
  $[\mathrm{C} +\!\!+ \mathtt{d}] \to \mathtt{cdad} \mid e\mathtt{d}$
- В последней грамматике тоже есть правило, длина которого больше 3, но оно состоит только из констант. Такие правила уже нет смысла разбивать дальше (т.е. можно, но не обязательно), т.к. это никак не повлияет на структуру AST.



Построение аппроксимации AST

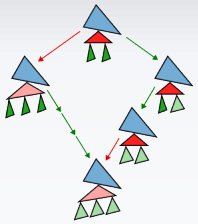
Алгоритм принимает на вход дерево разбора. Итогом работы этого алгоритма является дерево, в котором все узлы помечены терминальными символами. Алгоритм работает от корня вниз.

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



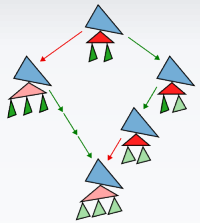
Построение аппроксимации AST

- Если потомки узла были построены по правилу $A \to T_1 T_2$ (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 $A \to T_2 T_1$.
- Если потомки узла были построены по правилу $A \to T_1 T_2 T_3$, 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 $A \to T_1 T_2 T_3$, 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 $A_i \to BC$, 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 леволинейности) $M = \{R_1, \ldots, R_k\}$, 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].

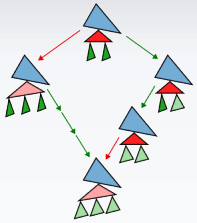
Синтаксис входных данных КС-грамматики (появилась возможность включать альтернативы в правила правой части):

⟨grammar⟩ ::= ⟨rule⟩$^{+}$

⟨rule⟩ ::= ⟨nterm⟩-> (⟨term⟩ | ⟨nterm⟩)*(|(⟨term⟩ | ⟨nterm⟩)$^{+}$)*

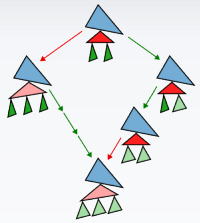
⟨term⟩ ::= [A–z] | [0–9]

⟨nterm⟩ ::= [[A–z]$^{+}$[0–9]*]



Слайд для красоты

- (Первый вариант)$\times$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.