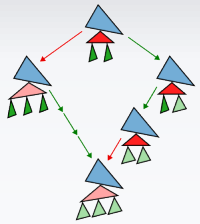
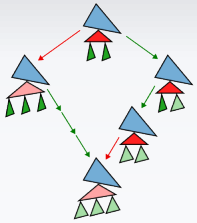


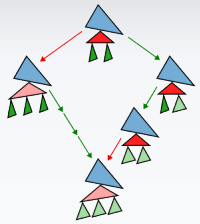
# Вариант 6, 7, 8

- Реализовать SLR(1)-разбор слова (входное данное 2) по грамматике (входное данное 1), синтаксическое дерево строить не обязательно. Грамматика может обрабатывать многострочные данные, для символа перевода строки в грамматике используется токен $. Для пробела — токен _.
- Реализовать обработку ошибок в режиме паники.
- Результат работы программы: сообщение об успешном разборе строки, либо сообщение о неуспешном разборе с указанием позиций ошибок (т.е. номеров символов в строке, на которых парсер перешёл в режим паники, либо пары номеров строки и позиций в строке, если слово многострочное). Ещё один возможный результат работы программы: сообщение о некорректности грамматики (в т.ч. если она не обладает SLR(1)-свойством).



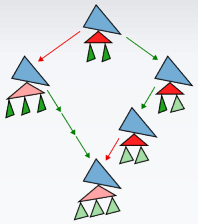
# Соло версия

- Реализовать стратегию восстановления после ошибок, при которой синхронизация осуществляется по первому попавшемуся правилу (без учёта старшинства), и только однострочных слов.
- Если в соло версии реализуется весь функционал ЛР, то она стоит 10 базовых баллов вместо 8.



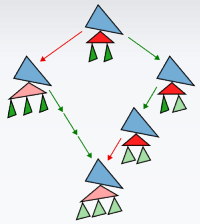
# Синхронизация и приоритеты

- Синхронизирующим токеном для $N_i$ считаем $\gamma \in FOLLOW(N_i)$.
- Нетерминал $N_i$ считаем старшим для $N_j$, если в любом дереве разбора, содержащем узел разбора $N_j$, какой-нибудь узел разбора $N_i$ обязательно является его предком.
- Правило $N_i \to \Phi_1 \bullet$ считаем старше правила $N_i \to \Phi_2 \bullet$, если $|\Phi_1| > |\Phi_2|$.
- Приоритет «от старшего к младшему»: для свёртки выбирается старшее правило для самого старшего нетерминала. От младшего к старшему — наоборот. Приоритет выбора должен определяться ключом запуска программы.



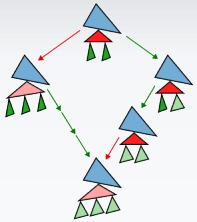
# Режим паники

- Если ячейка таблицы разбора, соответствующая состоянию $k$ и символу $\gamma$, пуста, тогда считаем, что SLR-автомат перешёл в состояние паники, позиции которого получаются из позиций состояния $k$ так: правило $N_i \to \Phi \bullet \Psi$ становится $N_i \to \Phi \bullet \bot$, где $\bot$ — особый символ «ошибки».
- В состоянии паники с ленты «впустую» читаются символы (без изменения стека) до тех пор, пока не будет прочитан синхронизирующий токен $\gamma$ такой, что $\exists i(\gamma \in \mathrm{FOLLOW}(N_i))$. При этом правило обрабатывается как свёртка по $N_i \to \Phi\bot\bullet$, где $\bot$ считается строкой нулевой длины (т.е. скидывается $|\Phi|$ символ со стека и осуществляется переход по GOTO $N_i$).
- Если оказалось, что можно осуществить свёртку по нескольким правилам для одного и того же $N_i$, тогда выбираем самое высокоприоритетное. Аналогично — если можно осуществить свертку для разных $N_i$, $N_j$.



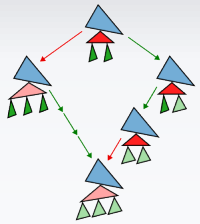
# Вариант 2, 4, 5

- Реализовать LL(1)-разбор слова $\omega_0$ (входное данное 2) по грамматике (входное данное 1) с построением синтаксического дерева. Слова могут быть и многострочными, см. условие предыдущего варианта.
- Реализовать инкрементальный разбор слова $\omega_1$, полученного из $\omega_0$ редактированием, в строгой либо экономной стратегии (контролируется ключом).
- Результат работы программы: деревья разбора для $\omega_0$ и $\omega_1$, в которых узлы помечены именами, и имена переиспользованных деревьев для $\omega_1$ совпадают с таковыми для $\omega_0$. Ещё один возможный результат: сообщение, что $\omega_0$ либо $\omega_1$ не принадлежит языку грамматики (без отчёта об ошибках), либо сообщение, что входная грамматика — не LL(1).



# Соло версия

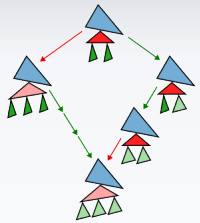
- Реализовать инкрементальный разбор с переходом только на самую правую ветвь общего суффикса, и только однострочных слов, и только в экономной стратегии.
- Если в соло версии реализуется весь функционал ЛР, то она стоит 10 базовых баллов вместо 8.



# Алгоритм инкрементального разбора

Позиция узла $N$ в синтаксическом дереве — позиция буквы в слове $\omega$, начиная с которой происходит разбор дерева с корнем в $N$.
Ближайший правый сосед узла $N$ — ближайший справа сиблинг $N$, либо, если $N$ самый правый потомок своего родителя, то ближайший справа сиблинг родителя $N$.

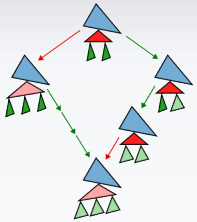
Обозначим за $s$ длину рассматриваемого общего суффикса (которая в первый момент равна $|z| - 1$). Пусть $xyz$ — слово из $\mathscr{L}(G)$ до коммита, порождающее дерево $T_0$; $xy'z$ — после коммита (для которого строим дерево $T_1$). Находим в $T_0$ узел с позицией $|x|$ и переносим левое поддерево до этой позиции включительно в $T_1$. Далее действуем итеративно.



# Алгоритм инкрементального разбора

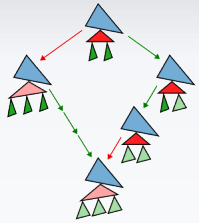
Обозначим за $s$ длину рассматриваемого общего суффикса (которая в первый момент равна $|z| - 1$). Пусть $xyz$ — слово из $\mathscr{L}(G)$ до коммита, порождающее дерево $T_0$; $xy'z$ — после коммита (для которого строим дерево $T_1$). Находим в $T_0$ узел с позицией $|x|$ и переносим левое поддерево до этой позиции включительно в $T_1$. Далее действуем итеративно.

- Производим LL(1)-разбор, достраивая $T_1$ до узла $N'_m$ с позицией $|xy'z| - s$ (т.е. $|xy'| + 1$ на первой итерации алгоритма).
- Находим в $T_0$ узел $N_m$ с позицией $|xyz| - s$ ($|xy| + 1$ на первой итерации). Если в узлах $N'_m$ и $N_m$ стоит один и тот же нетерминал, то переносим поддерево $T'$ его разбора из $T_0$ в $T_1$ и уменьшаем $s$ на длину строки, разобранной в дереве $T'$ (то есть переходим к ближайшему правому соседу корня $T'$ в $T_0$), после чего по необходимости повторяем итерацию.
- Если в $N'_m$ и $N_m$ стоят разные нетерминалы, тогда уменьшаем $s$ на 1 (строгая стратегия) или на длину инфикса, разобранного в поддереве с корнем $N_m$ (т.е. переходя к его ближайшему правому соседу), и повторяем итерацию.



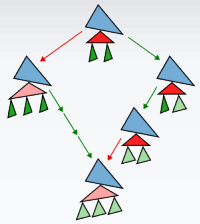
# Вариант 0, 1, 3, 9

- Реализовать LR(0)-разбор слова (входное данное 2) по грамматике (входное данное 1), синтаксическое дерево строить не обязательно. Слова могут быть и многострочными, см. условие предыдущего варианта.
- Реализовать обработку ошибок по анализу недопустимых инфиксов в двух стратегиях: с восстановлением и без него.
- Результат работы программы: сообщение об успешном разборе строки, либо сообщение о неуспешном разборе с указанием позиций ошибок (т.е. номеров символов в строке, начиная с которых обнаружились недопустимые префиксы, суффиксы или инфиксы — в последнем случае требуется вывести интервал, в котором находится проблемный инфикс). Ещё один возможный результат работы программы: сообщение о некорректности грамматики (в т.ч. если она не обладает LR(0)-свойством).



# Соло версия

- Обойтись без LR(0)-разбора (сделать только парсер Эрли) и проверки на LR(0), и без многострочных слов. Использовать только стратегию без восстановления.
- Если в соло версии реализуется весь функционал ЛР, то она стоит 10 базовых баллов вместо 8.

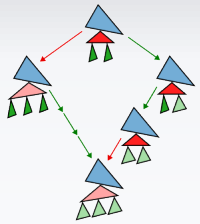


# Анализ недопустимых инфиксов

Разборы инфиксных и суффиксных (реверсированных) грамматик осуществляем посредством алгоритма Эрли. Здесь $\mathrm{FOLLOW}_\forall(G, \xi)$ — множество токенов, которые могут идти после токена $\xi$ в сентенциальной форме грамматики $G$.

- Осуществить LR(0)-разбор слова $\omega$ по грамматике $G$ до первой неудачи в позиции $i$. При этом положить список нетерминалов для разбора $M_0$ равным $M$, где $M$ — множество всех нетерминалов в левых частях позиций того состояния, где произошла ошибка, если стратегия — с восстановлением (стратегия 1); и $\{S\}$ (стартовый нетерминал) — в стратегии без восстановления (стратегия 2).
- Осуществить разбор реверса слова $\omega^R$ по реверсированной грамматике $G^R$ до первой неудачи в позиции $j$. Если $j > |\omega| - i + 1$, тогда положить $k_0 = k_{last} = |\omega| - j + 1$. Если $j \leqslant |\omega| - i + 1$, положить $k_0 = i$, $k_{last} = j$.

Сообщить о неудаче разбора в позиции $k_0$. Далее разбор переходит к анализу инфиксов.



# Анализ недопустимых инфиксов

Положить $i = 0$.

- Если $k_{last} \leqslant k_i$, то завершить поиск ошибок.
- В противном случае найти максимальный корректный инфикс $\omega_i$ языка нетерминалов из множества $M_i$, начинающийся в $k_i + 1$-позиции и заканчивающийся перед позицией $j$. Полагаем $k_{i+1} = j$.
  - (Стратегия 1) Если $\omega_i$ не является элементом языка суффиксов $M_i$, тогда объявляем о возможной ошибке в позиции $k_{i+1}$. Если $\omega_i$ гарантированно является суффиксом языка нетерминалов из множества $M_i$ (то есть не может являться точным инфиксом никакого нетерминала из $M_i$), то полагаем $M_{i+1} = \mathrm{FOLLOW}_\forall(G, M'_i)$, где $M'_i$ — подмножество $M_i$, допускающее $\omega_i$ в качестве суффикса. Иначе полагаем $M_{i+1} = M_i \cup \mathrm{FOLLOW}_\forall(G, M_i)$.
  - (Стратегия 2) Сообщаем об ошибке в позиции $j$. $M_{i+1} = \{S\}$ (то есть продолжаем отслеживать инфиксы всего языка грамматики $G$).