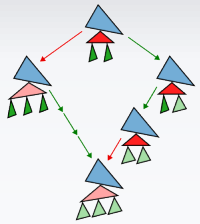
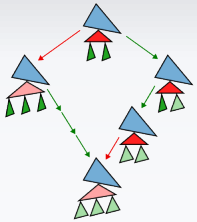


Лабораторная работа 1

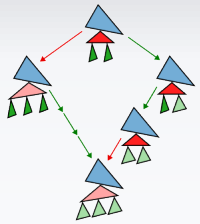
- (варианты 0, 1, 2, 3, 4) Реализовать алгоритм развертки на n шагов терма по правилам переписывания, определённым TRS. Развертка может быть неоднозначной, поэтому результат — список всех возможных результатов переписывания.
- (варианты 5, 6, 7, 8, 9) Реализовать алгоритм проверки неразрешимости соответствия Поста при анализе образов строк в линейной целочисленной арифметике.



Лабораторная работа 2

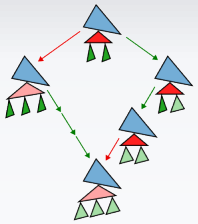
- (варианты 0, 1, 2, 3, 4) Дано регулярное выражение. Определить, является ли синтаксический моноид его языка апериодическим с периодом меньше k, и содержит ли он циклы размерности меньше k. Входные данные: регулярное выражение и значение k.
- (варианты 5, 6, 7, 8, 9) По конечному автомату методом исключения состояний построить регулярное выражение так, чтобы порядок исключения состояний был оптимальный (т.е. алфавитная длина итоговой регулярки была минимальной среди всех, получающихся из данного автомата методом исключения состояний в ином порядке). Конечный автомат описывается набором: стартовое состояние, список финальных состояний через запятую, список правил каждое с новой строки. Пример описания:

```
Q
Q1, Q3
(Q,a) -> Q1
(Q1,a) -> QQ
(Q1,b) -> Q3
(Q3,b) -> Q
(QQ,b) -> Q1
```



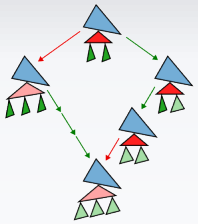
Лабораторная работа 4

- (варианты 0, 1, 2, 3, 4) По грамматике G и регулярке R построить описание слов до длины k в регулярном языке, принадлежащих языку грамматики.
- (варианты 5, 6, 7, 8, 9) По SLR(1)-грамматике построить PDA для языка её дополнения.



ЛР 1-1

- TRS задаётся как в лабораторной номер 1 варианты 2, 4, 5, но без ограничения на местность конструкторов.
- Значение n и входное слово для переписывания — дополнительные параметры.
- Исходный терм может переписываться с любой позиции, например, $g(x, x) \to h(h(x))$ может превращать за один шаг переписывания $g(g(t, t), g(t, t))$ как в $h(h(g(t, t)))$, так и в $g(h(h(t)), g(t, t))$. $g(g(t, t), h(h(t)))$.

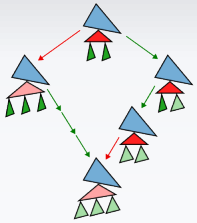


ЛР 1-2

- ПСП (списки домино) — список (Φ_1, Φ_2), перечисленных каждая с новой строки. Например
  `(ab,a)`
  `(aa,b)`
  `(bb,a)`
  Считаем, что нет домино с пустыми позициями (т.е. в каждой паре оба слова содержат хотя бы одну букву).
- Если ПСП имеет решение W, то в этом решении определены константы M_i — количество вхождений i-го домино, а также $N_{i,j}$ — количество соседств домино типа i от домино типа j слева. Их значения определяются, в том числе, уравнениями на число вхождений букв в слово W, а также пар букв в слово W. Например, для пары домино выше в каждом решении должны выполняться равенства:

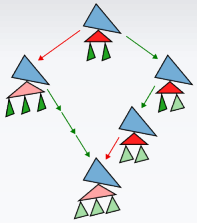
$$\begin{aligned} &(a) \quad 2 \cdot M_2 = M_3 \\ &(b) \quad M_1 + 2 \cdot M_3 = M_2 \end{aligned}$$

  В целых положительных значениях M_i решений этой системы не существует, поэтому ПСП решения не имеет.



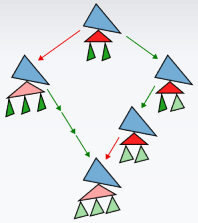
ЛР 1-2

- ПСП (списки домино) — список (Φ_1, Φ_2), перечисленных каждая с новой строки.
  Считаем, что нет домино с пустыми позициями (т.е. в каждой паре оба слова содержат хотя бы одну букву).
- Если ПСП имеет решение W, то в этом решении определены константы M_i — количество вхождений i-го домино, а также $N_{i,j}$ — количество соседств домино типа i от домино типа j слева. Их значения определяются, в том числе, уравнениями на число вхождений букв в слово W, а также пар букв в слово W.
- Используя SMT-решатель, проверить, существуют ли подходящие значения M_i и $N_{i,j}$, построив SMT-модель, описывающую уравнения на них в smtlib-языке (для $N_{i,j}$ ещё нужно определить уравнения, связывающие их число с M_i и M_j, и учесть, что крайние домино соседствуют только с одним).



ЛР 4-1

- Удалить в G недостижимые и непорождающие правила.
- Построить пересечение грамматики G с дополнением регулярного языка R. Если это пересечение непусто, вывести предупреждение, что язык G не полностью накрывается языком R.
- Пусть Σ — алфавит языка грамматики G. Задать на этом алфавите упорядочение $\preceq$ на буквах. Армейский (military) порядок $\preceq_{\mathrm{m}}$ на строках определяется следующим образом:
  - если $|w_1| < |w_2|$, тогда $w_1 \preceq_{\mathrm{m}} w_2$;
  - если $|w_1| = |w_2|$, тогда $w_1 \preceq_{\mathrm{m}} w_2 \Leftrightarrow w_1 \preceq_{\mathrm{lex}} w_2$ (т.е. в случае равенства длин слова сравниваются лексикографически)

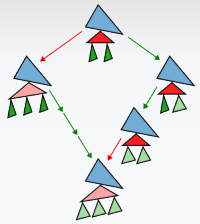


ЛР 4-1

- Удалить в G недостижимые и непорождающие правила.
- Построить пересечение грамматики G с дополнением регулярного языка R. Если это пересечение непусто, вывести предупреждение, что язык G не полностью накрывается языком R.
- Используя k в качестве верхней границы, вывести все слова меньшей или равной длины, принадлежащие одновременно G и R, в армейском порядке. При этом для всех степеней букв, больших 1, использовать обозначение свертки (т.е. указывать степень буквы). Например, для G, имеющей правила

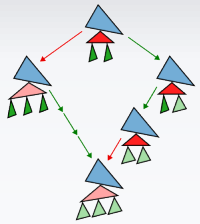
  ```
  S -> a S bb
  S -> a S b
  S ->
  ```

  и $R = a^*bb^*b$ подходящими словами для $k = 6$ и $a \preceq b$ ответом будут ab^2, a^2b^2, a^2b^3, a^3b^3, a^2b^4.



ЛР 4-2

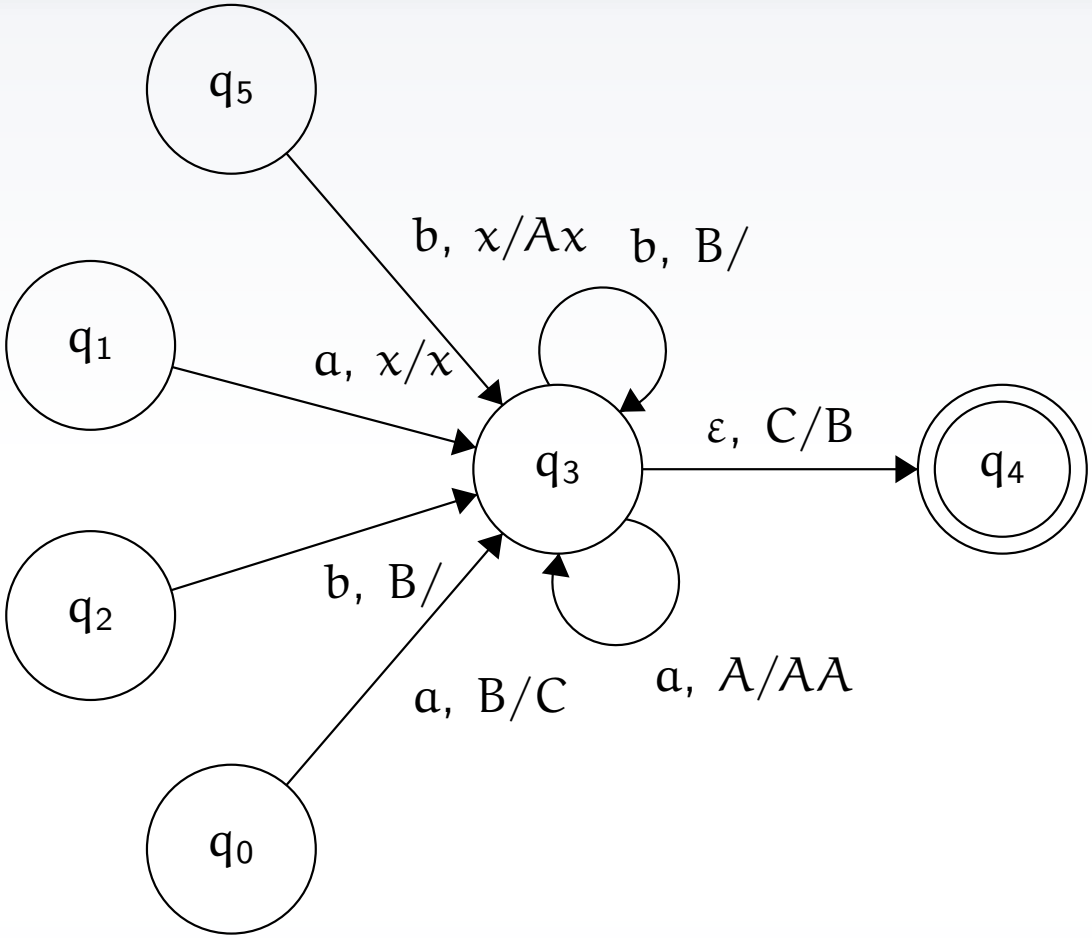
- Построить DPDA для SLR(1)-грамматики, испольщуя конструкцию LR(0)-автомата с разрешением конфликтов по 1-lookahead.
- Добавить в этот DPDA ловушку и обработать эпсилон-переходы, после чего инвертировать состояния.
- Результат — описание DPDA в формате dot.



Ловушка и обработка ε-переходов

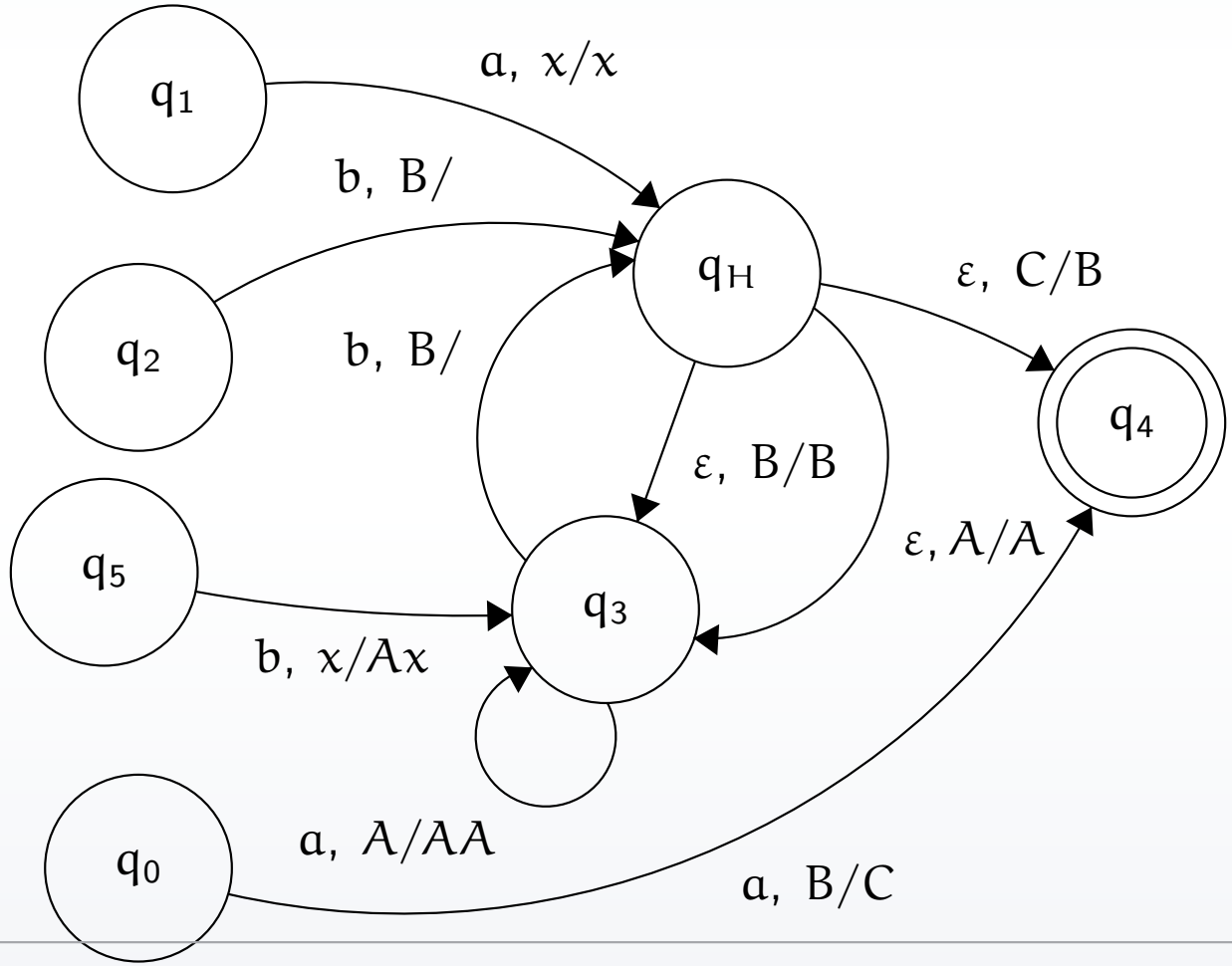
- Если $c \in \Sigma$ (то есть по символу c есть хотя бы один переход в исходном DPDA), но переходов по c из q_i нет, тогда добавляем переход из q_i по c и всем символам стека в ловушку.
- Если переходы по c из q_i есть, но не по всем символам стека, которые могут быть на его вершине в состоянии q_i, то по оставшимся стековым символам и c также нужно добавить переход в ловушку.
- ε-переход в исходном DPDA является проблемным, если он соединяет нефинальное и финальное состояния (то есть после смены финальности исходит из финального в нефинальное). В этом случае при формальной смене финальности состояний возникнет ошибка.

Рассмотрим проблемную ситуацию в следующем DPDA.

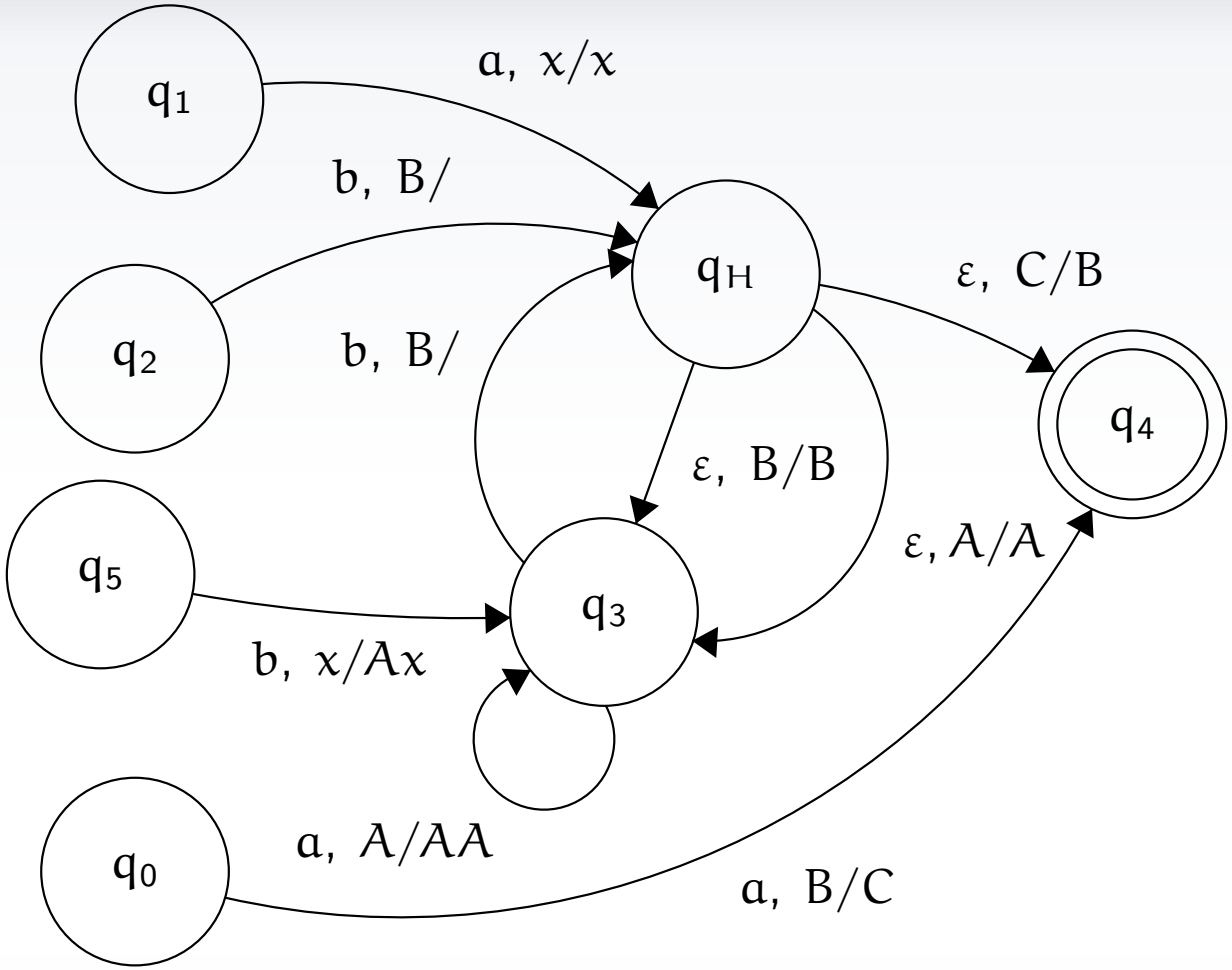


В дополнении к этому DPDA случаи, когда в q_3 на вершине стека находится C, должны распознаваться как нефинальные. Для этого нужно перенаправить все переходы, входящие в q_3 и имеющие возможность получить после них на вершине стека C, в новое нефинальное (в новом DPDA!) состояние, из которого будут только контролируемые стеком ε-переходы в состояния q_3 и q_4. Если на вершине стека после перехода точно находится C, то нужно перенаправить переход сразу же в q_4, минуя промежуточное состояние.

В дополнении к этому DPDA случаи, когда в q_3 на вершине стека находится C, должны распознаваться как нефинальные. Для этого нужно перенаправить все переходы, входящие в q_3 и имеющие возможность получить после них на вершине стека C, в новое нефинальное (в новом DPDA!) состояние, из которого будут только контролируемые стеком ε-переходы в состояния q_3 и q_4. Если на вершине стека после перехода точно находится C, то нужно перенаправить переход сразу же в q_4, минуя промежуточное состояние.
После преобразования получим следующую ситуацию:



После преобразования получим следующую ситуацию:



Состояние q_H в автомате-дополнении должно быть нефинальным, q_3 и q_4, как обычно, сменят финальность. Из q_H есть ε-переход в q_4 по C и ε-переходы по всем остальным символам стека (отличным от C) в q_3. Переход из q_5 и переход по a из q_3 в себя не перенаправляем, т.к. они кладут на вершину стека символ, точно не равный C. Переход из q_0 перенаправляем сразу в q_4, потому что он точно кладёт на стек C. Остальные (снимающие со стека либо стеконезависимые) перенаправляем в q_H.