Лабораторная работа 1 (обнуление)

1. $(\equiv 0 \mod 2)$ Реализовать алгоритм проверки выполнения лексикографического убывания на линейных комбинациях в TRS.
2. $(\equiv 1 \mod 2)$ Реализовать проверку пользовательской оценки SRS в полиномах.

Лабораторная работа 2 (обнуление)

1. $(\equiv 0 \mod 2)$ Построение пересечения КС-грамматики и праволинейной грамматики.
2. $(\equiv 1 \mod 2)$ LL-корректное удаление ε-правил.

Лабораторная работа 3 (обнуление)

1. ($\equiv 0 \mod 2$) Построение PDA по КС-грамматике с использованием LR(0)-автомата.
2. ($\equiv 1 \mod 2$) Построение DPDA для дополнения детерминированного КС-языка.

Лексикографические порядки

Если на n-ках натуральных чисел задан порядок, монотонный относительно сложения, тогда в нём нет бесконечных нисходящих цепочек.

$[x, y]$
$k(f(h(x))) \to h(f(x))$
$f(g(y)) \to h(h(y))$
Здесь, например, можно определить такой порядок: $\langle |k|, |f| \rangle$ — убывающий на сигнатуре.

Упорядочение на линейных комбинациях

Упорядочение на линейных комбинациях по построению не содержит бесконечных нисходящих цепочек $\Rightarrow$ свидетельствует о завершаемости.

$[x, y]$
$f(g(h(x))) \to h(h(x))$
$f(g(h(x))) \to g(g(x))$
$f(g(h(x))) \to f(f(x))$
$g(h(x)) \to h(h(x))$
Какое бы упорядочение мы ни задавали конструкторам, будет правило, в котором правая часть содержит больше вхождений самого «тяжёлого» конструктора, чем левая. Но можно задать порядок на линейной комбинации вхождений: $\langle |f| + |g| + |h|, |g| \rangle$.

Постановка задачи

- Необходимый кортеж линейных комбинаций, если он существует, имеет вид $\langle |f_1| \cdot c_{1,1} + \ldots |f_n| \cdot c_{1,n}, \ldots, |f_1| \cdot c_{m,1} + \ldots |f_n| \cdot c_{m,n} \rangle$. Считаем, что $m \leqslant n$. Также всегда полагаем, что $c_{i,j} \in \{0, 1\}$.
- Если кортеж линейных комбинаций, верифицирующий завершаемость TRS, найден, нужно его предъявить и вывести по каждому правилу TRS соответствующие n-ки оценок для левых и правые его частей. Иначе нужно вывести сообщение, что подходящей линейной комбинации не нашлось.

Проверка оценки в полиномах

- Из файла читается TRS, в которой все функции одноместны, и их интерпретация как полиномов. У полиномов могут быть и отрицательные коэффициенты (кроме старшего).
- Всякое правило имеет вид $f_1(f_2(\dots f_n(x))) \to g_1(g_2 \dots g_m(x))$. Чтобы гарантировать его завершаемость, нужно убедиться, что его левая часть растёт быстрее, чем правая, в заданной интерпретации. То есть необходимо проверить, является ли положительной на бесконечности функция

$$f_1 \circ f_2 \circ \dots f_n - g_1 \circ g_2 \circ \dots g_m$$

- Если это условие выполняется для всех правил TRS, то нужно сообщить о завершаемости TRS. Иначе вывести сообщение, на каких правилах переписывания нарушается убывание.

Синтаксис входных данных

Синтаксис записи входных данных для 1 задачи (лексикографические порядки):

⟨правило⟩	::=	⟨терм⟩ = ⟨терм⟩
⟨терм⟩	::=	x | ⟨конструктор⟩((⟨терм⟩,)* ⟨терм⟩)
⟨конструктор⟩	::=	[a–z]

Правил может быть одно или более. Подразумевается, что все конструкторы унарны, а переменная — только x.

Для проверки полиномиальной оценки дополнительно читается интерпретация (из другого файла) в следующем синтаксисе:

⟨интерпретация⟩	::=	⟨конструктор⟩ -> ⟨полином⟩
⟨полином⟩	::=	⟨моном⟩((+|-)⟨полином⟩)*
⟨моном⟩	::=	(-)?(⟨число⟩*)?x(^⟨число⟩)?
⟨число⟩	::=	[1–9][0–9]*

Например: `g -> 32*x^10-x+-12*x^2`. То есть мономы не обязаны быть упорядочены по убыванию степени.

Пересечение КС- и праволинейной грамматики

- Обе грамматики читаются из одного входа, синтаксис следующий:

  ⟨правило⟩ ::= ⟨нетерминал⟩ -> (⟨терминал⟩|⟨нетерминал⟩)*
  (|(⟨терминал⟩|⟨нетерминал⟩)$^+$)*

  ⟨ПР-правило⟩ ::= ⟨нетерминал⟩ -> ⟨терминал⟩(⟨нетерминал⟩)?
  (|⟨терминал⟩(⟨нетерминал⟩)?)*

  ⟨терминал⟩ ::= [a–z]

  ⟨нетерминал⟩ ::= [[A-z]$^+$]

  Общий вид входного файла:

  Context-free grammar:
  ⟨правило⟩$^+$
  Regular grammar:
  ⟨ПР-правило⟩$^+$
- Стартовым нетерминалом в обоих случаях является `[S]`.
- По праволинейной грамматике строится НКА, после чего можно пользоваться стандартным алгоритмом.

Ускорение и очистка

- Если в процессе построения пересечения возникло правило с нетерминалом в левой или правой части вида $[q_i, A, q_j]$, где A в исходной КС-грамматике всегда переписывался исключительно в терминал, и терминальных правил вида $[q_i, A, q_j]$ нет (а терминальные правила всегда стоит порождать первыми), то правило с таким нетерминалом можно смело выкидывать как непорождающее.
- В итоговой грамматике не должно быть непорождающих и недостижимых нетерминалов.

LL-корректное удаление ε-правил

1. Базовый алгоритм — в лекции 8. До его применения удалить все недостижимые и непорождающие нетерминалы и содержащие их правила. Учесть, что исходная грамматика допускает нетерминалы вида $[N_1 N_2]$, поэтому при присоединении контекста использовать разделитель (например, + или ->), отсутствующий в исходном алфавите нетерминалов.
2. Если грамматика не LL(k), тогда алгоритм может зациклиться. Произойти это может, по лемме Розенкранца и Стирнса, только если в результате присоединения обнуляемого контекста возникнет нетерминал вида $[\Phi_1 -> A(-> \Phi_2)? -> A(-> \Phi_3)?]$ (т.е. такой, в котором дважды присоединяется один и тот же обнуляемый нетерминал). Φ_i — произвольные последовательности присоединённых нетерминалов. В таких случаях прерывать исполнение, печатать промежуточную грамматику, в которой появился проблемный нетерминал, и сообщать, что исходная грамматика — не LL(k).

Синтаксис КС-грамматики

Синтаксис входной грамматики совпадает с синтаксисом КС-грамматики варианта 0. Стартовым нетерминалом считается `[S]`.

$$\begin{array}{rcl}
\langle\text{правило}\rangle & ::= & \langle\text{нетерминал}\rangle\ \texttt{->}\ (\langle\text{терминал}\rangle|\langle\text{нетерминал}\rangle)^* \\
 & & (\,|\,(\langle\text{терминал}\rangle|\langle\text{нетерминал}\rangle)^+)^* \\
\langle\text{терминал}\rangle & ::= & [\text{a–z}] \\
\langle\text{нетерминал}\rangle & ::= & [[\text{A-z}]^+]
\end{array}$$

Построение PDA по CFG

- Добавляем в грамматику «самое стартовое правило» (без эндмаркера) $S' \to S$ и строим LR(0)-автомат. Конфликты в данном случае ничему не мешают.
- Символами стека в PDA полагаем номера состояний LR(0)-автомата. Далее повторяем конструкцию, показанную в лекции 9 (добавляем ε-переходы на свёртках, ведущие в состояния, имеющие входящий переход по нетерминалу в левой части сворачиваемого правила, а затем удаляем переходы по нетерминалам).
- LR(0)-автомат и итоговый PDA выводим в dot-представлении.

Синтаксис КС-грамматики

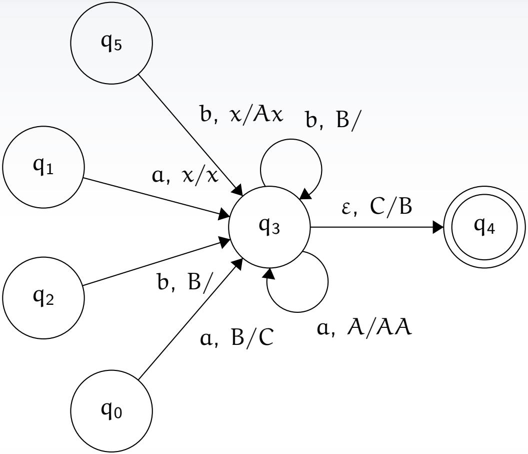
Синтаксис входной грамматики совпадает с синтаксисом КС-грамматик из второй обнулённой лабораторной. Стартовым нетерминалом считается `[S]`.

$$
\begin{array}{rcl}
\langle\text{правило}\rangle & ::= & \langle\text{нетерминал}\rangle \;\texttt{->}\; (\langle\text{терминал}\rangle|\langle\text{нетерминал}\rangle)^{*} \\
 & & (\,|\,(\langle\text{терминал}\rangle|\langle\text{нетерминал}\rangle)^{+})^{*} \\
\langle\text{терминал}\rangle & ::= & [\text{a–z}] \\
\langle\text{нетерминал}\rangle & ::= & [[\text{A-z}]^{+}]
\end{array}
$$

Построение дополнения для DPDA

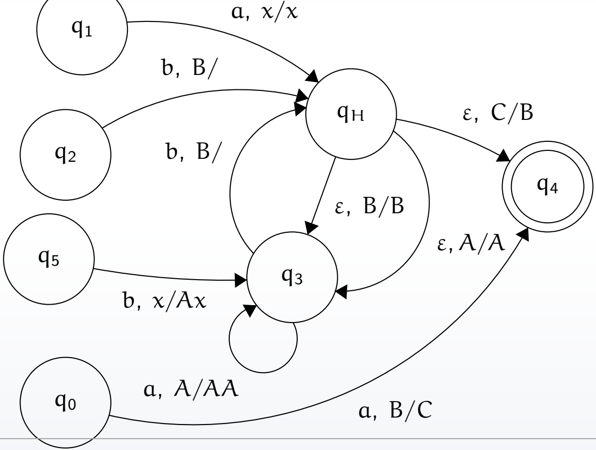
- Если на входе не DPDA, нужно сообщить об ошибке, в противном случае добавить ловушку, а затем обработать проблемные ε-переходы и сменить финальность состояний.
- Если c $\in \Sigma$ (то есть по символу c есть хотя бы один переход в исходном DPDA), но переходов по c из q_i нет, тогда добавляем переход 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.
- Итоговый PDA выводим в dot-представлении.

Рассмотрим проблемную ситуацию в следующем DPDA.

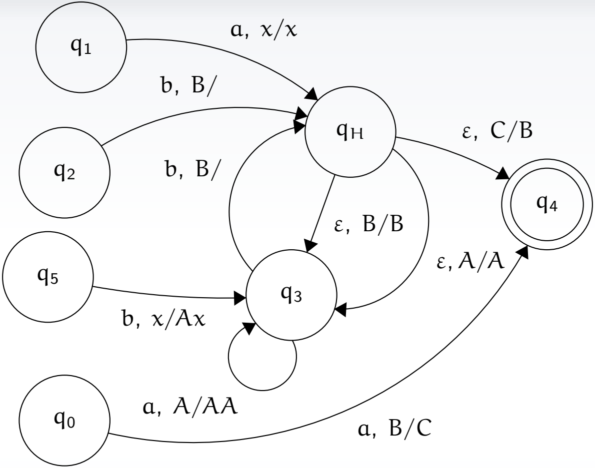


В дополнении к этому DPDA случаи, когда в q_3 на вершине стека находится C, должны распознаваться как нефинальные. Для этого нужно перенаправить все переходы, входящие в q_3 и имеющие возможность получить после них на вершине стека C, в новое нефинальное (в новом DPDA!) состояние, из которого будут только 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.

Синтаксис DPDA

DPDA записывается в следующей форме:

$$\text{finals} = \{\, \langle \text{state} \rangle (, \langle \text{state} \rangle)^{*} \,\}$$
$$\langle \text{transition} \rangle^{+}$$

Синтаксис переходов DPDA представлен ниже. Здесь !! — пустое слово:

$$\begin{array}{rcl}
\langle \text{transition} \rangle & ::= & <\langle \text{state} \rangle,(\langle \text{letter} \rangle \mid !!),\langle \text{stack_s} \rangle\text{>-><}\langle \text{state} \rangle,\langle \text{stack_s} \rangle^{*}> \\
\langle \text{state} \rangle & ::= & [\text{q–u}][0\text{-}9]? \\
\langle \text{stack_s} \rangle & ::= & [\text{A–Z}]^{+}[0\text{-}9]? \\
\langle \text{letter} \rangle & ::= & [\text{a–z}]
\end{array}$$

Начальным состоянием всегда полагаем `q0`, символом дна стека полагаем `Z0`.