Теория формальных языков. Лекции и практики. Заметки.

Семён Григорьев

2 апреля 2020 г.

Содержание

1 План лекций **3**

2 Лекция 1: Введение **3**

3 Практика 1 **6**

3.1 Григорьев С.В. 6

4 Лекция 2: Регулярные языки **7**

5 Практика 2 **8**

5.1 Григорьев С.В. 8

6 Лекция3. Контекстно-свободные грамматики **8**

7 Практика 3 **8**

7.1 Григорьев С.В. 8

8 Лекция 4 **9**

9 Практика 4 **9**

9.1 Григорьев С.В. 9

10 Лекция 5 **9**

10.1 Нормальня форма Хомского (НФХ) 9

10.2 Лемма о накачке для КС языков 10

11 Практика 5 **11**

11.1 Григорьев С.В. 11

12 Лекция 6 **12**

12.1 Свойства замкнутости КС языков 12

12.2 Алгоритм CYK 12

13 Практика 6 **12**

13.1 Григорьев С.В. 12

14 Лекция 7 **14**

14.1 Алгоритм на основе матричного произведения 14

14.2 Алгоритм на основе тензорного произведения 14

15 Практика 7 **14**

15.1 Григорьев С.В. 14

1 План лекций

1. Введение. Базовые определения. Обзор курса.
2. Регулярные языки, конечные автоматы (детерминированные, недетерминированные), регулярные выражения. Детерминизация, $\varepsilon-$замыкание, минимизация.
3. Взаимные преобразования способов задания.
4. Теоретико-языковые свойства регулярных языков. Лемма о накачке, замкнутость относительно операций.
5. Алгоритмы вычисления операций.
6. Лево(право)-линейные грамматики и регулярные языки.
7. Грамматики, переписывающие системы. КС-граммтики (обыкновенные граммтики). Вывод в граммтике, неоднозначные грамматики, существенно неоднозначные языки, дерево вывода.
8. Рекурсивные автоматы.
9. Лемма о накачке, замкнутотсть относительно операций, проверка пустоты.
10. Нормальная форма Хомского, CYK.
11. LL
12. LR
13. !!!
14. !!!
15. !!!
16. !!!
17. !!!

2 Лекция 1: Введение

Алфавит, язык. Операции над строками. Операции над языками.

Какие вопросы можно задавать о языках: о пустоте, универсальности, о построении пересечения, о пустоте пересечения, о вложенности, об эквивалентности.

Базовые способы задания: перечисление, генератор, распознаватель.

Взаимосвязь теории формальных языков с другими областями, области её применения.

- Синтаксический анализ языков программирования: в компиляторах, интерпертаторах, средах разработки, других инстументах.
- Анализ естественных языков. Активность в этой области несколько спала, так как на передний план сейчас вышли различные методы машинного обучения. Однако и в этой области ведуться работы. Примеры конференций:

- International Conference on Parsing Technologies (IWPT-2020)
- FG: Formal Grammar (FG-2020)

- Статический анализ кода.
  - Различные задачи межпроцедурного анализа. Основной подход — language reachability. Основоположник — Томас Репс. Примеры работ.
    * Thomas Reps. 1997. Program analysis via graph reachability. In Proceedings of the 1997 international symposium on Logic programming (ILPS '97). MIT Press, Cambridge, MA, USA, 5–19.
    * Qirun Zhang and Zhendong Su. 2017. Context-sensitive data-dependence analysis via linear conjunctive language reachability. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017). Association for Computing Machinery, New York, NY, USA, 344–358. DOI:https://doi.org/1
    * Kai Wang, Aftab Hussain, Zhiqiang Zuo, Guoqing Xu, and Ardalan Amiri Sani. 2017. Graspan: A Single-machine Disk-based Graph System for Interprocedural Static Analyses of Large-scale Systems Code. In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17). Association for Computing Machinery, New York, NY, USA, 389–404. DOI:https://doi.org/10.1145/3037697.3037744
    * Lu Y., Shang L., Xie X., Xue J. (2013) An Incremental Points-to Analysis with CFL-Reachability. In: Jhala R., De Bosschere K. (eds) Compiler Construction. CC 2013. Lecture Notes in Computer Science, vol 7791. Springer, Berlin, Heidelberg
  - Интерливинг (или шафл) языков для верификаци многопоточных программ.
    * Approximating the Shuffle of Context-free Languages to Find Bugs in Concurrent Recursive Programs
    * Flick N.E. (2015) Quotients of Unbounded Parallelism. In: Leucker M., Rueda C., Valencia F. (eds) Theoretical Aspects of Computing - ICTAC 2015. ICTAC 2015. Lecture Notes in Computer Science, vol 9399. Springer, Cham
  - Система типов Java: Radu Grigore, Java Generics are Turing Complete.

- Графовые базы данных. Поиск путей с ограничениями.
  - Maurizio Nolé and Carlo Sartiani. 2016. Regular Path Queries on Massive Graphs. In Proceedings of the 28th International Conference on Scientific and Statistical Database Management (SSDBM '16). Association for Computing Machinery, New York, NY, USA, Article 13, 1–12. DOI:https://doi.org/10.1145/2949689.2949711
  - Jochem Kuijpers, George Fletcher, Nikolay Yakovets, and Tobias Lindaaker. 2019. An Experimental Study of Context-Free Path Query Evaluation Methods. In Proceedings of the 31st International Conference on Scientific and Statistical Database Management (SSDBM '19). Association for Computing Machinery, New York, NY, USA, 121–132. DOI:https://doi.org/10.1145/3335783.3335791
  - Jelle Hellings. Querying for Paths in Graphs using Context-Free Path Queries.

- Биоинформатика. В основном это анализ геномных и белковых последовательностей.
  - Witold Dyrka, Mateusz Pyzik, Francois Coste, and Hugo Talibart. Estimating probabilistic context-free grammars for proteins using contact map constraints.

  - James WJ Anderson, Paula Tataru, Joe Staines, Jotun Hein, and Rune Lyngso. Evolving stochastic context-free grammars for RNA secondary structure prediction.
  - Ryan Zier-Vogel. Predicting RNA secondary structure using a stochastic conjunctive grammar.

- Машинное обучение.
  - Matt J. Kusner, Brooks Paige, José Miguel Hernández-Lobato. Grammar Variational Autoencoder. Опубликована в 2017 году и уже больше 200 цитирований.
  - TAG Parsing with Neural Networks and Vector Representations of Supertags. К разговору об оброаботке естественных языков.
  - Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, Owen Rambow. End-to-end Graph-based TAG Parsing with Neural Networks.

- Языки — это не только про строки.
  - Языки деревьев: Tree Automata Techniques and Applications.
  - Языки графов:
    * Graph Grammars
    * HYPEREDGE REPLACEMENT GRAPH GRAMMARS
    * (Re)introducing Regular Graph Languages
    * Hyperedge Replacement: Grammars and Languages
  - ...

- Теория групп. Как правило, это проблема слов группы или дополнение к ней.
  - Anisimov, A.V. Group languages. Cybern Syst Anal (1971) 7: 594.
  - David E. Muller, Paul E. Schupp, Groups, the Theory of ends, and context-free languages, Journal of Computer and System Sciences, Volume 26, Issue 3, 1983, Pages 295-310, ISSN 0022-0000
  - HOLT, D., REES, S., ROVER, C., & THOMAS, R. (2005). GROUPS WITH CONTEXT-FREE CO-WORD PROBLEM. Journal of the London Mathematical Society, 71(3), 643-657. doi:10.1112/S002461070500654X
  - Groups with Context-Free Co-Word Problem and Embeddings into Thompson's Group V
  - Kropholler, R. & Spriano, D. (2019). Closure properties in the class of multiple context-free groups. Groups Complexity Cryptology, 11(1), pp. 1-15. Retrieved 13 Feb. 2020, from doi:10.1515/gcc-2019-2004
  - Word problems of groups, formal languages and decidability

- Прочая забавная математика.
  - Немного топологии в теории формальных языков: Salvati S. On is an n-MCFL. – 2018.
  - Salvati S. MIX is a 2-MCFL and the word problem in Z2 is captured by the IO and the OI hierarchies //Journal of Computer and System Sciences. – 2015. – Т. 81. – №. 7. – С. 1252-1277.

– О том, как задачи из теории графов связаны с теорией формальных языков: Abboud, Amir & Backurs, Arturs & Williams, Virginia. (2015). If the Current Clique Algorithms are Optimal, So is Valiant's Parser. 98-117. 10.1109/FOCS.2015.16.

– A context-free grammar for the Ramanujan-Shor polynomials

3 Практика 1

Детали о том, как будет проходить практика.

3.1 Григорьев С.В.

Немного про описания языков. Пописать языковые уравнения, граммтики. Посмотреть на операции над языками.

Постановка задачи на весь семестр.

Запросы к графовым базам данных. Контекст задачи, примеры графовых БД (RedisGraph, Neo4j, ...), задача о путях впринципе.

Ссылка на второй конспект.

Задача: реализовать свою "графовую миниБД".

Реализация: оформление, инструменты, языки.

- Ограничений на язык реализации нет.
- Ограничений на использование библиотек нет. Главное — не нарушать лицензии и чтобы можно было вносить изменения в библиотеку (при необходимости).
- Каждый создёт под решение репозиторий на GitHub и снабжает его всем необходимым: readme, лицензия, CI-сборка с тестированием, инструкции по локальному развёртыванию.
- Разработка ведётся в отдельной ветке и когда очередная часть задачи готова к сдаче — делаем pull request в master и добавляем меня (gsvgit) в ревьюверы.

Задачи на дом.

1. Выбрать язык программирования, на котором будет вестись разработка.
2. Создать репозиторий на GitHub.
3. Настроить CI-сборку и тестирование.
4. Реализовать подгрузку графов из RDF используя готовые библиотеки.

4 Лекция 2: Регулярные языки

Иерархия Хомского. Проблемы с ней. Классы языков.

Грамматики. Системы переписывания.

Регулярные множества. Регулярные языки. Регулярные выражения.

$$V^* = \bigcup_{\infty}^{i=0} V^i$$

Конечные автоматы. Система переходов.

Язык, задаваемый автоматом.

Понятие выводимости ($\vdash^*$).

Конфигурация: $\langle$Состояние, Остаток$\rangle$.

Полный автомат и вершина-сток.

Детерминизация, алгоритм Томпсона.

НКА: $\langle \Sigma, Q, s \in Q, T \in Q, \delta : Q \times \Sigma \to 2^Q \rangle$

ДКА: $\langle \Sigma, Q_d, s_d \in Q_d, T_d \in Q_d, \delta_d : Q_d \times \Sigma \to Q_d \rangle$, где:

- $Q_d = \{q_d \mid q_d \in 2^Q\}$,
- $s_d = \{s\}$,
- $T_d = \{q \in Q_d \mid \exists p \in T : p \in q\}$,
- $\delta_d(q, c) = \{\delta(a, c) \mid a \in q\}$.

ε-замыкание.

1. Транзитивное замыкание отношения ε-перехода.
2. Обработка финальных состояний
3. Добавление переходов: если $\delta(v_0, \varepsilon) = v_1, \delta(v_1, c) = v_2$, то добавим $\delta(v_0, c) = v_2$.
4. Удалим ε-переходы.

Эквивалентность автоматов. Эквивалентность состояний: состояния эквивалентны ели нет различающей строки.

Минимизация.

Теорема Клини об эквивалентности автоматов и регулярных языков.

Построение автомата по регулярному выражению.

Построение регулярного выражения по автомату: устранение вершин.

5 Практика 2

5.1 Григорьев С.В.

Построение минимального ДКА по регулярному варажению.

Домашнее задание.

1. Реализовать функцию (можно с применением библиотек), которая принимает на вход регулярное выражение в виде строки и строит по нему минимальный ДКА.
2. Реализовать необходимые тесты на построение ДКА по регулярному выражению.
3. Реализовать (можно с применением библиотек) пересечение минимального ДКА и НКА без ε-переходов.
4. Реализовать необходимые тесты на пересечение ДКа и НКА.

6 Лекция3. Контекстно-свободные грамматики

Лево- и право-линейные грамматики и регулярные языки. Неразрешимость задачи проверки того, что граммтика задаёт регулярный язык. Статья на эту тему: Self-embedded context-free grammars with regular counterparts. Грамматика $\to$ регулярка и регулярка $\to$ грамматика.

Выовд цепочки в грамматике, левосторонний, правосторонний вывод, неоднозначные и однозначные грамматики. Примеры. Существенно неоднозначные языки.

Дерево вывода. Соотношение между деревьями и выводами. Примеры.

Расширенные контекстно-свободные грамматики.

7 Практика 3

7.1 Григорьев С.В.

Пересечение автоматов — это тензорное произведение матриц смежности. Пример.

Про коммутативность пересечения и некоммутативность тензорного произведения.

Домашнее задание.

1. Реализовать консольный клиент, позволяющий
   (a) загрузить RDF-файл
   (b) вывести список меток рёбер
   (c) задать к загруженному графу регулярный запрос с возможностью указать представление результата: пустота ответа, автомат сдампить в файл в формате DOT (`https://www.graphviz.org/doc/info/lang.html`), пара (кол-во рёбер, кол-во вершин) в результирующем автомате
   (d) выйти из клиента.
2. Подгрузку RDF и выполнение запросов реализовать на основе уже существующей функциональности.

3. Провести замеры производительности на графах из репозитория `https://github.com/JetBrains-Research/CFPQ_Data`. Графы брать из подпапки `data/graphs/RDF`. Так как в графах присутствуют одинаковые отношения, то можно один и тот же запрос выполнять на всех графах. Отчёт оформить в виде раздела в README репозитория в виде таблицы.

   Эти эксперименты проводятся локально! Не надо таскать репозиторий с графами за собой. Для тестов клиента использовать маленькие синтетические RDF.

4. Реализовать необходимые тесты на работоспособность клиента.

8 Лекция 4

Рекурсивные автоматы. Построение, интерпретация.

9 Практика 4

9.1 Григорьев С.В.

Больше подробностей про рекурсивные автоматы: тотальная минимизация. Как их применять для КС запросов. Тензоры + транзитивное замыкание.

Домашнее задание.

1. Реализовать выполнение регулярных щапросов через тензорное произведение. Для тензорного произведения использовать существующие библиотеки линейной алгебры. Обратите внимание на то, что матрицы должны быть разреженными. Скорее всего, удобно будет использовать представление в виде набора булевых матриц.

2. Интегрировать новую реализацию в клиент наравне со старой.

3. Провести замеры производительности на графах из репозитория `https://github.com/JetBrains-Research/CFPQ_Data`. Графы брать из подпапки `data/graphs/RDF`. Так как в графах присутствуют одинаковые отношения, то можно один и тот же запрос выполнять на всех графах. Отчёт оформить в виде раздела в README репозитория в виде таблицы. Сравнить с результатми предыдущей задачи.

   Эти эксперименты проводятся локально! Не надо таскать репозиторий с графами за собой. Для тестов клиента использовать маленькие синтетические графы и запросы.

4. Реализовать необходимые тесты на работоспособность алгоритма через тензорное произведение.

10 Лекция 5

10.1 Нормальня форма Хомского (НФХ)

Определение 10.1. КС грамматика находится в нормальной форме Хомского если любое правило имеет один из трёх видов:

1. $S \to \varepsilon$
2. $N_i \to t_j$
3. $N_i \to N_j N_k$, $N_j \neq S, N_k \neq S$

Важно: стартовый нетерминал не встречается в правых частях правил, ε-продукция только для стартового нетерминала.

Note. Любую КС грамматику можно преобразвать к нормальной форме Хомского.

Преобразование в НФХ. Шаги.

1. Устранение длинных правил.
2. Устранение ε-правил.
3. Устранение цепных правил.
4. Устранение бесполезных нетерминалов
   (a) Удаление непорождающих нетерминалов
   (b) Удаление недостижимых нетерминалов
5. Устранение продукций с правой частью длины 2, содержащей терминалы.

Надо не забыть добавить новый стартовый нетерминал, если нужно: чтобы вывести из него ε и чтобы не встречался в правых частях правил.

Важно. Порядок применения шагов преобразования важен.

1. Второй шаг можно поднять наверх, но это приведёт к более существенному разрастанию результирующей граммтики.
2. Подшаги шага 4 нельзя менять местами. Попробуйте поприменять их к граммтике:

$$\begin{aligned} S &\to AB \mid a \\ A &\to b \end{aligned}$$

Материалы по преобразованию в НФХ.

10.2 Лемма о накачке для КС языков

Теорема 10.1. Пусть L — контекстно-свободный язык над алфавитом Σ, тогда существует такое n, что для любого слова $\omega \in L$, $|\omega| \geq n$ найдутся слова $u, v, x, y, z \in \Sigma^*$, для которых верно: $uvxyz = \omega, vy \neq \varepsilon, |vxy| \leq n$ и для любого $k \geq 0$ $uv^k xy^k z \in L$.

Идея доказательства леммы о накачке.

1. Для любого КС языка можно найти грамматику в нормальной форме Хомского.

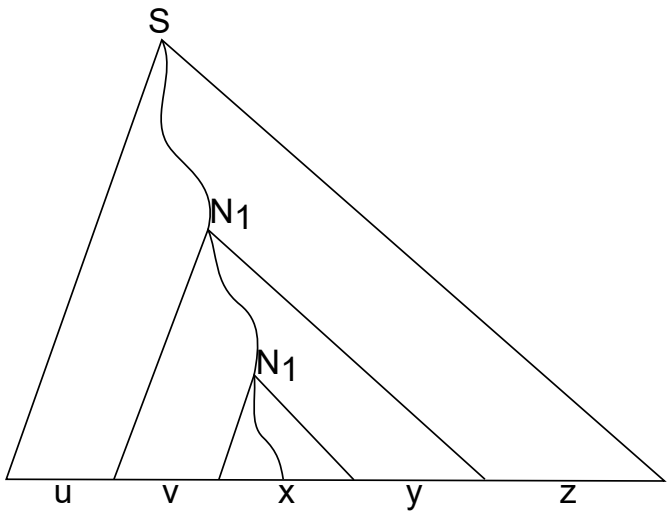


Рис. 1: Разбиение цепочки для леммы о накачке

2. Очевидно, что если брать достаточно длинные цепочки, то в дереве вывода этих цепочек, на пути от корня к какому-то листу обязательно будет нетерминал, встречающийся минимум два раза. Если m — количество нетерминалов в НФХ, то длины 2^{m+1} должно хватить. Это и будет n из леммы.

3. Возьмём путь, на котором есть хотя бы дважды повторяется некоторый нетерминал. Скажем, это нетерминал N_1. Пойдём от листа по этому пути. Найдём первое появление N_1. Цепочка, задаваемая поддеревом для этого узла — это x из леммы.

4. Пойдём дальше и найдём второе появление N_1. Цепочка, задаваемая поддеревом для этого узла — это vxy из леммы.

5. Теперь мы можем копировать кусок дерева между этими повторениями N_1 и таким образом накачивать исходную цепочку.

Надо только проверить выполение ограничений на длины.

Материалы по лемме о накачке для КС языков.

Проверить неконтекстно-свободность языка $L = \{a^n b^n c^n \mid n > 0\}$.

11 Практика 5

11.1 Григорьев С.В.

Преобразование в нормальную форму Хомского.

Формат входа:

1. Одна продукция на строку.

2. Продукция — это список терминалов и нетерминалов через пробел, начинающийся с нереминала (левая часть продукции).

3. Нетерминалы — заглавные буквы с опциональным числовым суффиксом.

4. Терминалы — строчные буквы с опциональным числовым суффиксом.

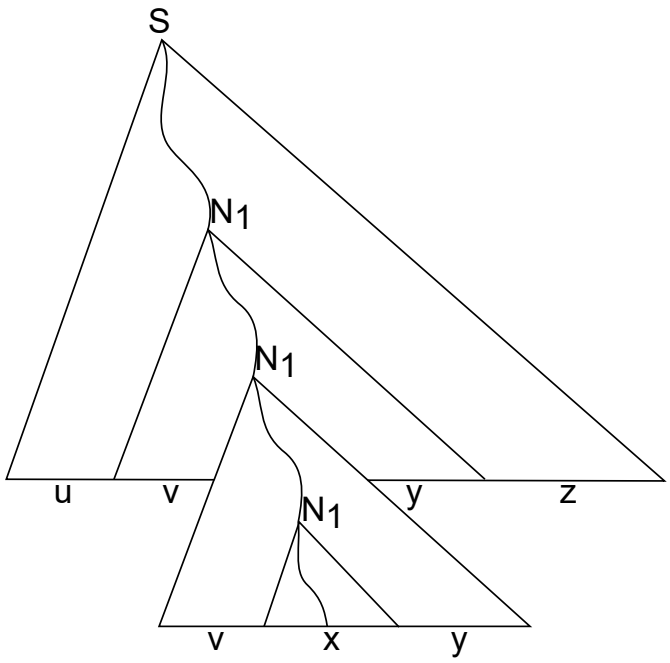


Рис. 2: Пример накачки цепочки с рисунка 1

5. Специальный символ `eps` для обобзначения ε.

Пример входя, описывающего граммтику $S \to aSbS \mid \varepsilon$:

```
S a S b S
S eps
```

Домашнее задание.

1. Реализовать преобразование в нормальную форму Хомского. На входе файл с граммтикой, на выходе — файл с граммтикой в НФХ в том же формате, что и вход.

12 Лекция 6

12.1 Свойства замкнутости КС языков

Глава 2.6 конспекта.

12.2 Алгоритм CYK

Глава 4.1 “Алгоритм CYK” конспекта.

13 Практика 6

13.1 Григорьев С.В.

Алгоритм CYK и алгоритм Хеллингса.

Формат входа для CYK:

1. Грамматика: смотри предыдущее ДЗ.

2. Входная строка: терминалы разделены пробелами.

Пример входа, описывающего граммтику $S \to aSbS \mid \varepsilon$:

```
S a S b S
S eps
```

Пример входной строки:

```
a a b a a b b b
```

Формат входа алгоритма Хеллингса:

1. Грамматика: тот же формат, что и для CYK. НО! ИСпользуем преобразование а ослабленную НФХ.

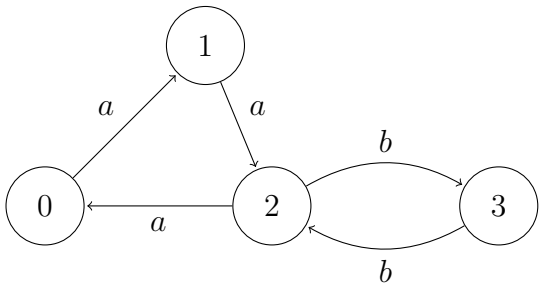
2. Входной граф: файл в ктором на каждой строке записано ребро в виде тройки

$$\langle \textit{вершина метка\_ребра вершина} \rangle.$$

Элементы тройки разделены пробелами.

3. Можно считать, что все вершины графа — числа от нуля, идущие подряд.

Пример входного графа:



Пример описания входного графа:

```
0 a 1
1 a 2
2 a 0
2 b 3
3 b 2
```

Домашнее задание.

1. Реализовать алгоритм CYK для линейного входа. На вход принимаются два файла: с граммтикой и входной строкой. Результат (выводится ли входная цепочка в грамматике) печатается в консоль.

2. Реализовать алгоритм Хеллингса. На вход принимается файл с граммтикой и файл с графом. В результирующий файл печатается граммтика в ослабленной НФХ (с которой непосредсвенно работал алгоритм) и множество пар достижимых вершин для стартового нетерминала (одна пара на строку, две вершины через пробел)

14 Лекция 7

Алгоритмы решения задачи контекстно-свободной достижимости, основанные на операциях линейной алгебры.

14.1 Алгоритм на основе матричного произведения

Глава 5.1 конспекта.

14.2 Алгоритм на основе тензорного произведения

Глава 6 конспекта.

15 Практика 7

15.1 Григорьев С.В.

Алгоритмы на основе линейной алгебры.

Для реализации предлагается использовать следующие библиотеки. Так как с булевыми не везде хорошо, то будем использовать те типы, которые поддерживаются: `Int`, `float` и т.д.

- Для языка Python — разреженные матрицы в scipy и соответствующие опреации работы с ними: **scipy.sparse.kron** и обычное матричное произведение. Предпочтительный формат разреженных матриц — CSR.
- Для языка Kotlin —la4j. Операции: кронекер и обычное умножение.

Поэлементное сложение есть и там и там.

Для алгоритма на матричном умножении всё точно так же, как и в предыдущей ДЗ для Хеллингса.

Для тензорного произведения расширим формат представления входной грамматики. Одна строка на нетерминал. Терминалы, нетерминалы, ε обозначаются как и раньше. Как и раньше, левая часть от правой отделена пробелом. В правой части можно использовать конструкции регулярных выражений: альтернатива, звезда клини, групперцющие скобки. Этот набор можно расширять по своему усмотрению.

Пример входа, описывающего грамматику $S \to (aSb)* \mid \varepsilon$:

```
S (a S b)* | eps
```

Домашнее задание. Время на выполнение — две недели. Один из алгоритмов — на первую, оставшийся и эксперименты — на вторую.

1. Реализовать алгоритм, основанный на матричном умножении. На вход принимаются два файла: с граммтикой и входным графом. В результирующий файл печатается граммтика в ослабленной НФХ (с которой непосредсвенно работал алгоритм) и множество пар достижимых вершин для стартового нетерминала (одна пара на строку, две вершины через пробел).

2. Реализовать алгоритм, основанный на тензорном произведении. На вход принимается файл с граммтикой и файл с графом. В результирующий файл печатается матрица смежности рекурсивного автомата (с которым непосредсвенно работал алгоритм, построчно, элементы разделены пробелом, пустая ячейка обозначается символом '.') и множество пар достижимых вершин для стартового нетерминала (одна пара на строку, две вершины через пробел).

3. Сравнить производительность трёх реализованных алгоритмов (Хеллингс, матричное произведение, тензорное произведение). Результат — описание эксперимента и таблица сравнения в readme.