Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Демьяненко В.С., Костур О.Г. Программная проверка непустоты языка бесконтекстной грамматики с визуализацией // Академия педагогических идей «Новация». Серия: Студенческий научный вестник. – 2025. – №4 (июнь) – АРТ 6-эл. – 0,2 п.л. - URL: http: //akademnova.ru/page/875550

РУБРИКА: ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.423.25

Демьяненко Виктория Сергеевна
студентка 2 курса, Физико-Технический институт
ФГАОУ ВО «КФУ им. В.И. Вернадского»
г. Симферополь, Российская Федерация
e-mail: vdem1716@yandex.ru

Костур Ольга Геннадьевна
студентка 2 курса, Физико-Технический институт
ФГАОУ ВО «КФУ им. В.И. Вернадского»
г. Симферополь, Российская Федерация
e-mail: olgadance20@gmail.com

ПРОГРАММНАЯ ПРОВЕРКА НЕПУСТОТЫ ЯЗЫКА БЕСКОНТЕКСТНОЙ ГРАММАТИКИ С ВИЗУАЛИЗАЦИЕЙ

Аннотация: В статье представлен алгоритм программной проверки непустоты языка контекстно-свободной грамматики, реализованный на Python. Научная новизна заключается в сочетании оптимизированного алгоритма с анимированной визуализацией графа зависимостей, включающей пояснения для каждой итерации, и подробным анализом продуктивных, непродуктивных и недостижимых нетерминалов.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Ключевые слова: Формальные грамматики, проверка непустоты, визуализация, Python, формальные языки, КС-грамматика, нормальная форма Хомского.

Demyanenko Victoria Sergeevna,
2nd year student, Physics and Technology Institute
FGBOU VO " V.I. Vernadsky Crimean Federal University"
Simferopol, Russian Federation
e-mail: vdem1716@yandex.ru
Kostur Olga Gennadevna,
2nd year student, Physics and Technology Institute
FGBOU VO " V.I. Vernadsky Crimean Federal University"
Simferopol, Russian Federation
e-mail: olgadance20@gmail.com

SOFTWARE VERIFICATION OF NON-EMPTINESS OF A CONTEXT-FREE GRAMMAR LANGUAGE WITH VISUALIZATION

Abstract: The article presents an algorithm for software verification of the non-emptiness of a context-free grammar language, implemented in Python. The scientific novelty lies in the integration of an optimized algorithm with animated visualization of the dependency graph, including explanations for each iteration, and a detailed analysis of productive, non-productive, and unreachable non-terminals.

Keywords: Formal grammars, non-emptiness verification, visualization, Python, formal languages, context-free grammar, Chomsky normal form.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Обоснование актуальности и цели исследования

Проверка непустоты языка формальной грамматики — фундаментальная задача теории формальных языков, находящая применение в компиляторах, обработке естественных языков и формальной верификации [1].

В условиях цифровизации образования и разработки программного обеспечения возрастает потребность в инструментах, которые делают сложные теоретические концепции доступными для студентов и эффективными для практического применения. Проверка непустоты языка грамматики является ключевым этапом в проектировании компиляторов и парсеров, где необходимо гарантировать, что грамматика порождает хотя бы одну терминальную цепочку. Традиционные подходы к этой задаче, основанные на рекурсивных или аналитических методах, часто сложны для понимания и не включают визуальных средств, что затрудняет их использование в образовательных целях. Кроме того, современные грамматики, используемые в языках программирования и обработке текстов, требуют масштабируемых алгоритмов, способных обрабатывать большие множества продукций. Данная работа решает эти проблемы, предлагая программный инструмент, который сочетает оптимизированную реализацию, информативный анализ и наглядную визуализацию процесса. Актуальность исследования подчеркивается необходимостью создания интерактивных образовательных решений и эффективных средств верификации грамматик для прикладных задач.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Целью работы является разработка и реализация алгоритма проверки непустоты языка формальной грамматики, который:

1.Обеспечивает высокую производительность за счет оптимизированных операций над множествами.

2.Предоставляет информативный отчет о продуктивных и недостижимых нетерминалах.

3.Включает анимированную визуализацию процесса с пояснениями для повышения образовательной ценности.

4.Демонстрирует корректность и применимость на примерах контекстно-свободных грамматик.

Контекстно-свободные грамматики в иерархии Хомского

Формальная грамматика G = (V_T, V_N, P, S) состоит из множества терминалов V_T, нетерминалов V_N, продукционных правил P и стартового символа S. Контекстно-свободные грамматики (КС-грамматики), или грамматики типа 2, характеризуются правилами вида $A \to \alpha$, где $A \in V\_N$ — нетерминал, а $\alpha \in V^+$ — непустая цепочка терминалов и/или нетерминалов. Для укорачивающихся КС-грамматик допускается $\alpha \in V^*$, включая ε-правила ($A \to \varepsilon$), позволяющие порождать пустую строку. Например, грамматика G = ({a, b}, {S}, {S → aSb | ε}, S) порождает язык {ε, ab, aabb, aaabbb, …}, иллюстрируя способность КС-грамматик моделировать парное соответствие символов [1]. Отличительная особенность КС-грамматик — независимость замены нетерминала от контекста, что упрощает их анализ по сравнению с контекстно-зависимыми грамматиками.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Н. Хомский предложил классификацию порождающих грамматик, выделив четыре типа (0–3), которые образуют строгую иерархию, где каждый последующий тип является подмножеством предыдущего [1]:

1.Тип 0 (неограниченные грамматики): Правила вида $\alpha \rightarrow \beta$, где $\alpha \in V^+$, $\beta \in V^*$. Эти грамматики порождают самые сложные языки, включая $\{a^n b^n c^n \mid n \geq 0\}$.

2.Тип 1 (контекстно-зависимые грамматики): Правила вида $\xi_1 N \xi_2 \rightarrow \xi_1 \gamma \xi_2$, где $N \in V\_N$, $\gamma \in V^+$, а замена нетерминала зависит от контекста ξ_1, ξ_2. Также включают неукорачивающиеся грамматики ($\alpha \rightarrow \beta$, $|\alpha| \leq |\beta|$).

3.Тип 2 (контекстно-свободные грамматики): Правила $A \rightarrow \alpha$, описанные выше, обеспечивают баланс между выразительностью и вычислительной эффективностью.

4.Тип 3 (регулярные грамматики): Правила праволинейные ($A \rightarrow t$ или $A \rightarrow tB$) или леволинейные ($A \rightarrow t$ или $A \rightarrow Bt$), порождающие простые языки, например, регулярные выражения.

КС-грамматики занимают промежуточное положение в иерархии Хомского. Любой регулярный язык является КС-языком, но существуют КС-языки, такие как $\{a^n b^n \mid n \geq 0\}$, которые не являются регулярными из-за необходимости учета парного соответствия. КС-языки, в свою очередь, являются подмножеством контекстно-зависимых языков, но не могут порождать языки с более сложной структурой, например, $\{a^n b^n c^n \mid n \geq 0\}$, требующие контекстной зависимости. КС-грамматики также являются неукорачивающими, если не содержат ε-правил, и входят в класс неограниченных грамматик [1]. Например, грамматика $G = (\{0, 1\}, \{S, A, B\}, \{S \rightarrow 0A \mid 1B, A \rightarrow 0 \mid 0S \mid 1AA, B \rightarrow 1 \mid 1S \mid 0BB\}, S)$ порождает КС-язык, где число нулей равно числу единиц, демонстрируя их выразительность.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

КС-грамматики играют центральную роль в компьютерных науках благодаря их способности описывать вложенные структуры, характерные для синтаксиса языков программирования, арифметических выражений и естественных языков. Они являются основой для алгоритмов синтаксического разбора, таких как CYK, LL и LR, которые широко применяются в компиляторах и парсерах [1].

Формально, задача опеределения непустоты сводится к проверке, является ли стартовый нетерминал S продуктивным, то есть способен ли он порождать терминальную цепочку прямо или через другие продуктивные нетерминалы. Нетерминал $A \in V\_N$ продуктивен, если существует вывод $A \Rightarrow^* w$, где $w \in V\_T^*$.

Для решения задачи проверки непустоты необходимо определить множество продуктивных нетерминалов. Нетерминал A продуктивен, если:

1.Существует правило $A \rightarrow w$, где $w \in V\_T^*$ (прямое порождение терминалов).

2.Существует правило $A \rightarrow \alpha$, где α содержит только терминалы и/или продуктивные нетерминалы, и через последовательные выводы $\alpha \Rightarrow^* w$, где $w \in V\_T^*$.

Алгоритм итеративно строит множество продуктивных нетерминалов, начиная с тех, что непосредственно порождают терминалы, и добавляя нетерминалы, зависящие от уже продуктивных. Язык непуст, если стартовый символ S входит в это множество.

В данной работе реализован алгоритм проверки непустоты КС-грамматик, который оптимизирован для эффективной обработки продукций с использованием операций над множествами в Python. Алгоритм не только определяет непустоту языка, но и классифицирует нетерминалы на продуктивные, непродуктивные и недостижимые, предоставляя

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

информативный отчет.

Описание разработанного алгоритма

Алгоритм проверки непустоты языка контекстно-свободной грамматики G = (V_T, V_N, P, S) определяет, может ли стартовый нетерминал S порождать терминальные цепочки. Основная идея заключается в итеративном построении множества продуктивных нетерминалов, то есть тех, которые прямо или косвенно порождают цепочки из терминалов V_T*. Алгоритм состоит из следующих шагов:

1.Инициализация: Создается пустое множество N_prev для хранения продуктивных нетерминалов (см. рисунок 1);

2.Итеративный поиск продуктивных нетерминалов: Для каждой продукции $(A \rightarrow \alpha) \in P$ проверяется, состоит ли α только из терминалов V_T или нетерминалов, уже входящих в N_prev. Если условие выполняется, нетерминал A добавляется в множество N_current (см. рисунок 2);

3.Обновление множества: N_prev обновляется копией N_current, и процесс повторяется, пока не перестают добавляться новые нетерминалы (см. рисунок 3);

4.Проверка результата: Язык непуст, если стартовый символ $S \in$ N_current.

```
start_time = time.time()
N_prev = set()
N_current = set()
all_non_terminals = grammar['V_N']
all_terminals = grammar['V_T']
productions = grammar['P']
start_symbol = grammar['S']
iterations = 0
```

Рисунок 1 – Инициализация

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)
Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Рисунок 2 – Итеративный поиск продуктивных нетерминалов

Рисунок 3 – Обновление множества

Дополнительно алгоритм выявляет недостижимые нетерминалы (см. рисунок 4).

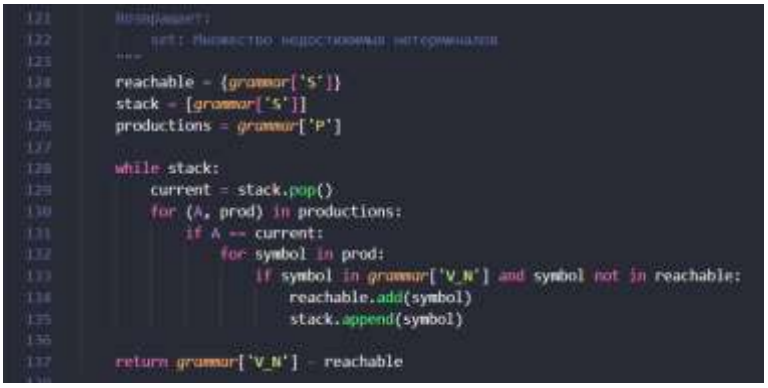

Рисунок 4 – Выявление недостижимых нетерминалов

Реализация на Python

Алгоритм реализован на языке Python с использованием библиотек networkx для построения графа зависимостей и matplotlib для анимации. Оптимизация достигнута за счет операций над множествами (set), которые обеспечивают эффективную проверку принадлежности символов. Основные компоненты реализации:

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

1.Обработка грамматики: Грамматика представлена словарем с ключами V_T (терминалы), V_N (нетерминалы), P (продукции в виде списка кортежей), S (стартовый символ);

2.Информативный вывод: Алгоритм формирует отчет, включающий продуктивные, непродуктивные и недостижимые нетерминалы, число итераций и время выполнения (см. рисунок 5);

3.Визуализация: Создается направленный граф, где узлы — нетерминалы, а ребра — зависимости в продукциях. Анимация показывает итерации алгоритма: узлы меняют цвет (красный → зеленый) при добавлении в продуктивное множество (см. рисунок 6). Каждое окно анимации содержит текстовое описание: номер итерации, продуктивные нетерминалы, добавленные нетерминалы и текущий статус языка (например, "Ожидается завершение" или "Язык существует").

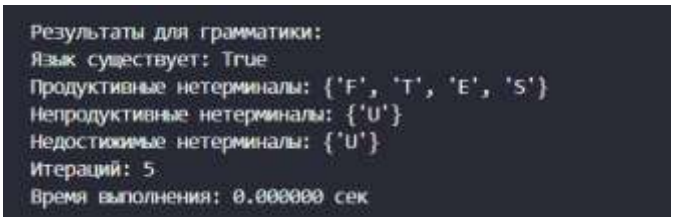

Рисунок 5 – Информация о нетерминалах в выводе программы

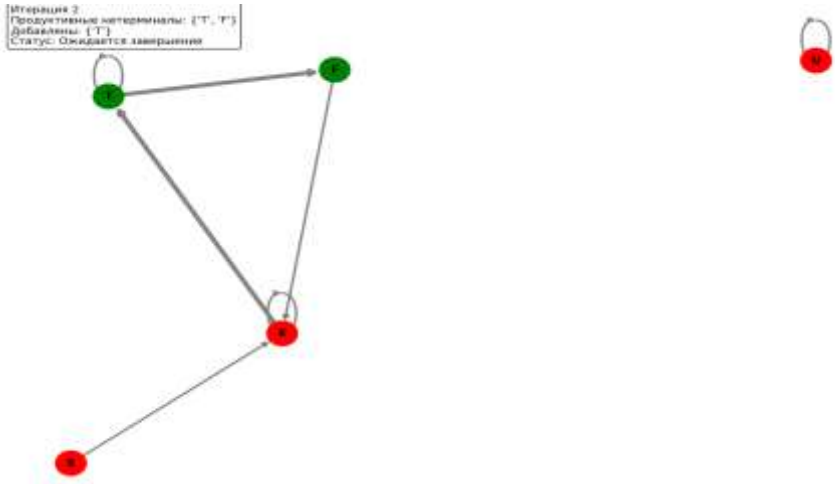

Рисунок 6 – Вторая итерация

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Визуализация реализована как анимация графа, где динамика процесса наглядно демонстрирует, как нетерминалы становятся продуктивными. На каждой итерации:

1.Узлы, соответствующие продуктивным нетерминалам, окрашиваются в зеленый цвет, а непродуктивные остаются красными;

2.Текстовое поле в левом верхнем углу окна описывает состояние: итерация, множество продуктивных нетерминалов, добавленные нетерминалы и предварительный или окончательный статус языка.

Полный код программы расположен в открытом репозитории GitHub одного из авторов [3].

Выводы

Разработанный алгоритм проверки непустоты языка контекстно-свободной грамматики представляет эффективный и наглядный инструмент.

Ключевые результаты:

1.Эффективность: Реализация на Python с временной сложностью O(|P| · |V_N|) успешно анализирует сложные КС-грамматики, такие как грамматика арифметических выражений с нетерминалами {F, T, E, S, U};

2.Информативность: Алгоритм предоставляет отчет о продуктивных ({S, E, T, F}), непродуктивных ({U}) и недостижимых ({U}) нетерминалах, числе итераций и времени;

3.Визуализация: Пошаговая анимация графа с пояснениями (итерация, продуктивные нетерминалы, статус) упрощает понимание и служит образовательным инструментом.

Научная новизна заключается в сочетании оптимизированного алгоритма с уникальной анимированной визуализацией и анализом нетерминалов. Алгоритм применим в обучении формальным языкам.

Всероссийское СМИ
«Академия педагогических идей «НОВАЦИЯ»
Свидетельство о регистрации ЭЛ №ФС 77-62011 от 05.06.2015 г.
(выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций)

Сайт: akademnova.ru
e-mail: akademnova@mail.ru

Список использованной литературы:

1. Лаздин, А. В. Формальные языки, грамматики, автоматы: учебное пособие [Электронный ресурс] / А. В. Лаздин. – ТарУ, 2022. – URL: https://lib.tau-edu.kz/wp-content/uploads/2022/04/Лаздин-А.-В.-Формальные-языки-грамматики-автоматы-1.pdf (дата обращения: 23.05.2025);
2. Хопкрофт, Дж. Введение в теорию автоматов, языков и вычислений / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман; пер. с англ. – М.: Вильямс, 2007. – 528 с.;
3. Grammar and Language Modeling Project [Электронный ресурс] // GitHub. – URL: https://github.com/vdemyy/grammar-language (дата обращения: 12.06.2025).

Дата поступления в редакцию: 13.06.2025 г.
Опубликовано: 18.06.2025 г.