Архитектура трансляторов

Лекция 2: От выражений к учебным тестам

Курс Б1.В.12

ИГУ, Кафедра информационных технологий

2025

Вчера и сегодня

Лекция 1

- Регулярные выражения
- ДКА/НКА
- Лексеры
- Токенизация

Лекция 2

- КС-грамматики
- Парсеры
- Деревья разбора
- Архитектура трансляторов

Следующая

Лекция 3: Семантический анализ, типы, генерация кода

Педагогический кейс: Приоритет операций

Проблема

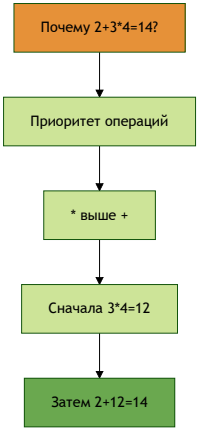
Почему калькулятор вычисляет:
2 + 3 * 4 = 14
а не 20?

Ограничение регулярок

Регулярные выражения не различают структуру и приоритеты

Решение

Нужны контекстно-свободные грамматики и деревья разбора



Педагогический кейс: Система тестирования

Задача

Автоматическая проверка ответов с разным форматом

GIFT-формат

- Столица Франции = Париж
- 2+2 = =4 =четыре
- Выберите столицы { Лондон =Париж Берлин}

Решение

Транслятор для формата тестовых заданий

Применение

- Генерация quiz
- Проверка домашних работ
- Создание вариантов

Архитектура транслятора

Этапы трансляции

Лексический анализ Разбиение входного текста на токены (лексемы) с помощью регулярных выражений и конечных автоматов

Синтаксический анализ Построение дерева разбора на основе контекстно-свободной грамматики, проверка структуры

Семантический анализ Проверка типов, областей видимости, контекстных ограничений

Генерация AST Создание абстрактного синтаксического дерева - промежуточного представления программы

Генерация промежуточного кода Создание абстрактного синтаксического дерева или другой промежуточной формы

Оптимизация Улучшение промежуточного кода для повышения эффективности

Генерация целевого кода Формирование конечного результата (машинный код, другой язык, структура данных)

Сквозные примеры лекции

Пример 1

Арифметические выражения

- Калькулятор
- Приоритет операций
- Переменные

Пример 2

GIFT-формат тестов

- Генератор quiz
- Проверка ответов
- Экспорт в Moodle

Пример 3

Учебный язык

- Простые конструкции
- Обучение основам
- Проверка решений

Инструменты сегодня

ANTLR4: https://www.antlr.org/

Промышленный генератор парсеров

- Поддержка многих языков
- Мощные возможности
- Профессиональный инструмент

Python + Lark: https://www.lark-parser.org/ide/

Быстрое прототипирование

- Простой синтаксис
- Быстрый старт
- Идеально для обучения

Mermaid

Визуализация грамматик и архитектуры

Цели лекции

Знать

- Архитектуру трансляторов
- Типы парсеров (LL/LR)
- Форматы представления AST

Применять

- Создавать учебные трансляторы
- Генерировать тесты
- Проверять решения

Уметь

- Проектировать грамматики
- Выбирать тип парсера
- Строить деревья разбора

Педагогический аспект

Объяснять ученикам структуру языков и алгоритмов

Ограничения регулярных языков

Не могут считать

- Скобки: ((()))
- Вложенность конструкций
- Баланс открывающих/закрывающих

Примеры

- if (x > 0) then y = 1
- while (i < 10) i = i + 1
- 2 + 3 * 4 (приоритет)

Не выражают структуру

- Условные операторы
- Циклы и блоки
- Приоритет операций

Вывод

Нужны более мощные грамматики!

КС-грамматики: формальное определение

Грамматика G

- $G = (N, \Sigma, P, S)$
- N - нетерминалы
- Σ - терминалы
- P - правила вывода
- S - начальный символ

Пример

- $N = \{expr, term, factor\}$
- $\Sigma = \{+, *, (,), number\}$
- $S = expr$
- P: правила вывода

Правила вывода

- $expr \to expr + term$
- $expr \to term$
- $term \to term * factor$
- $factor \to number$
- $factor \to (expr)$

Грамматика арифметических выражений

ANTLR4 нотация

```
expr : expr '+' term
     | expr '-' term
     | term
term: term '*' factor
     | term '/' factor
     | factor
factor: NUMBER
     | '(' expr ')'
NUMBER: [0-9]+
```

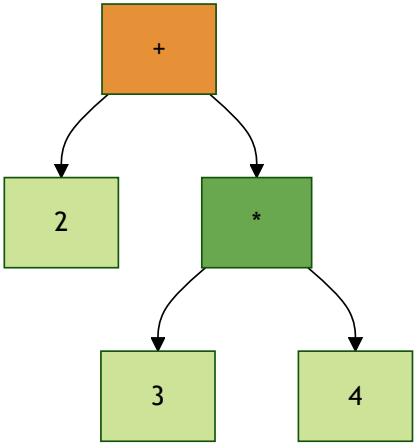
Особенности

- Рекурсивная структура
- Приоритет операций
- Группировка скобками
- Левая рекурсия

Важно для педагогов

Такая грамматика естественно отражает математические правила

Дерево разбора для 2 + 3 * 4



Вычисление

Леворекурсивные vs праворекурсивные грамматики

Леворекурсивные

$expr \longrightarrow expr$ ’+’ $term$

- Естественны для математики
- Проблемны для нисходящих парсеров
- Требуют преобразования

Праворекурсивные

$expr \longrightarrow term$ ’+’ $expr$

- Легко разбираются нисходящими методами
- Менее интуитивны
- Могут изменить ассоциативность

Педагогический аспект

Левая рекурсия соответствует естественному порядку вычислений ”слева направо”

Нормальные формы грамматик

BNF

Форма Бакуса-Наура

<**expr**> ::= <**expr**> "+" <**term**>
| <**term**>

Классическая нотация

EBNF

Расширенная BNF expr ⟶ term ('+' term)*
Более компактная запись

ANTLR4

Практическая нотация

expr: term ('+' term)* ;

Близка к EBNF

Преобразования

- Устранение левой рекурсии
- Левая факторизация
- Приведение к нормальной форме

МП-автоматы: исполнители КС-грамматик

Структура МП-автомата

- **Стек** - память для вложенных конструкций
- **Управляющее устройство** - правила переходов
- **Входная лента** - исходная строка

Пример: проверка скобок

- Открывающая скобка - push в стек
- Закрывающая скобка - pop из стека
- Ошибка если стек пуст или не совпадает

Практическое задание: грамматика с переменными

Задача

Напишите грамматику для выражений с переменными:

- x + 2 * y
- (a + b) * c - 5
- temp / 3.14

Вопросы

- Какие терминалы?
- Какие нетерминалы?
- Как обеспечить приоритет?

Решение

expr: expr ('+' | '-') term | term
term: term ('*' | '/') factor | factor
factor: NUMBER | ID | '(' expr ')'
NUMBER: [0-9]+ ('.' [0-9]+)?
ID: [a-zA-Z_][a-zA-Z_0-9]*

Два подхода к синтаксическому анализу

Нисходящий (Top-Down)

- LL-парсеры
- От начального символа к строке
- Рекурсивный спуск
- **ANTLR4**

Восходящий (Bottom-Up)

- LR-парсеры
- От строки к начальному символу
- Сдвиг-свертка
- **Bison/Yacc**

Аналогия

Разбор предложения «сверху вниз»

Аналогия

Сборка конструктора «снизу вверх»

LL(1) грамматики

Требования

- По одному токену впереди можно выбрать правило
- Нет неоднозначностей
- Предсказуемый разбор

Проблемы

- Левая рекурсия
- Неоднозначные грамматики
- Общие префиксы правил

Преобразования

- Устранение левой рекурсии
- Левая факторизация
- Вычисление FIRST/FOLLOW

Педагогическая ценность

LL(1) грамматики проще для понимания и отладки

Сравнение LL vs LR парсеров

Параметр	LL-парсеры	LR-парсеры
Сложность понимания	Проще	Сложнее
Мощность	Ограничены	Более мощные
Отладка	Проще	Сложнее
Инструменты	ANTLR4	Bison/Yacc
Левая рекурсия	Запрещена	Разрешена
Ошибки	Раннее обнаружение	Позднее обнаружение

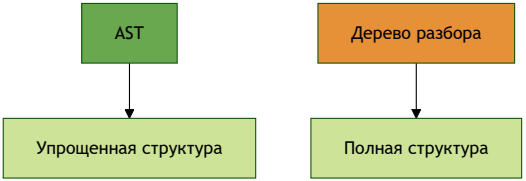
Рекомендация для педагогов

Начинать с LL-парсеров - они интуитивно понятнее для студентов

Абстрактное синтаксическое дерево (AST)

Отличие от дерева разбора

- Упрощенная структура
- Только существенные узлы
- Нет служебных символов



Пример: 2 + 3 * 4

- Бинарные операции
- Числовые литералы
- Сохранение приоритета

Зачем нужно AST

Удобное представление для семантического анализа и генерации кода

Построение AST в ANTLR4

С действиями

```
expr returns [ExprNode node]
  : left=expr '+' right=term
    { $node = AddNode($left.node, $right.node) }
  | term { $node = $term.node }
  ;
```

С помощью листенеров

```
class MathListener(ParseTreeListener):
    def exitAddExpr(self, ctx):
        left = self.get_node(ctx.expr(0))
        right = self.get_node(ctx.term())
        ctx.node = AddNode(left, right)
```

Структура узлов

```
class AddNode:
    def __init__(self, left, right):
        self.left = left
        self.right = right
        self.type = 'add'
```

Обработка синтаксических ошибок

Типичные ошибки

- 2 + * 3
- (x + y
- if then else
- Неизвестные символы

Педагогический аспект

- Понятные сообщения об ошибках
- Указание позиции ошибки
- Предложения по исправлению
- Обучение через ошибки

Стратегии восстановления

- Пропуск токена
- Вставка ожидаемого
- Переход к синхронизации

Пример 1: Транслятор арифметических выражений

Вход

"2 + 3 * (10 - x)"

Этапы обработки

1. Лексер: числа, операторы, переменные
2. Парсер: AST с приоритетами
3. Вычисление/трансляция

Применение

- Калькулятор для уроков математики
- Проверка алгебраических преобразований
- Генерация заданий

Полная грамматика выражений

ANTLR4 грамматика

```
grammar MathExpr;
expr: expr ('+' | '-') term
    | term
    ;
term: term ('*' | '/') factor
    | factor
    ;
factor: NUMBER
    | IDENTIFIER
    | '(' expr ')'
    ;
NUMBER: [0-9]+ ('.' [0-9]+)?;
IDENTIFIER: [a-zA-Z_][a-zA-Z_0-9]*;
WS: [ \t\r\n]+ -> skip;
```

Особенности

- Поддержка вещественных чисел
- Идентификаторы переменных
- Полный набор операций
- Группировка скобками

Педагогическое применение

Идеально для создания проверялок домашних работ по алгебре

Пример 2: GIFT-формат тестов

Стандарт GIFT

- Используется в Moodle, Quizlet
- Текстовый формат
- Разные типы вопросов

Преимущества

- Простота редактирования
- Совместимость с системами
- Разнообразие форматов

Примеры

Столица Франции {=Париж}
2+2 {=4 =четыре}
Выберите столицы {~Лондон =Париж ~Берлин}
Верно/Неверно {T} Столица РФ - Москва

Педагогическое применение

- Создание банка задач
- Генерация вариантов
- Автоматическая проверка

Грамматика GIFT-формата

ANTLR4 грамматика

```
grammar GiftFormat;
quiz: question+ ;
question: text '{' answer+ '}' ;
answer: '=' text weight?   # correctAnswer
      | '~' text weight?   # incorrectAnswer
      ;
text: TEXT ;
weight: '%' NUMBER '%' ;
TEXT: ~[{}]+ ;
NUMBER: [0-9]+ ;
WS: [ \t\r\n]+ -> skip ;
```

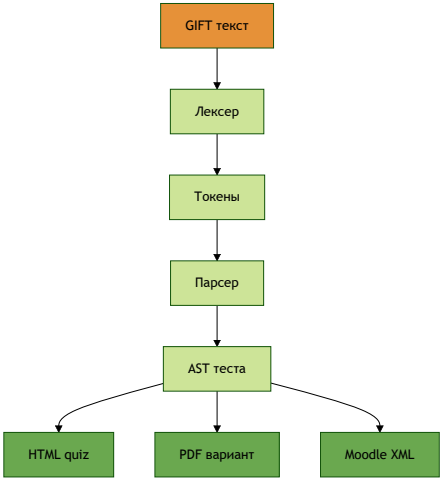
Типы вопросов

- Одиночный выбор
- Множественный выбор
- Верно/Неверно
- Короткий ответ
- Числовой ответ

Расширения

- Веса ответов
- Форматирование текста
- Изображения и медиа

Архитектура транслятора GIFT



Пример 3: Учебный язык программирования

Цели

- Обучение основам программирования
- Минимальный синтаксис
- Наглядное выполнение
- Пошаговая отладка

Педагогические преимущества

- Простота понимания
- Минимум синтаксического сахара
- Прямая связь с алгоритмами
- Легкость отладки

Пример программы

```
x = 10
sum = 0
while x > 0:
    sum = sum + x
    x = x - 1
print(sum)
```

Грамматика учебного языка

ANTLR4 грамматика

```
grammar EduLang;
program: statement+ ;
statement: assignment | while_loop | print ;
assignment: ID '=' expr ;
while_loop: 'while' expr ':' statement+ ;
print: 'print' expr ;
expr: expr ('+' | '-') term | term ;
term: term ('*' | '/') factor | factor ;
factor: NUMBER | ID | '(' expr ')' ;
NUMBER: [0-9]+ ;
ID: [a-zA-Z_][a-zA-Z_0-9]* ;
WS: [ \t\r\n]+ -> skip ;
```

Конструкции языка

- Присваивание переменных
- Цикл while с условием
- Вывод на печать
- Арифметические выражения
- Группировка скобками

Расширения

- Условные операторы
- Функции
- Массивы
- Строки

Педагогические применения

Для учителей математики

- Генератор вариантов контрольных
- Проверка алгебраических решений
- Создание интерактивных заданий
- Адаптация сложности

Для учителей информатики

- Система проверки задач
- Генератор тестовых данных
- Визуализация алгоритмов
- Автоматическая проверка ДЗ

Интеграция

- Moodle, Google Forms
- Jupyter Notebook
- LearningApps
- Word/PDF генерация

Преимущества

- Экономия времени
- Индивидуализация
- Объективность оценки
- Статистика успеваемости

Практическое задание

Задание

1. Выберите свой предмет
2. Спроектируйте грамматику
3. Опишите 3-5 правил в BNF
4. Приведите примеры

Критерии оценки

- Корректность грамматики
- Полнота покрытия
- Практическая ценность
- Ясность описания

Предметные области

- Математика: уравнения, задачи
- Информатика: алгоритмы, программы
- Тесты: вопросы, ответы
- Другие предметы

Пример: геометрия

<**task**> ::= "Найти" <**target**> "у" <**figure**>
<**figure**> ::= "треугольника" | "окружности"
<**target**> ::= "площадь" | "периметр"

Лабораторная 2: Парсер условий задач

Цель

Написать парсер для учебного формата заданий

Инструменты

- ANTLR4 + Python
- Регулярные выражения
- Построение AST
- Тестирование

Выходные данные

AST разобранной структуры задания

Этапы выполнения

1. Проектирование грамматики
2. Написание лексера
3. Создание парсера
4. Построение AST
5. Тестирование

Критерии оценки лабораторной

Качество грамматики (30%)

- Однозначность
- Полнота покрытия
- Отсутствие конфликтов
- Читаемость

Педагогическая ценность (25%)

- Применимость в школе
- Удобство использования
- Адаптивность сложности
- Документация

Корректность разбора (30%)

- Тестовые примеры
- Обработка ошибок
- Построение AST
- Валидация результатов

Качество кода (15%)

- Читаемость
- Структура
- Комментарии
- Тестирование

Что дальше?

Лекция 3

Семантический анализ

- Таблицы символов
- Проверка типов
- Области видимости
- Оптимизации

Лабораторная 3

Проверка корректности решений

- Семантический анализ
- Верификация
- Генерация отчетов

Фокус

От синтаксиса к смысловой проверке

Практическое применение

- Проверка домашних работ
- Анализ решений задач
- Генерация подсказок
- Адаптивное обучение

Ключевые выводы

Архитектурные

- Транслятор = Лексер + Парсер + Семантика
- КС-грамматики мощнее регулярных
- AST - удобное промежуточное представление
- LL vs LR - разные подходы к разбору

Педагогические

- Формальные языки - основа ИТ образования
- Можно создавать полезные учебные инструменты
- Понимание процессов важнее результатов
- Интеграция с школьной практикой

Главный результат

Практический навык создания трансляторов для реального использования в образовательном процессе в школе