Алгоритм Эрли

Позднякова Алиса Б05-924

4 октября 2022

Алгоритм Эрли - алгоритм, осуществляющий парсинг слова в условиях заданной КС грамматики и определяющий, принадлежит ли слово ей.

Синтаксический анализ (парсинг) – это процесс сопоставления линейной последовательности лексем (слов, токенов) языка с его формальной грамматикой.
Синтаксический анализатор (парсер) – это программа или часть программы, выполняющая синтаксический анализ.

Парсинг применяется прежде всего:
1. Для регулярных выражений;
2. Для разбора математических выражений, ЯП, структурированных форматов данных;
3. Для анализа естественных языков – орфография, пунктуация, грамматика, машинный перевод;

Алгоритм Эрли может использоваться на этапе компиляции для построения AST-дерева из исходного кода программы.

1 Необходимые математические понятия

Def: Формальная грамматика — способ описания формального языка, представляющий собой четверку
$\Gamma =< \Sigma, N, S \in N, P \subset ((\Sigma \cup N)^* N (\Sigma \cup N)^*) \times (\Sigma \cup N)^* >$, где:

Σ — алфавит, элементы которого называют терминалами;
N — множество, элементы которого называют нетерминалами;
S — начальный символ грамматики;
P — набор правил вывода $\alpha \to \beta$

Def: Контекстно-свободной грамматикой называется грамматика, у которой в левых частях всех правил стоят только одиночные нетерминалы.

Пример КС грамматики:

Правильные скобочные последовательности задаются КС грамматикой, где

$\Sigma = \{(,)\}$ — алфавит;
$N = \{S\}$ нетерминал;
S — стартовый нетерминал;
P — набор правил:

1. $S \to (S)S$
2. $S \to S(S)$
3. $S \to \epsilon$

Выведем слово (()(()))():
$S \Rightarrow (S)S \Rightarrow (S)(S)S \Rightarrow (S)()S \Rightarrow (S)() \Rightarrow (S(S))() \Rightarrow (S(S)(S))() \Rightarrow (S(S)(S(S)))() \Rightarrow (S(S)((S)))() \Rightarrow (S()((S)))() \Rightarrow (()((S)))() \Rightarrow (()(()))()$

2 Описание алгоритма Эрли

Алгормтм Эрли получает на вход КС грамматику и слово. Затем проверяет принадлежит ли это слово КС грамматике.

Алгоритм:

Пусть $G = <N, \Sigma, P, S>$ - КС грамматика; $w = w_0 w_1 \dots w_{n1}$ - слово из символов алфавита

Введем понятие ситуации:
Ситуация - объект вида $[A \to \alpha \cdot \beta, i]$,
где $A \to \alpha\beta$ - правило из грамматики, i - позиция буквы в слове w.

Алгорим Эрли - динамический алгоритм. Он состоит из n итераций построения списков ситуаций.

Список ситуаций на j итерации обозначим D_j. Наличие ситуации $[A \to \alpha \cdot \beta, i]$ в этом списке означает, что из стартового нетерминала выводится $w = w_0 w_1 \dots w_{i1} A\delta$, а из нетерминала A выводится $w = w_i w_{i+1} \dots w_{j1}$ (то есть часть слова до j буквы, возможно с добавлением нескольких терминальных и нетерминальных символов, выводится по правилам КС грамматики; причем часть слова от i до j буквы выводится из нетерминала A)

Суть алгоритма заключается в том, что на каждой итерацией мы проверям выводима ли часть слова до j буквы, то есть если на n итерации список ситуаций содержит $[S \to S_1 \cdot, 0] \in D_n$

D_j строится по следующим правилам:
1. Если $[A \to \alpha \cdot w_j \beta, i] \in D_{j-1}$, то $[A \to \alpha w_j \cdot \beta, i] \in D_j$
2. Если $[B \to \eta \cdot, i] \in D_j$ и $[A \to \alpha \cdot B\beta, k] \in D_i$, то $[A \to \alpha B \cdot \beta, k] \in D_j$.
3. Если $[A \to \alpha \cdot B\beta, i] \in D_j$ и $(B \to \eta) \in P$, то $[B \in \cdot\eta, j] \in D_j$.

Пример

Построим список разбора для строки $w = (a + a)$ в грамматике со следующими правилами:

- $S \to T + S$
- $S \to T$
- $T \to F * T$
- $T \to F$
- $F \to (S)$
- $F \to a$

D_0

Ситуация	Из правила
$[S' \to \cdot S, 0]$	0
$[S \to \cdot T + S, 0]$	3
$[S \to \cdot T, 0]$	3
$[T \to \cdot F * T, 0]$	3
$[T \to \cdot F, 0]$	3
$[F \to \cdot (S), 0]$	3
$[F \to \cdot a, 0]$	3

D_1

Ситуация	Из правила
$[F \to (\cdot S), 0]$	1
$[S \to \cdot T + S, 1]$	3
$[S \to \cdot T, 1]$	3
$[T \to \cdot F * T, 1]$	3
$[T \to \cdot F, 1]$	3
$[F \to \cdot (S), 1]$	3
$[F \to \cdot a, 1]$	3

D_2

Ситуация	Из правила
$[F \to a\cdot, 1]$	1
$[T \to F \cdot *T, 1]$	2
$[T \to F\cdot, 1]$	2
$[S \to T\cdot, 1]$	2
$[S \to T \cdot +S, 1]$	2
$[F \to (S\cdot), 0]$	2

D_3

Ситуация	Из правила
$[S \to T + \cdot S, 1]$	1
$[S \to \cdot T + S, 3]$	3
$[S \to \cdot T, 3]$	3
$[T \to \cdot F * T, 3]$	3
$[T \to \cdot F, 3]$	3
$[F \to \cdot (S), 3]$	3
$[F \to \cdot a, 3]$	3

D_4

Ситуация	Из правила
$[F \to a\cdot, 3]$	1
$[T \to F \cdot *T, 3]$	2
$[T \to F\cdot, 3]$	2
$[S \to T \cdot +S, 3]$	2
$[S \to T\cdot, 3]$	2
$[S \to T + S\cdot, 1]$	2
$[F \to (S\cdot), 0]$	2

D_5

Ситуация	Из правила
$[F \to (S)\cdot, 0]$	1
$[T \to F \cdot *T, 0]$	2
$[T \to F\cdot, 0]$	2
$[S \to T \cdot +S, 0]$	2
$[S \to T\cdot, 0]$	2
$[S' \to S\cdot, 0]$	2

Так как $[S' \to S\cdot, 0] \in D_5$, то $w \in L(G)$.

Библиотека и API:

Код: `https://github.com/alicepozd/Erley-algo`
В этом же репозитории есть doxygen документация
Чтобы начать использовать модуль:

1. Соберите библиотеку erley:
Для этого выполните код, подсавив вместо path_to_project путь до директории:

```
doxygen
mkdir build
cd build
cmake ./path_to_project
make
```

2. Алгоритм Эрли реализован в виде класса Erley.
И имеет 2 public функции:

```
1 void set_grammar(const std::set<std::string>& new_grammar)
```

Функция, которая устанавливает КС грамматику.
Вход функции: множество правил, заданных строками вида $A \to \alpha$,
где A - нетерминал, α - некоторая последовательность терминальных и нетерминальных символов

```
1 bool predict(std::string& word)
```

Функция проверяет принадлежность слова к установленной в алгоритме грамматике.
Вход: строка (последовательность терминальных символов)
Выход: 0 (если слово не лежит в грамматике)/ 1 (если слово лежит в грамматике)

Детали реализации:

Для реализации алгоритма дополнительно используются структуры Rule и Situation, соответствующие описанным выше.

Алгоритм инициализируется с помощью добавления в D_0 правила $[S' \to S]$, где S - стартовый нетерминал. Сам алгоритм динамический, каждая итерация - построение i списка ситуаций. Для этого используются 3 функции:

1. Функция Scan. Применяется в начале каждого построения нового списка: проходя по всем ситуациям списка D_{i-1} проверяет выполнения правила 1, после чего добавляет в список нужную ситуацию.

Затем пока D_i изменяется последовательно выполяем следующие функции:

2. Функция Complete. Дополняет D_i с помощью правила 3. (отличие от остальных функций проходит по всем спискам ситуаций, а не только по D_{i-1})

3. Функция Predict. Дополняет D_i с помощью правила 2.

3 Тестирование

Написаны тесты для грамматик:

1)Задающей арифметические выражения
2)Задающей правильные скобочные последовательности
3)Для грамматики из примера выше

Тесты собираются в объект erley_test.
Для запуска тестов из директории build воспользуйтесь командой:

```
./source/erley/erley_test
```

4 Ссылки

Article: Earley - An Efficient Context-Free Parsing Algorithm
http://staff.icar.cnr.it/ruffolo/progetti/projects/10.Parsing%20Earley/1970-An%20efficient%20context-free%20parsing%20algorithm-earley.pdf

Формальные грамматики:
https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8

КС грамматики:
https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE-%D1%81%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D1%8B%D0%B5_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8,_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%BB%D0%B5%D0%B2%D0%BE-_%D0%B8_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4,_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%80%D0%B0%D0%B7%D0%B1%D0%BE%D1%80%D0%B0

Алгоритм Эрли:
https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%AD%D1%80%D0%BB%D0%B8

Асимптотика:
https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%AD%D1%80%D0%BB%D0%B8,_%D0%B4%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE_%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8_O(n%5E2)_%D0%B4%D0%BB%D1%8F_%D0%BE%D0%B4%D0%BD%D0%BE%D0%B7%D0%BD%D0%B0%D1%87%D0%BD%D0%BE%D0%B9_%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B8