

Общий восходящий разбор

Поиск, алгоритм Эрли (Earley)

В. С. Полозов

Кафедра системного программирования СПбГУ



Теория автоматов и формальных языков

Восходящий анализ

Структура:

| Стек ($V_T \cup V_N$) | Остаток входной цепочки (V_T) |
|------------------------------------------|--------------------------------------|
| $t_g N_f t_e t_d N_c N_b t_a$ | $t_1 t_2 t_3 \dots$ |
| $\Delta \quad \quad \Delta \quad \Delta$ | |
| (частичные деревья) | |

Две операции:

- shift - сдвиг символа из цепочки в стек
- reduce - замена верхних символов в стеке на соответствующий нетерминал

(Алгоритм shift-reduce – свёртка-сдвиг)

Восходящий разбор поиском

Добавим пару операций к алгоритму
"shift-reduce" ("свёртка-сдвиг"):

- unshift - отмена действия "shift"
- unreduce - отмена действия "reduce"

Что позволит нам делать "откат" (backtracking).

Поиск

Рассмотрим простую грамматику:

$$1. S \rightarrow a S b$$

$$2. S \rightarrow S a b$$

$$3. S \rightarrow a a a$$

И цепочку: **aaaaab**

Два способа обхода при поиске: в глубину и в ширину.

Представим в виде графа

Поиск

Рассмотрим простую грамматику:

$$1. S \rightarrow a S b$$

$$2. S \rightarrow S a b$$

$$3. S \rightarrow a a a$$

И цепочку: **aaaaab**

Два способа обхода при поиске: в глубину и в ширину.

Представим в виде графа

Поиск

Рассмотрим грамматику:

$$S \rightarrow E$$

$$E \rightarrow E Q F$$

$$E \rightarrow F$$

$$F \rightarrow a$$

$$Q \rightarrow +$$

$$Q \rightarrow -$$

Цепочка $a-a+a$ разбирается однозначно.

Однако, граф разбора будет содержать 108 ребер для сдвига и 265 для свёртки.

Алгоритм Эрли

Идея: для уменьшения количества "бесполезных" действий надо добавить нисходящую компоненту разбора, чтобы ограничить свёртки только выводимыми из стартового символа.

Получится парсер нисходящий парсер с ограниченным поиском в ширину, с восходящим распознаванием.

Почему мы его относим к восходящим?

- работа с левой рекурсией
- сложности с ϵ -правилами

Алгоритм Эрли

Идея: для уменьшения количества "бесполезных" действий надо добавить нисходящую компоненту разбора, чтобы ограничить свёртки только выводимыми из стартового символа.

Получится парсер нисходящий парсер с ограниченным поиском в ширину, с восходящим распознаванием.

Почему мы его относим к восходящим?

- работа с левой рекурсией
- сложности с ϵ -правилами

Early item

Введем понятие *ситуации*:

$$A \rightarrow \alpha \bullet \beta @ i, \text{ где } \alpha, \beta \in V^*, A \rightarrow \alpha\beta \in P, i - \text{ позиция}$$

и *множества ситуаций* после p -го символа

$$itemset_p$$

Например, если правило

$$A \rightarrow \alpha \bullet \beta @ i \in itemset_p$$

значит α располагается в строке в диапазоне $\{i \dots p\}$

Алгоритм Эрли

Положим, что $itemset_{p-1}$ заполнен, и очередной символ на входе σ_p . Построим $itemset_p$

- Scanner: Для всех ситуаций вида $A \rightarrow \alpha \bullet \sigma_p \beta @i \in itemset_{p-1}$ добавить ситуацию $A \rightarrow \alpha \sigma_p \bullet \beta @i$ в $completed_p$, если $\beta = \epsilon$, и в $active_p$ иначе.
- Completer: Для всех ситуаций вида $A \rightarrow \gamma \bullet @m \in completed_p$ и $B \rightarrow \alpha \bullet A \beta @i \in itemset_{m-1}$ добавить ситуацию $B \rightarrow \alpha A \bullet \beta @i$ в $completed_p$, если $\beta = \epsilon$, и в $active_p$ иначе.
- Predictor: Для всех ситуаций вида $A \rightarrow \alpha \bullet B \beta @m \in (active_p \cup predicted_p)$, где $B \in V_N$ и $B \rightarrow \gamma \in P$ добавить ситуацию $B \rightarrow \bullet \gamma @p + 1$ в $predicted_p$

Начальное состояние: $active_0 = \{S \rightarrow \bullet \alpha @1 \mid S \rightarrow \alpha \in P\}$

Пример

Рассмотрим грамматику:

$$S \rightarrow E$$

$$E \rightarrow E Q F$$

$$E \rightarrow F$$

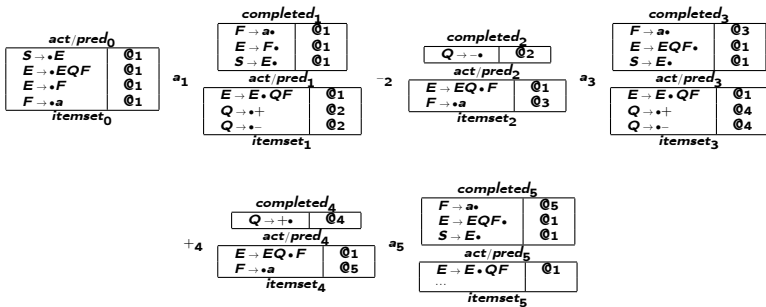
$$F \rightarrow a$$

$$Q \rightarrow +$$

$$Q \rightarrow -$$

Вход: $a-a+a$

Пример



Восстановление дерева разбора

Добавим для этого парсер типа Ангера.
По построенным множествам ситуаций $completed_p$ построим
дерево вывода.

Оценка сложности

Память

- Количество различных ситуаций I зависит от от грамматики.
- К каждой ситуации может приписано от $\Theta(1)$ до $\Theta(p+1)$.
- Соответственно $|itemset_p| \leq I \times (p+1)$.
- Во всех $itemset$ - порядка $I \times p^2/2$.
- Такого же порядка в $completed$.
- Т.е. для входа длины n требуется $O(n^2)$ памяти.

Время

- На позиции p может быть $O(p)$ ситуаций.
- Количество работы для построения пропорционально p^2 .
- Для всех множеств до p - примерно $I \times p^3/6$.
- Итого $O(n^3)$

Оценка сложности

На практике:

- Сложность линейна на многих грамматиках (как в нашем примере).
- Для неоднозначных грамматик $O(n^2)$.
- $O(n^2)$ и $O(n^3)$ появляются для грамматик, где почти все нетерминалы производят почти все цепочки. (Например: $S \rightarrow SS, S \rightarrow x$)

Дальнейшие улучшения

- Обработка ε -правил.
- Заглядывание при предсказании.
- Заглядывание при свёртке.

Вопросы