Общий восходящий разбор

Поиск, алгоритм Эрли (Earley)

В. С. Полозов

Кафедра системого программирования СПбГУ

Теория автоматов и формальных языков

Восходящий анализ

Структура:

Стек ($V_T \cup V_N$)	Остаток входной цепочки (V_T)
$t_g\ N_f\ t_e\ t_d\ N_c\ N_b\ t_a$	$t_1\ t_2\ t_3\ \ldots$
$\Delta \qquad \Delta\ \Delta$	
(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. $S \to aSb$
2. $S \to Sab$
3. $S \to aaa$

И цепочку: **aaaab**

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

Представим в виде графа

Поиск

Рассмотрим простую грамматику:

1. $S \to aSb$
2. $S \to Sab$
3. $S \to aaa$

И цепочку: **aaaab**

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

Представим в виде графа

Поиск

Рассмотрим грамматику:

$S \to E$

$E \to EQF$

$E \to F$

$F \to a$

$Q \to +$

$Q \to -$

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:

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

и множества ситуаций после p-го символа

$$itemset_p$$

Например, если правило

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

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

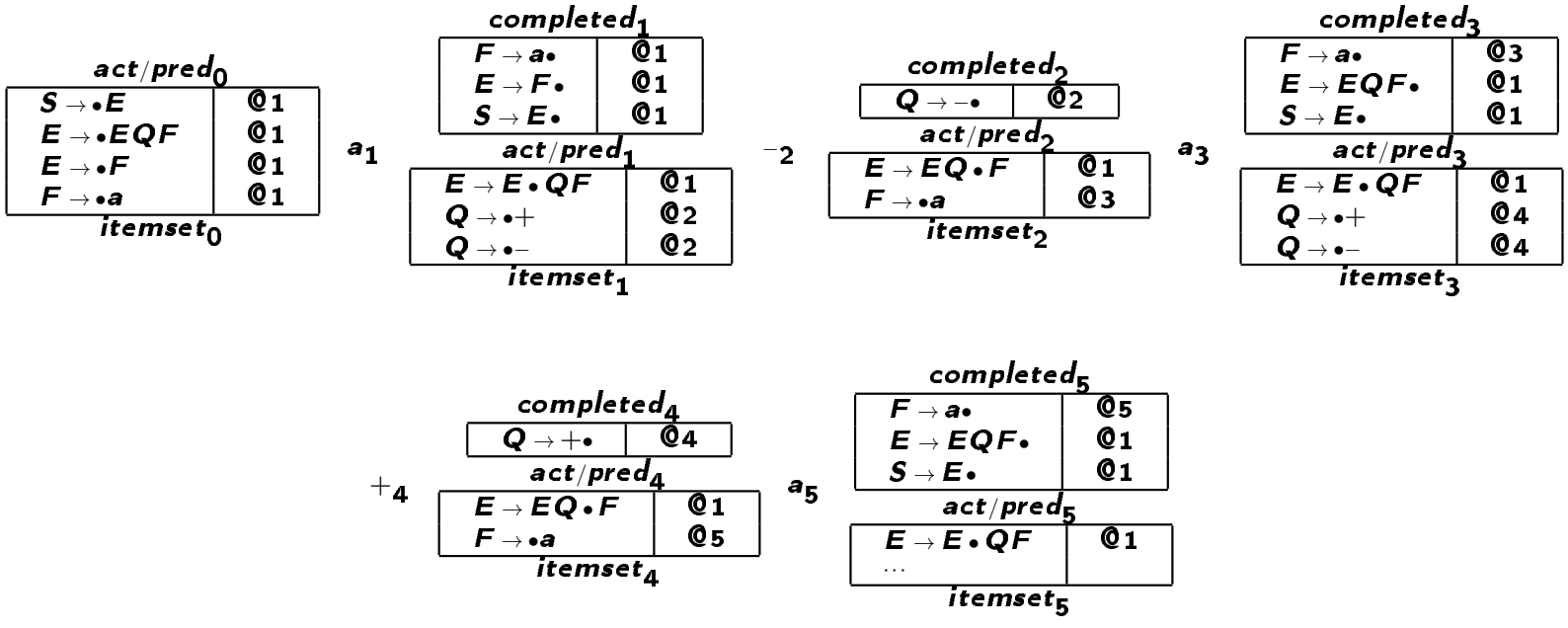
Алгоритм Эрли

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

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

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

Пример


act/pred0
S → •E @1
E → •EQF @1
E → •F @1
F → •a @1
itemset0
a1
completed1
F → a• @1
E → F• @1
S → E• @1
act/pred1
E → E • QF @1
Q → •+ @2
Q → •− @2
itemset1
−2
completed2
Q → −• @2
act/pred2
E → EQ • F @1
F → •a @3
itemset2
a3
completed3
F → a• @3
E → EQF• @1
S → E• @1
act/pred3
E → E • QF @1
Q → •+ @4
Q → •− @4
itemset3
+4
completed4
Q → +• @4
act/pred4
E → EQ • F @1
F → •a @5
itemset4
a5
completed5
F → a• @5
E → EQF• @1
S → E• @1
act/pred5
E → E • QF @1
...
itemset5

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 @1 до @$(p+1)$.
- 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: $S \to SS, S \to x$)

Дальнейшие улучшения

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

Вопросы