Синтаксические ошибки

Восстановление при ошибках

В. С. Полозов

Кафедра системого программирования СПбГУ

Теория автоматов и формальных языков

Определение ошибок

Корректность ввода.
Позиция
Свойство корректного префикса (что не значит, что ошибка именно там)
А остальные ошибки?

Восстановление от ошибок

Восстановление от ошибок

Цель:

- Нахождение всех ошибок
- Избавление от ложных ошибок

Коррекция ошибок

Коррекция ошибок: удаляем, вставляем или изменяем входной символ.

- Можем построить синтаксическое дерево
- Семантический код выполняется в правильном порядке

Ненаправленные методы

Метод Ангера - если есть ошибка, то мы не найдем основу
Рассмотрим грамматику:

$$Expr \to Expr + Term \mid Term$$

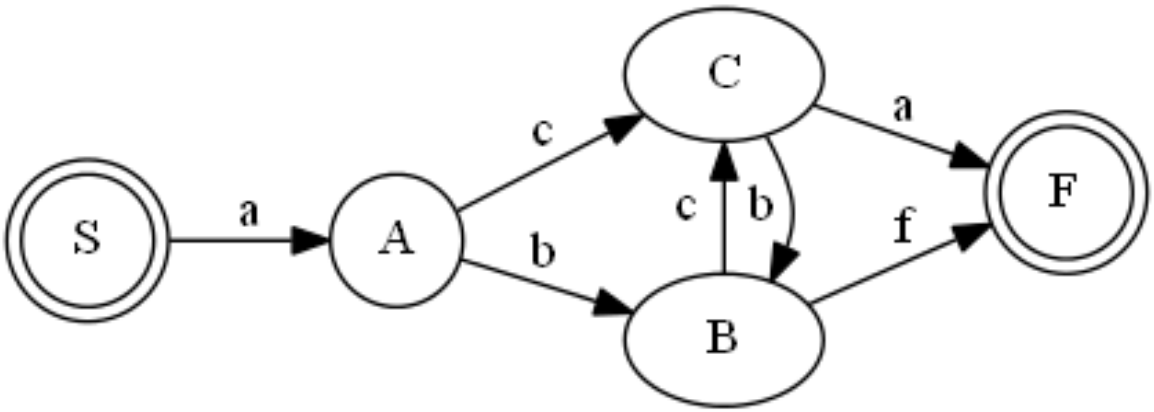
$$Term \to Term \times Factor \mid Factor$$

$$Factor \to (Expr) \mid i$$

И вход $+\times$
CYK - аналогично.
Вывод: без модификации работают плохо.

DFA

Конечные автоматы - очень хорошо.



Свойство корректного префикса.

Свойство немедленного определения ошибки.

Общие направленные нисходящие парсеры

Общий направленный нисходящий разбор поиском в ширину - свойство корректного префикса.
Поиск в глубину - нет.
(Однако, легко добавить - “отметка уровня воды”)

Общие направленные восходящие парсеры

Ошибка - не можем свернуться к стартовому символу.
Однако, при добавлении нисходящей компоненты добавляется свойство корректного префикса. Например, Earley

Линейные нисходящие

Сильные $LL(1)$ - свойство корректного префикса, но не немедленного определения ошибки.

Линейные восходящие

Ошибки в парсерах с приоритетами:

- пара символов
- ошибка свёртки
- ошибка стека

$LR(1)$ обладает свойством немедленного определения ошибки.
$SLR(1)$ и $LALR(1)$ - свойством корректного префикса.
Метод Томиты - такой же, как у базового метода

Восстановление от ошибок

Общие методы - предлагают “общие методы восстановления ошибок”.
Информация для воостановления:

- Восходящий парсер - стек
- Нисходящий - предсказание
- Входная строка и текущая позиция

Классы алгоритмов:

1. *Ad-hoc*
2. Интервальная (regional) обработка ошибок
3. Локальная (regional) обработка ошибок
4. Суффиксный метод

Восстановление от ошибок

Общие методы - предлагают “общие методы восстановления ошибок”.
Информация для воостановления:

- Восходящий парсер - стек
- Нисходящий - предсказание
- Входная строка и текущая позиция

Классы алгоритмов:

1. *Ad-hoc*
2. Интервальная (regional) обработка ошибок
3. Локальная (regional) обработка ошибок
4. Суффиксный метод

Общие методы восстановления ошибок

Метод “наименьшей коррекции ошибки”.

Ad-hoc методы

- Ошибочные продукции
- Табличные - вызывается пользовательская процедура.
- Ошибочный символ

Локальные методы

Основаны на “допустимом множестве”.

1. “Режим паники”
2. множество *FOLLOW*
3. Продолжения (continuations)
4. Локально-минимальные изменения.

Грамматики продолжений

Парсер со свойством корректного префикса находит ошибку после строки u. Назовём строку w - *продолжением*, если $uw \in L$.
Построим грамматику продолжений

- Положим для терминалов 0 шагов вывода, для нетерминалов - ∞
- Для правой части правила - сумма + 1 для левой
- Обновить значение для нетерминала, если получилось меньше
- Повторить, пока что-то меняется
- Выбрать те альтернативы, для которых меньше шагов

Продолжения и *LL*

Рассмотрим грамматику:

$$\begin{aligned} \mathit{Session} &\rightarrow \mathit{Facts}\ \mathit{Question} \mid (\mathit{Session})\ \mathit{Session} \\ \mathit{Facts} &\rightarrow \mathit{Fact}\ \mathit{Facts} \mid \varepsilon \\ \mathit{Fact} &\rightarrow !\ \mathit{STRING} \\ \mathit{Question} &\rightarrow ?\ \mathit{STRING} \end{aligned}$$

Грамматика продолжений:

$$\begin{aligned} \mathit{Session} &\rightarrow \mathit{Facts}\ \mathit{Question} \\ \mathit{Facts} &\rightarrow \varepsilon \\ \mathit{Fact} &\rightarrow !\ \mathit{STRING} \\ \mathit{Question} &\rightarrow ?\ \mathit{STRING} \end{aligned}$$

Вход: (? *STRING* ? *STRING*

Продолжения и *LL*

Рассмотрим грамматику:

$$\begin{aligned} \mathit{Session} &\rightarrow \mathit{Facts}\ \mathit{Question} \mid (\mathit{Session})\ \mathit{Session} \\ \mathit{Facts} &\rightarrow \mathit{Fact}\ \mathit{Facts} \mid \varepsilon \\ \mathit{Fact} &\rightarrow !\ \mathit{STRING} \\ \mathit{Question} &\rightarrow ?\ \mathit{STRING} \end{aligned}$$

Грамматика продолжений:

$$\begin{aligned} \mathit{Session} &\rightarrow \mathit{Facts}\ \mathit{Question} \\ \mathit{Facts} &\rightarrow \varepsilon \\ \mathit{Fact} &\rightarrow !\ \mathit{STRING} \\ \mathit{Question} &\rightarrow ?\ \mathit{STRING} \end{aligned}$$

Вход: (? *STRING* ? *STRING*

Интервальные методы

Восстановление на уровне фразы

- Применим к восходящим анализаторам
- Собираем некоторый контекст рядом с местом ошибки (дополняем стек)
- Сворачиваем (вместе с ошибкой)

Суффиксный разбор

У рассмотренных методов есть общие проблемы

- При обнаружении ошибки они изменяют вход или состояние, используя эвристики чтобы выбрать один из вариантов. Но почему выбранный вариант правильный?
- Выбор неправильного варианта восстановления приводит к лавине ошибок.

Вместо изменения ввода или состояния предположим, что остаток строки после ошибки является корректным суффиксом предложения языка.

Definition

Язык суффиксов языка назовём суффиксным языком.
(а грамматику - суффиксной грамматикой)

Суффиксные грамматики

По правилу

$$A \to X_1 X_2 ... X_n$$

Построим

$$\begin{array}{lcl} A' & \to & X_1' X_2 ... X_n \\ A' & \to & X_2' ... X_n \\ ... & ... & ... \\ A' & \to & X_n' \\ A' & \to & \varepsilon \end{array}$$

(если $X_i \in V_T$, то $X_i' = \varepsilon$)

$$S^{suffix} \to S \mid S'$$

Суффиксный разбор - обсуждение

Восстановление от ошибок: когда оригинальный парсер встречает ошибку, сообщаем о ней, и запускаем суффиксный парсер со следующего символа.
Преимущества:

- Все сообщения об ошибках разные, и ни одна ошибка не сообщается дважды.
- После каждой ошибки парсер перезапускается в правильном состоянии.
- Ввод, кроме символов ошибок, не пропускается.

Однако, суффиксная грамматика может оказаться не LL или LR.

Суффиксный разбор - обсуждение

Восстановление от ошибок: когда оригинальный парсер встречает ошибку, сообщаем о ней, и запускаем суффиксный парсер со следующего символа.
Преимущества:

- Все сообщения об ошибках разные, и ни одна ошибка не сообщается дважды.
- После каждой ошибки парсер перезапускается в правильном состоянии.
- Ввод, кроме символов ошибок, не пропускается.

Однако, суффиксная грамматика может оказаться не LL или LR.

Вопросы