Lemon Parser Generator

Stephan Schulze sschulze@informatik.hu-berlin.de

Gliederung

- Einleitung
- Erzeugen des Parsers
- Implementierung
- Syntax der Grammatikdatei
- Semantische Aktionen
- Konfliktlösung
- Minako

Einleitung

- entwickelt von D. Richard Hipp (drh@acm.org) (http://www.hwaci.com/sw/lemon/)
- letztes Update 1997
- LALR(1) Parser
- Stellt nur Funktionen zur Implementation eines Parsers bereit
- soll Alternative zu yacc/bison darstellen
- hat Geschwindigkeits- und Bedienungsvorteile gegenüber yacc/bison

Erzeugen des Parsers

lemon [Parameter] grammatikdatei.y

erstellte Dateien:

- grammatikdatei.c
- grammatikdatei.h
- grammatikdatei.out

Erzeugen des Parser

interessante Kommandozeilenparameter

- ***-c*** frühere Entdeckung von Syntaxfehlern
- ***-s*** Ausgabe Parser Statistik
- ***-m*** macht C-Datei kompatibel zu Makeheaders
- ***-g*** Ausgabe der Grammatikdatei
- ***-q*** unterdrückt die Ausgabe der Report Datei
- ***-b*** reduziert Ausgabe in Report Datei

Implementierung

allgemeiner Ablauf:

- erzeugen eines Pointers für den Parser
- allokieren von Speicher für den Parser
- Aufruf der Parse Funktion mit Übergabe des Pointers, des Integer- und des semantischen Wertes des aktuellen Symbols
- Freigabe des vom Parser allokierten Speichers

Syntax Grammatikdatei

allgemeiner Aufbau:

Spezielle Direktiven

Grammatikregeln

Spezielle Direktiven

- %include
- %destructor
- %name
- %parse_failure, %parse_accept
- %stack_size
- %stack_overflow
- %start_symbol
- %token_prefix, %token_type, %type
- %left, %right, %nonassoc

Grammatikregeln

allgemeiner Aufbau:

nonterminal ::= (TERMINAL | nonterminal). {...}?*

- Nicht-Terminale werden **immer** klein geschrieben
- Terminale werden **immer** gross geschrieben
- jede Regel endet mit einem **.**
- an jede Regel kann sich in C-Code in geschweiften Klammern anschliessen
- Lemon kennt keine (E)BNF

Semantische Aktionen

- In yacc/bison werden die Werte der Nicht- Terminale mit $1 ... $n angesprochen
- bei Lemon können Symbole mit einem „Namen“ versehen werden, in Klammern direkt hinter dem Symbol

  nt1(A) ::= nt2(B) PLUS nt3(C). { A = B + C;}

Konfliktlösung

- Shift-Reduce Konflikte
  - ohne Angabe von Präzendenzen shiften
  - Meldung eines „*Parse Conflict*“
- Reduce-Reduce Konflikte
  - ohne Angabe von Präzedenzen Auflösung nach Regel die in Grammatik zuerst kommt
  - Meldung eines „*Parse Conflict*“

Implementation Minaco

- Lösung des Shift-Reduce Konflikt durch Angabe von Assoziativität und Zuweisung einer Präzedenz an ein Nicht-Terminal


R R A G E N ? ?