The Lemon Parser Generator

Einleitung

Der Lemon Parser Generator erzeugt einen LALR(1) Parser und wurde vor über 10 Jahren von D. Richard Hipp entwickelt.
Er steht unter der Adresse http://www.hwaci.com/drh/ zum Download zur Verfügung.
Nichtsdestotrotz findet er auch heute noch Verwendung und zwar in dem freien Datenbankserver SQLite, welcher in PHP5 benutzt werden kann.

Der Lemon Parser Generator funktioniert ähnlich wie Yacc oder Bison, ist aber in punkto Geschwindigkeit und Handhabbarkeit besser. Er benutzt er genauso wie Yacc oder Bison einen DFA als Zustandsautomaten.

Installation

Zur Installation muss man sich den Quelltext des Lemon Parser Generator von der oben genannten Adresse herunterladen und mit einem C-Compiler übersetzen.
Das erzeugte Binary lässt sich auf eine gegebene Grammatikdatei anwenden und erzeugt aus dieser eine C-Quelltext Datei welche Funktionen zur Implementation des Parser bereitstellt.
Da nur C-Dateien mit den Funktionen für den Parser erzeugt werden, ist es durchaus möglich, mehrere von Lemon erzeugte Parser in einem Programm zu benutzen.

Erzeugen eines Parsers für eine Grammatikdatei

Laut Konvention enden Grammatikdateien mit der Endung **.y** .
Nachdem man eine Grammatikdatei entsprechend den Syntaxregeln erstellt hat, ruft man das bei der Installation erzeugte Binary (im folgenden *lemon* genannt) folgendermaßen auf:
lemon [Parameter] grammatikdatei.y

Standardmäßig werden dadurch drei Dateien erzeugt:

- grammatik.c ➔ diese Datei stellt die Funktionen zur Implementation des Parser bereit
- grammatik.h ➔ diese Datei enthält die Zuordnung eines Terminalsymbols zu einem Integerwert
- grammatik.out ➔ diese Datei enthält Informationen über den erzeugten DFA und seine Zustände

.

Syntax einer Grammatik Datei

Eine Grammatikdatei lässt sich grob in zwei Teile gliedern.
Der erste Teil enthält Direktiven für die Grammatik, der zweite Teil enthält die eigentlichen Grammatikregeln.

Eine Direktive beginnt generell mit einem „**%“** – Zeichen. Danach folgt der Name und dann die Parameter oder in geschweiften Klammern eingeschlossener C-Quelltext.
Diese Direktiven können die Abarbeitung der Grammatik und die Implementation der erzeugten Parser Funktionen erheblich beeinflussen.

Eine Grammatikregel hat immer den folgenden Aufbau:
nonterminal ::= (TERMINAL | nonterminal). {C-Quelltext}?*
Der Teil „*C-Quelltext*“ wird ausgeführt, wenn die entsprechende Regel reduziert wird.
Innerhalb dieses C-Quelltextes können beliebige semantische Aktionen ausgeführt werden.

Bei den semantischen Aktionen zeigt sich auch ein wesentlicher Vorteil des Lemon Parser Generator.
Um die semantischen Werte von Symbolen innerhalb einer Grammatikregel benutzen zu können, bietet Lemon die Möglichkeit, Symbole in Grammatikregeln mit Namen anzusprechen.
Dazu fügt man hinter ein Symbol, von welchem man den semantischen Wert benutzen möchte, in Klammern einen Namen für das Symbol ein.
Dieser Name steht dann im C-Quelltext hinter dieser Regel als Variable zur Verfügung.
Ein Beispiel findet sich in der Datei rechner.y in den Vortragsbeispielen.

Implementation der erzeugten Parser Funktionen
Der allgemeine Ablauf zur Benutzung eines von Lemon erzeugten Parsers ist der folgende:

- Inkludieren der von Lemon erzeugten Header Datei
- Reservieren von Speicherplatz für den Parser (Funktion ParseAlloc)
- Übergabe der vom Scanner erkannten Token an den Parser (Funktion Parse)
- Freigabe des reservierten Speichers (Funktion ParseFree)

Beispiele finden sich in den jeweiligen main.c Dateien der Vortragsbeispiele.