SableCC

Benjamin Daeumlich

Grundlegendes

SableCC (www.sablecc.org), geschrieben von Étienne Gagnon, ist ein JAVA Compilergenerator. Es werden ein LALR(1) Parser, ein auf einem DFA basierender Lexer, ein AST-Gernerator und „Treewalker-Klassen“ generiert.

Compilergenerierung

Der Prozess der Compilergenerierung mit SableCC läuft im Wesentlichen in 5 Schritten ab.
Nachdem man eine Grammatikdatei erstellt hat, in der sowohl die Lexik als auch die Grammatik definiert werden, wendet man SableCC auf diese an. Dadurch wird ein „Framework“erstellt, welches Klassen für Parser, Lexer, AST-Generator und die Analyse des AST („Treewalker-Klassen“) enthält.
Danach kann man noch eine Klasse mit semantischen Aktionen („working-class“) erstellen.
Anschließend benötigt man noch eine Main-Methode („driver-class“), die alles miteinander verknüpft, bevor man dann am Ende den Compiler mit JAVA kompilieren kann.

Grammatikdatei

Die Grammatikdatei ist in 5 Abschnitte eingeteilt:

1. `Package`: Ordner, in dem die Dateien generiert werden
2. `Helpers`: für Hilfstoken, z. B. Buchstaben oder Zahlen
3. `Tokens`: für Token
4. `Ignored Tokens`: Angabe von zu überlesenden Token
5. `Productions`: Regelteil

Zur Syntax ist folgendes zu Sagen: Rechte und linke Seite werden jeweils durch ein Gleichheitszeichen getrennt und durch ein Semikolon abgeschlossen. Auf der rechten Seite steht dabei der Name und auf der linken Seite die Definition in EBNF.
Im Regelteil muss man darauf achten, dass jede Alternative einer Regel einen eindeutigen Namen in geschweiften Klammern hinter dem Gleichheitszeichen bekommt. Außerdem sind Klammerungen auf der rechten Seite im Regelteil nicht erlaubt.

Framework

Das Frameworkt enthält neben den „Treewalker-Klassen“ und den Klassen für Parser und Lexer außerdem noch Klassen, die den AST definieren.
Dabei werden folgende Dateien generiert:

- für jedes Token: `T≪Tokenname≫.java`
- für jede Produktion: `P≪Produktionsname≫.java`
- für jede Alternative: `A≪Alternativenname≫≪Produktionsname≫.java`

semantische Aktionen

Klassen, die semantische Aktionen beinhalten, erweitern zumeist die generierte Klasse `DepthFirstAdapter.java`. Diese Klasse enthält zu jedem Knoten des AST folgende Methoden:

- `in≪Knotenname≫` (wird beim Eintritt in einen Knoten ausgeführt)
- `out≪Knotenname≫` (wird beim Verlassen eines Knotens ausgeführt)
- `case≪Knotenname≫` (besucht alle Kinderknoten, Code dazwischen möglich)

Diese Methoden können in den Klassen für die semantischen Aktionen neu definiert werden. Dadurch wird beim Traversieren des Baumes entsprechender JAVA-Code ausgeführt.

Bewertung

Vorteile von SableCC sind die Trennung zwischen generiertem Code und Benutzercode, die Unicode-Unterstützung, die Objektorientierung durch Java und die automatische AST-Generierung.
Allerdings gibt es auch einige schwerwiegende Nachteile. Man kann z. B. keine Präzedenzen definieren, was bei vielen Grammatiken ein Umschreiben zur Folge hat. Außerdem können keine sematischen Attribute definiert werden. Auch eine mangelnde Dokumentation erschwert die Benutzung von SableCC.