Exposé zur Studienarbeit

Ein Compiler für PQL mit zwei Backends (Codegeneratoren).
Betreuer: Prof. Ulf Leser

Thomas Trost
trost@informatik.hu-berlin.de

24. November 2005

1 Motivation

In der Biologie werden Interaktionen zwischen Molekülen mit Netzwerken dargestellt. Diese Netzwerke werden im Allgemeinen durch Graphen visualisiert. Die Anfragen an solche Netzwerke sind auch meistens wieder Graphen. Am Lehrstuhl „Wissensmanagement in der Bioinformatik“ wurde eine einfache Sprache namens „Pathway Query Language (PQL)“ entwickelt, um Anfragen an solche Netzwerke zu stellen. PQL setzt auf einer relationalen Datenbank auf, denn die Informationen von diesen biologischen Netzwerken werden in relationalen Datenbanken gespeichert. Die Syntax von PQL ähnelt der von SQL. Die Semantik ist aber verschieden von der Semantik von SQL [Les05]. Um die Anfragen, welche mit PQL gestellt werden, in einer relationalen Datenbank auszuführen, wird jedoch PL/SQL benötigt. Es ist also ein Compiler von PQL nach PL/SQL notwendig.
Es sind auch andere Methoden zur Berechnung der Ergebnisse denkbar. Eine weitere Methode wäre, das Ergebnis der Anfrage im Hauptspeicher zu berechnen. Für nicht allzu große Graphen ist dies ohne Probleme möglich. Diese Berechnung wäre vor allem von Vorteil, falls die Graphen nicht in einer Datenbank vorliegen. Als Szenario wäre eine strukturierte Textdatei oder eine XML-Datei denkbar. Für dieses Szenario könnte z.B. ein Javaprogramm die Berechnung übernehmen, somit wird in diesem Falle die Datenbank nicht gebraucht.

2 Zielsetzung

Das Ziel dieser Studienarbeit ist die Erstellung eines Compiler mit zwei Backends bzw. Codegeneratoren, einer von PQL nach PL/SQL und ein weiterer von PQL nach Java. Die Compiler werden in Java unter der Verwendung eines geeigneten Compilerbau-Werkzeugs erstellt. Ein sehr wichtiger Aspekt des Compilers soll die leichte Erweiterbarkeit sein. Der zweite Compiler soll eine PQL-Query in ein Java Programm übersetzen, welches die PQL Anfrage dann berechnet. Die Effizienz des erzeugten Java Programms steht nicht im Vordergrund dieser Arbeit.

3 Vorgehensweise

Als ersten Schritt zur Erstellung des Compilers muss die Grammatik von PQL genau spezifiziert werden. Diese Spezifizierung sollte in Backus-Nauer-Form bzw. erweiterter Backus-Nauer-Form (EBNF)erfolgen. Im zweiten Schritt ist zu untersuchen, welcher Compilergenartor sich anbietet. Bei der Auswahl

eines geeigneten Werkzeuges kommen nur solche in Betracht, die als Implementationssprache Java bereitstellen bzw. bei denen Javaklassen entstehen. Dabei sind unter anderem folgende Generatoren in Betracht zu ziehen:

- JavaCC
- JJTree
- ANTLR
- SableCC
- Grammatica

Nachdem ein Generator ausgewählt wurde, wird der Compiler von PQL nach PL/SQL mit diesem Generator implementiert. Anschließend wird der Compiler von PQL nach Java implementiert. Bei der Übersetzung von PQL nach Java, soll ein eigenständiges Javaprogramm entstehen. Dieses Javaprogramm soll die Berechnung der Anfrage vornehmen.
Die grobe Vorgehensweise ist dabei wir folgt: Zuerst muss der Graph aus der Datenbank in den Hauptspeicher geladen werden, danach wir die Berechnung durchgeführt. Zum Schluß werden die Ergebnisse in eine Datenbanktabelle geschrieben. Die Algorithmen zur Berechnung sind ebenfalls zu entwickeln, dabei ist aber nicht wichtig, die effizienteste Implementierung zu finden. Es entsteht für jede PQL-Query ein eigenes Javaprogramm.
Die Umschaltung des Backends erfolgt über einen Komandozeilenparameter, dabei wird als Defaultcodeerzeugung PL/SQL gewählt.
Falls die Sprache PQL erweitert oder verändert wird, sollte der Compiler „leicht“ an die neuen Voraussetzungen angepasst werden können. Zur der Studienarbeit gehört nur eine „einfache“ Übersetzung von PQL nach PL/SQL und Java. Das heißt, es wird kein Queryrewriting zur Optimierung vorgenommen.

Literatur

[Les05] Prof. Ulf Leser, *Paper:* A Query Language for Biolagical Networks, Institut für Informatik der Humboldt-Universität zu Berlin, Informatik-Bericht Nr.187 (2005), 38 Seiten;