Prototypische Implementierung der Diplomarbeit Denny Priebe

Denny Priebe
priebe@informatik.uni-rostock.de

14. Januar 1999

1 Überblick

In [Pri98] wurden eine Abbildungsfunktion von CROQUE-OQL in die vorgeschlagene variablenfreie Algebra sowie eine Regelverwaltung zur externen Verwaltung deklarativer Regelmengen über Ausdrücken dieser variablenfreien Algebra beschrieben. In den folgenden Abschnitten wird die vorgenommene Implementierung vorgestellt.

2 Module

Es existieren drei Komponenten:

cbe Die Komponente *cbe* realisiert das Parsen der OQL-Anfrage, die Transformation in einen Ausdruck der variablenfreien Algebra, die Anwendung von Rewriting-Regeln auf diesen Ausdruck sowie die Auswertung des Algebraausdrucks aufgrund der vorhandenen Daten und Schemainformationen.

cbed Aus technischen Gründen wurde *cbed* als Wrapper um die Komponente *cbe* realisiert. Die Aufgabe von *cbed* ist die Realisierung der Kommunikation zwischen cbe und externen Komponenten, wie einem User-Interface.

cfe Die Komponente *cfe* stellt das User-Interface dar und dient allein der Anzeige und Eingabe von Ergebnissen bzw. Daten. Die komplette Funktionalität enthält das Modul *cbe*.

3 Anwendung

Die Implementierung wurde als Client-Server-System konzipiert. Der Serverprozeß *cbed* ist mit der zu verwendenden Portnummer[1] als Argument zu starten:

```
user@host$ ./cbed 3068
user@host$
```

[1] Über diesen Port erfolgt die weitere Kommunikation etwa mit einem User-Interface

Der eigentliche Serverprozeß *cbe* muß sich dazu entweder im aktuellen Verzeichnis bei Start von *cbed* oder im `PATH` befinden.

Das User-Interface kommuniziert mit dem Serverprozeß über Sockets, so daß auch der Zugriff auf entfernte Rechner (Server) möglich ist. Ein Serverprozeß kann dabei mehrere Clients bedienen. Das User-Interface benötigt keine weiteren Parameter:

```
user@host$ ./cfe &
user@host$
```

Im entsprechenden Dialog sind sowohl Rechner als auch Portnummer eines Serverprozesses anzugeben. Das User-Interface präsentiert Ergebnisse bzw. akzeptiert Eingaben in den Bereichen *OQL*, *result* sowie *status*. Bei Start nicht sichtbar sind die Fenster *Schema*, *Data*, *Rules* sowie *Algebra*.

OQL editierbar; dient der Eingabe von OQL-Anfragen

result nicht editierbar; Anfrageergebnisse oder Fehlermeldungen werden hier angezeigt

status nicht editierbar; Statusinformationen werden angezeigt

Schema Anzeige über Menü Schema→Show; editierbar; dient der Eingabe einer Schemabeschreibung (Teilmenge von ODL)

Data Anzeige über Menü Data→Show; editierbar; dient der Eingabe der zu verwendenden Daten

Rules Anzeige über Menü Rules→Show; editierbar; dient der Eingabe der zu verwendenden Regeln; Formatbeschreibung siehe [Pri98]

Algebra Erscheint bei Notwendigkeit; nicht editierbar; dient der Anzeige der Algebrarepräsentation(en) einer OQL-Anfrage

Bei Start lädt das User-Interface die Schemabeschreibung, Datendefinition bzw. Regeln aus den Files `schema.default`, `data.default` bzw. `rules.default`, falls das betreffende File existiert. Das Einlesen der Daten ist nicht gleichbedeutend mit dem Absenden an der Serverprozeß, sondern bedeutet, daß diese Daten im entsprechenden Feld angezeigt werden. Erst durch den Submit-Button werden die Daten ausgewertet.

Wurde eine OQL-Anfrage eingegeben und an den Serverprozeß geschickt, finden die folgenden Aktionen statt:

- Überprüfung auf syntaktische Korrektheit und Korrektheit gegenüber dem spezifizierten Schema
- Transformation in die variablenfreie Algebra
- Typüberprüfung des Ausdrucks der variablenfreien Algebra
- Anwenden ggf. spezifizierter Rewriting-Regeln

- Auswerten des Originalausdrucks der variablenfreien Algebra aufgrund ggf. spezifizierter Daten
- Anzeige evtl. weitere Algebraausdrücke

Die Bearbeitung einer Anfrage durch den Server erfolgt asynchron.

4 Realisierung

Der Serverprozeß *cbe* wurde in SML implementiert. Der verwendete OQL-Parser ist eine angepaßte Version des Konstanzer Parsers. Entwickelt wurde unter SML 0.93.

Die Komponenten *cfe* und *cbed* wurden in C++ implementiert. Um die Benutzbarkeit des User-Interfaces auch zu gewährleisten, während der Server eine Anfrage bearbeitet, wurden threads eingesetzt. Das User-Interface wurde mittels `gtk+` und `gtk--` entwickelt. Die Kommunikation zwischen *cbe* und *cbed* erfolgt durch Pipes, zwischen *cbed* und *cfe* durch Sockets.

5 Protokollbeschreibung

Zur Kommunikation zwischen *cbed* und *cfe* wurde ein einfaches Protokoll eingeführt. Jede Nachricht besteht aus einem Header, einem optionalen Body sowie einer Endekennung. Ein Header hat das Format `#`*tag*`#\n`, Body kann jede Zeichenfolge außer der Endekennung enthalten. Die Endekennung selbst ist `#EOT#\n`. *tag* gibt den Typ der Nachricht an und kann folgende Werte annehmen:

`?` fordert den Server auf, seinen Status zu liefern; Body ignoriert; mögliche Antworten sind `READY` und `BUSY`

READY mögliche Antwort des Servers auf `?`; Body ignoriert

BUSY mögliche Antwort des Servers auf `?`; Body ignoriert

QUERY fordert den Server auf, die im Body enthaltene Anfrage zu bearbeiten

SCHEMA übergibt das im Body enthaltene Schema dem Server; bisheriges Schema, bisherige Daten und Regeln sind dann ungültig

DATA übergibt die im Body enthaltenen Daten dem Server; bisherige Daten sind dann ungültig

RULES übergibt die im Body enthaltenen Regeln dem Server; bisherige Regeln sind dann ungültig

ALGEBRA Antwort der Servers auf eine `QUERY`; enthält Algebrarepräsentationen der Anfrage; mehrere `ALGEBRA`-Nachrichten auf eine `QUERY` sind möglich.

RESULT Antwort des Servers auf eine `QUERY`; enthält Ergebnis der Operation oder Fehlermeldung

QUIT fordert Server auf, sich zu beenden; Server schickt `ACK`, wenn erfolgreich; Body ignoriert

ACK Antwort des Servers auf eine `QUIT`-Nachricht; Body ignoriert

5.1 Beispiel

```
user@host$ telnet server 3068
Escape character is '^]'.
#?#
#EOT#
#READY#
#EOT#
#QUERY#
1
#EOT#
#ALGEBRA#
(K 1)
#EOT#
#RESULT#
1
#EOT#
#STATUS#
Query successfully read
#EOT#
#QUIT#
#EOT#
#ACK#
Good bye.
#EOT#
Connection closed by foreign host.
```

6 Verfügbarkeit

Die Implementierung ist in Binärform für die Plattformen Linux (glibc2) und Solaris 2.5.1 per anonymous ftp unter

`ftp://laxos.informatik.uni-rostock.de:/pub/people/priebe/croque`

verfügbar.

Literatur

[Pri98] Denny Priebe. *Konzeption einer variablenfreien CROQUE-Algebra und Implementierung einer geeigneten Regelverwaltung.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, Dezember 1998