Über ein Werkzeug zur visuellen Auswertung funktionaler Programme

Rudolf Berghammer

Institut für Informatik
Christian-Albrechts-Universität Kiel
Olshausenstraße 40, D-24098 Kiel

Funktionale Programme sind in der Regel einfacher zu verstehen, leichter zu erstellen und zu verifizieren und folglich auch weniger fehlerhaft als ihre imperativen Gegenstücke. Deshalb wird das funktionale Programmierparadigma oft verwendet, um an Universitäten den Informatikstudierenden die grundlegenden Konzepte des Programmierens nahezubringen. Eine bekannte operationale Art, die Semantik von funktionalen Programmen zu definieren, verwendet Berechnungssequenzen $e_1, e_2, \ldots$ zur Beschreibung der Einzelschritte einer Auswertung. Dabei sind die e_i Terme und werden, ausgehend vom auszuwertenden Originalterm e_1, nacheinander dadurch erzeugt, daß man e_{i+1} aus e_i durch einen Expansionsschritt gefolgt von einem Vereinfachungsschritt gewinnt. Der Expansionsschritt ersetzt dabei gewisse Aufrufe einer Rechenvorschrift jeweils durch den Rumpf, wobei in diesem noch die formalen Parameter durch die jeweiligen Argumente substituiert werden. Welche Aufrufe zu ersetzen sind, wird durch eine Berechnungsregel (die Parameterübergabeart) bestimmt. Die bekanntesten Beispiele sind Leftmost-innermost-Ersetzung (Call-by-value), Leftmost-outermost-Ersetzung (Call-by-name) und volle Ersetzung (Herbrand-Regel). Der an die Expansion sich anschließende Vereinfachungsschritt führt die Operationen der primitiven Datentypen und Fallunterscheidungen soweit wie möglich aus.

Die eben skizzierte Semantik wertet ein funktionales Programm imgrundegenommen dadurch aus, daß bedeutungserhaltende Transformationen angewendet werden. Sie ist nicht nur für Anfänger besonders leicht verständlich, sondern auch relativ einfach zu implementieren. Die Vorteile, die sich aus einem entsprechenden Computersystem ergeben, sind vielfältig. Dies gilt insbesondere, wenn man mittels moderner GUI-Toolkits Terme geeignet graphisch darstellt und dem Benutzer die volle Kontrolle über den Auswertungsprozeß überläßt. So ein System kann etwa dazu dienen, termersetzungsartige Programmauswertungen und die Auswirkungen der Berechnungsregeln zu verdeutlichen. Man kann damit aber auch Programmabläufe schrittweise nachvollziehen, z.B. um Schwachstellen und logische Fehler zu erkennen. Eine Visualisierung der Auswertung kann schließlich noch dazu verwendet werden, die Ideen zu erkennen, die dem durch das funktionale Programm implementierten (abstrakten) Algorithmus zugrunde liegen, sowie den durch sie verursachten Aufwand.

Im folgenden beschreiben wir ein Computersystem, das es erlaubt, die Auswertung von funktionalen Programmen erster Stufe zu visualisieren, die in einer einfachen Teilsprache von ML geschrieben sind. Es heißt KIEL, was ein Akronym ist für "Kiel Interactive Evaluation Laboratory", und wurde in den letzten Jahren an der CAU Kiel speziell für den Unterricht im Informatik-Grundstudium an Universitäten konzipiert.

Die von KIEL unterstützte Teilsprache von ML entspricht, bis auf die Verwendung von höheren Funktionen und von Polymorphie, dem, was man üblicherweise zur Einführung in die funktionale Programmierung verwendet. Deklarationen werden wie in ML formuliert, mit zwei Ausnahmen. Die erste Ausnahme betrifft die Typisierung. Wegen des fehlenden Polymorphismus muß eine Rechenvorschrift vollständig typisiert sein. Dabei wird der Typ m eines formalen Parameters x innerhalb der Parameterliste in der Form $x : m$ angegeben und der Resultattyp wird, ebenfalls mit dem Doppelpunkt als Trennzeichen, der Parameterliste nachgestellt. Die zweite Ausnahme betrifft die Aufrufe von Basisoperationen und Rechenvorschriften. Aus methodischen Gründen wird hier grundsätzlich die klassische mathematische Notation $F(A_1, \ldots, A_n)$ verlangt. Hier ist ein kleines Beispiel:

```
fun flatten (s: int list list): int list =
  if null(s) then nil
             else hd(s) @ flatten(tl(s));
```

Die Rechenvorschrift `flatten` überführt eine Liste von Listen ganzer Zahlen durch die Konkatenation ihrer Elemente in eine einzige Liste ganzer Zahlen.

KIEL ist in der Programmiersprache C implementiert und verwendet die Bibliotheken GTK, GDK und GLib für die graphische Benutzerschnittstelle. Obwohl der Benutzer die Oberfläche des Systems in weiten Teilen selbst gestalten kann, zeigt Abbildung 1 eine typische Benutzeroberfläche mit den wichtigsten Fenstern. Aus diesem Bild ist erkennbar, daß KIEL Terme durch Bäume darstellt. So entspricht der Baum im „tree

window“ einem Zwischenergebnis bei der Auswertung von `flatten([[1,2,3],[1,1,1],[4,3,2,1],[2,2]])`. Die Baumdarstellung von Termen macht deren Struktur extrem deutlich. Weiterhin ermöglicht sie eine sehr einfache Auswahl der Anwendungsstellen bei schrittweisen oder teilweisen Auswertungen.

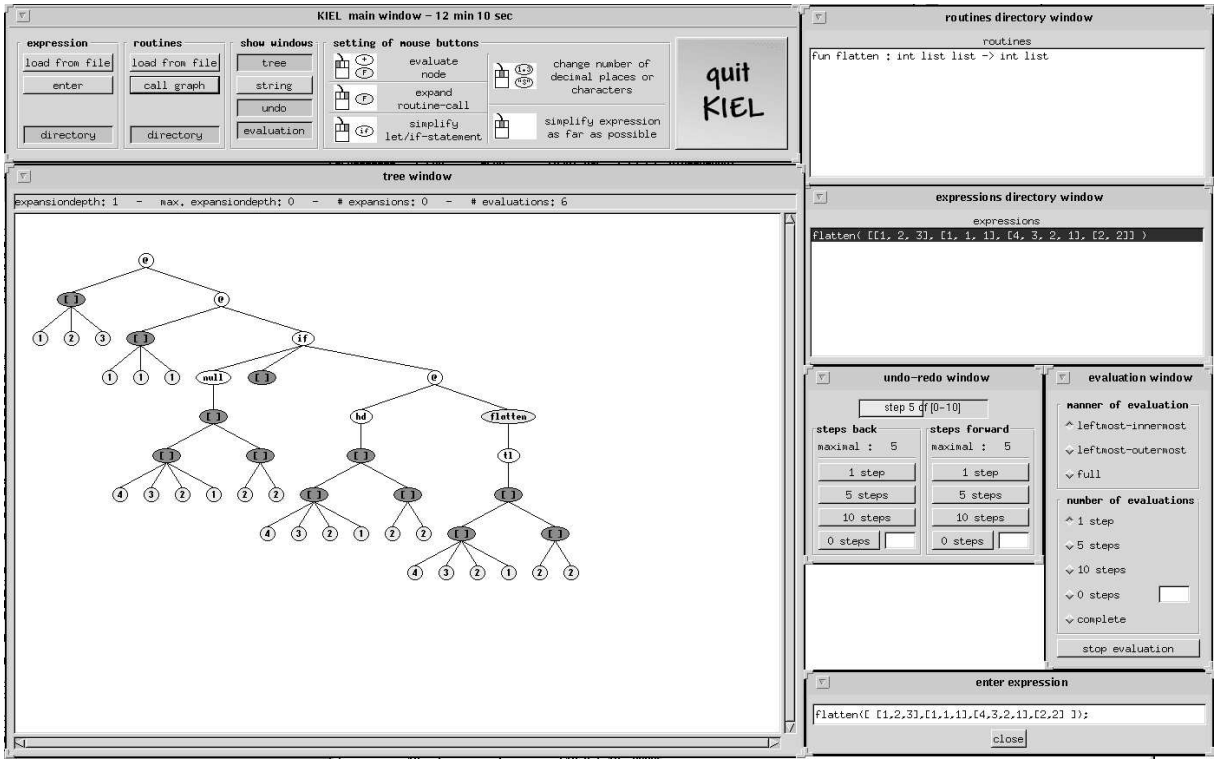

Abbildung 1: Die Benutzerschnittstelle des KIEL-Systems

Aus der Sicht des Benutzers besteht KIEL aus zwei Komponenten. Die erste Komponente ist der Arbeitsbereich. Er enthält die Programme und auszuwertenden Terme. Programme werden aus Dateien in das System geladen. Auch Terme können aus Dateien geladen werden; sie werden jedoch üblicherweise direkt mit Hilfe eines Fensters eingegeben. Die zweite Komponente des Systems ist die Auswertungseinheit. Durch sie werden Terme des Arbeitsbereichs berechnet. Die Auswertung kann schrittweise, semiautomatisch oder vollautomatisch erfolgen. Ersteres geschieht mittels einzelner Mausklicks im entsprechenden Fenster, in den beiden anderen Fällen kommen noch zusätzliche Fenster ins Spiel, mit denen etwa eine spezielle Berechnungsregel und die maximale Schrittanzahl festgelegt werden können. An dieser Stelle sollte auch erwähnt werden, daß KIEL einen Undo/Redo-Mechanismus besitzt, der es erlaubt, Auswertungen zu stoppen und Schritte rückgängig zu machen bzw. zu wiederholen. Dieses kann insbesondere benutzt werden, um durch eine Berechnungssequenz „zu wandern“, z.B. zu Zwecken der Visualisierung, der Fehlersuche oder um alternative Berechnungen zu testen.

Das KIEL-System kann auf allen SUN Workstations mit SUN-OS 5.x installiert werden, sowie auf allen PCs mit LINUX als Betriebssystem. Es ist via FTP frei erhältlich vom Server `ftp.informatik.uni-kiel.de`, wo man es im Verzeichnis `pub/kiel/progdev/KIEL` findet. Zusätzliche Informationen werden mittels der Web-Seite `http://www.informatik.uni-kiel.de/~progsys/kiel.html` verbreitet. Durch diese Web-Seite hat man auch Zugang zu einer Online-Version von KIEL. Diese steht, mit einem im Vergleich zur C-Applikation eingeschränkten Sprachumfang und etwas weniger Komfort in der Benutzerinteraktion, allen Anwendern des Internets als Java-Applet zur Verfügung.

Literatur

[1] Berghammer R.: KIEL – Ein Computersystem zur Visualisierung der Auswertung von funktionalen Programmen. In: Diehl S., Kerren A. (eds.): Proc. GI-Workshop „Softwarevisualisierung 2000“, Technischer Bericht A/01/2000, Fachbereich Informatik, Universität des Saarlandes, 53-64 (2000).

[2] Berghammer R., Milanese U.: KIEL – A computer system for visualizing the execution of functional programs. In: Hanus M. (ed.): Proc. International Workshop on Functional and (Constraint) Logic Programming (WFLP 2001), Bericht Nr. 2017, Institut für Informatik und Praktische Mathematik, Universität Kiel, 365-368 (2001).