Generierung textueller Strukturediteren

Sebastian Hagenkötter
abbath@uni-paderborn.de
Universität Paderborn
Fakultät für Elektrotechnik, Informatik und Mathematik

Betreuer der Arbeit: Prof. Dr. Uwe Kastens
Art der Arbeit: Studienarbeit
GI-Fachbereich: Softwaretechnik

Zusammenfassung

In dieser Studienarbeit ist ein Generator für Strukturediteren textueller Sprachen entworfen und implementiert worden. Aus Spezifikationen hohen Niveaus erstellt der Generator Strukturediteren, mit denen Programme der jeweils spezifizierten Sprache editiert werden können. Die Verwendung bekannter Grammatiknotationen seitens der Spezifikationssprache, sowie die Realisierung einer intuitiven, übersichtlichen Darstellung von textuellen Strukturen in den generierten Editoren standen hierbei im Vordergrund.

1. Einleitung

Strukturediteren werden eingesetzt, um die Benutzung von Sprachen zu erleichtern. Der Anwender eines Strukturediteors wird durch eine intuitive Benutzungsschnittstelle bei der Programmkonstruktion unterstützt, und durch strukturorientierte Editieroperationen sind Syntaxfehler bei der Programmerstellung ausgeschlossen.

In dieser Studienarbeit ist ein Generator für Strukturediteren textueller Sprachen entworfen und implementiert worden. Dieser verwendet Teile des VL-Eli Systems [SK02], einer Werkzeugsammlung zur Entwicklung visueller Sprachen, und erweitert dessen Funktionalität. Aus Spezifikationen hohen Niveaus erstellt der Generator Strukturediteren, mit denen Programme der jeweils spezifizierten Sprache editiert werden können (s. Bsp. Abbildung 1). Um eine intuitive, komfortable Bedienung der erstellten Editoren sicherzustellen, wurde das Konzept eines *direct manipulation* Editors umgesetzt, welches erstmals von B. Shneiderman in [Sh83] beschrieben und von Sten Minør in [Mi92] evaluiert wurde. Dieses Konzept stellt sowohl Anforderungen an die Benutzungsoberfläche an sich, als auch an die graphische Hervorhebung der dem Programm zugrundeliegenden Sprachstruktur. Letztere unterscheidet die generierten Editoren von klassischen Strukturediteren (z.B. die vom Synthesizer Generator [RT88] generierten Editoren), da in solchen lediglich textuelle Mittel, wie z.B. Einzüge und Zeilenumbrüche, zur Strukturhervorhebung eingesetzt werden.

Zentrale Bestandteile dieser Studienarbeit sind der Entwurf einer einfachen und zugleich mächtigen Spezifikationssprache und die Realisierung einer dem *direct manipulation* Konzept entsprechenden graphischen Hervorhebung von Sprachstrukturen.

2. Entwurf der Spezifikationssprache

Die Spezifikationssprache soll es dem Sprachentwickler ermöglichen, zum einen die Struktur seiner entwickelten Sprache und weiterhin deren graphische Darstellung zu spezifizieren. Letztere umfasst sowohl die Formatierung von Teilstrukturen, als auch graphische Eigenschaften, wie z.B. Farben. Das Ziel bestand darin, die Syntax der Spezifikationssprache so zu gestalten, dass ein Sprachentwickler Spezifikationsmittel seines Anwendungsgebiets, wie z.B. bestimmte Grammatiknotationen, wiederverwenden kann, und die Spezifikation textueller Anordnungen keine graphischen Details, wie z.B. Koordinaten, erfordert.

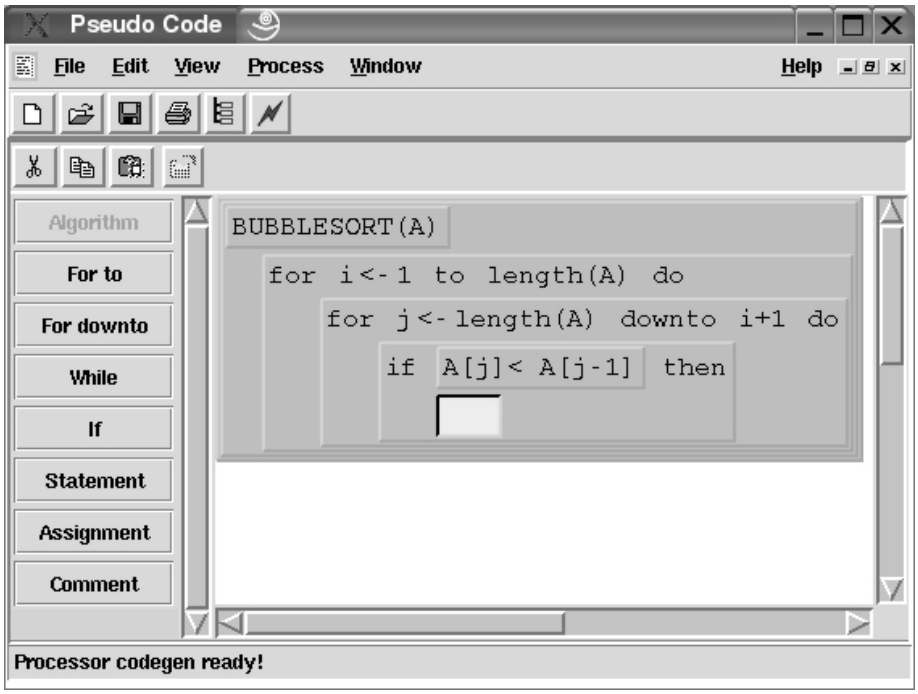

Abb. 1: Generierter Strukturediteur für „Pseudocode“ Programme

Abbildung 2 zeigt die Spezifikation einer einfachen Sprache in der entwickelten Syntax. Der Hauptbestandteil ist die Strukturbeschreibung der Sprache (STRUCTURE), in welcher die bekannte "Backus Naur Form" [Na60] kurz BNF, verwendet wird, mit der zusätzlichen Möglichkeit Listen (*, + oder {n}) und benannte Produktionen zu spezifizieren. Die Elemente von Teilstrukturen können mit Zeilenumbrüchen ([NL]) und Einzügen ([IN], [EX]) angeordnet werden. Die Spezifikation graphischer Eigenschaften, wie z.B. Hintergrundfarben (BC), Farben von Einfügestellen (IC), oder Schriftarten (F), wird mittels einer Textauszeichnungssprache in die Strukturbeschreibung eingebunden. Sollen graphische Eigenschaften nicht nur in einem speziellen Kontext gelten, so können sie in einer DEFAULTS Beschreibung für ein einzelnes Symbol oder auch global für alle Symbole spezifiziert werden. Diese Angaben gelten dann an Auftreten der entsprechenden Symbole, in denen sie nicht durch kontextspezifische Eigenschaften überschrieben werden.

Besonders hervorzuheben ist das realisierte Konzept zur Darstellung von Listen, welches an die Arbeit von Koen DeHondt [DeH93] angelehnt ist. Dieses bietet u.a. über einfache horizontale oder vertikale Anordnungen hinaus die Möglichkeit, Elemente einer Liste abhängig von deren Anzahl zu positionieren. So können je nach aktueller Komponentenzahl Listen mit keinen, wenigen und vielen Elementen übersichtlich dargestellt werden, indem die Liste entweder horizontal (wenige Elemente) oder vertikal (viele Elemente) angeordnet wird. Der Entwickler spezifiziert hierzu lediglich die maximale Anzahl der Elemente, die nebeneinander positioniert werden sollen.

```
LANGUAGE ("Simple","sim")
STRUCTURE [Program] (
                Program    ::= Stmt* .
   "IfStmt":    Stmt       ::= "if " Expr " then" [IN] [NL] Stmt[-IC green] [EX] [NL]
                              "else" [IN] [NL] Stmt[-IC red] .
   "Assignment": Stmt      ::= Identifier " = " Constant .
   "StringExpr": Expr      ::= String .
   "Identifier": Identifier ::= Ident .
   "Constant":   Constant  ::= Const .
)
TERMINALS (
   Ident:  ["a" "b" "c"]
   Const:  INTEGER
   String: STRING
)
DEFAULTS (
   *[-BACK_COLOR gray]
   Constant[-BC darkgray]
)
```

Abb. 2: Beispielspezifikation

3. Realisierung

Der mit dem Eli System [KPJ98], einer Werkzeugsammlung zur Sprachentwicklung, implementierte Generator erstellt aus den oben genannten Sprachspezifikationen VL-Eli Spezifikationen. Diese enthalten neben der abstrakten Struktur der textuellen Sprache auch die Berechnungen zur graphischen Darstellung der einzelnen Sprachelemente und sind daher um ein Vielfaches umfangreicher als die Eingabespezifikation. Mit dem VL-Eli System [SK02] können nun aus den generierten, auf attributierten Grammatiken basierenden Spezifikationen visuelle Strukturediteren generiert werden. Aus der Eingabespezifikation extrahiert der Generator die Struktur der Sprache und fügt ggf. die zur graphischen Darstellung notwendigen, zusätzlichen Produktionen ein. Entsprechend der spezifizierten textuellen Anordnung und den graphischen Eigenschaften werden Attributberechnungen zur Darstellung der Strukturen generiert. Der Generator stellt dabei sicher, dass auch bei semantisch fehlerhaften Eingaben übersetzbare VL-Eli Spezifikationen generiert werden. So kann der Entwickler visuell anhand des generierten Strukturediteors die Fehler seiner Strukturdefinition erkennen. Auf alle semantischen Fehler erhält der Entwickler Warn- bzw. Fehlermeldungen mit Hinweisen zur Behebung des Problems. Auf diese Weise wird der Sprachentwicklungsprozess durch den Generator unterstützt.

Die mit VL-Eli generierten Editoren besitzen bereits eine graphische Benutzungsoberfläche, welche *direct manipulation* Editieren unterstützt. Der Benutzer kann neue Elemente einer Sprache von einer Werkzeugleiste an passende Stellen des Programms ziehen und einfügen. Die einzelnen Elemente des editierten Programms können ausgewählt, bewegt oder auch gelöscht werden. Da diese Benutzungsschnittstelle den Anforderungen des *direct manipulation* Konzepts genügt, musste noch eine graphische Darstellung für die geforderte Hervorhebung syntaktischer Strukturen realisiert werden.

Abbildung 1 zeigt den Strukturediteor, welcher aus einer Spezifikation für die „Pseudocode“ Sprache generiert worden ist. Die Sprachstruktur wird mit visuellen Mitteln, wie z.B. perspektivischen Rechtecken, hervorgehoben. Ein Rechteck umgibt jeweils eine Teilstruktur des editierten Programms, während „abgesenkte“ Rechtecke Platzhalter darstellen, an denen der Benutzer weitere Strukturen der Sprache einfügen kann. Die herausragende Eigenschaft der gezeigten graphischen Darstellung ist ihre Einfachheit. Es gibt nur wenige unterschiedliche graphische Elemente, so dass der Anwender schnell die dargestellten Objekte identifizieren kann. Die Darstellung der Einfügestellen als „abgesenkte“ Rechtecke unterstützt zudem die intuitive Handlungsweise des Benutzers, an „leeren“ Stellen etwas einzufügen, sozusagen die Verwendung eines Baukastensystems. Trotz der Einfachheit kann mit diesem Layout eine große Anzahl textueller Anordnungen dargestellt werden.

4. Zum Einsatz des Generators

Im Rahmen dieser Studienarbeit wurde der Generator mit mehreren Beispielsprachen, darunter u.a. einem Ausschnitt aus Pascal und der Spezifikationssprache des Generators selbst, getestet. Die generierten Strukturediteoren sind aufgrund des realisierten *direct manipulation* Konzepts einfach und intuitiv zu bedienen und durch die Verwendung graphischer Mittel zur Hervorhebung ist die Struktur der editierten Programme immer deutlich zu erkennen.

Mit dem generierten Strukturediteor für die Spezifikationssprache kann ein Benutzer nun z.B. Spezifikationen erstellen, welche immer syntaktisch korrekt sind, und diese direkt als Eingabe für den Generator verwenden. Durch eine ebenfalls entwickelte Code Generierung wird dabei aus der visuellen Anordnung des Programms eine entsprechende textuelle Form zur Speicherung in eine Textdatei erzeugt.

Das Größenverhältnis von Eingabespezifikation zu generierten VL-Eli Spezifikationen verdeutlicht den Nutzen des Generators. Die Spezifikation des Ausschnitts aus Pascal umfasst z.B. 25 Zeilen, aus denen 3250 Zeilen VL-Eli Spezifikationen generiert werden. Im Durchschnitt lag das Verhältnis bei den getesteten Sprachen bei 1 zu 100. Das bedeutet, dass kleine Änderungen in einer spezifizierten Sprachstruktur oder auch nur in der graphischen Darstellung der Sprache umfangreiche Modifikationen in den erstellten VL-Eli Spezifikationen zur Folge haben. Für den Benutzer des Generators bleiben diese Berechnungen jedoch transparent.

Literatur:

[DeH93] De Hondt, K.: A Customizable Ergonomic, Hybrid Structure-Oriented Editor. Brussels Free University, Master Thesis, 1993.

[KPJ98] Kastens, U.; Pfahler, P.; Jung, M.: The Eli System. In Kai Koskimies, Hrsg., Proceedings 7th International Conference on Compiler Construction CC'98, No. 1383. In: Lecture Notes in Computer Science. Springer Verlag, März 1998, S. 294-297.

[Mi92] Minør, S.: Interacting with structure-oriented editors. In: International Journal of Man-Machine Studies 37, 1992, No. 4, S. 399-418. ISSN 0020-7373.

[Na60] Naur, P. (ed.): Revised Report on the Algorithmic Language ALGOL 60. In: Communications of the ACM, Vol. 3, No. 5, Mai 1960, S. 299-314.

[RT88] Reps, T. W.; Teitelbaum, T.: The synthesizer generator: A system for constructing language-based editors. In: Texts and monographs in computer science,Springer Verlag, 1998.

[Sh83] Shneiderman, B.: Direct Manipulation: A step beyond programming languages. In: IEEE Computer, August 1983, S. 57-68.

[SK02] Schmidt, C.; Kastens, U.: VL-Eli: A generator for visual languages. In: Electronic notes in theoretical computer science 65, No. 3, 2002.