Funktional-logische Graph-Parser mit Vervollständigung

Steffen Mazanek, Mark Minas
Institut für Softwaretechnik
Universität der Bundeswehr München

Zusammenfassung

Mit funktional-logischen Programmiersprachen lassen sich auf sehr elegante Weise Parser für Zeichenkettensprachen zusammenbauen (siehe etwa Caballero et al., 1999). Noch stärker können funktional-logische Sprachen ihre Vorteile jedoch bei der Syntaxanalyse von Graphen ausspielen. Graphen sind eine Datenstruktur mit vielen Anwendungen in der Informatik. Insbesondere bilden sie eine natürliche Basis visueller Sprachen. In diesem Anwendungsbereich ist die Syntaxanalyse erforderlich, um korrekte Diagramme erkennen und auswerten zu können.

Ein bekannter Formalismus zur Beschreibung von Hypergraphen, einem erweiterten Graphbegriff, sind so genannte *Hyperkantenersetzungsgrammatiken*. Hier werden nichtterminal markierte Hyperkanten gemäß einer Grammatik durch neue Hypergraphen ersetzt. Dabei werden bestimmte Knoten miteinander verklebt. Hyperkantenersetzungsgrammatiken teilen viele zentrale Eigenschaften kontextfreier Zeichenkettengrammatiken (insbesondere natürlich die Kontextfreiheit). Durch die größere Ausdrucksstärke ist das Parsen im Allgemeinen jedoch nicht mehr so effizient möglich, obwohl sich viele, gerade für die visuellen Sprachen relevante Graphsprachen als hinlänglich effizient zu parsen herausgestellt haben.

Bei der Realisierung einer Bibliothek von *Graph-Parser-Kombinatoren* haben wir festgestellt, dass zur Erstellung von Graph-Parsern sowohl Sprachelemente aus der funktionalen als auch aus der logischen Programmierung sehr nützlich sind:

- Kombinatoren sind *Funktionen höherer Ordnung*, die es ermöglichen, den "verbleibenden Input" bei der Konstruktion von Parsern weitgehend zu verstecken. Prologs *DCGs* hingegen setzen eine lineare Struktur voraus, die bei Graphen nicht gegeben ist.
- Auf der anderen Seite sind *freie Variablen* wie sie aus der Logikprogrammierung bekannt sind für unseren Ansatz sehr vorteilhaft. Wir benötigen diese, um mit noch unbekannten Knoten umzugehen.

Diese Kombination der Stärken funktionaler und logischer Sprachen ermöglicht es uns, die Parser in einer textuellen Notation zu spezifizieren, die der jeweiligen Hyperkantenersetzungsgrammatik 1:1 entspricht. Dabei ist eine leistungsstarke *Attributierung*, wie wir sie von herkömmlichen Parser-Kombinatoren für Zeichenketten kennen und schätzen, möglich. Unsere Parser haben sogar noch einen weiteren herausragenden Vorteil: Sie können bidirektional eingesetzt werden, d.h. wir können (in einem bestimmten Rahmen) Graphen einer Sprache generieren oder sogar bestehende Graphen vervollständigen lassen.

Unsere Bibliothek ist in der funktional-logischen Sprache Curry realisiert und als Komponente in den Diagrammeditorgenerator DiaGen (Minas, 2002) eingebunden. Hier ermöglicht sie, dass generierte Editoren die frei gezeichneten Diagramme nicht nur analysieren, sondern, wenn es der Nutzer anfordert, diese sogar vervollständigen können.