

# **AutoEdit - ein Werkzeug zum Editieren, Simulieren, Transformieren und Publizieren abstrakter Automaten**

Michael Hielscher, Christian Wagenknecht

FB Informatik  
Hochschule Zittau-Görlitz  
Brückenstr. 1  
02826 Görlitz  
mail@genesis-x7.de  
c.wagenknecht@hs-zigr.de

**Abstract:** AutoEdit ist eine Manipulations- und Simulationsumgebung für abstrakte Automaten. Gegenüber existierenden Systemen, wie etwa [JFLAP], richtet sich diese Softwarelösung vorwiegend an Autoren von Unterrichtsmaterialien mit abstrakten Automaten. Ein spezielles XML-Format ermöglicht außerdem vielseitige Weiterverarbeitungsmöglichkeiten.

## **1 Einleitung**

Unterrichtslehrpläne diverser Bundesländer enthalten in der Jahrgangsstufe 11/12, den Lernbereich theoretische Informatik, mit dem Schwerpunkt „Theoretische Grundlagen von Programmiersprachen“, welcher sich vorwiegend mit Sprachen und Automaten beschäftigt.

Die Hauptinhalte sind dabei reguläre und kontextfreie Sprachen, endliche Automaten, Kellerautomaten und Turingmaschinen. Der Aufbau und die Arbeitsweise dieser Automaten sollen mit einfachen Beispielen veranschaulicht werden.

Von Lehrerinnen und Lehrern wird heute zunehmend erwartet, dass sie schriftliche Unterrichtsmaterialien, wie Arbeitsblätter, Lehrtexte, Übungsaufgaben und Klausuren als Textdokumente am Computer produzieren. Diese werden ausgedruckt bzw. ins Web gestellt. Für InformatiklehrerInnen, die mit Computeranwendungen der entsprechenden Art vertraut sind, ist die Herstellung solcher Materialien dennoch ein großes Zeitproblem, denn die typografischen Anforderungen an die Texte sind hoch: Schüler begnügen sich nicht mit reinem Textformat im E-Mail-Stil, sondern erwarten typografisch anspruchsvolle Dokumente, die oftmals Formeln und grafische Darstellungen enthalten. Gängige Textprozessoren verfügen über Formeleditoren und Zeichenkomponenten, die den „Lehrmaterial-Redakteur“ unterstützen.

Dennoch gibt es Felder, in denen LehrerInnen bisher vergeblich nach einem geeigneten Werkzeug suchen. Beispielsweise fehlt es an einem Editor für Graphen abstrakter Automaten. Ein solcher Editor sollte die Definition abstrakter Automaten ermöglichen, zu-

gehörige Zustandsüberföhrungsgraphen generieren, bzw. automatisch anordnen und manuelle Nachbesserungen gestatten.

Hinzu kommt, dass Lehrende unmittelbar beim Entwurf eines Beispielautomaten dessen Anwendung auf Eingabewörter erproben bzw. simulieren möchten. Folglich sollte ein Entwurfswerkzeug für abstrakte Automaten nicht nur ein Editor sein, der gewisse Publikationsaufgaben unterstützt, sondern sollte darüber hinaus Simulationen und gängige Transformationen ermöglichen. Im übertragenen Sinne ist ein dementsprechendes Werkzeug auch für die Hand des Schölers geeignet.

Das von uns entwickelte Programm AutoEdit ist ein kostenlos verfügbares, didaktisches Tool dieser Art. Dabei handelt es sich um eine relativ kleine Datei mit sehr geringem Hardwareanspruch. Im Hinblick auf die Softwareumgebung werden ebenfalls keine nennenswerten Anforderungen gestellt. Die Installation geschieht automatisch, ein Wizard unterstützt den Anwender bei der Einarbeitung in den Systemgebrauch.

## 2 Vorhandene Software und ihre Grenzen

Auf dem Markt findet man bereits gute Softwarelösungen für die einzelnen Teilgebiete, wie Simulation, Transformation (zum Beispiel JFLAP) und Präsentation (zum Beispiel MS Visio). Dennoch findet man keine Gesamtlösung, die all diese Gebiete in einem Programm vereint. Durch ein fehlendes gemeinsames Datenformat ist der Austausch von Automatendefinitionen zwischen den vorhandenen Programmen auch sehr schwierig.

Für Autoren, die ihre Materialien mit  $\text{\LaTeX}$  verfassen, gibt es auch direkte Packages für das Zeichnen von Zustandsübergangsgraphen. Diese beschränken sich jedoch zum einen ausschließlich auf die Darstellungsaufgabe und zum anderen müssen diese auch in Kommandoform notiert werden, was umfangreiche Kenntnisse über das verwendete Package erfordert. Der fertige Graph wird erst nach dem Compilieren des  $\text{\LaTeX}$ -Dokuments sichtbar und kann somit nicht interaktiv bearbeitet werden. (Ein Beispiel für ein solches Package: [Vaucanson-G].)

Bildverarbeitungssoftware, wie etwa MS Visio, bietet die Möglichkeit, Graphen zu erstellen und in für Printmedien sinnvollen Auflösungen zu speichern. Der große Nachteil ist jedoch der hohe Zeitaufwand, da der Anwender bei der Anordnung und Ausrichtung kaum unterstützt wird. Weiterhin ist es nicht möglich, den gezeichneten Automaten zu simulieren oder zu transformieren. Eine Weiterverarbeitung in einem entsprechenden Tool ist ebenfalls unmöglich.

Tools, wie JFLAP, bieten eine Vielzahl von Transformations- und Simulationsmöglichkeiten, unterstützen den Anwender bei der Erstellung von Zustandsübergangsgraphen und erlauben somit ein schnelles Zeichnen und Testen von Automaten. Die Präsentation beschränkt sich aber auf den Bildschirm und es wird derzeit keine Möglichkeit geboten, einen erstellten Graphen in ein für Printmedien geeignetes Format zu exportieren. ([FLAT] bietet eine Liste von derzeit vorhandenen Tools.)

## 3 AutoEdit

### 3.1 Automatendefinition

Da Publikationswünsche sowohl die Printmedien als auch das Web betreffen, ist ein single-source-multiple-purpose-Ansatz angezeigt. Dies bedeutet, dass man einen Automaten in einer einzigen Datei (single source) vorhält, aus der nach Bedarf Dokumente unterschiedlichen Zielformats (multiple purpose) generiert werden. Modifikationen finden dann nur an einer Stelle, nämlich in der einen Quelle, statt, während die für Publikationen oder Verarbeitungen erforderlichen Transformationen (z.B. Überführung eines Automatentyps in einen anderen), mit eben diesem Quelldokument stattfinden. Von daher ist es nahe liegend, zur Repräsentation abstrakter Automaten, eine bestimmte XML-Sprache zu verwenden. Diese wird hier im Folgenden kurz als FADL (finite automaton definition language) bezeichnet. Wir haben FADL in AutoEdit mit einem XML-Schema definiert.

```
<AUTOMATON>
  <TYPE value="NEA" />
  <ALPHABET>
    <ITEM value="a" />
    <ITEM value="b" />
  </ALPHABET>
  <STATE name="Z1" finalstate="false">
    <TRANSITION target="Z4"><LABEL read="a" /></TRANSITION>
    <TRANSITION target="Z2"><LABEL read="EPSILON" /></TRANSITION>
  </STATE>
  <STATE name="Z2" finalstate="false">
    <TRANSITION target="Z5"><LABEL read="a" /></TRANSITION>
    <TRANSITION target="Z3"><LABEL read="b" /></TRANSITION>
  </STATE>
  <STATE name="Z3" finalstate="true" />
  <STATE name="Z4" finalstate="false">
    <TRANSITION target="Z2"><LABEL read="EPSILON" /></TRANSITION>
    <TRANSITION target="Z5"><LABEL read="b" /></TRANSITION>
  </STATE>
  <STATE name="Z5" finalstate="false">
    <TRANSITION target="Z6"><LABEL read="EPSILON" /></TRANSITION>
  </STATE>
  <STATE name="Z6" finalstate="false" />
  <INITIALSTATE value="Z1" />
</AUTOMATON>
```

Abbildung 1: FADL - XML Definition eines Automaten

In Abb.1 wird zunächst der Typ des Automaten festgelegt. Anschließend folgen das Eingabealphabet und die Zustände mit den entsprechenden Übergängen. Am Ende wird der Startzustand festgelegt. Je nach Typ des Automaten variiert der Umfang der XML Elemente.

Ein Automat kann attribuiert, d.h. mit Layoutdaten des Zustandsübergangsgraphen, oder als reine Definition gespeichert werden. Für Transformationen und als Quelle für andere Programme ist das Layout nicht relevant.

Besonderer Wert wurde auf die Definitionstreue gelegt. Im Gegensatz zu anderen Tools, wie JFLAP (welches vorherrschend darstellungsorientiert arbeitet), folgt ein in AutoEdit definierter Automat den formalen Beschreibungen aus der Automatentheorie.

Aus der FADL-Repräsentation eines Automaten können mit Hilfe von AutoEdit Bilder von Zustandsübergangsgraphen und Tabellen in Bitmapformaten (BMP, PNG, JPEG, GIF) wählbarer Auflösungen und in Vektorformaten (EPS, SVG) erzeugt werden. Darüber hinaus ist es möglich, für einen Automaten Schemequellcode<sup>1</sup> zu erzeugen, der in einem entsprechenden Interpreter ausgeführt werden kann. Transformationen je eines Automaten in einen anderen liefern wiederum entsprechende FADL-Repräsentationen.

Abb. 2 zeigt die Zustandsübergangstabelle für den in Abb. 1 dargestellten NEA:

$$M = (\{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6\}, \{a, b\}, \delta, Z_1, \{Z_3\})$$

	a	b	$\epsilon$
Z <sub>1</sub>	{ Z <sub>4</sub> }	{ }	{ Z <sub>2</sub> }
Z <sub>2</sub>	{ Z <sub>5</sub> }	{ Z <sub>3</sub> }	{ }
Z <sub>3</sub>	{ }	{ }	{ }
Z <sub>4</sub>	{ }	{ Z <sub>5</sub> }	{ Z <sub>2</sub> }
Z <sub>5</sub>	{ }	{ }	{ Z <sub>6</sub> }
Z <sub>6</sub>	{ }	{ }	{ }

Abbildung 2: Zustandsübergangstabelle

### 3.2 Zustandsübergangsgraphen

AutoEdit bietet eine Vielzahl von Manipulationsmöglichkeiten an einem Zustandsübergangsgraphen. Vom einfachen Ausrichten der Zustände bis hin zur Gestaltung von Linienstärken und Farben ist alles editierbar und erlaubt dem Lehrenden markante Stellen hervorzuheben.

Durch einfaches Drag and Drop lassen sich die graphischen Objekte für Zustände und Übergänge bewegen. Eine automatische Ausrichtung des Zustandsübergangsgraphen wird ebenfalls angeboten.

---

<sup>1</sup> Scheme ist eine Lisp-Sprache, die auch im FB Informatik an der Hochschule Zittau-Görlitz (FH) vor allem für didaktische Zwecke eingesetzt wird.

Abb. 3 zeigt den Zustandsübergangsgraph für den in Abb. 1 dargestellten NEA:

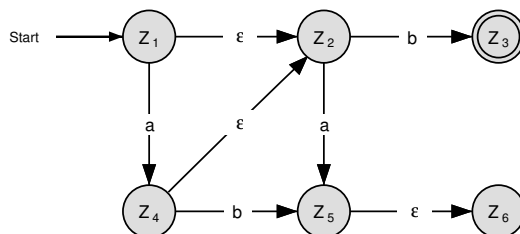


Abbildung 3: Zustandsübergangsgraph

### 3.3 Simulation und Transformation von Automaten

AutoEdit kann das Akzeptanzverhalten eines Automaten simulieren. Der Zustandsübergangsgraph wird dazu verwendet, eine Animation der Verarbeitung eines Eingabeworts darzustellen. Zusätzlich wird eine Tabelle angezeigt, aus der ersichtlich ist, welcher Zustandswechsel gerade vorgenommen wurde. Diese Tabelle stellt gleichzeitig eine Übersicht über alle möglichen Übergänge dar und zeigt, ob und auf welchem Weg das Eingabewort akzeptiert wurde.

Umwandlungen von nichtdeterministischen in deterministische Automaten, Minimierung oder Schaffung von Epsilonfreiheit werden angeboten, und können somit auch von Schülern zu Lern- und Kontrollzwecken eingesetzt werden.

## 4 Zusammenfassung

AutoEdit ist ein Werkzeug für die Hand von Lehrenden und Lernenden. Es unterstützt die Möglichkeit selbst- und aufgabengesteuerten Lernens und hilft Lehrenden beim Entwurf didaktischen Materials und dessen Publikation.

### Literatur

- |                    |  |
|--------------------|--|
| [JFLAP]            | <i>Java Formal Languages and Automata Package</i><br><a href="http://www.cs.duke.edu/~rodger/tools/tools.html">http://www.cs.duke.edu/~rodger/tools/tools.html</a><br>Susan H. Rodger, 2004                        |
| [FLAT]             | <i>FLAT-Werkzeuge</i><br><a href="http://www.inf.hs-zigr.de/~wagenkn/TI/Automaten/FLAT-Software/">http://www.inf.hs-zigr.de/~wagenkn/TI/Automaten/FLAT-Software/</a><br>Stand: 11.11.2003                          |
| [Vaucanson-G]      | <i>Vaucanson-G: A package for drawing automata and graphs</i><br><a href="http://www.liafa.jussieu.fr/~lombardy/Vaucanson-G/">http://www.liafa.jussieu.fr/~lombardy/Vaucanson-G/</a><br>Jacques de Vaucanson, 2003 |
| [Vorlesungsskript] | <i>Vorlesungsskript „Theorie der formalen Sprachen und Automatentheorie“</i><br>Prof. Dr. Christian Wagenknecht, 2001  |