Von ALGOL 60 zur symbolischen Simulation

Günter Riedewald

Fakultät für Informatik und Elektrotechnik, Institut für Informatik
Universität Rostock
Albert-Einstein-Str. 21
18051 Rostock
gri@informatik.uni-rostock.de

Abstract: Die Arbeitsgruppe „Programmiersprachen und Übersetzertechnik“ an der Universität Rostock existierte von 1966 bis Ende September 2007. Der Artikel zitiert die wichtigsten Forschungsvorhaben und Forschungsergebnisse aus dieser Zeit und geht auf das jeweilige Umfeld stärker ein.

1 Einleitung

Die Forschungsgruppe „Programmiersprachen“ wurde 1966 unter der Leitung von I. O. Kerner im Rahmen der Sektion Mathematik der Universität Rostock gegründet. Später wurde sie in Arbeitsgruppe „Programmiersprachen und Übersetzertechnik“ umbenannt und bestand unter diesem Namen bis zum Ende der Existenz des gleichnamigen Lehrstuhls am 31.9.2007. Zu den Pflichten der Arbeitsgruppe gehörten sowohl die Forschung als auch eine damit eng verknüpfte Lehre.

In der Forschungsarbeit waren die Hauptziele

- Beiträge zur Grundlagenforschung auf dem Gebiet der Programmiersprachen und der Sprachprozessoren zu liefern
- sowie durch studienbegleitende Forschung neueste Forschungsergebnisse anderen, insbesondere Studenten, so schnell wie möglich zugängig zu machen.

Die Realisierung des zweiten Ziels führte z.B. zur Einführung neuer Lehrveranstaltungen. So gehörte die Ausbildung von Mathematikern und später von Informatikern in Rostock zu den Studiengängen in Deutschland, die schon zeitig Vorlesungen wie Übersetzertechnik, Semantik von Programmiersprachen, Parallele Prozesse und Logische Programmierung einführten. Ebenso wurden Einführungslehrveranstaltungen zur Programmierungstechnik durch Gebiete wie Strukturierte Programmierung oder Abstrakte Datentypen frühzeitig modernisiert.

Da der Forschungsschwerpunkt in Rostock Computergraphik, früher Digitalgraphik genannt, war, ergaben sich für andere Arbeitsgruppen weniger Ressourcen. Das wurde unter anderem 1977 durch ein Schreiben von I. O. Kerner an die SED-Parteileitung der Universität zum Ausdruck gebracht. Leider hatte es kaum Wirkungen. Nicht desto trotz gingen aus der Arbeitsgruppe fünf Professoren hervor und hervorragend ausgebildete Assistenten und Studenten, die in der Regel später gute Arbeitsplätze fanden, leisteten wichtige Forschungsbeiträge.

Eine relativ gute Unterstützung gab es durch die Universität bei der Reisetätigkeit. Reisen ins westliche Ausland waren beschränkt auf die Reisen I. O. Kerners im Rahmen seiner Mitgliedschaft in der Working Group 2.1 (WG 2.1) der IFIP (**eine** Teilnahme bei zwei jährlichen Sitzungen) und auf wenige Reisen des Autors kurz vor der Wende zur Universität Linköping in Schweden und zum INRIA in Frankreich. Reisen ins östliche Ausland bzw. zu internationalen Konferenzen wie MFCS (Mathematical Foundations of Computer Science) und FCT (Foundations of Computational Theory), sofern sie im östlichen Ausland oder in der DDR stattfanden, konnten dagegen in größerer Anzahl realisiert werden. Das führte dazu, dass sehr gute Kontakte zur Universität Rīga, zur Tschechischen Technischen Hochschule (ČVUT) Prag und zu einer Informatik-arbeitsgruppe der Polnischen Akademie der Wissenschaften in Warschau, der z.B. die international renommierten Informatiker Blikle, Mazurkiewicz, Dembiński und Małuszyński angehörten, aufgebaut werden konnten. Diese Kontakte – abgesehen von der Mitgliedschaft I. O. Kerners in der WG 2.1 – stießen uns ein Fenster in westlicher Richtung auf, besonders, was den Zugang zu westlicher Literatur betraf, und ermöglichten damit überhaupt erst bestimmte Forschungen. Diese internationalen Kontakte waren aber nicht einseitig ausgerichtet. Z.B. gab es umfangreiche gemeinsame Diskussionen zum Thema attributierte Grammatiken. An der Universität Rīga wurden vom Autor Vorlesungen für höhere Semester über attributierte Grammatiken und die Semantik von Programmiersprachen gehalten. Unsere Erfahrungen gingen ebenfalls in die Mitarbeit in verschiedenen Arbeitsgruppen ein, wie z.B. WG 2.1 der IFIP oder RG 23 „Spezialsprachen und Methodik der Programmierung“ der mehrseitigen Kommission „Wissenschaftliche Probleme der Informatik“ der Akademien der sozialistischen Länder.

Mehrfach wurde uns von westlichen Kollegen Zusammenarbeit zu gemeinsam beforschten Gebieten angeboten, was leider nicht realisiert werden durfte. Z.B. arbeitete C. H. A. Koster aus Amsterdam an den Affixgrammatiken, wohingegen wir die Grammatiken syntaktischer Funktionen (GSF) aus der Taufe gehoben hatten. Kosters Angebot zur Zusammenarbeit musste abgelehnt werden, obwohl er sich zum damaligen Zeitpunkt mit Berlin, aber leider im falschen Teil der Stadt (Berlin West), recht nahe zu Rostock aufhielt Genauso durfte ein Angebot zur Zusammenarbeit auf dem Gebiet der syntaktischen Analyse und der Syntaxfehlererkennung nicht wahrgenommen werden. Nach der Wende wurden zwar noch Kontakte zur Universität Maribor und zum CWI Amsterdam aufgenommen, aber insgesamt ging die Reisetätigkeit wegen Mangels an Haushaltmitteln zurück.

Die Leitung der Arbeitsgruppe hatte, bis zu seiner Berufung am 1.9.1977 zum ordentlichen Professor an die Pädagogische Hochschule K.-F. Wander in Dresden, I. O. Kerner inne. Nach einer anschließenden kurzen Leitungsphase durch S. von Weber übernahm der Autor die Leitung. Die Ausrichtung der Forschung auf Programmiersprachen und deren Implementation blieb allerdings während der gesamten Zeit erhalten. Will man eine Klassifizierung der Forschungsgebiete der Arbeitsgruppe vornehmen, so könnte man folgende Hauptüberschriften wählen:

- Die Programmiersprache ALGOL60
- Die Programmiersprache ALGOL68
- Attributierte Grammatiken und Anwendungen
- Symbolische Simulation
- Programmtransformation.

Die Bearbeitung dieser Gebiete erfolgte ungefähr in dieser Reihenfolge, aber selbstverständlich mit Überlappungen. Da in der DDR die Forschung koordiniert wurde, waren die Rostocker Arbeiten in der Hauptforschungsrichtung „Mathematische Grundlagen der Informationsverarbeitung“ eingegliedert.

In den nachfolgenden Kapiteln wird zu den wichtigsten Gebieten ein Überblick gegeben. Eine Liste ausgewählter Veröffentlichungen kann vom Autor angefordert werden.

2 ALGOL 60, ALGOL 68

Der Beginn dieses Zeitraums war durch ALGOL 60 geprägt. Im Mittelpunkt standen das Studium und die Propagierung von ALGOL 60, seiner neuen Sprachkonstrukte und seiner neuen Methode der Syntaxdefinition (BNF), sowie seiner Compiler. Es entstanden Lehrbücher zur Einführung von ALGOL 60 (Kerner/Zielke 1969) und zur Einführung in die Informatik (Kerner 1970, 1973), welches vom Aufbau und den Prinzipien von Rechenanlagen, über die theoretische Basis von Programmiersprachen, Ideen der Implementation von Programmiersprachen bis hin zur Programmierung numerischer Verfahren in ALGOL 60 reichte. Erste praktische Erfahrungen mit der Implementation von Programmiersprachen konnten Mitarbeiter und Studenten an der Rechenanlage ZRA 1 sammeln.

Die Mitgliedschaft von I. O. Kerner in der Arbeitsgruppe WG 2.1 der IFIP ermöglichte unserer Arbeitsgruppe den schnellen Zugang zu neuestem Material über Definitionsmethoden von Programmiersprachen, insbesondere zur sich in Entwicklung befindlichen Programmiersprache ALGOL 68, die für viele heutige Programmiersprachen (prominentes Beispiel ist C) eine Vorreiterrolle spielte, sowie eine frühzeitige und intensive Beschäftigung mit dieser Sprache. In Seminaren - zunächst für Mitarbeiter und später ebenfalls für Studenten - wurde ein systematisches Studium der Sprache und ihrer neuen Konzepte betrieben. Eine enge Zusammenarbeit gab es dabei mit einer Arbeitsgruppe an der TU Dresden unter der Leitung von Prof. Stiller. Beide Gruppen repräsentierten in der DDR das geballte Wissen über ALGOL 68 und seine Implementation. Um dem deutschen Leser ALGOL 68 zugänglich zu machen, wurden durch

unsere Arbeitsgruppe eine deutsche Übersetzung des Sprachreports zu ALGOL 68 (Kerner 1972) und eine deutsche kommentierte Fassung des revidierten Sprachreports (Kerner 1978) erarbeitet.

Da es längere Zeit international keinen Compiler gab, wurde auch in der Forschungsgruppe über Möglichkeiten der Implementation von ALGOL 68 nachgedacht. Auf Grund der geringen Größe der Forschungsgruppe war eine vollständige Implementation ausgeschlossen. Stattdessen wurde ein Compiler für eine Untersprache von ALGOL 68 – ALGOL 60+8 – im Sprachumfang von ALGOL 60 für den Rechner R 300 im Rahmen von Qualifizierungsarbeiten (Dissertationen, Diplomarbeiten) realisiert (Kerner/Al-Sheik-Khalil 1974). Das Ergebnis war gleichzeitig ein Beitrag für die Untergruppe „Untersprachen von ALGOL 68“ der WG 2.1, die von I. O. Kerner geleitet wurde. Trotz des sehr positiven Echos kam der Compiler allerdings aus objektiven (Kompatibilitätsprobleme durch Hardware) und subjektiven (Bevorzugung von Pl/1 gegenüber ALGOL 68, Bevorzugung der IBM-Rechentechnik mit entsprechenden Softwarekonsequenzen) Gründen nur in beschränkten Einsatz.

Die Syntax und die statische Semantik von ALGOL 68 waren durch den neuen Formalismus der Zweistufengrammatiken (van Wijngaarden-Grammatik) beschrieben und stellten damit eine Herausforderung für die Forschung insbesondere zum so genannten Front-End von Compilern dar. In den Arbeitspapieren der WG 2.1 zu ALGOL 68 wurde ohne Beweis behauptet, dass ein entsprechender Parser für ALGOL 68 aus der formalen Definition ableitbar sei. Allerdings selbst in der WG 2.1 gab es keine konkreten Vorstellungen dazu. Unsere Arbeitsgruppe, die sich mit Methoden der Implementation von ALGOL 68-ähnlichen Sprachen beschäftigte, schrieb sich deshalb dieses Thema auf ihre Fahnen. Als Ergebnis wurde 1971 (Riedewald 1972, 1974, 1975) aus der Zweistufengrammatik eine neue Grammatikform – die Grammatik syntaktischer Funktionen (abgekürzt GSF) – als „ausführbare“ Grammatik (parametrisierte kontextfreie Grammatik) entwickelt. Zur Zweistufengrammatik von ALGOL 68 entstand auf der Basis intensiver Analysen ihrer Zweistufensyntax eine GSF für ALGOL 68, aus der ein Compiler-Front-End ableitbar war. Ende der 70-er Jahre stellte sich die enge Verwandtschaft der GSF sowohl zu attributierten Grammatiken als auch zu logischen Programmiersprachen – speziell Prolog – heraus. Als A. Colmerauer die Idee der GSF 1993 vorgetragen wurde, formulierte er das so: „Da haben Sie ja auch die logische Programmierung erfunden.“ Weiss bewies 1979, dass Grammatiken des Chomsky-Typs 0, Zweistufengrammatiken (van Wijngaarden), Verallgemeinerte attributierte Grammatiken (Małuszyński), Affixgrammatiken (Koster) und GSF äquivalent sind. Lässt man in der GSF die Terminale weg, bekommt man eine Grammatik, die eng mit den 1984 entwickelten relationalen attributierten Grammatiken (Deransart 1984, Courcelle/Deransart 1987) verwandt ist.

Für praktische Anwendungen musste für die GSF ein Algorithmus zur Berechnung der Parameterwerte (Attributwerte) entwickelt werden. Er entstand von 1973 bis 1974, wurde 1975 in einer Diplomarbeit getestet, aber erst 1977 in einem technischen Bericht (Riedewald 1977) veröffentlicht. Einige Jahre später stellte sich heraus, dass der Algorithmus eng verwandt mit der Argumentberechnung in Prolog, allerdings ohne Backtracking, ist.

1976 wurde ebenfalls in einem technischen Bericht (Riedewald 1976) ein Compilermodell auf der Basis der GSF vorgestellt, welches den Algorithmus verwendete, und durch eine Übersetzung aus der damals üblichen akademischen Beispielsprache ASPLE (Cleaveland/Uzgalis 1973) in FORTRAN 63 illustriert. Damit wurde nachgewiesen, dass die GSF als ein wirkungsvolles Beschreibungsmittel für Sprachprozessoren geeignet ist. In Zusammenhang mit Grammar Engineering taucht die GSF noch heute, z.B. bei R. Lämmel, in der Form der logischen Grammatik auf.

Mit der semantischen Synthese von ALGOL68-Compilern beschäftigte sich die Dissertation von Lorenzen (Lorenzen 1978). Vorgeschlagen wurde eine zweistufige Codeerzeugung, wobei zunächst einmal als Zwischenstufe Makrofolgen generiert werden. Es wird ebenfalls ein Modulkonzept vorgeschlagen, um Compiler (damals noch nicht modularisierter) umfangreicher Sprachen besser zu beherrschen.

3 Attributierte Grammatiken und Anwendungen

Aufbauend auf den praktischen Erfahrungen mit dem oben erwähnten Compilermodell wurde ein System zur automatischen Erzeugung von Compilern aus GSF-Spezifikationen – RÜGEN (Rostocker ÜbersetzerGENerator) – (Riedewald/U. Lämmel 1982) konzipiert und zu großen Teilen implementiert. Basis war der Algorithmus zur Berechnung der Parameterwerte einer GSF. Wegen des großen Umfangs wurde das Thema schließlich eingeschränkt auf die Entwicklung eines Systems zur Erzeugung von Compilern für Kommandosprachen – RUEGEN-KS (U. Lämmel 1984). Auch dieses System ging von GSF-Spezifikationen aus, aber es erleichterte dem Anwender den Entwurf der GSF. Der Anwender musste eine Art Signatur der Kommandosprache definieren, die vom System auf Vollständigkeit getestet und anschließend in Regeln einer GSF transformiert wurde. Voraussetzung hierfür war ein standardisiertes Format der Kommandos der Kommandosprachen. Zum damaligen Zeitpunkt war das der erste Vorschlag zur automatischen Erzeugung von Regeln einer attributierten Grammatik.

Grammatiken syntaktischer Funktionen sind - grob charakterisiert – parametrisierte kontextfreie Grammatiken und sehr eng verwandt mit den attributierten Grammatiken. Diese Verwandtschaft wurde uns während eines Forschungsaufenthalts an der ČVUT Prag bewusst, wo wir zum ersten Mal die Gelegenheit hatten, attributierte Grammatiken intensiv zu studieren. Attributierte Grammatiken unter besonderer Beachtung der GSF und ihre Anwendungen standen für längere Zeit im Mittelpunkt der Forschung der Arbeitsgruppe. Im Rahmen zweier durch uns organisierter Tagungen mit internationaler Beteiligung versuchten wir, die Forschungen zu attributierten Grammatiken und ihren Anwendungen besser zu propagieren, um sie einem breiteren Publikum zugängig zu machen. Zusammen mit einer von der Akademie der Wissenschaften der DDR organisierten Konferenz waren diese Tagungen Keimzellen der noch heute existierenden internationalen Tagung CC, wobei sich die Bedeutung dieser Abkürzung von Compiler Compiler zu Compiler Construction änderte. Diese Entwicklung wäre ohne die Unterstützung durch Kollegen aus dem westlichen Ausland, wie z.B. Deransart, Koskimies, Fritzson, Wilhelm, die Mitglieder in den Programmkomitees waren, nicht möglich gewesen.

Die äußere Ähnlichkeit attributierter Grammatiken und logischer Programme lässt den Gedanken einer inhaltlichen Verwandtschaft aufkommen. So wurde gezeigt (Deransart/ Małuszyński 1985), dass relationale attributierte Grammatiken und logische Programme unter bestimmten Voraussetzungen aufeinander überführbar sind. Der Vorteil dieser Verwandtschaft besteht darin, dass man Erkenntnisse des einen Gebiets auf das andere übertragen kann. Diese Idee bildete die Grundlage der von Deransart und Małuszyński aus der Taufe gehobenen Konferenz PLILP (Programming Languages Implementation and Logic Programming). Nachdem in Rostock ein damals noch auf der Embargoliste stehender Compiler für die Sprache micro-Prolog ohne Sprach- und Compilerbeschreibung aufgetaucht war, begannen wir uns sofort mit dieser Sprache intensiv zu beschäftigen, da wir natürlich auch von den Plänen der japanischen Informatiker bezüglich Prolog gehört hatten. Als Nebenprodukt entstand für die Lehre das System MICALG (Forbrig 1987), welches gestattete, algebraische Spezifikationen in micro-Prolog-Programme zu transformieren. Auf diese Weise wurde eine bessere Anschaulichkeit abstrakter Datentypen in der Lehre erreicht (später wurde eine Transformation in funktionale Programmiersprachen verwendet.).

Bei der Arbeit mit Prolog fiel die Ähnlichkeit – zunächst der Notation – zur GSF auf. Es wurde gezeigt, dass sich eine GSF nach gewissen Modifikationen auf ein Prolog-Programm überführen lässt (U. Lämmel 1987, Riedewald/U. Lämmel 1988). Dabei muss nicht einmal auf Terminale verzichtet werden, wie das bei Deransart und Małuszyński oder bei der Transformation von DCG (Pereira/Warren 1980) - eine Notation logischer Programme mit Terminalen, die als spezielle attributierte Grammatik betrachtet werden kann - in Prolog-Programme üblich ist. Unsere Transformation der GSF in Prolog-Programme behielt die grammatikalische Notation einschließlich Terminale weitgehend bei. Außerdem nutzten wir an Stelle von expliziten Parsern den Berechnungsprozess in Prolog für die (absteigende) syntaktische Analyse. Die Transformation von GSF in Prolog-Programme hat auch einen weiteren praktischen Vorteil, denn man erhält ausführbare Programme. Beschreibt man z.B. einen Compiler durch eine GSF, so stellt das entsprechende Prolog-Programm einen Prototyp-Compiler dar.

Die Beziehungen zwischen GSF und logischer Programmierung bestimmten für längere Zeit die Forschungsarbeit der Gruppe. Die schnelle Umsetzung von GSF-Spezifikationen in ausführbare Prolog-Programme legt den Gedanken nahe, dieses Vorgehen für die Entwicklung von Prototyp-Sprachen, Prototyp-Interpretern und allgemein Prototyp-Software zu nutzen (Riedewald 1991). Die Spezifikation von Software mit Hilfe attributierter Grammatiken wurde 1984 durch Rechenberg am so genannten Telegrammbeispiel demonstriert. Im gleichen Jahr wurde auch bei uns durch Forbrig (Forbrig 1984, Riedewald/Forbrig 1987) ein Vorschlag zur Softwarespezifikation mit attributierten Grammatiken, speziell GSF, gemacht. Dabei wird als Neuerung die Verknüpfung von datengesteuerter Softwareentwicklung mit abstrakten Datentypen unter Verwendung attributierter Grammatiken vorgeschlagen.

Natürlich wurde auch das ureigenste Forschungsthema der Arbeitsgruppe – die Entwicklung und Implementation von Programmiersprachen – nicht vergessen. An Stelle von RÜGEN, eines Übersetzer generierenden Systems im klassischen Sinn, tritt nun ein System – LDL (Language Development Laboratory) - zur Entwicklung von Programmiersprachen und zur Erzeugung von Sprachprozessoren, wobei die Spezifikationen durch GSF und die Implementationen durch Prolog erfolgen (Riedewald 1991, 1992, 1994, Harm/R. Lämmel/Riedewald 1997). Um die Entwicklung von Programmiersprachen zu unterstützen war eine Art Expertensystem vorgesehen, welches dem Anwender eine Datenbank mit Spezifikationsmustern für gängige Sprachkonstrukte einschließlich Anwendungsdemonstrationen bieten sollte. In einer Fallstudie im Rahmen einer Diplomarbeit wurden erste Spezifikationsmuster zusammen mit Operatoren zu ihrer Bearbeitung entwickelt. Insgesamt erwies sich das Problem aber als zu umfangreich für eine kleine Arbeitsgruppe.

In diesem Zusammenhang wurde auch die Korrektheit von Interpretern untersucht (Riedewald/R. Lämmel 1994). Ebenfalls zum Thema Korrektheit von Compilern/Interpretern – aber nicht unbedingt im Zusammenhang mit LDL - wurden Möglichkeiten zur automatischen Generierung von Testprogrammen, die sowohl syntaktisch richtig sind als auch die Kontextbedingungen einer Sprache erfüllen, untersucht (Harm 1997, Harm/R. Lämmel 2000). Ausgangspunkt waren durch den Autor durchgeführte Teste mit Prototypinterpretern in PROLOG, wobei die Prolog-Eigenschaft genutzt wurde, dass, je nach konkreter Eingabe, der gleiche Parameter einmal zur Eingabe und ein anderes Mal zur Ausgabe dienen kann. Beschreibt ein Prolog-Programm eine Sprache, kann das Programm unter bestimmten Bedingungen einmal zur Analyse und einmal zur Synthese verwendet werden.

Es gab auch eine Reihe signifikanter Anwendungen von LDL: Implementierung einer Spracherweiterung zur Beschreibung von Entwurfsmustern in Eiffel, Generierung von Nutzerschnittstellen, Implementierung einer domainspezifischen Sprache für symbolische Simulation, Entwicklung und Implementierung von Integritätsbedingungen einer Sprache für objektorientierte Datenbanken. Zusammenfassend lässt sich LDL als Kombination eines Expertensystems und einer Werkzeugbank zur Entwicklung von Programmiersprachen und ihrer Prozessoren betrachten. Damit war es einerseits ein schwergewichtiges System, aber andererseits fehlten einige wichtige Komponenten für einen breiteren Einsatz.

Die obigen Forschungsarbeiten erforderten eine intensive Auseinandersetzung mit der Theorie. So entstand Ende der 70-er Jahre ein algebraisches Modell von Programmiersprachen und deren Compiler (Riedewald 1981, 1985). Im Unterschied zu anderen algebraischen Modellen von Programmiersprachen wurden die Eigenschaften von Sprachkonstrukten nicht direkt sondern über syntaktische Operationen, die den kontextfreien Syntaxregeln zur Generierung der Konstrukte entsprechen, beschrieben. Nicht die Programmiersprache selbst, sondern ihre kontextfreie Syntax wird als Algebra dargestellt. Zum ersten Mal wurde hier auch die Definition von Kontextbedingungen von Programmiersprachen durch Termgleichungen dargestellt und an einem Beispiel demonstriert.

Ein weiteres Forschungsthema war das klassische Thema der Syntaxanalyse und der Behandlung syntaktischer Fehler (Forbrig 1979, 1980).

Zyklische attributierte Grammatiken wurden am Beispiel von einfachen Datenflussanalysen studiert, wobei Prolog als Implementierungssprache diente und Zyklen durch Iterationen aufgelöst wurden.

Das Problem von Syntax und Semantik spielt in der Informatik – nicht nur bei Programmiersprachen – eine wichtige Rolle. Es ging darum, dieses Problem in die Lehre zu integrieren, wobei sich Programmiersprachen aus praktischer und theoretischer Sicht als Demonstrationsobjekt anbieten. Neben einer Reihe von Forschungsartikeln gab es international erst 1981 zur Semantik von Programmiersprachen ein einziges Buch im englischen Sprachraum (Pagan 1981), welches dafür die Basis sein konnte. In dieser Situation boten Dembiński und Małuszyński die gemeinsame Herausgabe eines solchen Buches an, welches dann 1983 als erstes deutschsprachiges Buch zur Semantik (und Syntax) von Programmiersprachen erschien und – obwohl es kein Lehrbuch war – häufig in der Lehre genutzt wurde. Es erschien sowohl beim Akademie-Verlag Berlin als auch beim Oldenbourg Verlag München Wien.

Oben wurde bereits auf das System MICALG und seinen Einsatz in der Lehre hingewiesen. Ein weiteres Nebenprodukt der Forschung für die Lehre war das System FLR (Baum/Forbrig 1991), welches zur Textgenerierung aber auch zur Manipulation von Spezifikationen genutzt werden konnte. Die theoretische Basis war die durch eine reguläre Grammatik gesteuerte Ableitung von Wörtern mit Hilfe einer kontextfreien Grammatik.

4 Programmtransformationen, symbolische Simulation

Die in Kapitel 3 angeführten Forschungen reichen bereits über die Wendezeit hinweg und waren Fortsetzungen der zu Vorwendezeiten durchgeführten Forschungen. Die im Kapitel 4 beschriebenen Forschungen starteten zwar erst nach der Wende, sie bauten aber genauso auf den vorherigen Erfahrungen und Ergebnissen auf.

In den Jahren unmittelbar nach der Wende stagnierte die Forschung zunächst, da politische und fachliche Überprüfungen der Mitarbeiter, Umstrukturierungen an der Universität, Neuausschreibungen der verbliebenen Arbeitsplätze usw. das Leben aller Mitarbeiter stark belasteten. Nachdem dann 1992 eine Neueinstellung erfolgt war, begann allmählich der Universitätsalltag wieder Fuß zu fassen, auch wenn Strukturierungsprozess und ständige Diskussionen um knappe Haushaltmittel auch heute noch die Mitarbeiter verunsichern.

Wenden wir uns nun wieder der Forschung der Arbeitsgruppe zu. Sie behielt ihre hauptsächliche Orientierung bei, allerdings zogen neue Gebiete wie Programmtransformation und symbolische Simulation ein. Konzentrieren wir uns zuerst auf Programmtransformationen. Das System LDL zeigte bereits die Notwendigkeit der Beschäftigung mit Operationen der Metaprogrammierung. Es ging dabei um Operatoren zur Entwicklung und Verknüpfung von Fragmenten von Sprachspezifikationen. Dieses Thema wurde nun allgemeiner aufgegriffen, wobei zunächst Operationen auf Fragmenten beliebiger deklarativer Beschreibungen im Mittelpunkt standen (R. Lämmel/Riedewald 1995, 1997, R. Lämmel 1998, Lohmann 2000). Deklarative Beschreibungen schlossen hier unter anderem ebenfalls Grammatiken ein, wobei damit schon die Basis für Grammar Engineering und Language Engineering gelegt wurde. Diese Themen wurden später von R. Lämmel intensiv weiter geführt.

Die hier gewonnenen Erkenntnisse spielten eine wichtige Rolle bei Programmtransformationen allgemeiner Art (R. Lämmel/Riedewald 1998, 1999, R. Lämmel/Riedewald/Lohmann 1999, Lohmann/Riedewald 2003, Lohmann/Riedewald/ Stoy 2004). Z.B. wurde das Thema der Kommentarerhaltung bei Programmtransformationen sehr intensiv untersucht (Lohmann/Riedewald 2003). Da es nach wie vor nur als teilweise gelöst betrachtet werden kann, bleibt es ein aktuelles Forschungsthema.

Es wurde bereits erwähnt, dass LDL ein schwergewichtiges System zur Entwicklung von Sprachen und deren Prozessoren darstellt. In den letzten Jahren zeichnet sich ein Trend zur Bereitstellung leichtgewichtiger Systeme von Sprachprozessoren ab, die auch von Nichtspezialisten unter gewisser Anleitung verwendet werden können. Wir haben uns diesem Trend angeschlossen und das System Laptob (Language Processing Toolbox) entwickelt (R. Lämmel/Riedewald 2001). Dieses System ist ein Rahmenwerk mit Werkzeugunterstützung zur Manipulation von Sprachprozessoren in Form logischer Programme. Es gibt dabei typisierte logische Programme sowie logische Programme höherer Ordnung. Grammatiken können Bestandteile von Prozessoren sein. In der Werkzeugbank werden Operatoren (Metaprogramme) zur Adaption und Komposition von Prozessoren bereitgestellt. Durch Benutzung des Systems können seine Möglichkeiten auf der Basis der Wiederverwendung ausgebaut werden.

Durch die Zusammenarbeit mit der Universität Maribor (M. Mernik) profitierte nicht nur die Forschung an attributierten Grammatiken und deren Anwendung im Bereich der Sprachdefinitionen und Sprachimplementationen, sondern es begann eine intensive Beschäftigung mit aspektorientierter Programmierung. Als hervorragendes Ergebnis entstand ein Vorschlag zur aspektorientierten Programmierung in Prolog (Lohmann/Riedewald/Wachsmuth 2006, 2008). Es ist international der erste umfassende Vorschlag zur Aspektorientierung in Prolog überhaupt. Auf der Basis dieses Vorschlags gelang es, wohlbekannte Techniken der Prologprogrammierung zum ersten Mal formal und einheitlich zu beschreiben. Die Forschungen zur Aspektorientierung in Prolog sind erst am Anfang und daher fehlen zurzeit natürlich bequeme Vorgehensweisen für den Alltagsprogrammierer. So wäre es z. B. sinnvoll, dem Programmierer eine domainspezifische Sprache zur bequemen aspektorientierten Programmierung in Prolog zur Verfügung zu stellen.

Im Zusammenhang mit Untersuchungen zu attributierten Grammatiken ergab sich die Frage, inwieweit Sprachen wie SDL (Specification and Description Language) zur Beschreibung von Echtzeitsystemen mit Hilfe attributierter Grammatiken definierbar und implementierbar sind. Ausgangspunkt war, dass z.B. SDL-Programme Netze semantisch erweiterter endlicher Automaten definieren, wobei Instanzen von Automaten während der Arbeit mit den Netzen gelöscht oder generiert werden können. Die Kommunikation der Automaten geschieht durch Signale über Kanäle. Da Netze endlicher Automaten allgemein kontextfreie Sprachen akzeptieren, sollten attributierte Grammatiken zur Beschreibung ausreichen, wobei die semantische Erweiterung der Automaten über den Formalismus der Attribute und semantischen Regeln darstellbar sein müsste. Diese Idee erwies sich als tragfähiges Konzept, zumal durch den Zusammenhang zwischen attributierten Grammatiken und Prolog eine schnelle Prototypimplementierung möglich war.

Bei der Beschäftigung mit dieser Problematik stießen wir auf die zeitabhängigen Automaten (Alur/Dill 1994) und hybride Automaten/Systeme (Alur/Courcoubetis/Henzinger/Pei-Hsin-Ho). Diese beschrieben wir zunächst durch eine GSF, die dann in ein Prolog-Programm transformiert wurde. Wegen der in den Automaten allgegenwärtigen Bedingungen (bedingte Übergänge, Invarianten in Lokationen) und wegen der Verwendung der Automaten zur symbolischen Simulation erwies sich aber eine Beschreibung der Automaten durch eine logische Programmiersprache mit Nebenbedingungen – auch constraint-logische Programmiersprachen genannt - (CLP, Prolog III, Prolog IV) als bequemer (Urbina/Riedewald 1995, Riedewald 1995), wobei man natürlich nicht den Umweg über GSF oder andere attributierte Grammatiken gehen muss. Hierbei spielte unsere Gruppe international eine Vorreiterrolle.

Eine Beschreibung hybrider Systeme durch constraint-logische Programmiersprachen ist zwar leicht erlernbar, aber für den Nutzer nicht besonders bequem. So entstand die Idee, zwei Sprachen für eine einfachere Handhabung zu entwickeln: eine textuelle und eine graphische Sprache (analog zu SDL). Folgende Forderungen sollten von den Sprachen erfüllt werden:

- Modulare Sprachdefinition orientiert an den einzelnen Automaten eines Systems
- Wiederverwendbarkeit von Automatenbeschreibungen durch Einführung von Automatentypen und parametrisierten Automaten
- Möglichkeiten einer Automatenverfeinerung
- Nutzerfreundliche Darstellung von Anfragen
- Synchronisation über Signale

Als Ergebnis entstanden die Sprachen MODEL-HS (Tetzner/Riedewald 1998, 1999) und VYSMO (Tetzner/Brauch/Riedewald 2001), die obige Forderungen im Wesentlichen erfüllen. Beide Sprachen sind äquivalent und werden in constraint-logische Programme übersetzt (Tetzner/Riedewald 2003). Als umfangreiche Anwendung wurde die Modellierung von Studiengängen untersucht (Tetzner/Lantsman/Riedewald 2006). Unter der Voraussetzung, dass die zeitlichen Abhängigkeiten eines Studiengangs einschließlich der Bedingungen (z. B. Wiederholungsfristen von Prüfungen, absolvierte Lehrveranstaltungen vor einem Praktikum oder der Diplomarbeit,...) aus Prüfungs- und Studienordnung als hybride Automaten spezifiziert werden, kann man sich Vorschläge für komplette konkrete Studienabläufe oder Möglichkeiten der Fortsetzung eines Studiums in einer bestimmten Studiensituation (bestimmte Studienleistungen wurden zu einem gegebenen Zeitpunkt erbracht/nicht erbracht) durch symbolische Simulation generieren lassen. Da einige Teile des Gesamtsystems noch per Hand erzeugt werden, ist das System noch nicht fremdnutzbar.

5 Schlussbemerkungen

In den vorherigen Kapiteln wurde der Weg der Arbeitsgruppe „Programmiersprachen und Übersetzertechnik“ auf dem Gebiet der Forschung skizziert. Dieser Weg war zum einen mit Problemen durch das politische System der DDR und zum anderen mit lokalen Problemen gepflastert, die nur teilweise überwindbar waren. Sehen wir uns einmal einige davon an.

Literaturbereitstellung Hier war das größte Problem ein chronischer Mangel an Devisen zur Beschaffung westlicher Literatur. Teilweise ausgenommen davon waren Forschungsschwerpunkte.

Der Devisenmangel führte dazu, dass sowohl wichtige Zeitschriften als auch Monographien nicht gekauft werden konnten. International sehr erfolgreiche Monographien wurden manchmal allerdings von sowjetischen Verlagen in Russisch herausgegeben. Solche Bücher waren dann sogar zu relativ niedrigen Preisen erwerbbar, wenn man sich rechtzeitig darum bemühte. Durch den so entstandenen Mangel an westlicher Literatur waren die Forschung – und in gewissem Maße – auch die Lehre beeinträchtigt. Ein Ausweg aus dieser Situation war die Anforderung von Sonderdrucken bei den Autoren oder die Beschaffung von Kopien bei Auslandsreisen, wobei persönliche Kontakte eine hervorragende Rolle spielten. In den Jahren nach der Wende konnten die Bibliotheken ihre Bestände auf den neuesten Stand bringen, aber in den letzten Jahren begann erneut ein strenges Sparregime mit dem Streichen bei der Beschaffung von Zeitschriften.

Was die Lehrbuchbeschaffung betrifft, konnte man vor der Wende kaum westliche Literatur verwenden, falls sie nicht gerade in Russisch zur Verfügung stand, allerdings stand genug Literatur zu studentengemäßen Preisen ostdeutscher Autoren zur Verfügung.

Veröffentlichungen Das Hauptproblem vor der Wende ist, dass an Konferenzen im westlichen Ausland nur Reise- und Auslandskader (und auch das nur in bescheidenem Maße) teilnehmen konnten. Eine Veröffentlichung im westlichen Ausland über Proceedings war deshalb nur wenigen vorbehalten. Veröffentlichungen in westlichen Zeitschriften dagegen waren prinzipiell möglich, aber durch ein Genehmigungsverfahren, wo z.B. die Notwendigkeit der Veröffentlichung im westlichen Ausland begründet werden musste, erschwert. Auf diese Weise war eine Propagierung von Forschungsergebnissen in westlicher Richtung schwierig, so dass – mit Ausnahmen - weder die Forscher aus der DDR noch ihre Ergebnisse im Westen bekannt waren.

Ressourcen Das Problem der Bereitstellung von Ressourcen ist ein ständiges Problem der Hochschulen. Eine einfache Formel der Verteilung lautet: Ist dein Thema ein Schwerpunktthema, dann wirst du auch gefördert. Wegen der in der Summe knappen Mittel heißt das dann für andere Mangel an Ressourcen. Die Frage ist, wie die Schwerpunkte festgelegt werden. Vor der Wende wurden z.B. gemeinsame Projekte mit der Industrie gefördert. Heute ist es wichtig, besonders viele Drittmittel einzuwerben, wobei nicht immer nach dem Nutzen aus wissenschaftlicher Sicht, insbesondere der Grundlagenforschung, gefragt wird.

Persönliche Kontakte Auf die Bedeutung persönlicher Kontakte für die Forschung wurde bereits im Zusammenhang mit den Forschungen der Arbeitsgruppe eingegangen. Deshalb wollen wir an dieser Stelle darauf verzichten.

Trotz aller genannten Probleme kann die Forschungstätigkeit der Arbeitsgruppe insgesamt als erfolgreich sowohl vor als auch nach der Wende eingeschätzt werden. Aus der obigen Beschreibung der Forschung geht ebenfalls die Kontinuität von der Vor- zur Nachwendezeit hervor, was nicht heißt, dass immer wieder die gleichen Forschungsfragen bearbeitet wurden.