Lehrstuhl für Informatik II (Forschungsgruppe 2)

Thema: Programmier- und Dialogsprachen sowie Compiler

Leiter:

Prof. Dr. Hans Jürgen Schneider

Mitarbeiter:

Arius, Peter, Dipl.-Inf. (wiss. Angest., SFB)
Behnsen, Wolfgang (Programmierer)
Betz, Wolfgang, Dipl.-Inf. (wiss. Angest., SFB)
Fischer, Ingrid, Dipl.-Inf. (wiss. Hilfskr.) ab 01.12.93
Hasbargen, Thorsten, Dr.-Ing. (wiss. Hilfskr.)
Jacob, Christian, Dipl.-Inf. (wiss. Hilfskr.)
Keil, Martina, M.A. (wiss. Hilfskr.)
Lührs, Erni (Sekretärin - halbtags)
Minas, Mark, Dr.-Ing. (wiss. Angest.)beurlaubt ab 01.09.93
Schied, Georg, Dr.-Ing. (wiss. Angest., SFB) bis 31.01.93
Schörmal, Elfriede (Sekretärin - halbtags)
Schorr, Ruth, Dr.-Ing. (wiss. Angest.)
Uebler, Manfred (Programmierer)
Viehstaedt, Gerhard,Dipl.-Inf.(wiss. Hilfskr.)
Wilke, Peter, Dr.-Ing. (Akad. Rat)

Gastwissenschaftler:

Issa, Gamal M. Behiry, M.Sc. (Channel-Programm) bis 31. 08. 93

Lehrbeauftragte:

Feder-Andres, Christiane, Dr.-Ing., Fa. sd&m GmbH, München
Höfner, Gerd, Dipl.-Inf., Fa. Siemens AG, Erlangen
Messerer, Monika, Dr.-Ing., Fa. infoteam, Bubenreuth

Forschungsergebnisse

1 Maschinelle Analyse natürlicher Sprache (Hasbargen)

Für die Kommunikation zwischen Mensch und Maschine werden in vermehrtem Umfang natürlichsprachliche Schnittstellen eingesetzt. Dabei werden die verwendeten natürlichsprachlichen Grammatiken typischerweise einmal vom Experten erstellt und sind dann für den Benutzer fest. Dieser ist somit auf die vom Grammatikersteller vorgesehenen Sprachkonstrukte beschränkt; eine automatische Anpassung des Systems an Spracheigenheiten und Sprachkompetenz des Benutzers ist nicht möglich. Daher erscheint es wünschenswert, Systeme zu konstruieren, die Grammatiken zur Beschreibung natürlicher Sprachen lernen bzw. schon existierende Grammatiken nachträglich um neue Regeln erweitern können. Um dies zu ermöglichen, wurde ein Transformationsverfahren entwickelt und prototypisch implementiert, auf dem Methoden zum maschinellen Lernen natürlichsprachlicher Grammatiken sowie zum automatischen Erweitern solcher Grammatikenum neue Sprachkonstrukte aufbauen können.

Dazu wurden als Formalismus zur Darstellung des Zusammenhangs zwischen Syntax und Semantik natürlicher Sprache die sog. Klauselgrammatiken definiert. Eine solche Klauselgrammatik ist ein endliches Regelwerk, durch das der Zusammenhang zwischen einer Äußerung und der Äußerungssemantik auf deklarative Weise beschrieben werden kann. Dabei werden sprachliche Äußerungen durch Zeichenketten und die Äußerungssemantik durch gerichtete azyklische Graphen (sog. Merkmalsgraphen) dargestellt; die Auswertung der Regeln bei der Analyse bzw. Synthese sprachlicher Äußerungen findet (ähnlich wie in der Programmiersprache PROLOG) durch Unifikation und Tiefensuche statt. Zum Formalismus der Klauselgrammatiken wurde ein Parser entwickelt und implementiert, der anhand der Grammatikregeln einer Äußerung eine Menge von möglichen Bedeutungen zuordnen bzw. aus einem semantikdarstellenden Graphen eine ihm zugehörige sprachliche Äußerung synthetisieren kann.

Kernpunkt der durchgeführten Arbeiten war die Definition eines optimierenden Transformationsverfahrens auf diesen Klauselgrammatiken. Die Transformationen bilden dabei eine Klauselgrammatik auf eine neue Klauselgrammatik ab; Optimierung bedeutet hierbei, daß die entstehende Grammatik bezüglich eines Größenmaßes kleiner und daß sie genereller als das zugrundeliegende Urbild ist. Die Wirkungsweise der Transformationen beruht darauf, daß aus der Klauselgrammatik eine Menge einander \qähnlicher\q Regeln ausgewählt wird und diese einander ähnlichen Regeln durch ein gemeinsames \qMuster\q und die individuellen Abweichungen jeder Regel von diesem Muster ersetzt werden. Dabei führen verschiedene Arten von Ähnlichkeit zu verschiedenen Arten von Transformationen.

Mit Hilfe dieser Transformationen wurde ein Verfahren konstruiert, bei dem eine vorhandene Klauselgrammatik (die auch leer sein kann) anhand gegebener neuer Beispiele von Äußerungenund einer Beschreibung der Äußerungsbedeutungen maschinell erweitert wird; dabei werden inden Beispielen enthaltene generelle

Sprachprinzipien erkannt und die Klauselgrammatik wird um neue Regeln, die diese Prinzipien beschreiben, erweitert. Dies geschieht, indem jedes neue Beispiel der Form Äußerung/Äußerungsbedeutung als \qtriviale\q Regel zur Klauselgrammatik hinzugefügt und die so erweiterte Klauselgrammatik durch Transformationen optimiert wird. In umfangreichen Tests konnte gezeigt werden, daß durch Transformationen auf Klauselgrammatiken zumeist korrekte Generalisierungen vorgenommen werden; Übergeneralisierungen sprachlicher Regeln traten nur selten bei bewußt nicht-repräsentativ gewählten Lernbeispielen auf. Weiterhin wurde festgestellt, daß durch das Verfahren Klauselgrammatiken erzeugt werden, die von Menschen geschriebenen Grammatiken zur Beschreibung derselben Sprachkonstrukte recht nahekommen.

Literatur:

Hasbargen, T.:
"Ein Inferenzmechanismus zum beispielbasierten Lernen natürlicher Sprache", Dissertation an der Universität Erlangen-Nürnberg, 1993

2 Sonderforschungsbereich 182, Teilprojekt B1

(Arius/Betz/Schorr/Viehstaedt)

Der Lehrstuhl für Programmiersprachen ist mit dem Teilprojekt B1 "Programmierumgebung (für Multiprozessor- und Netzwerkkonfigurationen)" an dem seit 1986 in Erlangen bestehenden Sonderforschungsbereich 182 "Multiprozessor- und Netzwerkkonfigurationen" beteiligt. Wie der Name schon verdeutlicht, werden in diesem Teilprojekt Methoden und Werkzeuge zur Programmierung verteilter Rechnersysteme entwickelt.

Die Aufgabe einer Programmierumgebung besteht darin, Werkzeuge und Methoden für den Entwurf, die Implementierung und den Test von Programmsystemen zur Verfügung zu stellen. Den zentralen Punkt einer Programmierumgebung stellt die Programmiersprache dar. Für den Softwareentwicklungsprozeß sind noch weitere Komponenten von Bedeutung, beispielsweise Werkzeuge und Methoden zur Spezifikation von Anwendungsproblemen, für den Test von Programmsystemen, aber auch für die Generierung benutzerfreundlicher Entwurfshilfsmittel.

In den nun folgenden Abschnitten werden einige der Themengebiete vorgestellt, die zur Zeit Gegenstand der Forschung sind.

2.1 Die Programmiersprache HERAKLIT

Die Realisierung von Anwendungen für verteilte Systeme stellt besondere Anforderungen an die verwendete Programmiersprache, da Probleme wie die inhärente Parallelität solcher Anwendungen, deren meist hohe Komplexität und die nötige langfristige Wartung bewältigt werden müssen. Diese Gesichtspunkte prägten den Entwurf und die in den letzten Jahren verfolgte Weiterentwicklung der Programmiersprache HERAKLIT im Teilprojekt B1.

Mit der durchgehenden Verwendung des objektorientierten Programmiermodells unterstützt HERAKLIT die Strukturierung umfangreicher Softwaresysteme und bietet dem Programmierer ein hohes Abstraktionsniveau an. So ist z.B. die Verteilung der Objekte auf die Hardware transparent, können Prozesse als Objekte modelliert werden und stellen Objekte die interne Koordinierung ihrer nebenläufigen Aufträge bereit.

HERAKLIT erlaubt es, Objekte zur Laufzeit des Programms gegen typverträgliche neue Versionen auszutauschen, ohne dabei das gesamte Programm anhalten und anschließend neu starten zu müssen (Objektkonvertierung). Da ein auszutauschendes Objekt interne Prozesse besitzen kann, muß vorausgesetzt werden, daß diese vom Programmierer festgelegte Übergabepunkte besitzen, an denen ihre Aktivitäten gestoppt werden. Die neue Version übernimmt dann, nachdem der bisher erreichte Objektzustand übertragen und ggf. angepaßt worden ist, die Rolle seines Vorgängers. Vom Anmelden der Konvertierung bis zum Abschluß der Aktivierung der neuen Version werden alle Methodenaufrufe an das Objekt

zurückgestellt.

Die Sprache HERAKLIT erfüllt somit die Voraussetzungen, um ein komplexes verteiltes System schrittweise zu entwickeln, ohne dabei auf die Sicherheit der strengen Typprüfung verzichten zu müssen.

Die Arbeiten des letzten Jahres setzten die Entwicklung des Prototypen des HERAKLIT-Programmiersystems fort, der auf miteinander vernetzten UNIX-Workstations verteilt lauffähig ist.Im Rahmen einer Programmier-Intensivübung sowie von Studien- und Diplomarbeiten wurden so z.B. der HERAKLIT-Compiler um Klassenschablonen erweitert, eine Datenbasis implementiert, an die sich die im verteilten System arbeitenden Übersetzerprozesse anbinden, und begleitend zu der laufenden Forschung im Bereich des Debugging die Interaktionsmöglichkeiten des Anwenders mit dem HERAKLIT-Laufzeitsystem erweitert. Weiterhin wurden verschiedene aus der Literatur bekannte Verfahren zur Koordinierung nebenläufiger Prozesse daraufhin untersucht, wie sie sich mit der Objektkonvertierung von HERAKLIT kombinieren lassen, so daß die Koordinierungsbedingungen für die Objektkonvertierung verbessert werden konnten.

2.2 Generator für Diagrammeditoren

In Programmierumgebungen kommt der selektiven und leicht faßbaren Darstellung der vom Benutzer gewünschten Informationen eine besondere Bedeutung zu. Hierfür werden die gängigen Komponenten graphischer Benutzungsoberflächen wie z.B. Menüs, Eingabefelder usw. verwendet. Dies kann durch handelsübliche Werkzeuge unterstüzt werden. Für die Darstellung komplexer Zusammenhänge jedoch sind Diagramme notwendig. Die interaktive Manipulation der Diagramme ist ebenfalls wünschenswert. Eine angemessene Unterstützung durch Werkzeuge gibt es aber nicht.

Daher wird z. Z. ein Generator für Diagrammeditoren entworfen und in C++ implementiert, der es gestattet, einen speziellen Editor mit relativ geringem Aufwand zu erzeugen. Zur Spezifikation eines Editors für einen Diagrammtyp werden die Beschreibung der Struktur der Diagrammein Form einer Grammatik, das Aussehen und Layout der Diagrammelemente, die Interaktionsmöglichkeiten und Änderungsoperationen benötigt. Die theoretische Grundlage sind kontextfreie Hypergraphgrammatiken, mit denen graphische Zusammenhänge auf sehr natürliche Weise dargestellt werden können.

2.3 Ausnahmebehandlung

Die Ausnahmebehandlung ist das programmiersprachliche Konzept, um auf Ereignisse, deren Eintreten zur Laufzeit eines Programms erwartet wird, denen aber keine feste Position im Programmablauf zugeordnet werden kann, mit einer individuellen Bearbeitung zu reagieren.

Bei der Ausnahmebehandlung in verteilten Programmiersprachen ist zu beachten, daß aufgrund der Kommunikation zwischen Prozessen eine Ausbreitung von fehlerhaften Daten stattfinden kann. Diese Ausbreitung von fehlerhaften Daten hat

dann eine Ausbreitung von Ausnahmen zur Folge. Weiter besteht die Notwendigkeit, die Prozesse zu identifizieren und zu koordinieren, die an einer Ausnahmebearbeitung zu beteiligen sind. Aus diesem Grunde wurden Sprachmittel zur Ausnahmebehandlung erarbeitet, die der speziellen Situation in verteilten Systemen Rechnung tragen.

Unterstützt man die Erstellung zuverlässiger Programme mit Hilfe von Sprachmitteln zur Ausnahmebehandlung, so benötigt man eine Methode, um die Korrektheit von Programmen, die diese Sprachmittel verwenden, nachweisen zu können. Die Hauptprobleme bei der Verifikation von verteilten Programmen mit Ausnahmen sind die Einflußnahme der Ausnahmebehandlung auf den Kontrollfluß von Programmen und die Beteiligung aller notwendigen Prozesse an einer Ausnahmebearbeitung. Diese Probleme wurden gelöst, indem die klassische Verifikationsmethode von Hoare so erweitert wurde, daß anstelle einer Nachbedingung eine Standard- und mehrere markierte Ausnahmenachbedingungen verwendet werden.

2.4 Konfigurierung objektorientierter verteilter Systeme

Um die volle Leistungsfähigkeit eines verteilten Systems ausschöpfen zu können, müssen die auf der programmiersprachlichen Ebene definierten Einheiten - z.B. Objekte und Prozesse - in geeigneter Weise den Einheiten der Hardware zugeordnet werden, was im folgenden mit Systemkonfigurierung bezeichnet wird. Dabei sollten, unter Einhaltung physischer und logischer Restriktionen, Mittel wie Lastverteilung, Redundanz und dynamische Rekonfigurierung Anwendung finden, um Effizienz und Fehlertoleranz des Systems zu erhöhen.

Der Programmierer einer verteilten Anwendung benötigt angemessene Mittel, um die von ihm beabsichtigte Konfigurierung zu realisieren. Dabei verlangen bestehende Systeme oft, daß z.B. Anweisungen zur Migration von Objekten im Quelltext codiert werden, oder daß die Anwendung in einer bestimmten Weise modularisiert wird. Gerade in Verbindung mit objektorientierten Programmiersprachen können solche Konzepte jedoch nicht befriedigen, da sie den Programmierer in frühen Phasen des Entwurfs zu Entscheidungen zwingen, die nicht von der eigentlichen Problemstellung her motiviert sind, und so die Wiederverwendbarkeit von Softwarekomponenten stark einschränken.

Wir verfolgen daher den Ansatz, Programmierung und Konfigurierung zu entflechten, indem einer objektorientierten Sprache, die wie HERAKLIT von der Verteilung abstrahiert, ein eigenes Konfigurierungswerkzeug zur Seite gestellt wird. Graphen erscheinen dabei als ein geeignetes Mittel, die räumlichen Zusammenhänge im verteilten System darzustellen und Konfigurierungen zu visualisieren. Eine Versionsverwaltung soll es weiterhin erlauben, die verteilungstransparent programmierten Objektklassen unter verschiedenen Konfigurierungen wiederzuverwenden.

2.5 Fehlerermittlung

Der Debugger ist ein Werkzeug, das sowohl in der Implementierungs- als auch in der

Testphase der Softwareentwicklung zur Ermittlung einer Fehlerursache benötigt wird. Daher stellt er für die Erstellung fehlerfreier Software eine der wichtigsten Komponenten innerhalb einer Programmier- umgebung dar. Dies gilt vor allem, da trotz der Fortschritte in der Softwaretechnik die Fehler- beseitigung - insbesondere nach Änderungen in sehr großen Systemen - immer noch eine wichtige Rolle spielt.

Grundlage der eigentlichen Fehlerbeseitigung ist die Fehlererkennung; darunter versteht man:

(1) die Feststellung, daß ein Fehler, d.h. ein Abweichen vom spezifizierten Verhalten, vorliegt,

(2) die Lokalisierung der Stelle, an der sich dieses Abweichen erstmals manifestiert, und schließlich

(3) die Aufdeckung der Ursache für die Abweichung - die eigentliche Fehlerermittlung -, d.h. das Herausfinden der Programmanweisung, die die Weichen für das spätere Fehlverhalten

stellt.

Betrachtet man deterministische, sequentielle Algorithmen, so können die oben angesprochenen Fragestellungen als gelöst angesehen werden. Für verteilte Systeme, die im Rahmen dieses Projekts untersucht werden, müssen aufgrund der in solchen Systemen vorhandenen Phänomene wie Parallelität, Nichtdeterminismus oder räumliche Verteilung, allerdings für jeden einzelnen dieser Punkte, neue Lösungen gefunden werden.

Zur Zeit arbeiten wir an einem Konzept zur Fehlerermittlung in langlebigen, verteilten objektorientierten Systemen, indem wir versuchen, die Methode der \qFlowback-Analyse\q in den betrachteten Systemen anwendbar zu machen. Dabei können grundsätzlich drei Phasen unterschieden werden: In der ersten werden zunächst die einzelnen Objektklassen und -methoden einer statischen Analyse unterworfen und ein sog. \qstatischer Graph\q erzeugt. In der zweiten Phase, der \qAusführungsphase\q, wird Information über das Ausführungsverhalten des Programmsystems protokolliert. Wird ein Fehler festgestellt, so können in der dritten Phase, der \qDebuggingphase\q, ausgehend von der Stelle, an der sich der Fehler manifestiert, mit den gesammelten Informationen der beiden ersten Phasen der tatsächliche Kontrollfluß sowie die Abhängigkeiten der Variablen berechnet werden. Diese bilden den sog. \qdynamischen Graphen\q. Beachtenswert an unserem Konzept ist, daß es durch einfaches Überschreiben der in der Ausführungsphase gesammelten Information auch auf langlebige Systeme anwendbar ist. Dabei ist lediglich einzuschränken, daß Fehler, deren Ursachen zeitlich gesehen sehr weit zurückliegen, unter Umständen nicht vollständig analysiert werden können.

Für die Zukunft ist geplant, das oben kurz beschriebene Konzept weiter zu verfeinern und dessen Tragfähigkeit durch die Implementierung eines entsprechenden Debuggers für das Programmiersystem HERAKLIT zu überprüfen.

Gastaufenthalte

Prof. Dr. Ewa Grabska, Univ. Krakau (04. - 09. 10. 1993)
Grazyna Hliniak, Univ. Krakau (04. - 09. 10. 1993)

Dissertationen

Hasbargen, Thorsten:
Ein Inferenzmechanismus zum beispielbasierten Lernen natürlicher Sprache

Schorr, Ruth:
Ausnahmebehandlung in verteilten Programmiersprachen: Sprach- und Verifikationskonzepte

Diplomarbeiten

Berentz, Jürgen:
: Neuronales Netz zum Klassifizieren von Blechteilen

Böttcher, Dirk:
: Entwurf und Implementierung einer Codeerzeugungskomponente für ein Simulatorsystem für neuronale Netze (NeuroGraph)

Buschmann, Achim:
: Adaptive Fuzzy-Systeme - Gegenüberstellung und Vergleich verschiedener Verfahren

Feltens, Peter:
: Entwurf und Implementierung einer datenorientierten Programmierumgebung für SPS-Systeme

Friedmann, Andrea:
: Untersuchungen zur evolutionären Entwicklung konnektionistischer Systeme

Hilbig, Robert:
: Entwurf und Implementierung einer Klassenbibliothek für ein Simulatorsystem für künstliche neuronale Netze (NeuroGraph)

Hokamp, Gerald:
: Ein Simulationssystem zur evolutionären Entwicklung verteilter, lernfähiger Systeme

Käfferlein, Bernd:
: 3-dimensionale Anzeigefenster und Editoren für ein Simulatorsystem für künstliche neuronale Netze (NeuroGraph)

Kaiser, Anja:
: Entwurf und Implementierung einer parallelen Programmiersprache mit Ausnahmekonzept

Kuss, Oliver:
: Dialogmodellierung - Implementierung eines Codegenerators für das Zielsystem MS Windows und einer zielsystemunabhängigen Bausteinbibliothek

Richter, Anke:
: Entwurf und Implementierung einer Visualisierungskomponente für ein Simulatorsystem für neuronale Netze (NeuroGraph)

Vogel, Claudia:
: Bewertung der Programmiersprache HERAKLIT hinsichtlich Methoden der objektorientierten Softwareentwicklung

Studienarbeiten

Abhau, Dirk:
: Integration eines Ausnahmekonzepts in C++ - Optimierung der Einplanungen -

Agouros, Konstantin:
: Genetische Programmierung: Evolutionäre Ansätze zum Erlernen komplexer Bewegungsabläufe

Anke, Heike:
: Implementierung eines graphischen Editors für Merkmalstrukturen in C

Barsuhn, Bernd:
: Visualisierungs- und Analysemethoden für evolutionsbasierte Software-Systeme mit Mathematica

Bauer, Bodo:
: Entwurf und Implementierung eines Protokollierungsmoduls zu einem Simulatorsystem für künstliche neuronale Netze

Bauer, Ute:
: Effizienzsteigerung bei der Fehlererkennung mit Zeitconstraintnetzen

Beck, Rolf:
: Entwurf und Implementierung einer Steuerungskomponente für ein Simulatorsystem für künstliche neuronale Netze (NeuroGraph)

Billing, Gunnar:
: Entwurf und Implementierung einer Beschreibungskomponente für neuronale Netzstrategien für ein Simulatorsystem für künstliche neuronale Netze (NeuroGraph)

Burghof, Axel:
: Implementierung einer grafischen Bedienoberfläche zu GENESIS

Burkert, Axel:
: Korrektur und Aufbereitung des maschinenlesbaren Lexikons SADAW

Cherichi, Nadia:
: Biologische Mechanismen in künstlichen neuronalen Netzen am Beispiel adaptiver Bewegungskoordination: Modellentwicklung

Kriegel, Jörn:
: Graphgrammatiken und Euler-Operatoren

Landauer, Jürgen:
: Dialogmodellierung - Implementierung eines Codegenerators für das Zielsystem Open Windows

Langer, Jürgen:
: Lösung eines Rangierproblems mit Hilfe künstlicher neuronaler Netze

Nilson, Jörg:
: Implementierung von Funktionen und Strategien in NeuroGraph

Radziej, Michael:
: Erweiterung und Modifizierung eines regelgesteuerten Generators für Maschinencodeoptimierer

Rehder, Jan:
: Evolutionäre Bestimmung neuronaler Netztopologien und Lernverfahren durch hierarchische genetische Operationen

Schneider, Andreas:

Konzeption und Implementierung einer Klassenhierarchie graphischer Objekte

Scholz, Ralf:

Entwurf und Implementierung einer Parallelisierungskomponente für ein Simulatorsystem für künstliche neuronale Netze (NeuroGraph)

Scholz, Roland:

Konzeption und Implementierung eines Laufzeitsystems für Diagrammeditoren

Siemandel, Jörg:

Implementierung einer graphischen Bedienoberfläche zu \qNeuroGene\q: Eine evolutionäre Entwicklungsumgebung für konnektionistische Systeme

Stamminger, Marc:

Realisierung nebenläufiger Übersetzungsprozesse im HERAKLIT-System

Tlili, Naouel:

Biologische Mechanismen in künstlichen neuronalen Netzen am Beispiel adaptiver Bewegungskoordination: Implementierung

Utz, Wolfgang:

Realisierung von Typschablonen für HERAKLIT

Walthelm, Axel M.:

Untersuchung zum Einsatz neuronaler Netze in der zerstörungsfreien Werkstoffprüfung

Vorträge

Jacob, C.:
"Evolution of neural net architectures by a hierarchical grammar-based genetic system", ANNGA\q93, International Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck, 14. 04. 1993

Jacob, C.:
"NeXTGene: A graphical user-interface for GENESIS under NeXTStep", ANNGA\q93, International Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck, 16. 04. 1993

Jacob, C.:
"\qWenn Darwin programmieren würde ...\q, Evolutionäre Entwicklung neuronaler Netze", Tag der Erlanger Informatik, 30. 04. 1993

Jacob, C.:
"\qWenn Darwin programmieren würde ...\q, Evolutionäre Entwicklung neuronaler Netze", VDI/VDE-Forum, Erlangen, 14. 10. 1993

Keil, M.:
"\qDa geht dem Rechner der Hut hoch\q - Probleme maschineller Verarbeitung natürlicher Sprache - " Tag der Erlangen Informatik, 30. 04. 1993

Minas, M.:
"Modelling and monitoring real-time systems with temporal constraint nets", 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System (IEA/AIE-93), Edinburgh, Schottland, 04. 06. 1993

Schneider, H. J.:
"Object-oriented programming and the maintenance of distributed processes", Al-Azhar Universität Kairo, 08. 04. 1993

Schneider, H. J.:
"Neue Ansätze in der Software-Entwicklung", SYSTEMS-Seminar, München, 20. 10. 1993

Schneider, H. J.:
"Einführung in die kategorielle Theorie der Graphgrammatiken", TU Ilmenau, 03. 12. 1993

Schneider, H. J.:
"Aussagen über Ableitungssequenzen in Graphgrammatiken", TU Ilmenau, 10. 12. 1993

Schneider, H. J.:
"Objektorientierte Programmierung und die Pflege verteilter Prozeßsysteme", Univ. Ulm, 15. 12. 1993

Schorr, R.:
"Ein verifizierbares Ausnahmekonzept für verteilte Programmiersprachen", Mathematik/Informatik-Kolloquium der Univ. Osnabrück, 06. 07. 1993

Schorr, R.:
"A Verifiable Exception Handling Model for Parallel Programming", Second Intern. Conference on Parallel Computing Technologies, Obninsk, 03. 09. 1993

Schorr, R.:
"Ausnahmebehandlung in verteilten Realzeitprogrammiersprachen",

PEARL\q93 Workshop über Realzeitsysteme, Boppard, 02. 12. 1993

Viehstaedt, G.:
"Specification of diagram editors providing layout adjustment with minimal change", IEEE Symposium on Visual Languages, Bergen/Norwegen, 27. 08. 1993

Viehstaedt, G.:
"Spezifikation von Diagrammeditoren mit automatischer Layoutanpassung", 23. GI-Jahrestagung, Dresden, 01. 10. 1993

Wilke, P.:
"The NeuroGraph Neural Network Simulator", Int. Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS\q93, San Diego, CA, USA, 19. 01. 1993

Wilke, P.:
"Simulation und Visualisierung künstlicher neuronaler Netze unter Verwendung paralleler Hardware mit NeuroGraph", Economic Parallel Processing epp\q93, Wien, 19. 02. 1993

Wilke, P.:
"NeuroGraph - Ein Simulator für künstliche neuronale Netze auf paralleler Hardware", 3. PASA Workshop, Bonn, 02. 04. 1993

Wilke, P.:
"NeuroGraph - A Neural Network Simulation Environment", Workshop on Software and Programming Issues for Connectionist Supercomputers, Berkeley, CA, USA, 19. 04. 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", WIRN\q93 6th Italian Workshop on Artificial Neural Networks, Salerno, Italien, 14. 05. 1993

Wilke, P.:
"Simulation und Visualisierung neuronaler Netze", Nachrichtentechnisches Kolloquium, TU Braunschweig, 18. 05. 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", IWANN\q93 Int. Workshop on Artificial Neural Networks, Sitges, Barcelona, Spanien, 11. 06. 1993

Wilke, P.:
"Simulation of Neural Networks and Genetic Algorithms in a Distributed Computing Environment using NeuroGraph", World Congress on Neural Networks, Portland, OR, USA, 13. 07. 1993

Wilke, P.:
"Visualization of Neural Networks using NeuroGraph", IFIP Working Group 3.2 (Computers in University Education) Working Conference, UC California, Irvine, CA, USA, 29. 07. 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", Computation and Neural Systems\q93, Washington, DC, USA, 02. 08. 1993

Wilke, P.:
"NeuroGraph - Ein Simulator für künstliche neuronale Netze auf paralleler

Hardware", 8. ASIM Symposium Simulationstechnik, Berlin, 29. 09. 1993

Wilke, P.:
"Konzeption für einen erweiterbaren Simulator für neuronale Netze", Sitzung Unterausschuß 4.8.2 der GMA im VDE/VDI, Erlangen, 14. 10. 1993

Wilke, P.:
"Simulation und Visualisierung künstlicher neuronaler Netze unter Verwendung paralleler Hardware mit NeuroGraph", Euro-ARCH Architektur von Rechensystemen, München, 19. 10. 1993

Wilke, P.:
"Des Kaisers neue Kleider - oder - Was kostet der Umstieg auf Motif und C++?", Informationstag Fa. Astrum, Erlangen, 02. 11. 1993

Wilke, P.:
"NeuroGraph - Ein Simulator für künstliche neuronale Netze auf paralleler Hardware", 2. Neuro-Mini-Messe FORWISS, Erlangen, 24. 11. 1993

Veröffentlichungen

Geiger, H. / Greska, W. / Wilke, P. / Berentz, J.:

Erkennung von Blechteilen mit Neuronalen Netzen", Zeitschrift für wirtschaftl. Fertigung und Automatisierung, ZWF/CIM 11/93, S. 526 - 528, 1993

Hasbargen, Th.:

"Ein Inferenzmechanismus zum beispielbasierten Lernen natürlicher Sprache", Dissertation, Arbeitsber. des IMMD der Univ. Erlangen-Nürnberg, Band 26, Nr. 16 (1993)

Jacob, C. / Burghof, A.:

"NeXTGene: A graphical user-interface for GENESIS under NeXTSTep", ANNGA\q93, Proc. of the International Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck, S. 602 - 606, 1993

Jacob, C. / Rehder, J.:

"Evolution of neural net architectures by a hierarchical grammar-based genetic system", ANNGA\q93, Proc. of the International Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck, S. 72 - 79, 1993

Minas, M.:

"Modelling and monitoring real-time systems with temporal constraint nets", Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System (IEA/AIE-93), Edinburgh, Schottland, Juni 1993

Minas, M.:

"Spezifikation von Diagrammeditoren mit automatischer Layoutanpassung", Proceedings der 23. GI-Jahrestagung, Informatik aktuell, S. 334 - 339, 1993

Minas, M. / Viehstaedt, G.:

"

Specification of Diagram Editors Providing Layout Adjustment with Minimal Change", Proceedings of the \q1993 IEEE Symposium on Visual Languages\q, IEEE Computer Society Press, Los Alamitos, CA, USA, S. 324 - 329, 1993

Schorr, R.:

"Ausnahmebehandlung in verteilten Programmiersprachen: Sprach- und Verifikationskonzepte", Dissertation, Arbeitsber. des IMMD der Univ. Erlangen-Nürnberg, Band 26, Nr. 8, (1993)

Schorr, R.:

"A Verifiable Exception Handling Model for Parallel Programming", in: V. Malyshkin (Hrsg.) Proc. of the Second Intern. Conference on Parallel Computing Technologies (PaCT93), Vol. II, S. 345 - 358, 1993

Schorr, R.:

"Ausnahmebehandlung in verteilten Realzeitprogrammiersprachen", in: P. Holleczek (Hrsg.) Proc. of PEARL\q93, Workshop über Realzeitsysteme, Informatik aktuell, S. 33 - 42, 1993

Wilke, P.:

"The NeuroGraph Neural Network Simulator", Proc. Int. Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS\q93, Simulation Series Vol. 25,1, San Diego, CA, USA, S. 341 -

342, 1993

Wilke, P.:
"Simulating Neural Networks in a Distributed Computing Environment Using NeuroGraph", Proc. Int. Workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS\q93, Simulation Series Vol. 25,1, San Diego, CA, USA, S. 382 - 383, 1993

Wilke, P.:
"NeuroGraph - A Neural Network Simulation Environment", Technical Report, International Computer Science Institute ICSI, Berkeley, CA, USA, 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", Proc. IWANN\q93 Int. Workshop on Artificial Neural Networks, Sitges, Barcelona, Spanien, 1993

Wilke, P.:
"Visualization of Neural Networks using NeuroGraph", Proc. IFIP Working Group 3.2 Working Conference on Visualization in Scientific Computing: Uses in University Education, UC California, Irvine, CA, USA, Edited by S.D. Franklin, 1993

Wilke, P.:
"Simulation of Neural Networks and Genetic Algorithms in a Distributed Computing Environment using NeuroGraph", Proc. ICANN\q93 Int. Conference on Artificial Neural Networks, Amsterdam, Niederlande, Springer Verlag, 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", Proc. WIRN\q93 6th Italian Workshop on Artificial Neural Networks, Salerno, Italien, 1993

Wilke, P.:
"Simulation of Neural Networks and Genetic Algorithms in a Distributed Computing Environment using NeuroGraph", Proc. World Congress on Neural Networks, Portland, OR, USA, 1993

Wilke, P.:
"Simulation of Neural Networks in a Distributed Computing Environment using NeuroGraph", Proc. Computation and Neural Systems\q93, Washington, DC, USA, 1993

Wilke, P.:
"NeuroGraph - Ein Simulator für künstliche neuronale Netze auf paralleler Hardware", 8. ASIM Symposium Simulationstechnik, Berlin, 1993