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 (Progr.)
Betz, Wolfgang, Dipl.-Inf. (wiss. Angest., SFB)
Fritzke, Bernd, Dr.-Ing. (wiss. Angest.) bis 30.09.92
Hasbargen, Thorsten, Dipl.-Inf.(wiss. Hilfskr.)
Jacob, Christian, Dipl.-Inf. (wiss. Hilfskr.)
Keil, Martina, M.A. (wiss. Hilfskr.)
Kips, Detlef, Dr.-Ing. (wiss. Ass. a.Z.) z.Z. beurlaubt
Lührs, Erni (Sekr. - halbtags)
Minas, Mark, Dr.-Ing. (wiss. Angest.)
Schied, Georg, Dr.-Ing. (wiss. Angest., SFB)
Schörmal, Elfriede (Sekr. - halbtags)
Schorr, Ruth, Dipl.-Inf. (wiss. Angest.)
Uebler, Manfred (Progr.) ab 01.02.92
Viehstaedt, Gerhard, Dipl.-Inf.(wiss. Hilfskr.)
Wilke, Peter, Dr.-Ing. (Akad. Rat z.A.)

Gastwissenschaftler:

Issa, Gamal M. Behiry, M.Sc. (Channel-Programm)
Zanzinger, Michael, Dr.-Ing. (BMFT) bis 30.09.92

Lehrbeauftragte:

Feder-Andres, Christiane, Dr.-Ing. Fa. sd&m GmbH, München

Am 12. 06. 1992 konnte der Lehrstuhl seine 20-Jahrfeier festlich begehen. Am Vormittag fanden zwei Hauptvorträge statt und am Nachmittag stellte der Lehrstuhl die z. Z. laufenden Forschungsprojekte in mehreren Vorträgen und Kurzvorführungen dar.

Forschungsergebnisse:

1 Graphgrammatiken (Schneider)

Graphen zur Darstellung von statischen Strukturen und dynamischen Vorgängen haben ihren festen Platz in der theoretischen und angewandten Informatik, aber auch in anderen Wissenschaften, gefunden. Während die klassische Graphentheorie sich mit der Analyse statischer Eigenschaften von Graphen befaßt, führen typische Fragestellungen in der Informatik dazu, sich mit der Charakterisierung von Graphklassen und mit der Beschreibung dynamischer Veränderungen zu befassen. In den sechziger Jahren wurden für diese Zwecke an verschiedenen Stellen Graphsprachen entwikkelt. Dieser Umweg, graphische Zusammenhänge durch lineare Zeichenfolgen zu codieren, erschien uns nicht sehr sinnvoll. Wir entwickelten daher den Begriff der Graphgrammatik als ein System von Regeln, die die Gewinnung neuer Graphen aus bereits bekannten beschreiben. Die Beschreibbarkeit dynamischer Veränderungen ist damit unmittelbar einsichtig. Graphklassen können als die Menge aller Graphen definiert werden, die sich mit gegebenen Regeln aus einem bestimmten Graphen gewinnen lassen. Diese Ideen, die von uns initiiert wurden, haben weltweite Resonanz gefunden; an verschiedenen Stellen wurden Weiterentwicklungen vorgenommen.

Gegenwärtig wenden wir die Grundlagenergebnisse auf die Beschreibung von Vorgängen in hierarchisch strukturierten und in asynchronen und verteilten Systemen an. Hierzu war es erforderlich, Graphen zu betrachten, deren Knoten ihrerseits wieder mit Graphen markiert sind. Es zeigt sich, daß man eine Reihe praktisch relevanter Fragestellungen unmittelbar auf graphgrammatikalisch formulierbare zurückführen kann, die sehr allgemein lösbar sind und somit nicht mehr für jeden Einzelfall separat untersucht werden müssen. Beispiele hierfür sind etwa die gegenseitige Unabhängigkeit von Operationen in einem asynchronen System, die Einführung neuer Abstraktionsebenen durch Zusammenfassung von Schrittfolgen zu einem Schritt oder die Bestimmung des minimalen Umfeldes, das von einer Schrittfolge tangiert wird.

2 Ausnahmebehandlung in verteilten Programmiersprachen: Sprach- und Verifikationskonzepte (Schorr)

Bei der Programmierung von Prozeßrechnersystemen haben Ereignisse schon immer eine Rolle gespielt, die zur Laufzeit eines Programms erwartet werden, denen aber keine feste Position im Programmablauf zugeordnet werden kann. Beispiele hierfür sind der Überdruck in einem zu überwachenden Kessel, das Überschreiten der Maximaltemperatur bei einem zu steuernden chemischen Prozeß oder das Auftreten von Turbulenzen während der Flugzeugführung durch einen Autopiloten.

Inzwischen ist der Nutzen der oben beschriebenen Ereignisse in allen Bereichen der Program-

mierung allgemein anerkannt. Das programmiersprachliche Konzept, das es ermöglicht, auf das

Eintreten derartiger Ereignisse mit einer individuellen Bearbeitung zu reagieren, ist die Ausnahmebehandlung. Eine Ausnahme ist ein Ereignis, das zur Laufzeit eines Programms eintreten kann. Zu jeder Ausnahme läßt sich eine Ausnahmebedingung angeben, unter der das Ausnahmeereignis eintritt. Sobald eine Ausnahme eintritt, wird ein dafür zuständiges Programmstück ausgeführt.

Aufgrund der Fortschritte der Mikroelektronik und der Verfügbarkeit von schnellen Kanälen zur Kommunikation haben verteilte Systeme in den letzten Jahren zunehmend an Bedeutung gewonnen. Der Einsatz von verteilten Systemen zur Bearbeitung von Anwendungsaufgaben setzt voraus, daß geeignete Programmiersprachen vorhanden sind. Neben der effizienten Nutzung der zugrundeliegenden Hardware, müssen diese Sprachen auch die strukturierte Erstellung von zuverlässigen Programmen ermöglichen, d. h. unter anderem müssen auch Mechanismen zur Ausnahmebehandlung vorhanden sein.

Bisher gibt es zur Ausnahmebehandlung in verteilten Programmiersprachen nur sehr wenige Arbeiten. Dabei werden die Sprachmittel aus dem sequentiellen Umfeld ohne Rücksicht auf die spezielle Eigenschaft verteilter Systeme, d.h. die parallele Ausführung mehrerer Prozesse, die miteinander kommunizieren können, übernommen. Tritt in einem Prozeß eine Ausnahme ein, so wird diese prozeßlokal mit einem Konzept für sequentielle Sprachen behandelt. In [Sch93] wird der besonderen Situation verteilter Systeme durch die Verwendung der folgenden Sprachmittel Rechnung getragen:

Kommunikationsabgeschlossene Einheiten:

Problematisch ist für eine Ausnahmesituation, bei der die zugehörige Bedingung auf den Informationsaustausch zwischen Prozessen Bezug nimmt, die Identifikation der Prozesse, die an der Ausnahmebehandlung zu beteiligen sind. Zur Lösung wird das Strukturierungsmittel der kommunikationsabgeschlossenen Einheiten eingeführt, das eine Interpretation der Menge der zu beteiligenden Prozesse enthält.

Auflösungsfunktionen:

Treten in einem verteilten Programm gleichzeitig mehrere verschiedene Ausnahmen ein, so kann folgendes Dilemma entstehen: Ein Prozeß kann von mehr als einer Ausnahme betroffen sein. Die Auflösungsfunktion ist eine Vorschrift, die in derartigen Fällen eine übergeordnete Ausnahme bestimmt, die anstelle der eingetretenen bearbeitet wird. Auch hier erweisen sich die kommunikationsabgeschlossenen Einheiten als adäquates Hilfsmittel.

Die Aufgabe der Ausnahmebehandlung ist es, die Zuverlässigkeit und Strukturierung von Programmen zu unterstützen. Um diese Ziele zu erreichen, sind spezielle Sprachkonstrukte notwendig. Weiterhin muß die Korrektheit der Programme, die diese Konstrukte verwenden, nachweisbar sein. Betrachtungen zur Verifikation bezüglich Ausnahmemechanismen in verteilten Programmiersprachen fehlen bisher gänzlich. Selbst die Verifikationsproblematik in sequentiellen Programmiersprachen mit Ausnahmekonzept ist in bisherigen Arbeiten nur rudimentär oder mangelhaft gelöst worden.

Um diese Mängel zu beseitigen, wird in [Sch93] ein Verifikationskonzept präsentiert, das auf der axiomatischen Verifikationsmethode von Hoare [Hoa69] aufbaut. Der zentrale Unterschied zu Hoare ist hierbei, anstelle der Nachbedingung eine Standard- und mehrere markierte Ausnahmenachbedingungen einzuführen. Mit dem Konzept werden alle Sprachmittel der Ausnahmebehandlung beschrieben. Insbesondere zeigt sich, daß die herausragende Eigenschaft der Ausnahmebehandlung, die Einflußnahme auf den Kontrollfluß, prägnant formuliert werden kann. Pragmatischen Gründen folgend wurden die allgemeinen Ansätze zur Verifikation verteilter Programme im Hinblick auf eine Integration der Ausnahmebehandlung untersucht und der optimale ausgewählt. Mit Hilfe des ausgesuchten Ansatzes [Zwi89] werden alle signifikanten Elemente der Ausnahmebehandlung in verteilten Programmiersprachen beschrieben. Hierzu ist wie im Sequentiellen eine Separierung der Nachbedingungen erforderlich. Als Schwierigkeit ergibt sich, alle notwendigen Prozesse an einer Ausnahmebearbeitung zu beteiligen. Auch hier erweist sich die Idee der Ausnahmenachbedingungen - wenn auch mit einer anderen Bedeutung als im Sequentiellen - als sehr erfolgreich.

Literatur:

[Hoa69]
Hoare, C. A. R.: "An Axiomatic Basis for Computer Programming"; Communications of the ACM; 12(10), p. 576 - 586; 1969

[Sch93]
Schorr, R.: "Ausnahmebehandlung in verteilten Programmiersprachen: Sprach- und Verifikationskonzepte"; Dissertation an der Universität Erlangen-Nürnberg; erscheint 1993

[Zwi89]
Zwiers, J.: "Compositionality, Concurrency and Partial Correctness - Proof Theory for Networks of Processes, and Their Relationship"; Lecture Notes in Computer Science 321, 1989

3 ROSE - eine Programmierumgebung für rechnerorganisierte Software-Entwicklung (Zanzinger)

Im Rahmen einer Forschungskooperation mit der Fa. BASYS GmbH wurde innerhalb von drei

Jahren eine Programmierumgebung mit der Bezeichnung ROSE (rechnerorganisierte Software-Entwicklung) konzipiert und implementiert.

Mit ROSE steht dem Progammierer eine Entwicklungsumgebung zur Verfügung, die sich durch die Kombination vieler Merkmale auszeichnet:

Einheitliche Bedienung

Dies wurde durch die Einführung von sog. Datenklassen erreicht, mit denen eine Strukturierung aller im System vorkommenden Daten möglich ist. Ausgehend von einer Menge frei definierter und einiger systemdefinierter Datenklassen ist eine einheitliche Bedienung über das gesamte System gewährleistet. So unterscheiden sich z. B. die Bedienaktionen zum Löschen einer Datei nicht von denen zum Beenden eines Prozesses.

Automatisierte Verwaltungsfunktionen

ROSE stellt Funktionen zur Verfügung, die die eigentliche Entwicklungsarbeit effizient unterstützen. Hierzu zählen u. a. die Prozeßverwaltung, die Benutzerverwaltung, der Log-Dienst und die Versionskontrolle.

Sprachunabhängige, mächtige Entwicklungswerkzeuge

Durch das Prinzip der generischen Werkzeuge konnte gezeigt werden, wie eine Reihe von Werkzeugen, die typischerweise nur für bestimmte Programmiersprachen existieren, als sprachunabhängige Dienste in ROSE integriert wurden. Der Zeitaufwand für die Anpassung an eine neue Programmiersprache liegt bei diesen Programmen weit unter der benötigten Zeit für eine Einzelimplementierung. Die verwendeten Sprachbeschreibungen sind auf hohem Abstraktionsgrad, so daß mit wenigen Anweisungen die gewünschte Funktionlität erreicht wird.

Interaktive Auswertungsmöglichkeiten

Die Auswertung der Daten einiger ROSE-Werkzeuge kann interaktiv im Zusammenspiel mit dem Editor erfolgen. Neben einer statischen Analyse und der Auswertung von Testläufen ist besonders ein Werkzeug zur Visualisierung nahezu beliebiger Ausgabedaten zu erwähnen.

Anschauliches, formales Modell zur automatischen Ablaufsteuerung

Durch sog. Software-Projekt-Graphen (SPG) werden einzelne Entwicklungsschritte nachgebildet und automatische Abläufe ermöglicht. Im Gegensatz zu den herkömmlichen Makefiles können mit SPGen auch kreisförmige Abhängigkeiten und

hierarchische Strukturierungen anschaulich beschrieben werden. SPGen bilden die Basis für die Implementierung einer parallelen Generierungsfunktion (Make-Funktion). Über eine Visualisierungskomponente (GrafEddi) ist ein Überblick über die Daten und deren Aktualisierungszustand sowie die interaktive Manipulation der Struktur möglich.

Leichte Erweiterbarkeit

Neben der Vielzahl an bereits in ROSE integrierten Funktionen kann der Benutzer seine eigenen Werkzeuge auf einfache Weise über eine definierte Werkzeugschnittstelle in das System einbringen. Die Einbindung geschieht dabei interaktiv über eine komfortable Dialogmaske. Die Erweiterung der Programmierumgebung um zusätzliche (interne) Komponenten ist ebenfalls über eine festgelegte Schnittstelle möglich.

Verteiltes Arbeiten in einem Rechnerverbund

Durch einige speziell für Programmierumgebungen entwickelte Kommunikationsdienste wurde eine Kopplung der voneinander unabhängigen ROSE-Prozesse erreicht. Die zentrale Vermittlungsinstanz (der ROSE-Server) übernimmt neben der Aufrechterhaltung der Verbindungen zu den einzelnen ROSE-Prozessen einige globale Aufgaben, wie z.B. den Log-Dienst und die Rechteverwaltung. Ein gewisses Maß an Robustheit ist durch spezielle Vorkehrungen in Fehlerfällen gegeben. Ausfälle einzelner Prozesse oder sogar des zentralen ROSE-Servers sind in ihren Auswirkungen begrenzt; die Entwicklungsarbeit kann meist ohne nennenswerte Störung fortgesetzt werden. Die vorhandenen Rechner eines Verbundes können zur verteilten Ausführung der Werkzeuge genutzt werden. Der Benutzer hat dabei die Wahl zwischen der gezielten Angabe eines ausführenden Rechners und einer automatischen Verteilung nach Lastgesichtspunkten. Langdauernde Generierungsläufe werden so in der Regel erheblich verkürzt.

Portabilität

Alle ROSE-Werkzeuge sind in der Programmiersprache ANSI-C geschrieben. Ein Teil davon verwendet ausschließlich die Funktionen der Standard-C-Bibliothek und ist somit leicht portierbar. Die interaktiven Ein- und Ausgaben wurden über X-Windows (X11 Release 4) und OSF-Motif (Version 1.1) realisiert. Diese Kombination von graphischen Funktionen ist auf vielen UNIX-Rechnern vorhanden. Die Kommunikationsfunktionen verwenden einige spezialisierte UNIX-Systemfunktionen der Interprozeßkommunikation (Semaphore und gemeinsamer Speicher) sowie die Dienste der UDP-Funktionen für Internet-Sockets. Das ROSE-System ist derzeit auf folgenden Rechnern lauffähig: HP Serie 9000, Sun3 und Sun4.

Genauere Informationen sind in [Zan92] nachzulesen.

Literatur:

[Zan92]
Zanzinger, M.: " ROSE - Konzeption und Implementierung einer Progammierumgebung für rechnerorganisierte Software-Entwicklung", Dissertation an der Universität Erlangen-Nürnberg, 1992

4 Wachsende selbstorganisierende Netzwerke(Fritzke)

Selbstorganisierende Neuronale Netzwerke werden für verschiedene Probleme u. a. in den Bereichen Mustererkennung, Robotersteuerung, kombinatorische Optimierung und Datenkompression eingesetzt.

Eine wichtige Voraussetzung für den praktischen Einsatz solcher Netzwerke ist die Fähigkeit, mit ihrer Topologie beliebige Strukturen im n-dimensionalen Raum nachzubilden. Bisherige Ansätze, wie das Modell von Kohonen, sind aufgrund ihrer starren Netzwerktopologie dazu nur in geringem Umfang in der Lage. Das in [Fri92] vorgeschlagene Modell der "Wachsenden Zellstrukturen" führt in dieser Hinsicht zu wesentlich besseren Ergebnissen.

Zentrale Idee war die Einführung einer variablen Netzwerktopologie durch die beiden folgenden Mechanismen:

Kontrolliertes Einfügen von Neuronen und Verbindungen

Kontrolliertes Löschen von Neuronen und Verbindungen

Diese strukturverändernden Operationen werden durch die problemspezifische Wahrscheinlichkeitsverteilung der Eingabesignale gesteuert. Da im allgemeinen eine explizite Beschreibung der Wahrscheinlichkeitsverteilung nicht gegeben ist, muß diese nachgebildet werden. Zu diesem Zweck wurde ein Modellierungsverfahren entwickelt, mit dem zu jedem Zeitpunkt aufgrund der bis dahin eingetroffenen Eingabesignale lokale Schätzwerte für die Wahrscheinlichkeitsdichte abgeleitet werden können. Zeitpunkt und Position der Einfüge- und Löschoperationen können mit Hilfe dieser Schätzwerte bestimmt werden.

Bei der Realisierung variabler Netzwerkstrukturen tritt das grundlegende Problem der Wahl eines adäquaten Grundbausteines auf. Bisherige Ansätze verwenden einfache Neuronenketten oder rechteckige Neuronengitter. In der vorliegenden Arbeit wurden erstmals mehrdimensionale Hypertetraeder als Grundbausteine verwendet. Ausschlaggebend für die Wahl der Hypertetraeder ist deren Eigenschaft, sich besonders gut zu größeren Verbänden kombinieren zu lassen. Damit wird eine exakte Nachbildung von nahezu beliebig geformten n-dimensionalen Strukturen möglich.

Da die Dimension des Netzwerks unabhängig von der Dimension der Eingabedaten ist, steht eine neuartige Methode zur Visualisierung hochdimensionaler Daten zur Verfügung. Voraussetzung dafür ist die Fähigkeit des Modells, die räumliche Struktur der Wahrscheinlichkeitsverteilung der Eingabesignale in der Netzwerktopologie zu repräsentieren. Aufgrund dieser Eigenschaft lassen sich durch Verwendung ein-, zwei- oder dreidimensionaler Netzwerke Zusammenhänge innerhalb hochdimensionaler Daten darstellen.

Die praktische Einsatzfähigkeit des Modells der Wachsenden Zellstrukturen wurde durch verschiedene Anwendungen nachgewiesen. Einerseits wurde mit dem neuen

Modell ein extrem schnelles Verfahren zur Berechnung von Näherungslösungen für ein kombinatorisches Optimierungsproblem entwickelt. Andererseits ist das Modell in der Lage, große Datenmengen selbständig in Gruppen untereinander ähnlicher Daten (sog. "Cluster") zu strukturieren.

Anhand verschiedener Beispiele wurde ein direkter Vergleich mit dem wegweisenden Ansatz von Kohonen durchgeführt. Dabei ergab sich bezüglich verschiedener praxisrelevanter Kriterien wie beispielsweise Fehlerminimierung und Modellierung der Wahrscheinlichkeitsverteilung eine deutliche Überlegenheit des Modells der Wachsenden Zellstrukturen.

Das neue Modell ist grundsätzlich für dieselben Anwendungen geeignet wie die Selbstorganisierenden Merkmalskarten von Kohonen. Die Ergebnisse des durchgeführten Vergleichs beider Modelle deuten darauf hin, daß auch für die Problemstellungen, auf die bisher die Wachsenden Zellstrukturen noch nicht angewendet worden sind, bessere Ergebnisse als mit Kohonens Ansatz zu erwarten sind.

Erste Resultate zur Zeit laufender Untersuchungen in den Bereichen Bildkompression und Robotersteuerung bestätigen diese Vermutung.

Literatur:

[Fri92]
Fritzke, B.: "Wachsende Zellstrukturen - ein selbstorganisierendes neuronales Netzwerkmodell"; Dissertation an der Universität Erlangen-Nürnberg, 1992

5 Maschinelle Analyse natürlicher Sprache (Hasbargen/Keil)

5.1 Analyse mit Hilfe selbstlernender Grammatiken

Menschliches Wissen in allen Gebieten wird seit Generationen in schriftlicher Form gespeichert. Solche Information ist neuerdings oft nicht nur auf Papier, sondern auf Digitalrechnern leichter zugänglicheren Medien (z.B. CD-ROMs) verfügbar. Es stellt sich daher die Frage, ob die für Expertensysteme erforderliche Fachinformation dem Rechner nicht direkt aus solchen Quellen zugänglich gemacht werden könnte, anstatt sie mühsam "per Hand" einzugeben.

Die Umwandlung von natürlichsprachlich vorliegendem Wissen in eine dem Digitalrechner verständliche Form ist jedoch kompliziert: Da alle potentiell vorkommenden natürlichsprachlichen Konstrukte durch die verwendete Grammatik abgedeckt sein müssen und zudem noch die Semantik eines Satzes nach der Analyse als Datenstruktur vorliegen soll, ist die Entwicklung solcher Grammatiken generell langwierig und fehlerbehaftet.

Ein alternativer Ansatz geht hier den umgekehrten Weg: Es werden dem spracherkennenden System "Beispiel"-Sätze und die ihnen jeweils entsprechende semantische Übersetzung zum Lernen vorgegeben; aus diesen Paaren lernt das System durch Generalisierung, Mustervergleich und Regelvereinfachung die der natürlichen Sprache zugrundeliegende Grammatik selbständig.

5.2 Unifikationsbasierte Syntax- und Semantikanalyse

Um natürlichsprachliche Anfragen an digitale Rechenanlagen zu ermöglichen und damit auch Laien den Umgang mit Computern zu erleichtern, muß ebenso sprachliches wie außersprachliches Wissen formalisiert werden. Fachwissen aus den Bereichen Computerlinguistik, formale Linguistik, Logik und Informatik sind notwendig, um diese interdisziplinäre Aufgabe zu bewältigen.

Bei der Entwicklung eines Beschreibungsformalismus ist vor allen aber auch darauf zu achten, daß dieser möglichst universell für alle natürlichen Sprachen einsetzbar ist und nicht nur die Sprachbeschreibung einer Einzelsprache liefert.

Die unifikationsbasierten Grammatikansätze versuchen diesem Anspruch gerecht zu werden.Sie dienen der Repräsentation und Verarbeitung linguistischer Information syntaktischer und semantischer Natur.

Ein Grammatikformalismus, der in jüngster Zeit innerhalb dieser Gruppe der unifikationsbasierten Grammatikformalismen entstanden ist und viele Elemente früherer Grammatikformalismen vereint, ist die von Carl Pollard entwickelte Head-driven Phrase Structure Grammar (HPSG). In diesem Formalismus werden explizit universalgrammatische Prinzipien und sprachspezifische Grammatikregeln getrennt. Zentrales Sprachbeschreibungsmittel bei HPSG sind die Attribut-Wert-Matrizen.

Sämtliche Eigenschaften von Sprachstrukturen werden mittels Attribut-Wertpaaren, den sogenannten Merkmalen, beschrieben. Grundlegende Operation auf diesen Merkmalstrukturen ist die Unifikation, die es ermöglicht, aus kleineren Sprachsegmenten größere Konstituenten (bis hin zum Satz) zu bilden.

Eine Erweiterung erfährt der HPSG-Formalismus durch logische Operatoren, durch Mengen und Listen, durch funktional abhängige Werte und durch Typen und Sorten.

Zukünftige Untersuchungen sollen nun zeigen, welche linguistischen Phänomene sich mit Hilfe dieses Formalismus beschreiben und verarbeiten lassen, und ob zusätzliche Erweiterungen nötig sind, um eine angemessene Verarbeitung natürlicher Sprache zu ermöglichen.

Gastaufenthalte

Prof. Dr. M. Nabil Allam Univ. Damietta/Ägypten (19.-27.09.1992)
Dr. M. Slusarek Univ. Krakau/Polen (02.-07.11.1992)

Dissertationen

Fritzke, Bernd:
: Wachsende Zellstrukturen - ein selbstorganisierendes neuronales Netzwerkmodell -

Minas, Mark:
: Überwachung technischer Prozesse mit Zeitconstraintnetzen

Schied, Georg:
: Über Graphgrammatiken, eine Spezifikationsmethode für Programmiersprachen und verteilte Regelsysteme

Zanzinger, Michael:
: ROSE - Konzeption und Implementierung einer Programmierumgebung für rechnerorganisierte Software-Entwicklung

Diplomarbeiten

Arnold, Stefan:
: Parallele verteilte Garbage-Collection im HERAKLIT-System

Bothe, Andreas:
: Dialogmodellierung - Implementierung einer Komponente für CASE-Umgebungen

Bouillon, Peter:
: Untersuchung des Einflusses von Ausnahmekonzepten auf Optimierungsalgorithmen

Coßmann, Helmut:
: Effiziente Fehlerbehandlung in Dateitransferprotokollen

Debes, Norbert:
: Parallelisierung von Algorithmen für künstliche neuronale Netze

Gessnitzer, Daniel:
: Generierung von C-Quelltexten aus DICOL-Programmen

Kiczko, Stefan:
: Anwendung und Bewertung des Autorensystems "Authorware Professional for Windows"

Kustermann, Klaus:
: Vergleich verschiedener Methodologien zum Entwurf objektorientierter Softwaresysteme - Eine Fallstudie aus dem Bereich betriebswirtschaftlicher Anwendungen -

Lammermann, Detlef:
: Dialogmodellierung - Konzeption einer Komponente für CASE-Umgebungen

Mauser, Hans:
: Eine Experimentierumgebung für Genetische Algorithmen

Nachtrab, Robert:
: Ein neues Lernverfahren für die visuomotorische Koordination eines Roboterarms

Preckel, Elke:
: Datenvisualisierung mit selbstorganisierenden Neuronalen Netzen

Prisille, Kay:
: Entwicklung und Implementierung einer Bedienoberfläche eines Programmes zur Visualisierung und Simulation von Neuronalen Netzen (NeuroGraph)

Pöschner, Jürgen:
: Eine verteilte System-Verwaltung für HERAKLIT

Schneiders, Ludger:
: Implementierung eines Rahmenprogramms für eine Programmierumgebung

Vay, Albert:
: Effiziente Implementierung eines Chart-Parsers in SCHEME

Wagner, Gerhard:
: Entwurf und Implementierung einer Datenbasis zur Verwaltung von Programm-Ressourcen

Studienarbeiten

Adler, Werner:
Realisierung des HERAKLIT-Laufzeitsystems - Benutzungsoberfläche -

Ailinger, Bernd:
Entwicklung und Implementation eines Unifikationsmechanismus für gerichtete Graphen in SCHEME

Andres, Michael:
Beschreibung "mobiler Prozesse" mit Graphgrammatiken

Blaa, Helmut:
Ein Compiler-Oberteil zur Analyse von HERAKLIT-Direktanweisungen

Brennecke, Dirk:
Aufbereitung und graphische Darstellung von strukturierten Ausgabedaten

Buschmann, Achim:
Vektorquantisierung mit wachsenden Zellstrukturen

Do, Huu Trung:
Entwurf einer verteilten Graphersetzungsmaschine

Feltens, Peter:
Analyse und Implementierung dynamischer Programmkommunikation

Fischer, Ingrid:
Entwicklung einer objektorientierten Repräsentation für kompositionelle Aspekte der Wortsemantik

Friedmann, Andrea:
Visualisierung zeitlich variierender Parameter in künstlichen neuronalen Netzen

Frings, Sandra:
Entwurf und Implementierung eines Demonstrationsmoduls zu einem Simulatorsystem für künstliche neuronale Netze

Holve, Rainer:
Generierung von Testinstrumentierern

Käfferlein, Bernd:
Entwurf und Implementierung grafischer Darstellungen für künstliche neuronale Netze

Kaiser, Anja:
Verifikationsregeln für ausgewählte Mechanismen der Ausnahmebehandlung

Kuss, Oliver:
Implementierung eines Editors für hierarchische bipartite Graphen

Mansfeld, Christian:
Aufbau eines Regelwerks zur Optimierung des erzeugten Maschinencodes

Menke, Andreas:
Realisierung des HERAKLIT-Laufzeitsystems - Semantikroutinen für Objektkonvertierung und Objektkopie

Pohl, Matthias:
Realisierung des HERAKLIT-Laufzeitsystems - Datenstrukturen und Zugriffsfunktionen des Objektsegments -

Reitzner, Stephan:
Realisierung des HERAKLIT-Laufzeitsystems - lokale Kommuni kation und Server-Dienste -

Richter, Anke:
Beschreibung "mobiler Prozesse" mit Graphgrammatiken

Riechmann, Thomas:
Realisierung des HERAKLIT-Laufzeitsystems - verteilte Kommunikation und Server-Dienste -

Schmidt, Harald:
Vergleich von Algebra-markierten Graphgrammatiken und attributierten Zeichenkettengrammatiken

Spohr, Lionel:
Realisierung des HERAKLIT-Laufzeitsystems - Freispeicherverwaltung des Objektsegments -

Vorträge

Arius, P.:
"HERAKLIT: Eine objektorientierte Programmiersprache für verteilte Systeme", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Fritzke, B.:
"Using a library of efficient data structures and algorithms as a neural network research tool", ICANN, Brighton, 06. 09. 1992

Fritzke, B.:
Poster-Presentation, ICANN Brighton, 06. 09. 1992

Hasbargen, T./Keil, M.:
"\q... denn Sie wissen nicht, was Sie tun!\q: Grundlegende Probleme und Lösungsansätze der maschinellen Verarbeitung natürlicher Sprache", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Jacob:
"Simulation künstlicher neuronaler Netze" 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Jacob:
"Evolutionäre Programmierung - Was kann der Software-Entwickler von Darwin lernen" Internes Seminar der Firma BASYS, Tennenlohe, 14. 11. 1992

Minas, M.:
"Erkennung von Fehlern im Betrieb technischer Anlagen", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Schied, G.:
"Spezifikation der Semantik verteilter Programmiersprachen mittels Graphgrammatiken", Mathematik-Kolloquium der Univ. Mainz, 27. 02. 1992

Schied, G.:
"Konfuzius, Einstein und Graphgrammatiken oder: Grundphänomene in verteilten Systemen", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Schneider, H. J.:
"Graphgrammatiken", Gastvorlesung an der Techn. Hochschule Ilmenau, 21. 05., 11. 06., 18. 06. u. 25. 06. 1992

Schorr, R.:
"Reliability and Efficiency Aspects of Real-Time Exception Handling", Brügge, 23. 06. 1992

Viehstaedt, G.:
"Visuelle Sprachen: Ein bewährtes Prinzip mit neuen Perspektiven", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Wilke, P.:
"Was bieten Neuronale Netze?", Seminarreihe "Neue Technologien für den Mittelstand" der IHK Nürnberg, 14. 05. 1992

Wilke, P.:
"Künstliche Neuronale Netze: Nürnberger Trichter oder Elektronengehirn", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Wilke, P.:
"Simulation künstlicher Neuronaler Netze", Arbeitsgemeinschaft

Funktionalanalysis Univ. Erlangen-Nürnberg, 22. 06. 1992

Wilke, P.:
"Simulating Neural Networks in a Distributed Computing Environment", CORTEX Workshop Saint Louis, Frankreich, 01. 07. 1992

Wilke, P.:
"NeuroGraph: Ein Simulator für künstliche Neuronale Netze", Fa. Kratzer GmbH, Unterschleißheim, 13. 08. 1992

Wilke, P.:
"Entwicklungswerkzeuge für Neuronale Netze und Genetische Algorithmen", Fa. MEDAV GmbH., Uttenreuth, 11. 12. 1992

Zanzinger, M.:
"ROSE: Die integrierte, sprachunabhängige Programmierumgebung", 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 12. 06. 1992

Veröffentlichungen

Arius, P./Betz, W./Schied, G./Schorr, R./Viehstaedt, G.:
"Programmierumgebung (für Multiprozessor- und Netzwerkkonfigurationen)", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Feder-Andres, C./Schorr, R.:

"Reliability and Efficiency Aspects of Real-Time Exception Handling", Proc. of the 18th IFAC/IFIP Workshop on Real-Time Programming (WRTP\q92), pp. 25 - 30, 1992

Feder-Andres, C./Schorr, R.:

"Realzeitprogrammierung mit PEARL" in der Informatikausbildung der Universität Erlangen-Nürnberg, PEARL 92 Workshop über Realzeitsysteme, Informatik-Aktuell, pp. 157 - 169, 1992

Fritzke, B.:
"Growing Cell Structures - a Self-organizing Network in k Dimensions," in Artificial Neural Networks II, ed. J. Taylor, pp. 1051 - 1056, North-Holland, Amsterdam, 1992

Fritzke, B.:
"Using a Library of Efficient Data Structures and Algorithms as a Neural Network Research Tool," in Artificial Neural Networks II, ed. J. Taylor, pp. 1273 - 1276, North-Holland, Amsterdam, 1992

Fritzke, B.:
"Selbstorganisierende Neuronale Netzwerkmodelle", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Hasbargen, T./Keil, M.:
"Maschinelle Analyse natürlicher Sprache", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Jacob, C./Wilke, P.:
"Simulation und Visualisierung künstlicher Neuronaler Netze unter Verwendung paralleler Hardware", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Minas, M.:
"Erkennung von Fehlern im Betrieb technischer Anlagen", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Schneider, H. J.:
"Graphgrammatiken", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992

Viehstaedt, G./ A.L.:
"Visual Representation and Manipulation of Matrices", in: Journal Am of Visual Languages and Computing, Vol. 3, Nr. 3, pp. 273 - 298, Academic Press, London, 1992

Wilke, P.:
"Simulating Neural Networks in a Distributed Computing Environment", Proc. CORTEX Workshop, 01./02. 07. 1992, Saint Louis, Frankreich

Zanzinger, M.:

"ROSE - eine Programmierumgebung für rechnerorganisierte Software-Entwicklung", in: Festschrift zur 20-Jahrfeier des Lehrstuhls für Programmiersprachen, 1992