T U M

I N S T I T U T   F Ü R   I N F O R M A T I K

Graduiertenkolleg
Kooperation und Ressourcenmanagement in verteilten Systemen

Arbeits- und Ergebnisbericht
zum ersten Fortsetzungsantrag auf Weiterführung des Graduiertenkollegs

TUM-I9707
März 97

T E C H N I S C H E   U N I V E R S I T Ä T   M Ü N C H E N

TUM-INFO-03-I9707-100/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Graduiertenkolleg

Kooperation und Ressourcenmanagement in verteilten Systemen

Arbeits- und Ergebnisbericht zum ersten Forsetzungsantrag im Frühjahr 1997

Herausgeber:

Prof. Dr. P. P. Spies

Sprecher des Graduiertenkollegs

Institut für Informatik

Technische Universität München

80290 München

Tel.: +49 - 89 - 289 - 28252

Fax: +49 - 89 - 289 - 22037

e-mail: gkolleg@informatik.tu-muenchen.de

http://hp0.informatik.tu-muenchen.de/gkolleg/gkolleg.html

Vorwort

Das Graduiertenkolleg „Kooperation und Ressourcenmanagement in verteilten Systemen“ wurde Anfang 1995 an der Technischen Universität München eingerichtet. Mit den entsprechenden Bewilligungen der Deutschen Forschungsgemeinschaft konnten für die Teilprojekte des Kollegs, die am Institut für Informatik und am Institut für Informationstechnik durchgeführt werden, im Jahre 1995 elf und im Jahre 1996 sechs weitere Promotionsstipendien vergeben werden.

Das Graduiertenkolleg hat sich die Aufgabe gestellt, die Einsatz– und Nutzungsmöglichkeiten verteilter Systeme für parallele und kooperative Anwendungsproblemlösungen sowie die dazu erforderlichen Methoden, Konzepte und Verfahren weiterzuentwickeln. Die vielfältigen Fragestellungen, die sich aus dieser Aufgabe ergeben, werden in den Teilprojekten des Kollegs, die zu zwei Themenbereichen zusammengefaßt sind, bearbeitet. Unter dem Themenbereich „Kooperierende Agenten in verteilten Anwendungen“ sind Arbeiten zusammengefaßt, mit denen von den spezifischen Anforderungen ausgewählter Anwendungsgebiete ausgehend Lösungsmethoden und –verfahren für die Probleme dieser Anwendungen entwickelt werden. Dabei werden insbesondere Lösungsansätze mit Systemen intelligenter, flexibel kooperierender Agenten verfolgt. Unter dem Themenbereich „Programmiermodelle und Ressourcenmanagement für verteilte Systeme“ sind Arbeiten zusammengefaßt, mit denen Methoden und Verfahren zur Realisierung paralleler und kooperativer Problemlösungen auf Parallelrechnern und auf Workstation–Netzen entwickelt werden. Die behandelten Themen reichen von Grundlagenproblemen bis hin zu unmittelbar einsetzbaren Verfahren.

Der vorliegende Bericht, der zusammen mit dem ersten Fortsetzungsantrag vorgelegt wird, gibt einen Überblick über die Arbeit des Kollegs und über den gegenwärtigen Stand der Arbeiten der Kollegiaten. Er zeigt die Vernetzung zwischen den Arbeiten des Kollegs und den Arbeiten, die in der näheren und weiteren Umgebung des Kollegs durchgeführt werden; er macht deutlich, daß das Kolleg eine wichtige integrierende und Synergien erzeugende Institution ist.

München, im März 1997 — Prof. Dr. P. P. Spies

Inhaltsverzeichnis

Graduiertenkolleg: Kooperation und Ressourcenmanagement in verteilten Systemen

Übersicht über die Teilprojekte

Leitung: Prof. Dr. P. P. Spies

Themenbereich 1:	**Kooperierende Agenten in verteilten Anwendungen**
Teilprojekt 1.1:	Kooperierende Agenten in verteilten Systemen – Grundlagen
Leitung:	Prof. Dr. W. Brauer
Kollegiaten:	Dirk Ormoneit, Angela Bücherl
Teilprojekt 1.2:	Ressourcenmanagement in breitbandigen ATM–Kommunikationsnetzen
Leitung:	Prof. Dr. J. Eberspächer
Kollegiat:	Bernhard Quendt
Teilprojekt 1.3:	Kooperierende Agenten im Netz– und Systemmanagement
Leitung:	Prof. Dr. H. G. Hegering
Kollegiat:	Maria–Athina Mountzia
Teilprojekt 1.4:	Kooperierende Agenten und autonome Robotersysteme
Leitung:	Prof. Dr. H.-J. Siegert
Kollegiat:	Florian Fuchs
Teilprojekt 1.5:	Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit
Leitung:	Prof. Dr. J. Schlichter
Kollegiat:	Martina Nöhmeier
Teilprojekt 1.6:	Bildverstehen in verteilten Systemen
Leitung:	Prof. Dr. B. Radig
Kollegiaten:	Dietrich Büsching, Maximilian Lückenhaus
Themenbereich 2:	**Programmiermodelle und Ressourcenmanagement für verteilte Systeme**
Teilprojekt 2.1:	Verteilte Algorithmen – Spezifikation, Modellierung, Korrektheit
Leitung:	Prof. Dr. J. Esparza, Prof. Dr. W. Reisig
Kollegiat:	Richard Mayr
Teilprojekt 2.2:	Programmentwicklung für Parallelrechner und vernetzte Architekturen
Leitung:	Prof. Dr. A. Bode
Kollegiat:	Christian Röder
Teilprojekt 2.3:	Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme
Leitung:	Prof. Dr. M. Paul
Kollegiaten:	Maximilian Frey, Markus Podolsky
Teilprojekt 2.4:	Konstruktion heteromorph paralleler Systeme
Leitung:	Prof. Dr. P. P. Spies
Kollegiat:	Sascha Groh
Teilprojekt 2.5:	Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für verteilte Programmierung
Leitung:	Prof. Dr. J. Eickel
Kollegiaten:	Boris Reichel, Barbara König
Teilprojekt 2.6:	Verteilte Inferenzsysteme
Leitung:	Prof. Dr. E. Jessen
Kollegiaten:	Marc Fuchs
Teilprojekt 2.7:	Verteilte numerische Algorithmen auf Bäumen
Leitung:	Prof. Dr. Chr. Zenger
Kollegiat:	Martin Backschat

1 Umsetzung der Zielsetzung und der Konzeption des Kollegs

Am Anfang seiner Einrichtung stand für die Arbeiten des Kollegs folgender Plan:

- Der Tragweite und der Vielschichtigkeit des Themas „Kooperation und Ressourcenmanagement in verteilten Systemen“ entsprechend sollen die Arbeiten mit den beiden Themenschwerpunkten „Kooperierende Agenten in verteilten Anwendungen“ sowie „Programmiermodelle und Ressourcenmanagement für verteilte Systeme“ von den Kollegiaten mit den Themen der einzelnen Teilprojekte durchgeführt werden.

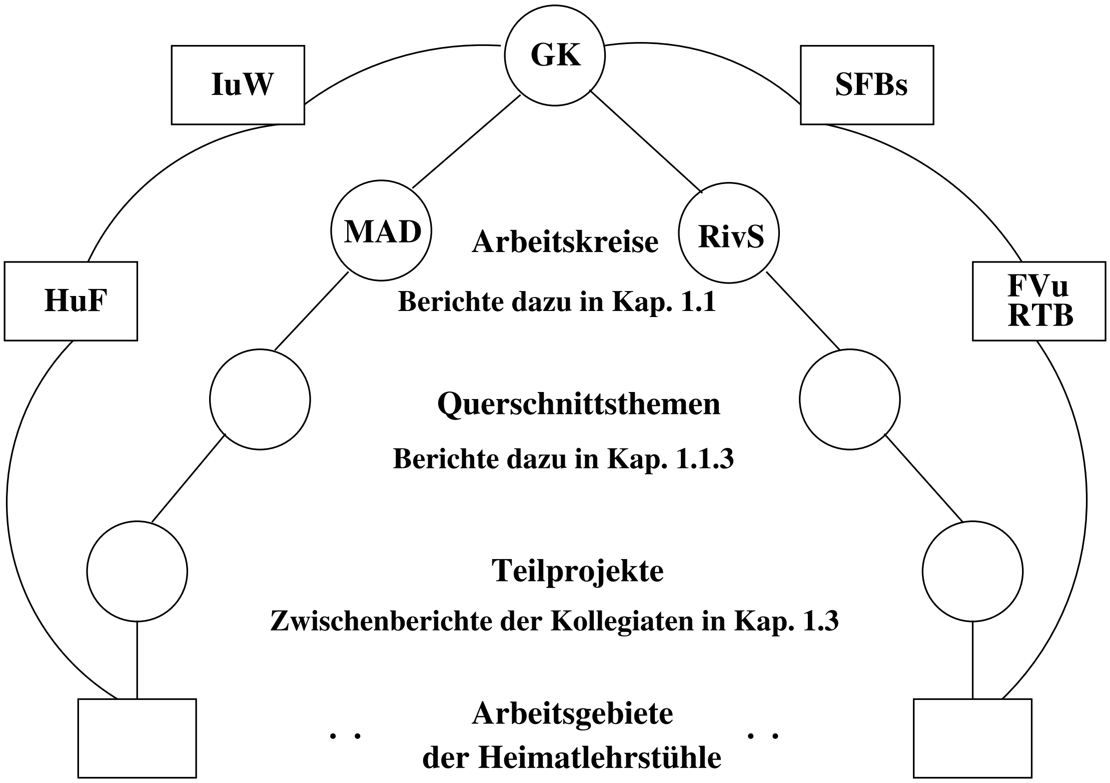
- Die Kollegiaten sollen in die Arbeitsgruppen der Heimatlehrstühle der Teilprojekte, deren Themen sie behandeln, integriert werden, so daß sie einerseits als Mitglieder des Kollegs und andererseits als Mitglieder der Gruppen ihrer Lehrstühle Anleitung, Anregungen und Diskussionspartner für ihre Arbeiten erhalten.

Diesem Plan entsprechend wurden bald nach seiner Einrichtung zwei Arbeitskreise des Kollegs mit den beiden Themenschwerpunkten gebildet. Sie waren die Ausgangsbasis für die weitere Entwicklung des Kollegs.

Die dem Startplan entsprechende Konzeption für die Arbeiten des Kollegs hat sich bewährt. Sie hat wesentlich dazu beigetragen, daß sich die Zusammenarbeit im Kolleg sowie die Zusammenarbeit der Kollegiaten mit den Gruppen der Lehrstühle und über diese mit weiteren Institutionen intensiv weiterentwickelt hat. Das Kolleg ist ein wichtiger Träger der Vernetzung zwischen den Arbeiten dieser internen und externen Gruppen geworden.

Der Graph gibt einen vergröberten Überblick über die Vernetzungen der Arbeiten des Kollegs. Er skizziert die Themen, die von den Kollegiaten im Kolleg bearbeitet wurden und werden, sowie die Zusammenarbeit der Kollegiaten mit weiteren Gruppen und mit externen Institutionen.

Im folgenden wird über diese Arbeiten berichtet, und zwar in Kapitel 1.1 über die beiden Arbeitskreise und über die Querschnittsthemen des Kollegs sowie in Kapitel 1.2 über die Zusammenarbeit mit externen Institutionen. In Kapitel 1.3 folgen die Zwischenberichte der Kollegiaten.



Vernetzung der Arbeiten des Kollegs

1.1 Arbeitskreise und Querschnittsthemen des Kollegs

Um die Kooperation und Zusammenarbeit zwischen den einzelnen Kollegiaten zu fördern, wurden im ersten Jahr des Bestehens des Graduiertenkollegs zwei Arbeitskreise ins Leben gerufen. Die allgemeine Zielsetzung der Arbeitskreise war, ein regelmäßiges Treffen der Kollegiaten zur Diskussion von allgemeinen, für alle Kollegiaten interessanten Themen und die Absicht, über die einzelnen Arbeitsschwerpunkte der Kollegiaten in einem etwas größeren Kreis zu reden. Die Aufteilung dieses Diskussionsforums in zwei Teile ist aus den beiden Themenbereichen des Graduiertenkollegs hervorgegangen. Daneben existiert ein Diskussionsforum für alle Kollegiaten, in dem themenbereichübergreifende Diskussionen geführt werden.

1.1.1 Arbeitskreis MAD

In den letzten Jahren wurde das Thema Agenten sehr intensiv und kontrovers diskutiert. Trotz vieler ungelöster Fragen waren die Prognosen für die weitere Entwicklung von Agenten oft euphorisch. Demgegenüber gab es jedoch nicht unberechtigte Befürchtungen, daß sich der Agentenbegriff immer mehr zu einem vieldeutigen und zunehmend inhaltslosen Mode-Schlagwort entwickeln könnte.

Innerhalb des Graduiertenkollegs beschäftigen sich zahlreiche Teilprojekte mit dem Thema Agenten, jedoch werden sehr unterschiedliche Einsatzgebiete bzw. Teil-Aspekte von Agenten behandelt. Diese Problematik ist gleichzeitig eine Herausforderung, da die Möglichkeit besteht, Agenten aus unterschiedlichen Sichtweisen zu beleuchten. Der Arbeitskreis MAD (Multi-Agent Discussion) wurde in diesem Zusammenhang geschaffen, um aus diesen heterogenen Sichtweisen gemeinsame Views zu extrahieren und dadurch eine wichtige Voraussetzung für die Kooperation zwischen den Teilprojekten zu schaffen. Im folgenden sind 5 Themenbereiche kurz beschrieben, die im Rahmen der Arbeitskreistreffen zur Erreichung des o.g. Ziels behandelt wurden.

1.1.1.1 Motivation für den Einsatz von Agenten

In vielen Anwendungsgebieten wird nicht nur der Begriff Agent, sondern auch die Beweggründe für den Agenteneinsatz bzw. die erwartete Bedeutung von Agenten für die verschiedenen Anwendungsgebiete kontrovers diskutiert.

Eines der ersten Themen, die innerhalb der Arbeitskreisdiskussionen untersucht wurden, war deshalb die Fragestellung, welches aus der Sicht der beteiligten Anwendungsgebiete die Motivationen sind, Agententechnologie einzusetzen. Hierbei wurden für die Anwendungsbereiche Telekommunikation, Netzmanagement, CSCW, Robotik und Bildverarbeitung sowohl Mißstände und unerfüllte Technologiewünsche diskutiert, die die Entwicklung neuer Technologien wie Agenten motivieren und vorantreiben, als auch Erwartungen und Anforderungen, die sich daraus für die Agententechnologie ergeben. Aus diesen projektspezifischen Sichtweisen wurde daraufhin eine projektübergreifende Übersicht erstellt, die wichtige, in mehreren Anwendungsgebieten relevante Motivationskriterien zusammenfaßt.

Die folgende Aufstellung enthält einige Beispiele für Antriebsfaktoren und nicht zufriedenstellende Ausgangssituationen, die nach Meinung der Teilnehmer am Arbeitskreis dazu geführt haben, daß in vielen Anwendungsgebieten zunehmend der Einsatz von Agententechnologie in Betracht gezogen wird:

1. Viele Problemlösungen sind durch einen hohen Ressourcenaufwand, aber gleichzeitig auch durch hohe Anforderungen an Effizienz gekennzeichnet. Dies übersteigt häufig die Leistungsfähigkeit sequentieller Architekturen, so daß eine nebenläufige Ausführung bzw. Verteilung auf mehrere Hardwareeinheiten erforderlich wird, die schon beim Entwurf der Problemlösungen berücksichtigt werden sollte.

2. Viele Aufgabenstellungen haben selbst schon einen inhärent verteilten Charakter oder beinhalten sogar Konfliktsituationen, die nicht zentral gelöst werden können. In diesen Aufgabenbereichen existiert der Wunsch nach einer möglichst natürlichen Repräsentation der gegebenen Verteiltheit in der Problemlösung.

3. In einigen Anwendungsgebieten ergibt sich aus der gleichzeitigen Forderung nach Verteiltheit, Kooperation und Effizienz der zusätzliche Wunsch nach Mobilität der kooperierenden Komponenten.

4. In vielen Aufgabenbereichen (z.B. bei Problemstellungen, die für globale Szenarien gelöst werden müssen) ist die zu lösende Aufgabenstellung durch große Heterogenität bzw. Dynamik gekennzeichnet. Hier werden leistungsfähige, nebenläufig arbeitende Modellierungs- und Ausführungskomponenten benötigt, die die Strukturierung, aber auch die Anpassungsfähigkeit der modellierten Lösung unterstützen.

5. Des weiteren erhofft man sich häufig durch den Einsatz kooperierender, nebenläufig arbeitender Software-Einheiten Vorteile bezüglich Offenheit, Skalierbarkeit und — in Kombination mit dem Einsatz redundanter Software-Einheiten — eine bessere Robustheit der Gesamtsysteme.

1.1.1.2 Der Agentenbegriff aus der Sicht verschiedener Forschungsgebiete

Für eine Softwareeinheit, die weitgehend selbständig komplexe Aufgaben verrichtet, wurde in vielen Teil- und Anwendungsgebieten der Informatik der Begriff des Agenten geprägt. Die Mannigfaltigkeit und Komplexität diverser meist anwendungsspezifischer Aufgabenstellungen, die erfolgreich von Agenten bearbeitet wurden, ist beachtlich. Bei oberflächlicher Betrachtung ensteht daher oft der Eindruck, Agenten und die zugehörige Agententechnologie seien eine Art universeller Problemlöser. Um dieses trügerische Scheinbild abzumildern, wurde im Arbeitskreis untersucht, welche technologischen Neuerungen sich konkret hinter den Agenten einzelner Anwendungsgebiete verbergen, durch welche der im vorigen Abschnitt angesprochenen Erwartungen diese jeweils motiviert waren und welche Probleme damit einhergehen.

Die folgende Zusammenstellung zeigt den Agentenbegriff daher aus der Sicht verschiedener Forschungsgebiete, in welchen sich Agententechnologie entwickelt hat. Es werden

Agententyp	**Erwartung**	**Technologische Neuerung**
Intelligenter Agent	Komplexitätsreduktion beim Programmentwurf, Kooperation von Expertensystemen.	Mentalistische Logik, Programmstruktur, Agentenzyklus, Strukturierte Kommunikationssprache.
Mobiler Agent	Neue Anwendungen und Produkte: Network-PC (NC), Smart Messaging, Flexible Agenten im Netzmanagement	Agentensprachen und Interpreter: z.B. Telescript, Java.
Benutzeragent	Kooperation mit menschlichem Benutzer, Abnahme von Routineentscheidungen.	Autonome, lernende, benutzeradaptierte Werkzeuge.

Tabelle 1: Agententypen, Erwartungen und Neuerungen der Agententechnologie.

drei wichtige Agententypen aufgelistet, sowie Beispiele für Errungenschaften, die für oder mit dem jeweiligen Agententyp realisiert wurden.

Eine weitere Möglichkeit, Agenten über wiederholt anzutreffende Eigenschaften zu definieren wird im nächsten Abschnitt beschrieben.

Die mentalistische Logik wurde u.a. mit dem Ziel entworfen, eine Komplexitätsreduktion beim Entwurf von Softwaresystemen durch die Möglichkeit einer Art intuitiven Programmierung zu erreichen. Kern der Logik bilden dabei Begriffe, mit denen der Mensch intuitiv bestimmte Vorstellungen verbindet, etwa Glaube (Belief), Wunsch (Desire) und Absicht (Intention). Wichtigster Vertreter dieser Logik bildet die BDI-Logik (Belief-Desire-Intention Logik) [Sho93].

Agentenprogramme weisen häufig eine charakteristische, zumeist dreistufige Struktur, den Agentenzyklus auf. Man unterscheidet im wesentlichen die Stufen Wahrnehmung (Sensorik), Planung und Handlungsausführung (Aktorik). Mit dieser Strukturierung eines Agentenprogrammes ist ebenfalls die Erwartung auf Komplexitätsreduktion beim Entwurfsprozeß verbunden. Die Tatsache, daß das Agentenprogramm einen Planungprozeß durchläuft, führt allerdings oft zu langsamer Performanz. Aus diesem Grunde wurden alternativ reaktive Programmarchitekture vorgeschlagen.

Die Kooperation von Agenten ist Forschungsgegenstand der Verteilten Künstlichen Intelligenz (VKI). Motiviert wurde dieser Teilbereich der Informatik ursprünglich durch die Erwartung, komplexe Probleme durch kooperierende Expertensysteme effizienter lösen zu können. Als Voraussetzung wurden verschiedene strukturierte Kommunikationssprachen entwickelt, ein Hauptvertreter ist KQML/KIF. Im Unterschied zur Objektkommunikation sind bei der Agentenkommunikation mit Hilfe einer strukturierten Kommunikationssprache nur standardisierte Nachrichtentypen zulässig.

Mentalistische Logik, der Agentenzyklus und eine strukturierte Agentenkommunikationssprache sind häufig gemeinsame Kennzeichen von Agenten aus dem Bereich KI/VKI. Zur Abgrenzung von anderen Agententypen werden sie als **Intelligente Agenten** bezeichnet.

Einen zweiten Agententyp bilden die **Mobilen Agenten** — Programmcode, der während seiner Ausführung von Rechner zu Rechner migrieren kann. Dies setzt das Vorhandensein eines speziellen Agentenspracheninterpreters auf den Rechnern voraus. Die Erwartungen an Mobile Agenten sind vielschichtig und hauptsächlich durch potentielle Anwendungen geprägt. Drei Anwendungen seien kurz skizziert, der NC (Network-PC oder Internet-PC), Intelligente Mailnachrichten und sog. 'Flexible Agenten' im Netzmanagement.

Der NC bezieht den Großteil seiner Software in Form von Mobilen Agenten über das Internet. Software ist somit nicht fest installiert, sondern wird bei Bedarf leihweise gegen Gebühr von speziellen Internetservern zur Verfügung gestellt.

In intelligenten Mailnachrichten (smart messaging) werden herkömmliche Nachrichten durch Programmcode ergänzt. Dies eröffnet eine Reihe von neuen Gestaltungsmöglichkeiten für derartige Nachrichten, z.B Animation.

Im Netzmanagement werden einfache Routineaufgaben (Erfassung von Ressourcendaten) von Agenten und übergeordnete, koordinative Arbeiten von sogenannten Managern durchgeführt. Zur Entlastung der Manager erweist sich das Konzept der flexiblen Agenten als vorteilhaft. In Form mobilen Programmcodes werden Teilaufgaben der Manager an flexible Agenten delegiert.

Unabhängig von speziellen Anwendungen ist mit Mobilen Agenten die Hoffnung verbunden, durch Migration von Programmteilen eine effiziente Lastverteilung in Mehrrechnersystemen zu erreichen.

Als wesentliche technologische Neuerung im Zusammenhang zu Mobilen Agenten sind Agentensprachen zu sehen, die die Migration von Programmcode ermöglichen. Anhand des Ausgangspunktes der Migration lassen sich zwei Arten unterscheiden. Zum einen Sprachen, bei denen der Anlaß zur Migration vom Quellrechner ausgeht (z.B Telescript), zum anderen Sprachen, bei denen der Zielrechner die Migration von Programmcode einleitet (z.B. Java).

Den dritten Agententyp bilden die **Benutzeragenten**. Bei ihnen steht weniger die Kooperation mit anderen Agenten, als vielmehr die Kooperation mit einem menschlichen Benutzer im Vordergrund. Damit verbunden ist die Erwartung, daß der Benutzeragent letztendlich dem Menschen gewisse Routineaufgaben abnimmt. Voraussetzung für den Einsatz von Benutzeragenten ist die Existenz wiederkehrender Routineaktionen im menschlichen Handeln.

Bekannt ist das Mailwerkzeug von P. Maes [Mae94] als typischer Benutzeragent. Er durchläuft drei Zustände, die durch zunehmende Autonomie gekennzeichnet sind. Zu Beginn, in der Lernphase, erkennt und speichert der Benutzeragent wiederkehrende Aktionsmuster. Dabei vergleicht der Agent fortwährend Aktionen, die er vorschlagen würde, mit tatsächlich vom Benutzer erbrachten. Sobald ein gewisser Grad an Übereinstimmung erreicht ist, macht der Agent selbständig Handlungsvorschläge, die aber zunächst noch vom Benutzer quittiert werden müssen. Erst wenn der Agent das Benutzerverhalten weitgehend korrekt vorhersagt, ist auch eine vollständig autonome Vorgehensweise möglich.

1.1.1.3 Eigenschaften zur Charakterisierung von Agenten

Wenn es einen Begriff aus der Informatik gibt, der mehr als jeder andere in verschiedensten Teilbereichen und auch angrenzenden Gebieten dieser Wissenschaft in den letzten Jahren Verbreitung gefunden hat, so trifft dies auf den Begriff 'Agent' zu. Bis zum heutigen Zeitpunkt ist diesem Begriff aber keine eindeutige Bedeutung zugeordnet — eine verbindliche Definition existiert nicht.

Wenn man bisherige Definitionsansätze aus der Literatur betrachtet, so kann man zwei unterschiedliche Wege zu einer solchen Definition erkennen, die beide ihre eigene Problematik besitzen. Auf der einen Seite findet man Ansätze, die anwendungsspezifische Anforderungen als Maßstab für Agenten ansetzen. Dieser ist dann gewöhnlich nicht auf andere Anwendungsbereiche übertragbar. Somit resultieren daraus unvereinbare Agentenbegriffe, die im wesentlichen eine Menge von Maßnahmen beinhalten, von denen man sich eine Verbesserung für die jeweilige Anwendung erhofft.

Auf der anderen Seite gibt es Ansätze, die versuchen, zu beschreiben, wie der Agent schlechthin aussehen soll. Dies wiederum ist als problematisch anzusehen, weil Agenten stets mit an die Anwendung angepaßten Fähigkeiten ausgestattet sein werden. Die Realisierung eines Universalagenten ist nicht erfolgversprechend.

Im Arbeitskreis wurde daher der Weg eingeschlagen, nicht einen bestimmten Katalog von Eigenschaften zu postulieren sondern vielmehr nur das Typische für einen Agenten herauszustellen. Eigenschaften, die ein Agent potentiell besitzt — z.B. Autonomie, Mobilität oder Rationalität, um nur einige der in der Literatur häufig genannten zu zitieren (siehe z.B. [WJ95]) — sind näher zu charakterisieren. Dieser Ansatz endet zwar nicht in einer Definition des Begriffes 'Agent', verspricht aber umso mehr, zu einem breiten Konsens bezüglich dieses Begriffes führen zu können.

Als typisch für einen Agenten müssen vor allem die im folgenden kurz charakterisierten Merkmale genannt werden:

Kapselung von Fähigkeiten in einer Rolle. Ein Agent übt eine spezifizierte Rolle aus und kapselt in sich die hierzu notwendigen Fähigkeiten. Dabei ist es nicht entscheidend, ob erforderliches Wissen tatsächlich lokal im Agenten gespeichert ist oder ob der Agent über geeignete Methoden verfügt, sich dieses Wissen aus seiner Umwelt zu beschaffen. Ein Agent kann mehrere Rollen in seinem Repertoire haben und diese je nach Erfordernis ausüben.

Agentenzyklus. Typisch für Agenten ist, daß sie einen Zyklus aufweisen, der im wesentlichen die Stufen *Perzeption*, *Planung* und *Aktion* durchläuft. Beispielsweise führt Levi in [Lev89] einen derartigen Autonomiezyklus ein, der aus den elementaren Übergangsfunktionen *plan*, *decide*, *effect* und *modify* und den drei Sensorfunktionen *see*, *do* und *monitor* besteht. Durch diesen Zyklus wird das Problemlösungsverhalten eines Agenten entscheidend bestimmt.

Explizite Wissensrepräsentation. Agenten repräsentieren typischerweise bestimmte Anteile ihres Wissens explizit. Dies ist ein wesentlicher Faktor für die Realisierung von Lernfähigkeit. Dabei muß differenziert werden, ob ausschließlich Domänenwissen explizit repräsentiert wird, oder zusätzlich auch noch Kontrollwissen, was weitergehende Konzepte zur dynamischen Adaption von Agenten an wechselnde Anforderungen ermöglicht.

Fähigkeit zur komplexen Kommunikation. Kommunikation spielt naturgemäß eine große Rolle in einem Multi-Agenten System. In Hinblick auf die kommunikativen Fähigkeiten besitzen Agenten Möglichkeiten, die über das einfache *send/receive* der Objekte hinausgeht — sie sind in der Lage, komplexe Verhandlungen zu führen.

Darüberhinaus wurde im Rahmen des Arbeiteskreises ein Konzept zur Einteilung von Agentencharakteristika in verschiedene Kategorien erarbeitet, das einen Klassifizierungsansatz für Agenten bietet. Desweiteren wurden noch andere Fragen diskutiert, die in Verbindung mit dieser Thematik stehen. Beispielsweise wurde erörtert, inwiefern die Entwicklung von Programmiersprachen — speziell Agentenprogrammiersprachen — einen prägenden Einfluß auf den Agentenbegriff besitzt.

1.1.1.4 Lernen in Agentensystemen

Eine häufig wünschenswerte Eigenschaft eines Agenten stellt seine Lern- bzw. Adaptionsfähigkeit dar. Multiagentensysteme (MAS) zeichnen sich gewöhnlich durch beträchtliche Komplexität sowie nichtdeterministische Dynamik aus. Dementsprechend ist es für die Konstruktion eines effizienten Gesamtsystems unerläßlich, daß die einzelnen Agenten sich den Veränderungen ihrer Umwelt anpassen.

Entsprechend wurden innerhalb des Arbeitskreises existierende Lernverfahren in Hinblick auf ihre Anwendbarkeit für MAS untersucht. Dabei stellte sich heraus, daß u.a. künstliche neuronale Netze (KNN) für die sich ergebenden Problematiken vielversprechend erscheinen. Hierbei unterscheidet man zwischen überwachtem, unüberwachtem und Verstärkungslernen:

Überwachtes Lernen zeichnet sich durch die Gegenwart eines Lehrers aus, welcher das korrekte Verhalten in einer gegebenen Situation spezifiziert. Ziel des Lernens ist es, ein Verhaltensmuster zu entwickeln, welches der vorgegebenen Spezifikation möglichst genau entspricht. Als Beispiel hierfür haben wir den Backpropagation-Algorithmus betrachtet.

Unüberwachtes Lernen: Hier ist kein Lehrer vorhanden, sondern das geeignete Verhalten muß auf der Basis von stochastischen Suchalgorithmen und selbstorganisierenden Prozessen herausgefunden werden. Beispiele: Kohonen-Karten, Mischdichten mit EM-Algorithmus.

Verstärkungslernen: Als Feedback erhält der Lernende lediglich ein Signal, welches den momentanen Nutzen heutiger sowie vergangener Aktionen zusammenfaßt. In Hinblick auf MAS erscheinen derartige Verfahren erfolgversprechend, da das MAS ja eine wichtige zeitliche Dimension besitzt, so daß zur Beurteilung des Nutzens einer Aktion zum Zeitpunkt t die gesamte Vorgeschichte berücksichtigt werden muß.

Speziell im Zusammenhang mit MAS sind die beschriebenen Lernverfahren erst seit kurzem Gegenstand intensiver Forschung. Im Gegensatz zum Single-Agent-Learning oder isolierten Lernen muß bei Multiagentenlernen berücksichtigt werden, daß das Wissen sowie das Lernen eines Agenten durch Wechselwirkungen mit den anderen Agenten beeinflußt wird. Es existieren allerdings Grenzsituationen, in denen keine klare Trennung zwischen den beiden Lernkategorien möglich ist. Wir hoffen, im Rahmen des Arbeitskreises verstärkt eine Integration in die Anwendungen aus anderen Teilprojekten, sowie eine entsprechende Erweiterung der dort angewandten Methoden zu erreichen.

1.1.1.5 Agentensprachen

Neben vielen theoretischen Aspekten von Agenten sind für viele der Agententeilprojekte des Graduiertenkollegs auch Fragen bezüglich der Realisierung von Agenten von Interesse. Erst wenn die in den vorigen Abschnitten beschriebenen Eigenschaften auch tatsächlich effizient mittels geeigneter Programmiersprachen spezifizierbar sind, kann Agententechnologie die beschriebenen Erwartungen erfüllen.

Als Folge der raschen Entwicklung von Agentenmodellen und -prototypen sind entsprechend viele Agentensprachen entstanden. Im Rahmen des Arbeitskreises haben wir uns mit einigen davon beschäftigt. Diese beinhalten unter anderen:

ABLE: Agent Behaviour Language. ABLE erlaubt die Beschreibung und Spezifikation von Agentenverhalten durch atomare Verhaltensweisen, Produktionsregeln und temporale Datenbanken. Der Unterschied zu normalen Produktionensystemen besteht darin, daß Regeln parallel anwendbar, dynamisch, beliebig schachtelbar sind und Voraussetzungen von Regeln mit Zeitbedingungen versehen sein können.

April: Agent Process Interaction Language. April ist eine Sprache, die für die Entwicklung von verteilten KI Anwendungen geeignet ist. Sie stellt Kern-Elemente zur Verfügung, die für die Realisierung der meisten Agentenarchitekturen benötigt werden z.B. Multi-Tasking: Prozeß ist first-class object, mit fork Möglichkeit), Kommunikation (Netzwerk-transparente Unterstützung von Agent-to-Agent Kommunikation), Pattern-Matching- und SymbolProcessing-Fähigkeiten, begrenzte Realtime-Features (nur insoweit, daß Applikation auf externe Stimuli in Echtzeit antworten kann, da Echtzeit-Anforderung und Symbolverarbeitungsfähigkeiten oft kollidieren).

Mai2l. Mai2l stellt eine reichhaltige Menge von vordefinierten Abstraktionen zur Verfügung, insbesondere repräsentiert es Konstrukte, die in dem Agenten-Basiszyklus (Modell der Anwendung $rightarrow$ Zielaktivierung $rightarrow$ Planen $rightarrow$ Synchronisation $rightarrow$ Ausführung mit Auswirkung auf Modell der Anwendung) benötigt werden. Neben Komponenten für den Agenten-Grundzyklus stellt Mai2l Kooperationsprimitive zur Verfügung, wobei die Primitive selbst als Pläne repräsentiert sind. Aus den Kooperationsprimitiven lassen sich Kooperationsmethoden erstellen (z.B. komplexe Meta-Pläne, um domänenspezifische Multi-Agenten-Pläne zu konstruieren und auszuführen, z.B. Kooperationsmethoden wie contract net).

Mekka. Mekka ist eine Mehragentensprache und Entwicklungsumgebung zur Konstruktion kooperativer Anwendungen und besteht aus: Mai2l-Modellierungsformalismus (Agenten, Interaktionsformalismen), Bibliotheken (für vordefinierte Agenten und Kooperationsmethoden) sowie Entwicklungswerkzeugen. Beispiele für vordefinierte, generische Agenten sind Agent directory service, Monitor Agent und User-Interface-Agent.

MAGSY. Magsy ist eine Entwicklungsplattform für regelbasierte Multiagentensysteme. Der Begriff Agent steht hierbei für eine selbständig arbeitende, mit bestimmten Problemlösungsfähigkeiten ausgestattete Recheneinheit, die innerhalb ihrer Fähigkeiten über Informations- und Kontrollautonomie verfügt. In Magsy wird das Verhalten der Agenten in einem OPS5-Dialekt beschrieben. Zur Laufzeit wird jeder Agent durch einen OPS5-Interpreter (in je einem UNIX-Prozeß) dargestellt. Dieser Interpreter ist eine Reimplementierung der OPS5-Lisp-Version in C. Desweiteren ist es jedem Agenten möglich, neue Agenten zu starten und mit ihnen Kontakt aufzunehmen. Hierdurch kann sich der Aufbau des Agentennetzes dynamisch ändern.

Trotz der Vielfalt existierender Agentensprachen ist es immer noch schwierig, für ein zu realisierendes Agentenprojekt eine passende Realisierungssprache zu finden. Viele der existierenden Agentensprachen sind ursprünglich in engem Zusammenhang mit einem zu realisierenden Prototypen entstanden und sind daher auf spezielle Agentenmodelle zugeschnitten. Diese Sprachen liefern daher für Projekte mit abweichendem Agentenmodell nicht die geeignete Unterstützung. Als Alternative zu typischen Agentensprachen existieren jedoch auch Programmiersprachen, die zwar nicht explizit für Agenten geschaffen wurden, die jedoch immer mehr Elemente bereitstellen, die für die Realisierung von Agenten wesentlich sind (z.B. Java).

1.1.1.6 Fazit zur Arbeit des Arbeitskreises MAD

Als abschließendes Fazit der bisherigen Treffen ist anzumerken, daß sicherlich den einzelnen Mitgliedern durch die Diskussionen Ausmaß und Tragweite der Heterogenität der Agenten-Welt und der dadurch entstehenden Probleme noch deutlicher bewußt geworden sind.

Gerade das detaillierte Erkennen und Verstehen dieser Vielfalt hat jedoch dazu beigetragen, daß die Vertreter der verschiedenen Teilprojekte ein objektiveres Bild von Agententum gewonnen haben. Dies trägt sowohl dazu bei, die eigenen Arbeiten besser einordnen und objektiver beschreiben zu können, hilft aber auch dabei, durch das Verständnis von Agentenmodellen und Sichtweisen anderer Agenteneinsatzgebiete redundante Untersuchungsbemühungen stärker zu vermeiden und den Informationsaustausch zu erleichtern.

Literatur

[Lev89] P. Levi. Architectures of individual and distributed autonomous agents. In *Proc of the* 2^{nd} *Intelligent Autonomous Systems*, Amsterdam, 1989.

[Mae94] P. Maes. agents that reduce Work and Information Overload. *Communications of the ACM*, 37(7), 1994.

[Sho93] Y. Shoham. Agent-oriented Programming. *Artificial Intelligence*, 60(1), 1993.

[WJ95] M. Wooldridge und N. R. Jennings. Intelligent Agents: Theory and Practice. In *Knowledge Engineering Review*. 1995.

1.1.2 Arbeitskreis RivS

Der Arbeitskreis „Ressourcen in verteilten Systemen“ bietet allen Interessierten an dem Themengebiet „Programmiermodelle und Ressourcenmanagement für verteilte Systeme“

ein Diskussionsforum, um über ihre Arbeiten, als auch über Schwerpunkte und Begriffe die diesen Bereich betreffen, zu reden. In dem zurückliegenden Zeitraum wurden folgende Themen behandelt:

1.1.2.1 System- und Ressourcenbegriff

Die Begriffe „System“ und „Ressource“ sind zentral für die Teilnehmer des Arbeitskreises. Es wurde versucht, unterschiedliche Sichten auf diese Begriffe zu vereinen und Gemeinsamkeiten herauszuarbeiten.

- **Systeme und Systembeschreibung:**
  Der Begriff „verteiltes System“ war Ausgangspunkt für eine Diskussion zur Beschreibung und Begriffsbstimmung von „System“ im allgemeinen Sinn. In den verschiedenen Teilprojekten wird dieser Begriff i.a. unterschiedlich verwendet. Trotz unterschiedlicher Auffassungen, wurden zwei Komponenten als wesentliche Bestandteile identifiziert: ein System setzt sich zusammen aus

  - einer Menge autonomer Objekte *und*
  - einer Menge von Relationen über diesen Objekten

  Interpretationen über diesen beiden Mengen ermöglichen semantische Aussagen bzgl. der Systeme.

  Im weiteren Verlauf wurden verschiedene Ausprägungen und Varianten von Systemen und Systembeschreibungen andiskutiert, z.B.:

  - offene versus geschlossene Systeme
  - die Beziehung von Systemen zu ihrer Umwelt (das „Komplementärsystem“)
  - Vollständigkeit versus Handhabbarkeit von Systemen

- **Ressourcen und Ressourcenmanagement in verteilten Systemen:**
  Die Diskussion beschäftigte sich vor allem mit der Frage, von welchen Faktoren die Qualität eines Ressourcenmanagements abhängt und wie man Ressourcen und Ressourcenmanagementsysteme klassifizieren kann.

  - Verteilte Systeme bewegen sich im Gegensatz zu sequentiellen in einem globalen, schwer kontrollierbaren, Zustandsraum. Für ein optimales Ressourcenmanagement werden Informationen über die Entwicklung des gesamten Raums benötigt. Die einzelnen aktiven Komponenten eines Systems verfügen nur über einen Ausschnitt—i.a. nur lokale Informationen—der Vergangenheit. Die Qualität der Verwaltung hängt somit stark von der verwendeten Heuristik ab. Als Basis für eine Bewertung könnte die Antwortzeit der Komponenten verwendet werden.
  - Ressourcen lassen sich nach unterschiedlichen Gesichtspunkten klassifizieren. Für die Entwicklung eines allgemeinen Ressourcenmanagementmodells eignen sich Funktionalität und Struktur. Die Frage nach einer adäquaten formalen Beschreibungsmethode wurde separat behandelt (siehe Abschnitt 1.1.2.2).

  - Ressourcenmanagementsysteme lassen sich nach ihrem Abstraktionsgrad klassifizieren. Die beiden Extremfälle sind systemintegriertes und anwendungsintegriertes Management. Bei Mischsystemen besteht die Gefahr, daß sich die verwendeten Verwaltungsstrategieen gegenseitig negativ beeinflussen. Dieses Thema wurde in einer weiteren Diskussionsrunde genauer behandelt:

- **Ressourcen auf unterschiedlichsten Abstraktionsniveaus:**
  Der Begriff Ressourcen wird in jedem Teilprojekt für andere Betriebsmittel verwendet. Dies liegt an den unterschiedlichen Abstraktionsebenen, mit denen sich die einzelnen Projekte beschäftigen. Im wesentlichen können folgende Ebenen innerhalb des Graduiertenkollegs unterschieden werden:

  - **Anwendungsebene (anwendungsintegriertes Ressourcenmanagement):**
    Im Vordergrund steht eine spezielle Anwendung die effizient auf einem gegebenen System ausgeführt werden soll. Die zur Verfügung stehenden Ressourcen sind demzufolge die Betriebssystemdienste und anwendungsspezifische Objekte. Die Güte der Ressourcenverwaltung wird durch Messung der Ausführungszeiten von Funktionen oder Zugriffshäufigkeiten auf gemeinsame Objekte beschrieben.
  - **Betriebssystemebene mit Ressourcenverwaltung (systemintegriertes Ressourcenmanagement):**
    Auf der Ebene der Betriebssysteme treten die Hardwareressourcen mehr in den Vordergrund. Das Managament versucht, den Anforderungen der Anwendungen in Form von Betriebssystemdiensten zu genügen. Die für diese Dienste zur Verfügung stehenden Ressourcen sind die Hardwareobjekte. Messungen erfolgen entsprechend auf Basis von Prozessorzyklen oder Zugriffshäufigkeiten auf Speicherzellen.

1.1.2.2 Systembeschreibungen und -analysen

Zur formalen Beschreibung verteilter Systeme gibt es in der Informatik verschiedene Ansätze. Im Arbeitskreis stellten die Teilnehmer die von ihnen betrachteten Modelle vor. Dabei wurden die Vor- und Nachteile der einzelnen Ansätze sowie deren Zusammenhang diskutiert.

- **Adäquate formale Beschreibungsmethoden für Ressourcenmanagementsysteme:**
  Es wurden verschiedene Methoden diskutiert, die für die Beschreibung von Ressourcenmanagement geeignet erscheinen:

  - **Attributierte Grammatiken** eignen sich zur Beschreibung der hierarchischen Struktur von Ressourcen. Hierbei muß jede Ressource durch eine attributierte Produktion beschrieben werden. Die Produktion spezifiziert ihre Struktur, d.h. welche Ressourcen zur Realisierung benötigt werden. Der konkrete Ableitungsbaum einer Ressourceninstanz veranlaßt die Auswertung der den einzelnen Komponenten zugeordneten Attribute (z.B. Antwortzeit).

   - Die Funktionalität von Ressourcen kann durch **algebraische Spezifikationen** beschrieben werden. Hierbei zeigte sich, daß für die Bewertung einzelner Funktionalitäten eine Kopplung der algebraischen Spezifikation mit einer attributierten Grammatik sinnvoll wäre.

- **Graphgrammatiken und Ihre Anwendungsgebiete:**
  Ein Vortrag diente als Einführung in die Thematik „Graphgrammatiken und Graphersetzungssysteme". Hierzu wurde im speziellen auf die Besonderheiten im Vergleich zu Wortgrammatiken eingegangen. Anhand von Proteinen wurde erläutert, wie die syntaktische Struktur eines massiven strukturierten Systemes durch eine Graphgrammatik spezifiziert werden kann.

  Im Anschluß an den Vortrag wurde die Verwendung von Graphersetzungssystemen für die Spezifikation dynamischen Verhaltens diskutiert.

- **Ein Graph-Kalkül für verteilte Programmierung:**
  In einem Vortrag wurde ein Graph-Kalkül zur verteilten Programmierung vorgestellt. In diesem Kalkül werden verteilte Systeme durch hierarchische Graphen dargestellt, wobei die Dynamik eines verteilten Systems durch Graphtransformationen beschrieben wird. Die Eigenschaften des Kalküls wurden anhand des Beispiels der „Dining Philosophers" erläutert.

  Bei der anschließenden Diskussion ging es um die Frage, ob sich verteilte Systeme mit Hilfe des Kalküls einfach und adäquat beschreiben lassen. Außerdem wurden Effizienzgesichtspunkte bei der Implementierung diskutiert.

- **Ein Tableauverfahren zum Model-Checking von Petri-Netzen mit dem linearen μ-Kalkül:**
  Es wurde in einem Vortrag ein Tableauverfahren vorgestellt, das das Model-Checking Problem für Petri-Netze und eine Untermenge des linearen μ-Kalkül (ohne den starken nexttime-Operator) löst. Dieses Problem ist zwar auch für den vollen linearen μ-Kalkül entscheidbar, doch ist dieser Algorithmus nicht primitiv rekursiv und auch nicht als Beweisverfahren geeignet. Dagegen kann das Tableauverfahren als Beweisverfahren dienen und liefert eine gute Intuition dafür, warum die bewiesene Eigenschaft gilt.

1.1.2.3 Ressourcenmanagement: Last und Arbeitsleistung

Bei der konkreten Realisierung von Ressourcenmanagement-Verfahren gibt es zahlreiche Problemstellungen: Wie definiert man die zentralen Begriffe „Last" und „Arbeit"? Wie erhält man sinnvolle Meßgrößen? Wie sieht ein skalierbares Ressourcenmanagement-System aus? Welche Probleme ergeben sich in heterogenen Systemen?

- **Mögliche Definitionen der Begriffe „Last" und „Arbeit":**
  Ausgangspunkt für die Diskussionsrunde war die Zusammenstellung verschiedener Definitionen für die beiden Begriffe „Last" und „Arbeit". Beide Begriffe sind im Bereich der Ressourcenverwaltung (insbesondere der Lastverwaltung) von grundlegender Bedeutung. Allerdings werden diese Begriffe in der Literatur, falls überhaupt,

nur ungenau erklärt. Es stellte sich heraus, daß eine einheitliche Definition von „Last“ und „Arbeit“ aufgrund der vielfältigen Blickwinkel, unter denen man diese Begriffe betrachten kann, nur schwer zu finden ist. An dieser Stelle sei beispielsweise darauf verwiesen, daß im Hinblick auf verschiedene Problemstellungen ein zeitlicher Aspekt der beiden Begriffe nicht festgelegt werden sollte: wird die Last eines Systems durch dessen augenblickliche oder zukünftige Arbeit bestimmt? Für die Methoden der Lasterfassung haben diese beiden Ansätze unterschiedliche Auswirkungen.

- **Ressourcenmanagement bei der Performance-Evaluierung:**
  Ausgehend von der Diskussion über eine formale Definition von Ressourcen (siehe Abschnitt 1.1.2.2) wurde im Arbeitskreis versucht, herauszuarbeiten, welche **Eigenschaften von Ressourcen** für die einzelnen Teilnehmer von Interesse sind. Diese Eigenschaften spielen zum Beispiele eine zentrale Rolle für die Performance-Evaluierung von verteilten Anwendungen und für die Strategiedefinition bei der Entwicklung von Ressourcenmangementeinheiten.

  - Um den Standpunkt aus dem Blickwinkel der Performance-Analyse genauer zu betrachten, wurde ein Vortrag über die Verwendung von Spezifikationen zur **Lokalisierung von Leistungsengpässen** gehalten. In diesem Vortrag wurde dargestellt, daß nur solche Eigenschaften von Ressourcen eines verteilten Systems für die Performance-Evaluierung einer verteilten Anwendung interessant sind, die der Benutzer auch tatsächlich beeinflussen kann. Hierbei ergab sich in der späteren Diskussion ein Unterschied zu den Eigenschaften, die für Ressourcenmanagementstrategien interesssant sind. Ähnliche Ergebnisse ergaben sich auch später in der Diskussion zu einem Vortrag über die **Definition von Last** (siehe weiter oben). So ist es für die Perfomance-Analyse ausreichend, die Meßgrößen so zu verwenden, wie sie gemessen wurde, während bei den Strategien von Ressourcenmangementsystemen immer noch abgeschätzt werden muß, inwieweit die gemessenen Größen Extrapolationen für die Zukunft darstellen.
  - Des weiteren wurde in einem Vortrag eine **Definition von atomaren Meßgrößen** vorgestellt, die einen Bezug zwischen der Verwendung der Ressource und dem gerade vorhandenen Zustand der Anwendung herstellt. Dieser Bezug zwischen Verwendung der Ressource und dem Programmcode ist nötig, um eine Lokaliserung der Ursachen von Leistungsengpässen durchführen zu können.

    Außerdem wurde eine Möglichkeit vorgestellt, basierend auf **Zustands-Aktions-Netzen** und Ablaufzeiten für einzelne Aktionen nicht nur den ursprünglichen Programmablauf sondern eine Klasse von Programmabläufen bezüglich ihrer Performance-Eigenschaften zu untersuchen. Dabei spielt die Definition von Interleavings von Zustands-Aktions-Netzen bzgl. einer bestimmten Konfiguration von Ressourcen eine zentrale Rolle. Im Anschluß an den Vortrag wurde diskutiert, inwieweit die vorgeschlagene Bildung von Interleavings die tatsächlichen Abläufe auf realen Architekturen repräsentiern könnten.

- **Neue und unkonventionelle Ansätze zur dynamischen Lastverteilung:**
  In dieser Diskussion wurden die **Probleme konventioneller Lastverteilungssy-**

steme in Workstationnetzen und Multiprozessorsystemen erörtert und exemplarisch in einem Vortrag neuartige Lösungsansätze vorgestellt und anschließend bewertet. Ausgangspunkt waren dabei die **Anforderungen**, die heutzutage und in Zukunft von den Programmierern und Benutzern verteilter und paralleler Anwendungen an ein Lastverteilungssystem gestellt werden:

- Ein Lastverteilungssystem sollte so organisiert sein, daß seine Leistung und sein Verwaltungsaufwand nur graduell mit der Größe (Rechnerzahl und topologischer Umfang) des verteilten Systems und der Dynamik des Lastaufkommens abnimmt. Die konventionellen Lastverteilungssysteme sind meistens zentral oder dezentral (und mit lokaler Sicht) organisiert und nur eingeschränkt skalierbar.
- Ein Lastverteilungssystem sollte die Heterogenität des verteilten Systems berücksichtigen, in Workstationnetzen beispielsweise die verschiedenen Architekturen, die Rechnerleistung etc. Noch beschränken sich viele Ansätze auf homogene Rechnerknoten.

Als Maßnahmen, diesen Anforderungen gerecht zu werden, wurden zwei neuartige Ansätze vorgestellt:

- Hierarchisch organisierte **Makler** zur Aggregation von Lastinformation und zur Lastverteilung
- Einführung eines **Preismechanismus** für die Ressourcennutzung

Ein Vortrag stellte dann exemplarisch ein Lastverteilungssystem vor, das diese Ansätze durch eine „Computational Economy“ realisiert. Es besteht aus **Diensterbringern** (Makler und Rechner) und **Konsumenten** (Rechenaufträge). Dienste und Ressourcen legen einen ihrer Fähigkeiten und ihrer Nachfrage seitens der Konsumenten entsprechenden Nutzungspreis fest. Dies erlaubt dann den Konsumenten, diejenigen Ressourcen auszuwählen, die ihnen gemäß ihrem Budget und ihrer Präferenzen am wichtigsten erscheinen. Die Erörterung dieser neuen Ansätze stellte auch den Bezug der Lastverteilung in verteilten Systemen mit der Problematik der Ressourcenallokation her, wie sie Gegenstand der mikroökonomischen Theorie in den Wirschaftswissenschaften ist.

- **Entwicklung paralleler Anwendungen auf gekoppelten Arbeitsplatzrechnern:**
  Zu Beginn des Vortrags wurden die grundlegenden Probleme erörtert, die der Entwickler einer parallelen Anwendung auf dieser speziellen Hardwareplattform zu berücksichtigen hat: einer insgesamt **hohen Rechenleistung** steht eine vergleichsweise **geringe Kommunikationsleistung** gegenüber. Als Konsequenz ergibt sich für den Anwendungsentwickler die Vermeidung häufig auftretender Kommunikation mit geringer Datenmenge, sowie die Vermeidung häufiger globaler Synchronisationsoperationen.

  Im zweiten Teil des Vortrags wurde ein Fallbeispiel für eine nachrichtenorientierte, parallele Programmierbibliothek vorgestellt: **PVM** (Parallel Virtual Machine). Anhand von PVM wurden die verschiedenen Möglichkeiten der Kommunikation (blockierend—nicht blockierend), der Synchronisation und der Prozeßsteuerung erläutert. Aufbauend auf dem Vortrag wurden die angesprochenen Punkte rege diskutiert.

1.1.3 Querschnittsthemen

1.1.3.1 Mathematische Methoden

Zur formalen Beschreibung von Multiagentensystemen (MAS) werden seit kurzem verstärkt mathematische Methoden eingesetzt [Woo96]. Der Grund dafür liegt in der Ausdruckstärke, der konzeptionellen Klarheit sowie der Verfügbarkeit einer wohlfundierten Theorie mathematischer Modelle. Mathematische Methoden werden sowohl auf Mikro- als auch Makroebene eingesetzt [Sho92]. Hierbei beschäftigen sich Mikrotheorien mit Eigenschaften und Fähigkeiten des einzelnen Agenten, während sich Makrotheorien mit der Organisation des Kooperationsverhaltens des gesamten Multiagentensystems auseinandersetzen.

In diesem Zusammenhang konzentriert sich die Kooperation beider Kollegiaten auf die Integration der Mikro- und Makroperspektive. Obwohl bereits zahlreiche Arbeiten zu den einzelnen Bereichen existieren, wurden bisher noch wenige Versuche unternommen, Ergebnisse aus den beiden besagten Perspektiven zu integrieren. Eine derartige Integration ist zur Erstellung einer allgemeinen, umfassenden Theorie für MAS jedoch unverzichtbar. Hierbei ist wiederum das Konzept der Faserbündel besonders vielversprechend, da es einen natürlichen Mechanismus zum Wechsel zwischen lokaler und globaler Perspektive bereitstellt.

Teilprojektthema „Computational Economies, Wahrscheinlichkeitsdichten und Neuronale Netze zur Entscheidungsmodellierung in Multiagentensystemen“

Entsprechend dieser Aufteilung befaßt sich dieses Thema mit der mathematischen Charakerisierung eines einzelnen Agenten auf Mikroebene. Speziell das wichtige Lernproblem läßt sich in geeigneter Weise durch statistische Methoden analysieren [Whi89]. Darüberhinaus können statistische Schätzverfahren verwendet werden, um die Lernfähigkeit der Agenten in realen Systemen zu implementieren.

Teilprojektthema „Ein logikbasierter Ansatz zur Modellierung kooperierender Agenten“

Dieses Thema beschäftigt sich dagegen mit der formalen Modellierung von Agentengesellschaften auf Makroebene. Ein wichtiger Gesichtspunkt dabei betrifft die Modellierung von Kooperations- und Interaktionsbeziehungen zwischen den einzelnen Agenten und ihre Auswirkungen auf das globale Systemverhalten. Es wird untersucht, inwieweit das mathematische Konzept der Faserbündel ([Pfa91]) zu diesem Zweck benutzt werden kann beziehungsweise hinsichtlich der Anwendung auf Multi-Agentenysteme adaptiert werden muß.

1.1.3.2 Remote Code

Das Agentenparadigma hat neue Dimensionen fuer das Client / Server Programming Modell gebracht. Dabei entstand das Konzept der mobilen Agenten, das die Nachteile von Remote Procedure Call (RPC) kompensieren sollte.

Bei RPC betrachtet man Computer- zu Computer-Kommunikation als die Moeglichkeit eines Computers bei einem anderen Prozeduren aufzurufen. Jede Nachricht, die vom Netz transportiert wird ist entweder ein Request oder ein Acknowledgement fuer eine Prozedur. Ein Request enthält Daten, die der aufgerufenen Prozedur als Argumente übergeben werden, die Antwort beinhaltet Ergebnisse. Ein Client Computer, der einen Auftrag an einen Server geben möchte, tut dies mit einer Serie von RPCs.

Die alternative Technologie, die die Agenten anbieten, ist das sogenannte „remote programming“ Paradigma. Unter remote programming versteht man bei der Computer-Computer Kommunikation die Möglichkeit einen Computer in der Lage zu versetzen, auf dem anderen nicht nur eine Prozedur aufzurufen, sondern auch noch den Code für die auszuführende Prozedur mitzuliefern. Das bedeutet für das Konzept von mobilen Agenten, daß ein Client-Computer aus einer Applikations-Umgebung besteht, die eine oder mehrere Anwendungen zur Interaktion mit einer Remote Server enthält. Mobile Agenten sollten also als eine Erweiterung von bereits wohlbekannten Methoden gesehen werden wie:

- remote dispatch von script programs
- remote submission von batch jobs

Vorteile dieses Verfahrens gegenüber RPC sind bessere Performance und die individuelle Anpassungsmöglichkeit. Wenn ein Client Rechner Arbeit für einen Server hat, schickt er einen Agenten zum Server und delegiert dadurch die Aufgabe lokal zum Server. Dadurch reduziert sich die Netzlast durch weniger versandte Messages. Agenten ermöglichen es Herstellern von Clientsoftware die Funktionalität von anderen Herstellern angebotener Software zu erweitern.

Teilprojekt „Kooperierende Agenten im Netz- und Systemmanagement“

In diesem Teilprojekt wird das „remote programming“ Paradigma verwendet, um Verteilung und Flexibilität bei der Realisierung von komplexen Aufgaben im Bereich von Netz- und Systemmanagement zu ermöglichen. Das wird durch die dynamische Delegierung von Managementfunktionalität auf die verteilten Managementagenten erreicht. Die delegierte Funktionalität ist in diesem Sinne eine Art mobiler Agent. Ein Agentenprogramm könnte aus prozeduralen Komponenten oder aus Klassen von Objekten bestehen. Es könnte sowohl in Maschinensprache oder in einer interpretierten (virtuellen Maschine) Sprache geschrieben sein. Um die heterogene Umgebungen besser zu unterstützen, sind in interpretierten Sprachen geschriebene Agenten vorzuziehen. Der Performancenachteil, der aus dieser Methode resultiert ist tolerierbar. Interpretierte Sprachen haben auch den Vorteil des „latebinding“. Das ermöglicht es dem Agenten Referenzen zu Funktionen oder Klassen zu enthalten, die auf dem System, wo er gestartet wird, überhaupt nicht vorhanden sind, sondern erst auf seinem Zielsystem. Auch der Sicherheitsaspekt ist mit interpretierten

Sprachen leichter in den Griff zu bekommen, da der Sprachenentwickler explizite Kontrolle welche Systemressourcen er im Interpreter zur Verfügung stellt.

Bei Agentensprachen ist Plattformunabhängigkeit eine sehr starke Anforderung. Mechanismen, die die Migration von mobilen Agenten ermöglichen, sind von Sprachen wie Telescript oder Java angeboten. Bei Telescript wird ein Teil oder das gesamte Program vom Client auf den Server übertragen, und alle Funktionsaufrufe finden nun lokal statt. Bei Java wird ein Clientprogramm zum Server geschickt, sondern der Server sendet einen Service Agenten an den Client. Der Client kommuniziert nun mit diesem Agenten, und erhält die Informationen von ihm. Diese Sprachen werden auf ihre Tauglichkeit für die Delegierung von Netz- und Systemmanagementfunktionen hin untersucht.

Teilprojekt „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen“

1.1.3.3 Vermittlungskonzepte

Vermittlungsmechanismen spielen in verteilten Systeme eine wichtige Rolle, da sie im Gegensatz zu statischen Zuweisungen die Dynamik des Systems und der Anwendungen berücksichtigen. Die Vermittlung kann sich, wie im Teilprojekt „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen“, auf die Zuweisung und Reservierung von physikalischen Ressourcen beziehen, oder, wie in den beiden Teilprojekten „Verteilte numerische Algorithmen auf Bäumen“ und „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“, auf das Zusammenbringen von Anbietern und Nutzern von im verteilten System zur Verfügung gestellten Diensten.

In unseren Teilprojekten liegt der Schwerpunkt auf der indirekten Vermittlung mittels einer dedizierten Instanz. Diese kann von mehreren Anwendungen genutzt werden, so daß diese also nicht selbst Vermittlungsfunktionen wahrnehmen müssen. Zum anderen sprechen Transparenz-, Wartungs- und Sicherheitsgründe für die indirekte Lösung.

Im Rahmen der Vermittlung kommen der Vermittlungsinstanz verschiedene Aufgaben zu. Zum einen muß sie Kenntnis über die im verteilten System verfügbaren Ressourcen bzw. Dienste besitzen. Dies erfordert geeignete Mechanismen für das Leistungs- und Fehlermanagement, vorallem zur Registrierung und der Aktualisierung des gehaltenen Zusandsdatenbestands. Zum anderen muß die Vermittlungsinstanz bei der Zuweisung einen von möglicherweise vielen potentiellen Kanditaten mit Hilfe geeigneter Bewertungs-, Auswahl- und Optimierungsstrategien selektieren. Dieser Vorgang erfordert geeignete Beschreibungsmittel (z.B Ressourcen- bzw. Dienstattribute und Funktionalitäten).

Die Selektion resultiert schließlich in der Benachrichtigung einer oder beider Seiten über ihre Zuteilung. Hierbei kommen der Vermittlungsinstanz ggf. noch Übersetzung- und Weiterleitungssaufgaben zu, z.B. das Aufbereiten und Formatieren der Anfrage bei automatischer Weiterleitung an den zugeteilten Dienst.

Neben der Untersuchung von geeigneten Lösungen für die obigen Aufgaben steht in den Teilprojekten auch die Wahl einer geeigneten Organisationsstruktur der Vermittlungsinstanz im Vordergrund.

Teilprojekt „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen“

Die derzeitigen Deregulierungsbestrebungen lassen deutlich die beginnende Umgestaltung des Telekommunikationsmarktes nach marktwirtschaftlichen Gesichtspunkten erkennen. Dienstzugangsforderungen privater Netzbetreiber, sowie wirtschaftswissenschaftliche Ansätze für eine effizientere Tarifierung, weisen auf die Notwendigkeit weiterer Veränderungen hin. Der Zielzustand ist idealerweise der offene Telekommunikationsmarkt, der durch die Beteiligung mehrerer Netzbetreiber an einem Kommunikationsdienst, eine nachfrageorientierte Tarifierung und die freie Netzbetreiberwahl durch den Teilnehmer gekennzeichnet ist.

Der Initiator eines Kommunikationsdienstes hat auf dem offenen TK-Markt die Möglichkeit, die beteiligten Netzbetreiber frei auszuwählen. Dies erfordert vom Initiator die Kenntnis von Preis und Qualität potentieller Ressourcen diverser Netzbetreiber. Angesichts der Vielzahl auszuwertender Daten benötigt er hierzu umfangreiche Unterstützung durch das Signalisierungssystem. Ausgehend von einer modernen Signalisierungsarchitektur für Multimedia-Dienste mit getrennter Verbindungs-, Ressourcen- und Rufsteuerung wird in Teilprojekt eine agentenbasierte Architektur für die Ressourcensteuerung entworfen. Als Voruntersuchung werden verschiedene Methoden der Ressourcenerfassung und Ressourcenselektion betrachtet. Um den strengen Anforderungen an die Rufaufbauzeit von Multimedia-Diensten gerecht zu werden, bildet eine von der Ressourcenselektion getrennte, autonome Ressourcenerfassung eine Basisfunktionalität der Ressourcensteuerung. Die Ressourcenselektion erfolgt durch ein verteiltes System aus organisierten, kooperierenden Agenten. Die Organisation kann zentral oder dezentral strukturiert sein. Aus Gründen der Verwaltbarkeit der Ressourcen, der Skalierbarkeit der Architektur und der Genauigkeit der ermittelten Ressourcenkonfigurationen eignen sich vorallem Agentenhierarchien.

Teilprojekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“

In diesem Teilprojekt wird ein Vermittlungskonzept für WWW-basierte Informationsdienste und Orientierungshilfen entwickelt. Die hier untersuchenten Besonderheiten im Hinblick auf die Verwendbarkeit von existierenden Vermittlungskonzepten anderer Anwendungsgebiete sind folgende: Die Eigenschaften, die den Nachfragewunsch des Benutzers beschreiben, sind nicht standardisiert und beziehen sich oft auf inhaltliche Eigenschaften der Information. Zudem besitzen die derzeit angebotenen, nutzbaren Informationsdienste keine einheitlichen Anfrageschnittstellen nach außen, oder gar standardisierte Leistungsbeschreibungen. Auch die Bewertung der erbrachten Informationsdienste ist nicht objektiv möglich, da viele Bewertungskriterien dem subjektiven Ermessen des Benutzers unterliegen und nur wenige seiner Bewertungskriterien objektivierbar sind. Diese heterogene Vermittlungssituation wird zusätzlich dadurch erschwert, daß die Menge Anbieter und die Eigenschaften des Angebotes sehr dynamisch sind (d.h. es können jederzeit neue Anbieter auftreten oder existierende wegfallen, wobei die angebotenen Informationsdienste selbst ebenfalls kontinuierlichen Änderungen unterworfen sind).

Teilprojekt „Verteilte numerische Algorithmen auf Bäumen“

Dieses Teilprojekt untersucht und entwickelt Vermittlungskonzepte, die sich auf einen offenen Markt von Diensten beziehen. Hierbei stehen als vorallem Rechendienste, wie numerische SPMD-Applikationen, im Vordergrund, die sich durch ihre Qualität und Funktionalität kennzeichnen lassen. Ausgangspunkt ist, wie auch im Teilprojekt „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen“, die Schaffung eines Szenarios nach marktwirtschaftlichen Gesichtspunkten. Hierzu gehört die Einführung eins Preissystems, nach dem Diensterbringer ihre angebotenen Dienste geeignet (dynamisch nach Angebot und Nachfrage) tarifieren und somit entsprechende Anreizmöglichkeiten bieten können. Dies ist Grundlage eines Verhandlungsmechanismuses für Dienstnutzungsvertäge, der die bindenden Bedingungen an die Dienstqualität und weitere Modalitäten spezifiziert.

Einer Vermittlungsinstanz kommt hierbei eine Optimierungsaufgabe zu. Sie muß für eine Anfrage einen (oder mehrere) Dienstanbieter finden, so daß dabei die Anforderungen, Präferenzen und Spezifikationen beider Seiten möglichst optimal berücksichtigt sind. Desweiteren muß sie im Anschluß daran die Aushandlung des Dienstnutzungsvertrages initieren. Von Interesse ist hierbei einerseits, wie sich die Vermittlungsinstanz im verteilten System organisiert, damit sie bezüglich Effizienz und Komplexität mit der Größe des Systems (und der Zahl darin angebotenen Dienste) skaliert. Zweitens sollten Dienstanbieter und -nutzer ihre Angaben in einer unabhängigen, flexiblen Spezifikationssprache formulieren und dem Vermittler mitteilen können. Hierbei gilt es zu berücksichtigen, daß viele Angaben zeitlich variieren können (z.B. der Nutzungspreis).

In einem offenen Markt von Diensten ist auch die Vermittlung nur ein (Meta-)Dienst. Neben den Maklern sollten auch evolutionär Dienstkomplexe entstehen, zum Beispiel Auktionen und Delegatoren bzw. Rekrutierer.

1.1.3.4 Modelle für Agentenkooperation und Organisationsstrukturen von Agentengesellschaften

Eine wesentliche Eigenschaft von Agenten im Anwendungsgebiet Verteilte Systeme ist die Fähigkeit zur Kooperation. Als Kooperationspartner können hierbei je nach Anwendungsgebiet sowohl andere Agenten, als auch menschliche Benutzer sowie 'Nicht-Agenten-Systeme' in der Umgebung von Agenten auftreten. Eng damit in Verbindung stehend ist die Problematik der für die angestrebte Kooperation geeigneten Organisationsstruktur der Agenten.

Teilprojekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“

In diesem Teilprojekt werden kooperierende Agenten zur Unterstützung von Informationsaustausch eingesetzt. Hierbei wird vor allem die Kooperation auf der Basis strukturierter Kommunikation betrachtet. Darauf aufbauend können konversationsorientierte Kooperationsmechanismen modelliert werden, wobei für den Bereich des Informationsaustausches komplexe Verhandlungsmechanismen keinen Schwerpunkt darstellen, zumal diese speziell im Teilprojekt „Kooperierende Agenten und autonome Robotersysteme“ untersucht werden. Wichtig sind ferner Koordinations- und Synchronisationsmechanismen, wobei im

Teilprojekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“ Koordination auf der Basis von organisatorischen Strukturen innerhalb der Agentengemeinschaft (z.B. Rollen) aber auch durch Verhaltensvorgaben für eingehende Nachrichten und Events erfolgt. Konflikte werden weitgehend durch die Rollenmodellierung vermieden. Dies wird zudem durch die Tatsache begünstigt, daß es sich bei den für Informationsaustausch benötigten Ressourcen oftmals um keine beschränkten sondern um vervielfältigbare Ressourcen handelt. Als Alternative zur Konfliktvermeidung wären Konfliktauflösungsstrategien möglich. Zu dieser Thematik existieren Untersuchungen in der von F. Fuchs bearbeiteten Thematik der Konfliktlösung in Konkurrenzszenarien.

Teilprojekt „Kooperierende Agenten und autonome Robotersysteme“

Hier bildet die Untersuchung von Organisationsstrukturen für Multi-Agenten Systeme und den damit in Verbindung stehenden Thematiken der Koordination und Synchronisation einen Schwerpunkt. Insbesondere werden hierbei die Kooperationsmöglichkeiten unter Zugrundelegung einer Konkurrenzsituation unter den Agenten untersucht. Dabei auftretende Konflikte zwischen den Agenten werden über Verhandlungen gelöst, in denen diese bestimmte Rollen übernehmen — ein Konzept, das in ähnlicher Weise im Projekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“ von M. Nöhmeier eingesetzt wird. Die Basis für entsprechende Konfliktlösungsstrategien bildet dabei ein marktorientierter Ansatz (vgl. hierzu auch die von M. Backschat bearbeitete Thematik des agentenbasierten, wirtschaftlich orientierten Lastverteilungs- und Ressourcen-Managementsystems für hierarchisch strukturierte numerische Algorithmen in verteilten Systemen).

Teilprojekt „Bildverstehen in verteilten Systemen“

In diesem Teilprojekt werden u.a. Möglichkeiten untersucht, kooperierende Agenten zur Planung und Steuerung paralleler Bildanalyse einzusetzen. Da hierbei speziell hinsichtlich der Bearbeitungszeiten sehr strenge Einschränkungen bestehen, wird eine weitgehend konfliktfreie Zusammenarbeit innerhalb einer flach struktuierten Agentengemeinschaft angestrebt. Hinsichtlich der Untersuchung von Kooperationsformen und Verhandlungsstrategien ergeben sich Verknüpfungspunkte zum Teilprojekt „Kooperierende Agenten und autonome Robotersysteme“.

Teilprojekt „Kooperierende Agenten im Netz- und Systemmanagement“

In diesem Teilprojekt wird der Einsatz von *flexiblen Agenten* vorgeschlagen, die gemeinsam verteilte Aufgaben im Bereich von Netz- und Systemmanagement erledigen. Hauptmerkmale dieser Agenten sind ihre dynamische Erweiterbarkeit bezueglich ihrer Funktionalität und die dadurch notwendig werdende Kooperation. In diesem Rahmen werden verschiedene Organisationsstrukturen und Kooperationsmechanismen für Agentensystemen untersucht und deren Anwendbarkeit für das verteilte Netz- und Systemmanagement bewertet. Ein wichtiger Gesichtspunkt besteht darin, Gruppen von flexiblen Agenten zu bilden, die einerseits die Abwicklung der Domänenbezogenen Aufgaben möglichst lokal halten. Andererseits sollte die Abwicklung von Domänenübergreifenden Aufgaben durch Kooperation zwischen sogenannten Gruppenleiter stattfinden. Die Lösung von dabei entstehenden Zielkonflikten zwischen Agenten ergibt Verknüpfungspunkte zum Teilprojekt „Kooperierende Agenten und autonome Robotersysteme“und „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“.

Desweiteren ergeben sich natürlicherweise noch Verknüpfungspunkte zum logikbasierten Ansatz zur Modellierung von kooperierenden Agenten, der von A. Bücherl im Rahmen des Teilprojektes „Kooperierende Agenten in verteilte Systemen – Grundlagen“ verfolgt wird.

1.1.3.5 Parallele Bildanalyseanwendungen

Digitale Bildanalyse stellt hohe Anforderungen an das verarbeitende System bzgl. Ressourcen (insbesondere Speicherplatz) und Rechengeschwindigkeit. Dies resultiert einerseits aus der großen Menge an Daten, die es zu verarbeiten gilt und andererseits aus den strengen Zeitvorgaben, die für zahlreiche Anwendungsgebiete — wie z.B. in der Robotik oder auch im Bereich der Medizin — gelten. Um diesen Anforderungen gerecht zu werden, kommen in der Bildanalyse vermehrt verteilte Systeme bzw. Verfahren zum Einsatz.

Das Teilprojekt „Bildverstehen in verteilten Systemen“ untersucht verschiedene Fragestellungen im Zusammenhang mit dieser Thematik. Es gliedert sich in die zwei Einzelprojekte „Verteilte Generierung von Objekterkennungsprogrammen“ und „Konzepte zur agentenbasierten Parallelisierung in der Bildanalyse“, die in enger Beziehung zueinander stehen.

Hier wird die automatischen Generierung von Bildanalyseseprogrammen zur Objekterkennung untersucht. Während deren Erzeugung innerhalb eines verteilten Systems geschieht, sind die Programme selbst sequentiell. Um sie automatisch zu parallelisieren, können die Werkzeuge und Methoden des zweiten Projektes eingesetzt werden. Durch diese Verzahnung ergeben sich viele Punkte der Zusammenarbeit, die z.B. den geforderten Leistungsumfang der Programmparallelisierung — also die verwendeten Methoden der Parallelverarbeitung und die dadurch erzielte Verarbeitungsgeschwindigkeit — betreffen. In beiden Projekten findet das Bildanalysesystem HORUS (entwickelt am Lehrstuhl für Informatik IX) als Implementierungsgrundlage Verwendung. Die generierten Objekterkennungsprogramme verwenden HORUS-spezifische Bildanalyse-Operatoren. Die Planung und Durchführung einer parallelen Bearbeitung dieser Operatoren geschieht mit Hilfe einer Erweiterung des HORUS-Systems um ein Multiagentensystem. Diese Erweiterung wird innerhalb des zweiten Einzelprojekts implementiert, wobei die Anforderungen der Anwendungsseite (insbesondere die des ersten Einzelprojekts) miteinfließen. Dadurch ergeben sich auch auf der Implementierungsebene Verknüpfungspunkte zwischen den zwei Einzelprojekten.

1.1.3.6 Spezifikation und Programmiersprachen

Spezifikationssprachen werden in der gesamten Informatik zur Beschreibung von Modellen verwendet [vL90]. Mit einer bestimmten Spezifikationssprache kann man eine ganze *Klasse von Modellen* beschreiben, während im Unterschied dazu eine Spezifikation ein konkretes *Modell* beschreibt.

Zentrale Bestandteile einer Spezifikationssprache sind *Syntax* und *Semantik* [Gun92], so daß eine formale Beschreibung von Modellen möglich ist.

Bei der Spezifikation von *Verteilten Systemen* ergeben sich zusätzliche Aspekte, wenn es darum geht, Nebenläufigkeit adäquat darzustellen und Unterstützung für die Beschreibung von Synchronisation, Kommunikation und anderen Konzepten bereitzustellen (siehe auch [Mil89, AWY93]).

Eine *Programmiersprache* ist eine Spezifikationssprache mit der man *ausführbare* Spezifikationen beschreiben kann.

Im Teilgebiet Spezifikation befassen sich die Teilbereiche „Objektorientierte Spezifikation verteilter Systeme" und „Agentengestützter Informationsaustausch in globalen Informationsräumen" mit *Requirements Engineering*, d.h. der Umsetzung der Benutzeranforderungen in eine Spezifikation.

Teilprojektthema „Objektorientierte Spezifikation verteilter Systeme"

In diesem Teilbereich des Teilprojektes „Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme" wird eine Entwurfsmethodik für verteilte objekt-orientierte Programmierung entwickelt. Es soll eine Toolunterstützung für die Umsetzung von Analyse zu Design bereitgestellt werden, die insbesondere die Beschreibung von Begriffen erlaubt, die zentral für verteilte Systeme sind, wie z.B. gegenseitiger Ausschluß und Synchronisation.

Teilprojekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit"

Dieses Teilprojekt beschäftigt sich damit, für eine Klasse von CSCW-Systemen aussagekräftige Modellierungskomponenten zur Beschreibung von (a)synchron nebenläufig arbeitenden Systemeinheiten zur Verfügung zu stellen. Diese sollen den Übergang zwischen der Beschreibung der Benutzeranforderungen bzw. der darauf abgestimmten Unterstützungsstrategie und dem konkret dafür entwickelten technischen CSCW-System erleichtern.

Teilprojektthema „Ein Kalkül für verteilte Programmierung, basierend auf Graph-Ersetzung"

Zur Realisierung von Spezifikationen verteilter Systeme benötigt man verteilte Programmiersprachen. Mit SPIDER wurde in diesem Teilbereich des Teilprojekts „Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für die verteilte Programmierung" eine einfache graphische Programmiersprache für verteilte Systeme entwickelt. Mit SPIDER lassen sich für Programmiersprachen wichtige Konzepte, wie z.B. Typ-Systeme, Verhaltensäquivalenz von Programmen, etc. betrachten.

Teilprojektthema „Verteilte attributierte Graphersetzung"

Graphersetzungssyteme können zur formalen Beschreibung der Semantik strukturierter dynamischer Systeme, z.B. graphischer Programmiersprachen, verwendet werden. Hierzu wird die operationelle Semantik des Systems durch eine Graphgrammatik spezifiziert. Dieser Teilbereich des Teilprojekts „Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für die verteilte Programmierung“ beschäftigt sich mit der verteilten Realisierung einer Plattform zur Implementierung derartiger Spezifikationen. Hierbei wird im besonderen die Nutzung des implizit in der Spezifikation enthaltenen Parallelismus untersucht.

Ist ein Programm geschrieben, so ist es wünschenswert, es zu verifizieren, d.h. die Übereinstimmung mit der ursprünglichen Spezifikation zu überprüfen. Mit dieser Fragestellung beschäftigen sich die Teilbereiche „Modellprüfung paralleler und verteilter Programmabläufe“ und „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen“.

Teilprojektthema „Modellprüfung paralleler und verteilter Programmabläufe“

Der Teilbereich „Modellprüfung paralleler und verteilter Programmabläufe“ beschäftigt sich mit der Entwicklung einer Modellklasse zur Beschreibung von Programmabläufen im Hinblick auf die Lokalisierung von Fehlern und Leistungsengpässen. Ein Modell dieser Klasse soll dabei mehrere Programmabläufe beschreiben obwohl es aus einem einzigen Programmablauf generiert wird. Ein weiterer Schwerpunkt dieses Teilbereichs liegt in der Entwicklung einer Spezifikationssprache basierend auf temporaler Logik. Diese Sprache soll die Definition von Eigenschaften ermöglichen, die von fehlerhaften bzw. ineffizienten Programmabläufen nicht erfüllt werden und damit Fehler bzw. Leistungsengpässe lokalisieren.

Teilprojektthema „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen“

Dieser Teilbereich des Teilprojekts „Verteilte Algorithmen – Spezifikation, Modellierung, Korrekheit“ beschäftigt sich mit der Entscheidbarkeit und Komplexität der Verifikation. Dazu werden verschiedene Modellklassen untersucht, die zwar unendliche, aber nicht Turing-mächtige Modelle beschreiben. Je mächtiger diese Modelle sind, und je ausdrucksstärker die verwendete Temporallogik ist, desto schwieriger ist dann im allgemeinen die Verifikation.

1.1.3.7 Verifikation

Unter Verifikation versteht man das Zeigen der Korrektheit eines Programms bezüglich einer formalen Beschreibung des Problems, das das Programm lösen soll. Ein Ziel dabei ist es, die Verifikation zu automatisieren.

Model Checking wird dabei als ein Teilgebiet der Verifikation betrachtet, das als Sprache zur formalen Spezifikation temporale Logik verwendet. Das Ziel ist dabei, automatisch zu überprüfen ob ein gegebenes Programm Modell einer temporallogischen Spezifikation ist. Ein Schwerpunkt der Zusammenarbeit liegt im Bereich Model Checking.

Beispiele für Methoden zur Verifikation finden sich u.a. in [Mil89] und [Eme90]. Grundlegende theoretische Betrachtungen finden sich in [Jan94] und [Esp96].

Teilprojektthema „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen"

Die Zusammenarbeit besteht darin, daß im Teilbereich „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen" theoretische Grundlagen der Verifikation bearbeitet werden, wie beispielsweise Entscheidbarkeit und Komplexität des Verifikationsproblems für unterschiedliche Beschreibungstechniken und unterschiedliche Arten von Systemen.

Teilprojektthema „Modellprüfung paralleler und verteilter Programmabläufe"

Dieser Teilbereich untersucht Verifikationsmethoden zur Überprüfung von Programmabläufen. Solche Methoden können beim Testen und Debugging paralleler und verteilter Systeme eingesetzt werden. Des weiteren wird in diesem Teilbereich versucht formale Beschreibungstechniken zu finden, die es ermöglichen Eigenschaften die beim Testen und Debugging wichtig erscheinen zu formulieren.

Teilprojektthema „Objektorientierte Spezifikation verteilter Systeme"

In diesem Teilbereich M. Podolskyuntersucht, welche formalen Methoden nötig sind, um eine von der Analyse bis zum Testen durchgängige Entwicklungsumgebung zu entwickeln. Dabei stellt unter anderem auch die Glaubwürdigkeit der resultierenden Implementierungen einen wichtigen Punkt dar. Zu diesem Zweck wird in diesem Teilbereich untersucht, welche Anforderungen an Verifikationsmethoden zu stellen sind und inwieweit bereits existierende Verifikationsmethoden diesen Anforderungen entsprechen beziehungsweise erweitert werden müssen. Insofern ergibt sich hieraus eine Rückkopplung zu den Teilbereichen „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen" und „Modellprüfung paralleler und verteilter Programmabläufe".

Teilprojektthema „Ein Kalkül für verteilte Programmierung, basierend auf Graph-Ersetzung"

Dieser Teilbereich untersucht unterschiedliche Kalküle zur verteilten Programmierung. Eine Aufgabenstellung dabei ist die Übersetzung dieser Kalküle ineinander. Diese Übersetzung muß semantisch korrekt sein. Dazu müssen geeignete Verhaltensäquivalenzen auf den Ausdrücken der jeweiligen Kalküle gefunden werden. Ein Querbezug ergibt sich dabei zu Bisimulationsäquivalenzen wie sie auch in der Verifikation verteilter Systeme verwendet werden. Somit können theoretische Ergebnisse über Verhaltensäquivalenzen verteilter Systeme aus dem Teilbereich „Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen" verwendet werden.

Teilprojektthema „Steuerung kooperativer paralleler Theorembeweiser unter Ausnutzung von Ähnlichkeit"

In diesem Teilbereich geht es darum, die Leistungsfähigkeit vorhandener Theorembeweiser durch Parallelisierungsansätze zu steigern. Theorembeweiser lassen sich im Rahmen eines

Entwicklungsprozesses eines Programms insbesondere zur Erstellung einer formalen Spezifikation verwenden. So ist ein Einsatz zur Verifikation von Verfeinerungsvorgängen einer Spezifikation (zeige daß wichtige Safety- und Livenesseigenschaften erfüllt bleiben) möglich. Beziehungen bestehen insbesondere zum Teilbereich „Objektorientierte Spezifikation verteilter Systeme“, indem untersucht wird, inwieweit sich Theorembeweiser als Hilfsmittel zur Erstellung formaler Spezifikationen einsetzen lassen.

1.1.3.8 Lastmanagement

Ressourcenmanagement stellt die Funktionalitäten und Mechanismen zur Verfügung, um die Ressourcen eines Systems entsprechend den anstehenden Aufgaben zu verwalten [Gos91]. Vorrangiges Ziel ist es, einen funktionsfähigen Zustand des Systems aufrecht zu erhalten. Zu den Mechanismen zählt man beispielsweise die Zuordnung von Ressourcen an die Aufgaben und deren Freigabe oder die Vermeidung von Verklemmungen in Konkurrenzszenarien.

Typischerweise stehen in einem verteilten System mehrere Ressourcen zur Verfügung, die zur Erledigung einer Aufgabe zwar geeignet, aber nicht alle notwendig sind. Andererseits stehen entsprechend mehrere Aufgabe zur Bearbeitung an. Somit existiert ein Freiheitsgrad bei der Auswahl der Ressourcen und ihrer Zuordnung zu den Aufgaben. Werden für das Ressourcenmanagementsystem zusätzliche Ziele in Form von Optimierungsproblemen formuliert – z.B. die anstehenden Aufgaben sind (jede für sich) möglichst schnell zu erledigen – so werden diese Ziele durch die Mechanismen des **Lastmanagement** erreicht [Lud93, SHK95]. Ein Lastmanagementsystem trifft also aufgrund des Optimierungskriteriums die Entscheidungen darüber, wann die Mechanismen des Ressourcenmanagers angestoßen werden. Für die Funktionalität eines Lastmanagementsystems ist es notwendig die Belastung der einzelnen Ressourcen zu erfassen, zu bewerten und miteinander vergleichen zu können. Wie die Belastung einer Ressource konkret zu definieren ist, hängt u.a. von ihrer Funktionalität (als Eigenschaft der Ressource) ab.

Allgemein versteht man unter dem Begriff der Lastbalanzierung die Überführung von unausgeglichener Belastung der Ressourcen in einen ausgeglichenen Zustand. Aufgrund der Verteiltheit und der unterschiedlichen Leistung der Ressourcen im System ist ein perfekter Lastausgleich in der Praxis nicht effizient realisiserbar. Durch den ansonsten notwendigen hohen Kommunikationsaufwand erfährt das Gesamtsystem selbst eine hohe Lesitungseinbuße. Zusätzlich sind die Entscheidungen durch die Veraltung der Lastwerte (von ihrer tatsächlichen Erfassung bis zur Ausführung der getroffenen Entscheidung) mit einer gewissen Unschärfe behaftet. Es wird deshalb in den zwei Teilprojekten versucht, einen Kompromiß zwischen dem durch das Lastmanagementsystem induzierten Leistungsverlust und der Qualität des Lastausgleichs zu finden. Dazu wird die Last auf die Ressourcen so verteilt und ggf. dynamisch umverteilt, daß keine der verfügbaren Ressourcen ungenutzt ist.

Abhängig vom Umfang und der topologischen Beschaffenheit des verteilten Systems ist die räumliche Verteilung des Lastmanagementsystems festzulegen, insbesondere der untergeordneten Komponenten wie des Lasterfassers und des Lastumverteilers. Hierbei lassen sich drei Strukturen identifizieren: zentrales, dezentrales und hierarchisch organisiertes Lastmanagement. Besonderer Bedeutung kommt der hierarchischen Struktur zu, da sie die Vorteile der zentralen und dezentralen Organisation in sich vereinigt. Einerseits erlaubt sie eine konsistente Gesamtsicht auf den (Last-)Zustand des verteilten Systems, zum anderen birgt sie nicht die Gefahr einer Überbeanspruchung der zentralen Instanz.

Teilprojekt „Verteilte numerische Algorithmen auf Bäumen“

Das Teilprojekt „Verteilte numerische Algorithmen auf Bäumen“ beschäftigt sich in diesem Zusammenhang mit Lastmanagementmechanismen für die Zuteilung von Rechenaufträgen an Rechendienste in einem verteilten, heterogenen System, so daß es zu einer möglichst ausgewogenen Nutzung aller verfügbaren Dienste kommt. Darunter fallen etwa parallele SPMD-Applikationen. Hierbei startet die Anwendung auf mehreren Rechern denselben Berechnungsdienst, an die es anschließend, um die Berechnungsdauer minimal zu halten, Berechnungsaufträge so verteilen muß, daß alle Rechner ausgewogen an der Berechnung teilnehmen können.

Dazu werden in diesem Teilprojekt die oben aufgeführten Lastmanagementaufgaben in einem allgemeineren Kontext betrachtet, nämlich einem offenen Markt von Diensten in einem verteilten System, der nach marktwirtschaftlichen Prinzipien funktioniert. Ein Preissystem dient als Grundlage zur Tarifierung von Diensten. Grundsätzlich werden Nutzungspreise nach Angebot und Nachfrage durch die Anbieter festgelegt. Durch Vermittlungsdienste und Verhandlungsmechanismen geschieht eine optimale Zuteilung. Der Vermittlerinstanz kommt hierbei die Aufgabe der dynamischen Informationsaggregation und -bewertung zu (etwa Lastinformationen), und den Dienstanbietern die Spezifikation der Dienstqualität und -funktionalität, die sich zeitlich ändern kann und entsprechend in der Vermittlerinstanz aktualisiert werden muß. Zu den dynamischen Attributen zählen z.B. Rechnercharakteristiken wie Leistung, Last und freier Hauptspeicher, sowie Preisinformationen. Dienstnutzer müssen schließlich ein Anforderungsprofil des gesuchten Rechendienstes erstellen. Auch während der Nutzungsphase müssen Dienstnutzer in der Lage sein, nach alternativen Dienstangeboten Ausschau zu halten und ggf. die bestehende Nutzung aufzukündigen, so daß sich die Dienstnutzung (Lastverteilung und -umverteilung) der Dynamik im verteilten System und dem Angebot und der Nachfrage auf dem offenen Markt anpassen kann.

Teilprojekt „Konstruktion heteromorph paralleler Systeme“

Bei der Top-Down-Konstruktion heteromorph paralleler Systeme werden Ressourcen auf verschiedensten Abtraktionsebenen unterschieden. Die hierbei auftretende Klassifizierung bildet die Grundlage für die Optimierungskritieren die bei einem Lastmanagement berücksichtigt werden müssen. Neben den Aufgaben die in den beiden anderen Arbeitsgebieten auftreten, gilt es bei einem Ansatz, der die Anwendungen, das Betriebssystem und die Hardware als ein Gesamtsystem betrachtet, weitere Möglichkeiten des Lastmanagements zu berücksichtigen. Zum einen wird bei diesem Ansatz die Granularität der vom Lastmanagement betrachteten Objekte erweitert. So kommen sowohl kleinste Aktivitäten (leichtgewichtige Prozesse), als auch Komplexe von Aktivitäten als Managementobjekte in Frage. Daraus abgeleitete neuartige Einflußgrösen wären zum Beispiel die Kommunikation innerhalb dieser Komplexe und nach aussen. Zum anderen treten andere Kriterien bei dem Lastmanagement in den Vordergrund. Nicht mehr alleine die Rechenkapazität ist ein Kriterium beim Lastmanagement, sondern auch Einflußfaktoren wie Kooperation mit anderen Komplexen, der Speicherbedarf des Komplexes oder die geschätzte Lebzeit eines Komplexes.

Teilprojekt „Programmentwickung für Parallelrechner und vernetzte Architekturen“

Im Teilprojekt „Programmentwickung für Parallelrechner und vernetzte Architekturen“ wird ein systemintegriertes Lastverwaltungssystem für parallele Anwendungen entwickelt, daß besonders auf zwei Aspekte der Ausführungsumgebung Rücksicht nimmt: Heterogenität der gekoppelten Arbeitsplatzrechner und Mehrbenutzer-/Mehrprozeßbetrieb des Systems. Im Vergleich zu dedizierten Parallelrechnern mit homogenen Rechnerknoten und statischer Knotenzuteilung erweitern diese beiden Aspekte die Aufgabenstellungen und Lastmodellierung des Lastverwaltungssystems.

Der Aspekt der Heterogenität gliedert sich in zwei Bereiche: die Architektur der Arbeitsplatzrechner schränkt die Auswahl bei der initialen Prozeßplazierung ein, falls die ablauffähigen Versionen der Anwendung auf gewisse Architekturklassen eingeschränkt ist. Die Ausbaustufe der Rechner einer Architekturklasse bestimmt deren Leistungsfähigkeit und muß bei der Modellierung der Last berücksichtigt werden. Mehrbenutzer-/ Mehrprozeßbetrieb erweitert zusätzlich die Lastmodellierung um den Aspekt fremder Benutzerprozesse, d.h. von Prozessen die im System ablaufen, aber zunächst nicht unter der Kontrolle des Lastverwaltungssystems stehen. Die Anforderungen der externen Benutzer sind und des Benutzers der parallelen Anwendung sind fair zu behandeln. Die Freigabe eines Rechnerknotens durch dessen lokalen Benutzer eröffnet dem Lastverwaltungssystem die Möglichkeit, während des Ablaufs einer Anwendung deren Teile auf solche Rechnerknoten zu verschieben auf denen aktuell keine Anwendungsteile ablaufen. Während der Beobachtungsradius (die Rechnerknoten auf denen die Last ermittelt wird) statisch ist, kann sich der Aktionsradius (wohin sind welche Anwendungsteile zu verschieben) des Lastverwaltungssystems dynamisch ändern.

1.1.3.9 Charakteristika von Ressourcen, Last und Performance

Entsprechend der allgemeinen Definition von Ressourcen und den vielschichtigen Betrachtungsmöglichkeiten ist das Spektrum der Eigenschaften von Ressourcen sehr groß und betrachtungsabhängig. Entsprechend der beiden unterschiedlichen Hauptabstraktionsniveaus, die in dem Graduiertenkolleg bearbeitet werden, können folgende beiden wesentlichen Entscheidungskriterien unterschieden werden:

1. Eingenschaften für die Ressourceneinsparung
   Hierbei werden Ressourcen betrachtet, die für den Anwendungsprogrammierer sichtbar sind, wie z.B. Semaphoren, gemeinsame Objekte etc. Diese Ressourcen werden explizit durch den Programmierer verteilt. Demzufolge ist es wichtig, ihm Hilfsmittel an die Hand zu geben, mit denen er in die Lage versetzt wird, Aussagen über diese Ressourcen treffen zu können.
   Mögliche Eigenschaften sind:

   - Anzahl der Zugriffe auf Objekte

- Erkennung von Deadlocksituationen
- Erkennung von programmbedingten Wartezuständen
- etc.

2. Eigenschaften für die Ressourcenverteilung und das Ressourcenmanagement
   Gegenüber den Maßnahmen, die ein Programmierer treffen kann, um Ressourcen zu sparen, bleibt dem Ressourcenmanagement auf Betriebssystemebene im wesentlichen nur noch die bestmögliche Zuteilung der Ressourcen. Hierzu sind folgende Eigenschaften von Bedeutung:

   - Zugriffszeiten auf Speicherzellen
   - CPU-Leistung
   - etc.

Ziel des Ressourcenmanagement ist es, alle Ressourcen unabhängig von ihrer Abstraktionsebene zu verwalten. Um dies für das gesamte System effizient durchführen zu können, müssen sowohl Einsparungsmaßnahmen, als auch Maßnahmen zur Ressourcenverwaltung ergriffen werden. Unter diesem Gesichtspunkt ist auch eine allgemeine Charakteristik des Begriffes Last und Performance zu verstehen, über die ausführlich im Rahmen der Treffen diskutiert wurde.

Ressourcencharakteristika im Lastmanagement

Für das Lastmanagement ist von Bedeutung welche Objekte Last erzeugen und welche Objekte Last verringern können. Die Grundlage hierzu bieten die von den darunterliegenden Schichten als Black-Box-Ressourcen bereitgestellten Objekte. Hierbei handelt es sich überwiegend um Ressourcen von niedrigen Abstraktionsebene, weswegen sich die relevanten Eigenschaften an Hardwarecharakteristika orientieren.

Ressourcencharakteristika im Bereich Performance Analyse

Zur Performance Analyse wird eine Anwendung bezüglich der Auslastung der von ihr verwendten Ressourcen untersucht. Um eine solche Analyse durchführen zu können müssen bestimmte Charakteristika der unterschiedlichen Ressourcen durch bestimmte Messgrößen dargestellt werden. Beispiele für solche Meßgrößen sind die Rechenzeit der einzelnen Komponenten der Anwendung, die Anzahl der I/0-Operationen, oder die Anzahl der Kommunikationen. Die Performace Analyse hat das Ziel Leistungsengpässe in einem Ablauf der Anwendung zu finden und dann den Programmcode so zu ändern, daß dieser Leistungsengpaß bei weiteren Abläufen nicht mehr auftritt. Insofern reicht es aus, die tatsächlichen Werte von Meßgrößen während eines Ablaufs zu bestimmen. Zur Lokalisierung von Leistungsengpässen sollten nur Meßgrösen der der Anwendung bekannten Ressourcen verwendet werden. Außerdem ist es nötig, die Verwendung dieser Ressourcen in Bezug zum Programmcode zu setzen.

1.1.3.10 Graphgrammatiken / Graphersetzungssysteme

Im folgenden wird die Eignung von Graphersetzungssystemen zur Modellierung verteilter Systeme und ihre Relevanz innerhalb einzelner Teilprojekte des Graduiertenkollegs dargestellt.

Graphen eignen sich zur formal korrekten Darstellung strukturierter Systeme. In der Praxis werden sie häufig in der SW-Entwurfsphase zur Veranschaulichung der Abhängigkeiten einzelner Komponenten eingesetzt - OO-Entwurfsschemata.

Graphgrammatiken eignen sich zur deklarativen Spezifikationen statischer und dynamischer Systemeigenschaften. Wenn ein Graph einen Systemzustand repräsentiert, so kann seine syntaktische Struktur durch eine Graphgrammatik als statische Eigenschaft spezifiziert werden. Das dynamische Verhalten, d.h. der Übergang zwischen einzelnen Systemzuständen kann ebenfalls durch eine Graphgrammatik ausgedrückt werden [BA78, JR91, SWZ95]. Hierbei ist in vielen Fällen die oben erwähnte intuitive Darstellungsform zur operationellen Beschreibung von Vorteil.

Graphersetzungssysteme eigenen sich zur formalen Definition der Semantik verteilter Systeme [Kre80, Rei80, Loy92].

Im Rahmen der Teilprojekte des Graduiertenkollegs werden Graphersetzungssysteme unter den im folgenden dargelegten Aspekten betrachtet.

Teilprojekt „Konstruktion heteromorph paralleler Systeme“

Für die Entwicklung eines Ressourcenmanagemnents für verteilte Systeme werden wegen der komplexen Zusammenhänge mächtige Beschreibungsinstrumentarien benötigt. Sie müssen in der Lage sein, wichtige Beziehungen zwischen den einzelnen Komponenten einer parallelen Anwendung, den Einzelanwendungen eines komplexen Systems sowie den Anwendungen und der dem Management zur Verfügung stehenden Hardware zu modellieren.

Graphen bieten die Möglichkeit sowohl Software- als auch Hardwarekomponenten zueinander in Beziehung zu setzen. Die Aufgaben eines Ressourcenmanagers bestehen im wesentlichen im Verändern solcher Beziehungen. Da sich Graphgrammatiken zur formalen Beschreibung des dynamischen Verhaltens komplexer Systeme eignen, werden sie zur operationellen Beschreibung der Managementfunktionalität herangezogen.

Teilprojektthema „Ein Kalkül für verteilte Programmierung, basierend auf Graph-Ersetzung“

Für verteilte Systeme eignen sich Programmiersprachen, die nicht auf einer sequentiellen Zeichenketten-Syntax, sondern auf Graph-Syntax beruhen. In diesem Fall entspricht ein Programm einem Graphen.

Dies bringt den Vorteil, daß man verteilte Komponenten graphisch auf natürliche Art und Weise beschreiben kann. Im Gegensatz zu vielen verteilten Kalkülen werden in einer geeigneten graphischen Notation keine globalen Port-Namen zur Modellierung der Kommunikation benötigt. Sie können in Graphen durch entsprechende Verbindungsstrukturen ausgedrückt werden.

Es wurde der verteilte Kalkül SPIDER entwickelt, der auf Hypergraph-Ersetzung basiert und miteinander kommunizierende Prozesse beschreibt. Er kann als Basis für eine graphische verteilte Programmiersprache dienen.

Teilprojektthema „Verteilte attributierte Graphersetzung“

Die Anwendbarkeit von Graphgrammatiken zur Spezifikation komplexer strukturierter Systeme wurde bereits durch einige Arbeiten untermauert. Ein wesentlicher Vorteil dieser Spezifikationsform zur Beschreibung verteilter Systeme liegt in der intuitiven deklarativen Darstellungsform. Sie vermeidet die explizite - i.a. fehlerträchtige - Modellierung von Kommunikation zwischen einzelnen Verarbeitungseinheiten.

Nebenläufigkeiten sind implizit in der Spezifikation, definiert durch die Semantik des Graphersetzungssystems, enthalten. Derart spezifizierte Anwendungen bieten somit die Möglichkeit, inherenten Parallelismus aufzufinden und automatisch für eine effiziente verteilte Implementierung zu nutzen. Eine verteilte Realisierung eines Graphersetzungssystems kann somit als universelle Plattform zur Ausführung konkreter Spezifikationen verwendet werden.

1.1.3.11 Ressourcenmanagement

Zusätzlich zu den bekannten Problemen im Bereich des Ressourcenmangements von sequentiellen Systemen müssen bei verteilten Systemen zusätzliche Ressourcenklassen wie beispielsweise das Netzwerk, Vervielfältigung bereits aus der sequentiellen Welt bekannter Ressourcen und die Tatsache berücksichtigt werden, daß kein globaler Zustand existiert.

Um dem Ziel eines effizient arbeitenden verteilten System näher zu kommen sind Arbeiten auf verschiedlichsten Ebenen notwendig. Zum einen müssen Anwendungen entwickelt werden, die die potentiellen Möglichkeiten eines verteilten Systems nutzen können, zum anderen müssen diese Anwendungen auf einem gegebenen System verteilt und verwaltet werden.

Zur Beschreibung der unteschliedlichen Aspekte von Ressourcen und Ressourenmangement in verteilten Systemen wurden folgende allgemein gültigen Definitionen im Graduiertenkolleg verwendet:

Ressourcen: Alle Objekte/Komponenten, die als Hilfsmittel für die Durchführung der Aufgaben eines Objektes benötigt werden können, sind Ressourcen.

Ressourceneigenschaften: Sind diejenigen Eigenschaften (Attribute) einer Ressource, die sie eindeutig charakterisieren und sie damit von anderen Ressourcen unterscheidet und vergleichbar macht.

Ressourcenmanagement: ist die Auswahl von potentiellen Ressourcen und deren Bindung an Objekte als gebundene Ressourcen. Die Grundlage hierzu bilden die Ressourceneigenschaften. Dadurch stellt das Ressourcenmanagement Funktionalitäten und Mechanismen zur Verfügung die Ressourcen eines Systems entsprechend den anstehenden Aufgaben zu verwalten.

Ein Beispiel für Ressourcenmanagement stellt ein Lastmanagement dar. Darunter versteht man die Überführung von unausgeglichener Belastung der Ressourcen in einen ausgeglichenen Zustand.

Generell können die unterschiedlichen Blickwinkel mit denen sich die einzelnen Arbeiten mit Ressourcen und Ressourcenmanagement beschäftigen aufgrund der unterschiedlichen Abstraktionsebenen von denen aus Ressourcen betrachtet werden und der Ressourcen, die auf diesen Ebenen zur Verfügung stehen unterteilt werden.

Charakteristika von Ressourcen, Last und Performance

Im Zusammenarbeitspunkt Charakteristika von Ressourcen wird versucht die unterschiedlichen Ressourcen hinsichtlich ihrer Charakteristiken zu klassifizieren und Meßgrößn anzugeben, die diese Charkteristika beschreiben. Dabei wird untersucht aus welchen unterschiedlichen Standpunkten heraus Ressourcen beim Lastmanagement bzw. bei der Performance Analyse betrachtet werden.

Ein anderer Schwerpunkt der Zusammenarbeit beschäftigt sich mit der Spezifikation von Ressourcen sowie Ressourcenmanagement.

Graphgrammatiken und Graphersetzungssysteme

Im Zusammenarbeitspunkt "Graphgrammatiken" werden unterschiedliche Grammatiken als Beschreibungsmitteln von potentiell unendlichen Graphen betrachtet. Da sich Objekte/Komponenten und deren Beziehungen untereinander anschaulich durch Graphen modellieren lassen, können die Bindungen von Ressourcen an Objekte eines Resssourcenmanagements als Relationen zwischen verschiedenen Komponenten betrachtet werden. Insofern können Graphgrammatiken als Beschreibungsmittel für Operationen des Ressourcenmanagements (Mutationen des Komponentengraphen) verwendet werden.

Literatur

[AWY93] Gul Agha, Peter Wegner und Akinori Yonezawa, Hrsg. *Research Directions in Concurrent Object-Oriented Programming.* MIT Press, Cambridge, Massachusetts, 1993.

[BA78] C. Batini und A.d'Atri. Rewriting systems as a tool for relational data base design. In *Graph-Grammars and Their Application to Computer Science and Biology*, Nr. 73 aus LNCS, S. 139–153. Springer, 1978.

[Eme90] E.A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, Hrsg., *Handbook of Theoretical Computer Science*, Bd. B, S. 996–1072. Elsevier Science Publishers, Amsterdam, 1990.

[Esp96] J. Esparza. More Infinite Results. In B. Steffen und T. Margaria, Hrsg., *Proceedings of INFINITY'96*, Nr. MIP-9614 aus Technical report series of the University of Passau. University of Passau, 1996.

[Gos91] Andrzej Goscinski. *Distributed Operating Systems - The Logical Design.* Addison-Wesley Publishing Company, Inc., Sidney, 1991.

[Gun92] Carl A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. MIT Press, Cambridge, Massachusetts, 1992. Foundations of Computing Series.

[Jan94] P. Jančar. Decidability questions for bisimilarity of Petri nets and some related problems. In *Proceedings of STACS'94*, Bd. 775 aus *LNCS*. Springer Verlag, 1994.

[JR91] D. Janssens und G. Rozenberg. Graph Grammar-Based Description of Object-Based Systems. In *Foundations of Object-Oriented Languages*, Nr. 489 aus LNCS. Springer, 1991.

[Kre80] Hans-Jörg Kreowski. A Comparison between Petri-Nets and Graph Grammars. In *Graphtheoretic Concepts in Computer Science*, Nr. 100 aus LNCS. Springer, 1980.

[Loy92] Joseph Patrick Loyall. *Specification of Concurrent Systems using Graph Grammars*. Dissertation, University of Illinois at Urbana-Champaign, Department of Computer Science, May 1992.

[Lud93] Thomas Ludwig. *Automatische Lastverwaltung für Parallelrechner*. Reihe Informatik, Band 94. BI-Wissenschaftsverlag, 1993.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Pfa91] J. Pfalzgraf. Logical Fiberings and Polycontextural Systems. In P. Jorrand und J. Keleman, Hrsg., *Fundamentals of Artificial Intelligence Research*, Bd. 535 aus *Lecture Notes in Computer Science*. Springer, 1991.

[Rei80] Wolfgang Reisig. A Graph Grammar Representation of Nonsequential Processes. In *Graphtheoretic Concepts in Computer Science*, Nr. 100 aus LNCS. Springer, 1980.

[SHK95] Behrooz A. Shirazi, Ali R. Hurson und Krishna M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, 1995.

[Sho92] Y. Shoham. Micro and Macro Theories of Artificial Agents. In A. Cesta, R. Conte und M. Miceli, Hrsg., *MAAMAW*, 1992.

[SWZ95] A. Schürr, A. Winter und A. Zündorf. Visual Programming with Graph Rewriting Systems. In *IEEE Symposium of Visual Languages Darmstadt*, 1995.

[vL90] Jan van Leeuwen, Hrsg. *Formal Models and Semantics, Handbook of Theoretical Computer Science*, Bd. B. Elsevier, 1990.

[Whi89] H. White. Learning in Artificial Neural Networks: A statistical Perspective. *Neural Computation*, 1, 1989.

[Woo96] Michael Wooldridge. Temporal Belief Logics for Modeling Distributed Artificial Intelligence Systems. In G.M.P. O'Hare und N.R. Jennings, Hrsg., *Foundations of Distributed Artificial Intelligence*. Wiley, 1996.

1.2 Zusammenarbeit mit externen Institutionen

Externe Kooperationen bestehen zwischen dem Graduiertenkolleg und folgenden Einrichtungen:

- Industrie und Wirtschaft
- Sonderforschungsbereiche am Institut
- andere Hochschulen und Forschungseinrichtungen
- Forschungsverbunde und RTB

Im folgenden wird die Zusammenarbeit mit diesem Einrichtungen näher beschrieben:

1.2.1 Zusammenarbeit mit Industrie und Wirtschaft

Zusammenarbeit mit der Industrie besteht mit der Hewlett Packard (HP), der Firma MV-Tec Software GmbH und der Siemens AG. Diese Kooperationen bestehen dabei von seiten des Graduiertenkollegs innerhalb des Themenbereichs „Kooperierende Agenten“ und betreffen insbesondere den Bereich der Modellierung von Agenten.

Von großer Bedeutung für das Teilgebiet „Kooperierende Agenten im Netz- und Systemmanagement“ ist die Kooperation mit Hewlett Packard Laboratories in Bristol in der Form des Austausches von Ideen und Erfahrungen im Bereich von intelligenten Agenten für Netz- und Systemmanagement. Dabei wurde von Hewlett Packard die Teilnahme von Kollegiaten an der Erstellung von Konzepten über agentenbasiertes, verteiltes Management ermöglicht. Diplomarbeiten von Studenten der TU München finden im Rahmen dieser Kooperation in Bristol statt.

Ein wichtiger Aspekt des Projekts „Konzepte zur agentenbasierten Parallelisierung in der Bildanalyse“ ist die Untersuchung der dort entwickelten Konzepte auf ihre Praxistauglichkeit hin. Hierfür wird deren Integration in das am Lehrstuhl für Informatik IX (TUM) entwickelte Bildanalysesystem HORUS angestrebt, das sowohl in Industrie als auch in der Forschung erfolgreich eingesetzt wird. Aus der Erweiterung von HORUS um Konzepte der Parallelverarbeitung ergeben sich zahlreiche Zusammenarbeitspunkte mit der MVTec Software GmbH, die die künftige Weiterentwicklung von HORUS übernommen hat. So beeinflußt sowohl das System-Design von HORUS (Schichtung, Kommunikation der einzelnen Moduln, bereits integrierte Vorarbeiten für eine Parallelisierung) stark die Funktionalität des im Rahmen des Kollegs in HORUS zu integrierenden Multiagentensystems. Zum anderen muß auch das (bislang sequentiell ausgerichtete) „HORUS-Gesamtsystem“ von MVTec stark umgestellt werden, damit es sich zur agentenbasierten, parallelen Bildverarbeitung einsetzen läßt.

Im Bereich des Projekts „Kooperierende Agenten in verteilte Systemen – Grundlagen“ besteht eine Zusammenarbeit mit der Forschungsgruppe „Neuronale Netze“ aus dem Bereich „Zentrale Forschung und Entwicklung“ der Siemens AG. Diese Forschungsgruppe befaßt sich mit den verschiedensten industriellen Anwendungen von neuronalen Netzen. Auf dem Gebiet theoretischer Grundlagen besteht eine enge Zusammenarbeit im Bereich der Untersuchung von Methoden zur Regularisierung von Gaussian Mixture Modellen zur

Schätzung von Wahrscheinlichkeitsdichten. Ein Ergebnis ist ein experimenteller Vergleich bestehender sowie von uns neu entwickelter Methoden. Desweiteren besteht eine Kooperation betreffend der Charakterisierung von Transitionsdichten in der Zeitreihenanalyse. Gegenstand dieser Kooperation ist vor allem die Untersuchung von neuronalen Modellen zur Beschreibung von bedingten Wahrscheinlichkeitsdichten und deren Anwendung zur Verbesserung von Regressionsschätzern. Die Zusammenarbeit wird durch gegenseitige Besuche und einen intensivem E-Mail Kontakt durchgeführt. Resultat der Arbeit sind u.a. mehrere gemeinsame Veröffentlichungen.

1.2.2 Zusammenarbeit mit Sonderforschungsbereichen

Die größte Zahl an externen Kooperationen besteht zwischen dem Graduiertenkolleg und Teilprojekten von Sonderforschungsbereichen. Im Vergleich zu den anderen externen Kooperationen sind diese auch als am wichtigsten einzuschätzen, da einerseits diese Kooperationen beide Themenbereiche des Graduiertenkollegs betreffen und andererseits im Rahmen der Kooperationen zumeist wichtige Fragestellungen aus Querschnittsthemen des Graduiertenkollegs behandelt werden. Konkret bestehen Kooperationen zwischen Teilnehmern des Graduiertenkollegs und den Sonderforschungsbereichen 331 und 342:

1.2.2.1 SFB 342

Eine besonders ausgeprägte Zusammenarbeit findet sich zwischen Teilnehmern des Graduiertenkollegs und dem SFB 342 „Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen". Hier besteht eine Kooperation von fünf Teilprojekten aus dem Themenbereich „Programmiermodelle und Ressourcemanagement in verteilten Systemen" mit dem SFB, die im wesentlichen aus einem regen Informations- und Gedankenaustausch zwischen einzelnen Teilprojekten des SFB und des Kollegs besteht. Insbesondere die Themen Verifikation und Ressourcen-/Lastmanagement werden im Rahmen der Kooperationen vertieft.

Im Bereich Verifikation besteht eine enge Zusammenarbeit des Teilprojekts „Verteilte Inferenzsysteme" mit dem Teilprojekt A5 des SFB 342. Im Rahmen des Teilprojekts „Verteilte Inferenzsysteme" entwickelte Verfahren (zur Bestimmung von Ähnlichkeit zwischen Beweiszielen und Beweisfakten) können (und werden) innerhalb der im SFB entwickelten Beweiser eingesetzt werden. Zudem besteht die Möglichkeit zum Erfahrungsaustausch betreffend der Entwicklung paralleler Beweiser. Im Rahmen des Teilprojekts „Verteilte Algorithmen – Spezifikation, Modellierung, Korrekheit" besteht eine enge Zusammenarbeit mit dem Teilprojekt A3. Diese Zusammenarbeit konzentriert sich insbesondere auf die Themengebiete Petri Netze, Komplexitätstheorie, Model checking und Semantik nebenläufiger Systeme. Die Zusammenarbeit des Projekts „Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme" mit dem Teilprojekt A1 des SFB besteht im Bereich des spezifikationsorientierten Testens verteilter und paralleler Programme. Dabei steht die Beherrschung des Nichtdeterminismus paralleler Programme während der Testphase im Vordergrund. Ziel dieser Zusammenarbeit ist es, eine Werkzeugumgebung zu realisieren, die es erlaubt durch eine geeignete Spezifikation von Testfällen, den Nichtdeterminismus so einzuschränken, daß in mehreren Abläufen unter Verwendung der Testfallspezifikation temporallogische Formeln, wie sie im Teilprojekt „Modellprüfung paralleler und verteilter Programmabläufe“ vorkommen, immer erfüllt bzw. unerfüllt sind. Zur Überprüfung der Erfülltheit temporallogischer Formeln in einem Testfall wird die im Teilprojekt „Modellprüfung paralleler und verteilter Programmabläufe“ entwickelte Methode verwendet.

Im Bereich des Ressourcenmanagements besteht im Teilprojekt „Konstruktion heteromorph paralleler Systeme“ eine wechselseitige Zusammenarbeit mit dem SFB 342 (Teilprojekt A8): Die Berührungspunkte mit den Projekten, die im Rahmen des Teilprojektes A8 entstehen, sind sehr vielschichtig. Zum einen dienen die in diesen Projekten entwickelten Verfahren und Konzepte als Grundlage für die Arbeiten im Rahmen des Graduiertenkollegs, zum anderen ist es für das Teilprojekt A8 möglich, die Ergebnisse im Bereich Ressourcenmanagement in heteromorph paralleler Systeme einzusetzen. Im Teilprojekt „Programmentwickung für Parallelrechner und vernetzte Architekturen“ des Kollegs besteht eine Zusammenarbeit mit dem SFB hinsichtlich der Klassifikation von Lastmodellen. Innerhalb des Querschnittsprojekts ALV (Anwendungsintegrierte Lastverteilung) wurde die Klassifikation eines Lastmodells vorgestellt und diskutiert. Da die Mitglieder des Projekts verschiedenste Sichtweisen auf das Problem der Lastverwaltung besitzen, konnte die Klassifikation in einigen Details verfeinert werden. Die Vollständigkeit der Klassifikation wurde anhand von Beispielen real existierender System zur Lastverwaltung überprüft.

1.2.2.2 SFB 331

Der Sonderforschungsbereich 331 „Informationsverarbeitung in autonomen, mobilen Handhabungssystemen“ beschäftigt sich mit der Weiterentwicklung von mobilen Systemen in Fertigungs- bzw. Büroumgebungen. Eine externe Kooperation mit dem SFB wird im Bereich „Kooperierende Agenten“, konkret von den Teilprojekten „Kooperierende Agenten und autonome Robotersysteme“ und „Bildverstehen in verteilten Systemen“ durchgeführt.

So werden in dem Teilprojekt „Kooperierende Agenten und autonome Robotersysteme“ Erkenntnisse eingebunden, die im SFB 331 innerhalb des Teilprojektes Q4 „Dezentrale kooperative Aktionsplanung von Robotern und Maschinen auf taktischer Ebene“ gewonnen wurden. Im Rahmen des Teilprojekts „Bildverstehen in verteilten Systemen“ besteht eine Zusammenarbeit mit dem Teilprojekt L9 des SFB. Im Hinblick auf die Entwicklung von Methoden zur Parallelverarbeitung in der Bildanalyse und zum Agentendesign ergeben sich die Möglichkeiten zum Erfahrungsaustausch zwischen Kolleg und dem SFB.

1.2.3 Hochschulen und Forschungseinrichtungen

Ebenso wie mit Sonderforschungsbereichen finden sich zahlreiche Kooperationen zwischen Kollegiaten und Hochschulen bzw. Forschungseinrichtungen. Im Vergleich zur Zusammenarbeit mit Sonderforschungsbereichen ist diese Zusammenarbeit meist enger auf die einzelnen Teilprojekte des Graduiertenkollegs abgestimmt. Diese Kooperationen fördern i.d.R. eher die spezifische Arbeit in den einzelnen Teilprojekten und sind weniger von kollegsübergreifendem Interesse. Im folgenden sind die einzelnen Kooperationen getrennt nach den zwei Teilbereichen des Graduiertenkollegs aufgelistet.

1.2.3.1 Kooperierende Agenten und autonome Robotersysteme

Im Rahmen des Teilprojekts „Kooperierende Agenten in verteilte Systemen – Grundlagen" (Thema: „Computational Economies, Wahrscheinlichkeitsdichten und Neuronale Netze zur Entscheidungsmodellierung in Multiagentensystemen") besteht eine Kooperation mit dem Department of Economics, UC San Diego, wobei Kontakte hauptsächlich mit Prof. Halbert White gepflegt werden. Prof. White von der UCSD arbeitet seit langem auf dem Gebiet der Zeitreihenanalyse, insbesondere angewandt auf Daten ökonomischer Natur. Dieses Themengebiet beinhaltet im besonderen die Analyse von Transitionsdichten. Die Zusammenarbeit mit Prof. White erstreckt sich vor allem auf Modelle zur Schätzung von Wahrscheinlichkeitsdichten hoher Flexibilität (Parametrisierung durch Momente erster bis vierter Ordnung) sowie hoher informationstheoretischer Plausibilität (hohe Entropie). Diese Modelle werden dann experimentell vor allem zur Charakterisierung und Vorhersage von Finanzzeitreihen sowie zur Schätzung von sogenannten Risk-Neutral-Distributions verwendet. Als Resultat dieser Arbeit liegen im Wesentlichen ein Journal-Paper (zur Zeit in Review) sowie ein Entwurf für ein weiteres Paper vor.

Weiterhin findet eine enge Zusammenarbeit im Rahmen des Teilprojektes „Kooperierende Agenten in verteilte Systemen – Grundlagen" (Thema: „Ein logikbasierter Ansatz zur Modellierung kooperierender Agenten") mit Prof. Dr. Jochen Pfalzgraf statt, der seit 1.11.1996 am Lehrstuhl „Wissensbasierte Systeme und lernende Systeme" am Institut für Computerwissenschaften und Systemanalyse der Universität Salzburg beschäftigt ist. Davor war er der Projektleiter der MEDLAR-Gruppe (Mechanising Deduction in the Logics of Practical Reasoning) am RISC-Institut (Research Institute for Symbolic Computation) der Universität Linz. Im Rahmen des MEDLAR-Projektes entwickelte er die sogenannten logischen Faserungen und untersuchte ihre Anwendbarkeit zur Steuerung kooperierender Roboter. Gemeinsam wird an der Weiterentwicklung des Konzeptes der logischen Faserungen zur formalen Modellierung von Multiagentensystemen gearbeitet. Dabei liegt vorerst ein Schwerpunkt auf der Modellierung von Kooperations- und Interaktionsbeziehungen zwischen einzelnen Agenten. Ausgangspunkt hierfür sind die Ergebnisse, die im MEDLAR-Projekt gewonnen wurden.

Im Bereich des Teilprojekts „Kooperierende Agenten im Netz- und Systemmanagement" findet eine Zusammenarbeit mit der Universität von Pretoria, South Africa, statt. Dabei ist u.a. ein Papier über den Einsatz von intelligenten Agenten für die verteilte Implementierung von Management Policies entstanden. Momentan findet auch eine gemeinsame Untersuchung von unterschiedlichsten Standardisierungsansätzen für delegierbare Management-Funktionalität statt.

Ein wichtiger Partner für das Teilprojekt „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit" ist der Rank Xerox Research Center (RXRC), Grenoble (Abteilung: Cooperation Technologies, Projekt-Thema: Constraint Based Knowledge Brokers CBKB). Das Projekt Constraint Based Knowledge Brokers beschäftigt sich mit der Entwicklung von leistungsfähigen Methoden für effizientes Information Retrieval und Kombination von Wissen. Dabei wird ein agentenbasierter Ansatz verfolgt, der zur Spezifikation und Bearbeitung von Suchanfragen Constraints einsetzt. Im Rahmen eines viermonatigen Gastaufenthaltes im RXRC Grenoble konnten Erfahrungen aus dem Bereich agentenbasierter Informationsaustausch bei der Weiterentwicklung eines constraint-basierten Brokersystems eingebracht werden. Andererseits lieferte die Tätigkeit in diesem Projekt für das Graduiertenkollegprojekt wertvolle Erfahrungen über die Auswirkungen verschiedener Agenteneigenschaften und constraint-basierter Wissensrepräsentation bei der Realisierung von Brokersystemen.

1.2.3.2 Programmiermodelle und Ressourcemanagement in verteilten Systemen

Für das Teilprojekt „Verteilte Algorithmen – Spezifikation, Modellierung, Korrekheit“ wurde eine Kooperation mit der Forschungsgruppe von Prof. Bouajjani in Grenoble (Frankreich) vereinbart. Es geht hierbei hauptsächlich um Model Checking Probleme, semantische Äquivalenzen und Tableauverfahren. Wechselseitige Besuche werden Anfang/Mitte 1997 stattfinden.

Auf dem Gebiet der Lastverwaltung (Definition einer architekturunabhängigen Schnittstelle zur Lasterfassung auf heterogenen Rechensystemen) findet eine Zusammenarbeit der Technischen Universität Braunschweig mit dem Teilprojekt „Programmentwickung für Parallelrechner und vernetzte Architekturen“ statt: Die unterschiedlichen Konzepte von heuristischen Lastverwaltungssystemen und der Einsatz neuronaler Netze zur Lastbalancierung wurden während mehrerer Treffen diskutiert. Die Methoden zur Lasterfassung auf heterogenen UNIX Rechnern basiert auf der zur Verfügung stehenden Systemunterstützung. Neben der Diskussion über die Details der Implementierung wurde ein allgemeine Schnittstelle definiert, welche Lastwerte zur Beurteilung der Last eines Rechners notwendig ist. Die Definition der Schnittstelle fließt in das Lehrstuhl-interne Projekt OMIS (Online Monitoring Interface Specification) ein.

Im Teilprojekt „Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme“ gibt es eine Zusammenarbeit mit dem Department E6, Institut Jožef Stefan, Ljubljana, Slowenien (Ansprechpartner: Dr. M. Pučko). Diese Zusammenarbeit entwickelte sich aus der Mitarbeit an dem von der EU geförderten Projekt COST14. Diese Mitarbeit bezog sich auf das Teilprojekt „CSCW and Software Engineering“. Nachdem die Themenstellung nach dem Ausscheiden etlicher Mitarbeiter geändert wurde und sich nicht mehr wie ursprünglich mit der Entwicklung vom Methoden und Werkzeugen zur Entwicklung von CSCW-Applikationen sondern mit der Anwendung von CSCW-Konzepten im Bereich Software Engineering beschäftigte, spaltete sich eine kleinere Gruppe von diesem Teilprojekt und COST14 ab und beschäftigte sich weiterhin mit Methoden und Werkzeugen zur Entwicklung von CSCW-Applikationen. Die Zusammenarbeit besteht im Bereich Entwicklung von formalen Methoden zur Spezifikation und Validierung von CSCW-Applikationen. CSCW-Applikationen werden dabei als Beispiele von inhärent verteilten Systemen betrachtet, so daß die dabei entwickelten Methoden durchaus auch für andere inhärent verteilte Probleme angewendet werden können. Diese Zusammenarbeit stellt somit eine Vorarbeit zum Thema „Objektorientierte Spezifikation verteilter Systeme“ dar, das sich mit formal fundierten Methoden zur Entwicklung von verteilten Systemen beschäftigt. Die im Laufe der Zusammenarbeit entwickelte Kombination aus temporaler Logik und algebraischer Spezifikation zur Beschreibung von CSCW-Systemen wird als formale Grundlage für die Spezifikation von verteilten Systemen im Teilprojekt „Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme“ verwendet.

Zudem bestehen seit Dezember 1996 Kontakte mit dem Bremer Institut für sichere Systeme an dem Technologiezentrum Informatik der Universität Bremen (Ansprechpartner: Dr. H. Schlingloff). Dr. Schlingloff war vor seinem Wechsel nach Bremen Mitarbeiter am Institut für Informatik der TU München und leitete das Forschungsprojekt „Temporale Spezifikation reaktiver Systeme“. Die Zusammenarbeit besteht zum einen im Bereich Spezifikation von Eigenschaften mit temporaler Logik und zum anderen im Bereich Modellprüfung verteilter und paralleler Systeme. Diese Zusammenarbeit hatte erheblichen Einfluß auf die Arbeit im Teilprojekt „Modellprüfung paralleler und verteilter Programmabläufe“ speziell bei der Diskussion verschiedener Verifikationsmethoden im Hinblick auf ihre Verwendbarkeit bei verteilten Systemen sowie auf die Überprüfung von Performance-Eigenschaften durch Modellprüfungsmethoden.

1.2.4 Forschungsverbunde und RTB

Kooperationen mit Forschungsverbunden finden zur Zeit nur im Teilprojekt „Bildverstehen in verteilten Systemen“ statt.

Im Rahmen des Teilprojekts „Medizinische Bild- und Signalverarbeitung“ des RTB Bayern wurden Bildverarbeitungsalgorithmen zur Erkennung von normalen und pathologischen Strukturen in Endoskopiebildern des oberen Verdauungstraktes entwickelt. Die im Teilprojekt untersuchten Verfahren zur Generierung von Objekterkennungsprogrammen wurden dabei mit Erfolg eingesetzt.

Zusätzlich besteht eine Kooperation innerhalb der Arbeitsgemeinschaft Bayerischer Forschungsverbünde (ABayFOR) (Forschungsgruppe Kognitive Systeme am Bayerischen Forschungszentrum für wissensbasierte Systeme FORWiss) hinsichtlich des Einsatzes von Parallelisierungstechniken in der Bildanalyse.

1.3 Teilprojektsberichte

1.3.1 Ein logikbasierter Ansatz zur Modellierung kooperierender Agenten

Teilprojekt: Kooperierende Agenten in verteilten Systemen – Grundlagen
Angela Bücherl

1 Einführung

In der Verteilten Künstlichen Intelligenz beschäftigt man sich in letzter Zeit verstärkt mit formalen Methoden zur Modellierung von Multiagenten-Systemen.
Die Entwicklung formaler Methoden für Multiagenten-Systeme ist notwendig, da (vgl. [Woo92]):

- sie ein Werkzeug zur Verfügung stellen, das den Entwurf auf einer konzeptionellen Ebene unterstützt, was aufgrund der Komplexität von Multiagenten-Systemen unerläßlich ist.
- sie eine Basis für die Verifikation und Validierung des Systemverhaltens und der Systemeigenschaften liefern.

- sie die Grundlage für die Entwicklung und Untersuchung von Kooperations- und Interaktionstheorien bilden.

Ein wesentliches Problem bei der Entwicklung einer geeigneten formalen Modellierungsmethode stellt die Integration der beiden Perspektiven dar, die man in einem Multiagenten-System unterscheiden kann. Einerseits kann man eine lokale Perspektive einnehmen und sich mit den Eigenschaften und Fähigkeiten der einzelnen Agenten beschäftigen, andererseits kann man eine globale Perspektive betrachten und sich mit der Organisation des Kooperationsverhaltens des gesamten Multiagenten-Systems auseinandersetzen. Shoham spricht in diesem Zusammenhang von Mikro- und Makrotheorien [Sho92]. Auf beiden Ebenen wurden schon mehrere formale Modellierungsmethoden entwickelt, es wurden jedoch bisher erst wenig Versuche unternommen, die beiden Perspektiven in einem Ansatz zu integrieren.
In dieser Hinsicht erscheint das Konzept der logischen Faserungen geeignet, da es auf einfache Weise einen Wechsel von einer lokalen Agentenperspektive zur globalen Systemperspektive erlaubt.

2 Logische Faserungen

Das Konzept der logischen Faserungen wurde von J. Pfalzgraf in [Pfa91] basierend auf der klassischen Theorie der Faserbündel entwickelt. Faserbündel (auch Faserungen genannt) sind ein ausdrucksstarkes mathematisches Modellierungswerkzeug, mit deren Hilfe lokal-global Beziehungen verschiedener Strukturen in einem Konzept ausgedrückt werden können. Da zum Verständnis von logischen Faserungen keine Grundlagen aus der Faserbündeltheorie benötigt werden, wird für Interessierte auf [Por77], [Ste51] und [Hus94] verwiesen.

Ausgangspunkt für eine logische Faserung ist ein indiziertes System von disjunkten logischen Räumen $(L_i)_{i \in I}$. Jedes L_i ist beispielsweise eine klassische zweiwertige Logik. Die L_i sollen zu einem globalen logischen Raum E („Totalraum“) zusammengefaßt werden unter Berücksichtigung der Struktur von E als disjunkte Vereinigung der L_i und ihrer Beziehung zur Indexmenge I. Dies wird mathematisch formalisiert mit Hilfe einer logischen Faserung $\mathcal{L} = (E, \pi, I)$. Die Abbildung $\pi : E \to I$, die den Totalraum E auf die Indexmenge I („Basisraum“) abbildet, wird so definiert, daß die Urbildmenge jedes Punktes i, $i \in I$ genau der zugehörige logische Raum L_i ist. Diese Urbildmengen werden Fasern genannt. Eine Faser L_i über dem Punkt $i \in I$ besteht demnach aus allen Elementen des Totalraums E, die durch π auf i abgebildet werden.

Faser über I: $\quad i \in I: \quad L_i := \pi^{-1}(i) = \{x \in E | \pi(x) = i\}$

Eine logische Faserung liefert dadurch eine mathematische Beschreibung für ein verteiltes, indiziertes, logisches System. Man muß unterscheiden zwischen der lokalen Wahrheitswertemenge Ω_i jeder Faser L_i, $i \in I$, und der Menge der globalen Wahrheitswerte Ω^I, die die Vereinigung aller lokalen Wahrheitswerte ist, d.h $\Omega^I = \bigcup_{i \in I} \Omega_i$.

Innerhalb jeder Faser können unabhängig und parallel lokale logische Ausdrücke gebildet werden und Schlußfolgerungsprozesse ablaufen. Andererseits können auch globale logische Ausdrücke φ gebildet werden, die als Familie von lokalen Ausdrücken φ_i definiert werden.

$$\varphi \in \mathcal{L}: \quad \varphi := (\varphi_i)_{i \in I}, \quad \varphi_i \in L_i$$

Die Menge aller Ausdrücke, die auf diese Art und Weise gebildet werden können, stellen die Sprache der logischen Faserung $\mathcal{L}$ dar.

In jeder Faser können lokal logische Operationen ausgeführt werden. Außerdem können lokale Operationen der einzelnen Fasern zu einer globalen logischen Operation zusammengefaßt werden, so daß bei der Anwendung einer globalen logischen Operation parallel in jeder Faser eine lokale logische Operation durchgeführt wird.

Beipiel:
Eine globale bivariate logische Operation

$$\vartheta: \quad (x_i, y_i)_{i \in I} \longmapsto (z_i)_{i \in I}$$

bekommt als Eingabe jeweils zwei Ausdrücke aus jeder Faser. Jedes lokale Paar $(x_i, y_i) \in L_i \times L_i$ wird durch ϑ auf einen Ausdruck $z_i \in L_i$ abgebildet.

Eine Besonderheit bei logischen Faserungen ist, daß man auch logische Operationen definieren kann, die lokale Ausdrücke nicht nur innerhalb ihrer Faser abbilden, sondern die Bilder über verschiedene Fasern verteilt. So eine Operation wird Transjunktion genannt.

Beipiel:
Eine bivariate Transjunktion Θ

$$\Theta : \begin{cases} L_i \times L_i & \longrightarrow \quad E \\ \Omega_i \times \Omega_i & \longrightarrow \quad \Omega^I \end{cases}$$

bekommt als Eingabe zwei Ausdrücke aus dem Subsystem L_i und verteilt die Bilder über die gesamte Faserung. Mit Wahrheitswerten ausgedrückt bildet Θ ein Paar von lokalen Wahrheitswerten aus Ω_i auf die Menge der globalen Wahrheitswerte Ω^I ab.

3 Ein einfaches Beispiel

Anhand eines einfachen Beispiels (aus [PS92]) wird nun gezeigt, wie logische Faserungen zur modularen Modellierung von Multiagenten-Systemen benutzt werden können. Das zu modellierende Multiagenten-System besteht aus drei Agenten, R_0, R_1 und R_2, die eine einfache Montagearbeit kooperativ ausführen sollen. Die Aufgabe des Agenten R_0 besteht darin, jeweils ein Bauteil des Typs A und B zusammenzufügen. R_1, der das Lager mit Bauteilen des Typs B verwaltet und R_2, der das Lager mit Bauteilen des Typs A verwaltet, sind dafür verantwortlich, R_0 mit Bauteilen zu versorgen. R_0 prüft mit Hilfe eines Sensors, ob sich ein Bauteil A und ein Bauteil B auf seinem Arbeitstisch befinden. Da nur R_0 Zugang zu den Sensordaten hat, muß er den zuständigen Agenten informieren, wenn ihm ein bestimmtes Bauteil fehlt.

3.1 Logische Modellierung

Zur logischen Modellierung dieses Beispiels wird die Menge der Agenten $I = \{R_0, R_1, R_2\}$ als Indexmenge der logischen Faserung verwendet. Jedem Agenten wird als Faser ein individueller lokaler logischer Zustandsraum $L_i, i \in I$ zugeordnet. Alle Fasern zusammengefaßt ergeben den globalen Zustandsraum E (Totalraum). Die Faserung $\mathcal{L}$ ist somit ein logisches Modell des gesamten Multiagenten-Systems.

Der logische Zustandsraum des Agenten R_0 besteht aus zwei logischen Kontrollvariablen x_0 und y_0. x_0 bezeichne das Prädikat on(table,A) und y_0 bezeichne das Prädikat on(table,B). Die lokalen Wahrheitswerte sind T_0 und F_0. Die logische Variable z_0 wird zur Steuerung der Aktionen verwendet, die R_0 in einem konkreten Zustand ausführen soll. Wenn $z_0 = T_0$, soll R_0 die $AKTION_0$: „Füge A und B zusammen“ ausführen. Wenn $z_0 = F_0$ ist, soll er nichts tun. Der logische Zustandsraum des Agenten R_1 besteht nur aus einer Aktionssteuerungsvariable z_1. Die lokalen Wahrheitswerte sind T_1 und F_1. Wenn $z_1 = F_1$, dann soll R_1 die Aktion $AKTION_1$: „Bringe ein Bauteil B zu R_0“ ausführen, ansonsten, wenn $z_1 = T_1$ soll er inaktiv bleiben. Analog besteht der logische Zustandsraum des Agenten R_2 ebenfalls nur aus einer Aktionssteuerungsvariable z_2. Wenn $z_2 = F_2$, dann soll R_2 die Aktion $AKTION_2$: „Bringe ein Bauteil A zu R_0“ ausführen, ansonsten soll er inaktiv bleiben. Die lokalen Wahrheitswerte sind T_2 und F_2.

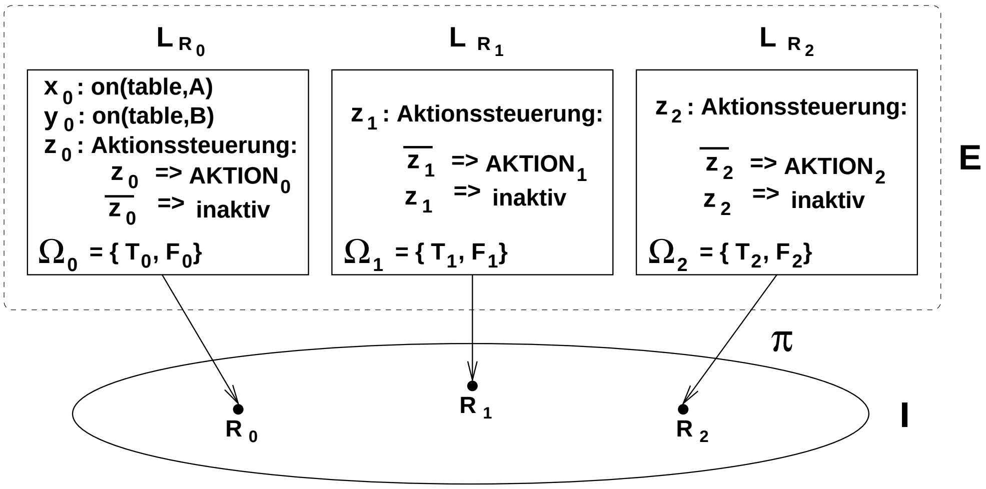


Abbildung 1: Modellierung als logische Faserung

Die Menge der globalen Wahrheitswerte Ω ist die Vereinigung aller lokalen Wahrheitswerte. D.h. die logische Faserung stellt global gesehen eine 6-wertige Logik dar.

Die Steuerung des Montageprozesses ist abhängig vom Zustand (d.h. von der Belegung) der beiden Kontrollvariablen x_0 und y_0, der die auszuführenden Aktionen der Agenten

bestimmt. Das bedeutet, daß abhängig von der Belegung von x_0 und y_0 die Aktionssteuerungsvariablen z_i entsprechend belegt sein müssen. Da die Belegung von x_0 und y_0 durch den Sensor bei R_0 erfolgt und nur R_0 direkten Zugang zu den Sensordaten hat, muß die semantische Bedeutung dem entsprechenden Agenten irgendwie übermittelt werden.
Der Fall, wenn sowohl x_0 als auch y_0 „false“ sind, d.h. überhaupt nichts liegt auf dem Arbeitstisch von R_0, wird so gelöst, daß zuerst R_1 ein Bauteil B zu R_0 bringen soll, der danach eine erneute Sensorauswertung durchführt.

Die Belegung der Aktionssteuerungsvariablen z_i wird mit Hilfe einer Transjunktion Θ modelliert. Die Transjunktion Θ kann als logische Steuerung der Kooperation interpretiert werden oder als Modellierung der zur Kooperation nötigen Kommunikation. Die Transjunktion Θ erhält als Eingabe die beiden Kontrollvariablen x_0 und y_0 (also zwei logische Ausdrücke aus L_{R_0}) und verteilt ihre Bilder, je nach Zustand von x_0 und y_0, über die drei logischen Subsysteme L_{R_0}, L_{R_1} und L_{R_2}. Die z_i nehmen den Wert auf, falls das Bild in ihrem Subsystem liegt. Mit Wahrheitswerten ausgedrückt: Θ bekommt zwei Wahrheitswerte aus Ω_0 und bildet sie auf einen Wahrheitswert aus Ω ab.

$$\Theta : \begin{cases} L_0 \times L_0 & \longmapsto & E \\ \Omega_0 \times \Omega_0 & \longmapsto & \Omega \end{cases}$$

Wertetabelle :

x_0 \ y_0	T_0	F_0
T_0	T_0	F_1
F_0	F_2	F_1

3.2 Informationsverarbeitung

Die lokale Informationsverarbeitung, die in jedem Agenten stattfindet, kann nun folgendermaßen beschrieben werden:

Agent R_0:
„Sensorauswertung $\Rightarrow$ Aktualisierung von x_0 und y_0“
$\Theta(x_0, y_0)$
IF $z_0 = T_0$
THEN „A und B zusammenfügen“
$z_0 := F_0$

Agent R_1:
IF $z_1 = F_1$
THEN „Bringe ein Bauteil B zu R_0
$z_1 := T_1$

Agent R_2:
IF $z_2 = F_2$
THEN „Bringe ein Bauteil A zu R_0“
$z_2 := T_2$

Natürlich müssen die lokalen Informationsverarbeitungsprozesse der einzelnen Agenten noch zeitlich koordiniert werden, um das gewünschte globale Systemverhalten zu erhalten.

4 Zusammenfassung und Ausblick

Das Beispiel illustrierte, wie das Konzept der logischen Faserungen benutzt werden kann, um Systeme kooperierender Agenten logisch zu modellieren. Dabei steht die Idee im Vordergrund, daß jeder der beteiligten Agenten einen eigenen lokalen logischen Zustandraum besitzt. Zur Modellierung von Kooperation ist es notwendig, Kommunikation zwischen den Agenten zu beschreiben und Beziehungen zwischen Objekten und Tatsachen, auf die sich die Agenten in ihrem individuellen Zustandsraum beziehen, zu identifizieren und darzustellen. Dazu bieten sich Transjunktionen an, wie im Beispiel gezeigt wurde.
In der weiteren Arbeit gilt es zunächst, die formale Handhabung von Transjunktionen im Hinblick auf ihre Anwendung zur Modellierung von Kooperation weiterzuentwickeln. Das Ziel ist die Entwicklung eines allgemeinen logischen Kalküls, der den formalen Umgang mit multivariaten logischen Operationen beschreibt. Diese sollten so weit wie möglich fasernweise abgearbeitet werden können.

Wie oben schon erwähnt, müssen in einer Modellierung zeitliche Abhängigkeiten erfaßt werden, um das Verhalten der einzelnen Agenten zu koordinieren. Auf welche Art und Weise zeitliche Aspekte in das Konzept der logischen Faserungen integriert werden können ist ein weiterer wichtiger Gegenstand der zukünftigen Arbeit. Einen wertvollen Hinweis auf eine Lösung dieses Problems liefert dabei der Ansatz zur Integration räumlicher Abhängigkeiten in das Faserungenkonzept, der in [PS95] vorgestellt wurde. Die Idee dahinter ist, daß sich der Wahrheitswert bestimmter Bedingungen abhängig von der räumlichen Position des Agenten verändern kann.

Literatur

[Hus94] Dale Husemoller. *Fibre Bundles.* Springer Verlag, third. Auflage, 1994.

[Pfa91] J. Pfalzgraf. Logical Fiberings and Polycontextural Systems. In P. Jorrand und J. Keleman, Hrsg., *Fundamentals of Artificial Intelligence Research*, Bd. 535 aus *Lecture Notes in Computer Science*. Springer, 1991.

[Por77] Richard D. Porter. *Introduction to Fibre Bundles.* Marcel Dekker, Inc., 1977.

[PS92] J. Pfalzgraf und K. Stokkermans. Scenario construction continued and extended with a view to test and enhancement of reasoning methods. Technical Report 92-27.0, RISC-Linz, J. Kepler Universität, Linz, Mai 1992.

[PS95] J. Pfalzgraf und K. Stokkermans. On Robotics Scenarios and Modeling with Fibered Structures. In Jochen Pfalzgraf und Dongming Wang, Hrsg., *Automated Practical Reasononing: Algebraic Approaches*, S. 53–80. Springer-Verlag, 1995.

[Sho92] Y. Shoham. Micro and Macro Theories of Artificial Agents. In A. Cesta, R. Conte und M. Miceli, Hrsg., *MAAMAW*, 1992.

[Ste51] Norman Steenrod. *The Topology of Fibre Bundles.* Princeton Mathematical Series. Princeton University Press, 1951.

[Woo92] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems.* Dissertation, UMIST, Manchester, October 1992.

1.3.2 Verteilte Generierung von Objekterkennungsprogrammen

Teilprojekt: Bildverstehen in verteilten Systemen
Dietrich Büsching

1 Einleitung und Motivation

Der Entwicklungsaufwand für heutige Bildverarbeitungssysteme ist relativ hoch und ein wesentliches Hindernis für ein weiteres Vordringen dieser Technik in praktische Anwendungen. Es wird intensiv nach Möglichkeiten gesucht, diese Kosten durch bessere Tools und lernende Systeme zu senken. Die Objekterkennung ist dabei ein besonders komplexes Teilproblem.

Die übliche Vorgehensweise bei der ingenieursmäßigen Entwicklung von Bildanalysesystemen ist der Aufbau eines Algorithmus aus einer Verkettung von vorhandenen Operatoren (Glättung, Kantendetektion, etc.). Da diese Operatoren bereits implementiert sind, beschränkt sich der Entwicklungsaufwand auf die Auswahl einer geeigneten Operatorfolge. In früheren Arbeiten wurden häufig anhand eines (CAD-) Modells des zu erkennenden Objektes analytisch die für die Erkennung geeigneten Merkmale bestimmt. Im Rahmen meiner Dissertation möchte ich einen anderen stärker an empirischen Untersuchungen der Eignung von Merkmalen oder Operatoren orientierten Ansatz verfolgen.

Beim Entwurf eines solchen Systems zur automatischen Erzeugung von Objekterkennungsprogrammen muß selbstverständlich der gegenwärtige Stand der Forschung in der Objekterkennung berücksichtigt werden. Hier sind universelle Algorithmen zur Erkennung beliebiger Objekte noch längst nicht in Sicht (falls sie überhaupt existieren). Vielmehr existieren lediglich Teillösungen für Spezialfälle. Beispielsweise gibt es Systeme, die anhand von Texturmerkmalen Unregelmäßigkeiten auf der Oberfläche von Objekten erkennen. Andere Systeme können Gesichter von Menschen in komplexen Szenen detektieren und sie dann auch individuellen Personen zuordnen. Die im Rahmen dieses Promotionsvorhabens im Berichtszeitraum durchgeführten Arbeiten waren (in ihrem wesentlichen Teil) ebenfalls auf eine solche Klasse von ausgewählten Erkennungsaufgaben - die Erkennung planarer Objekte - konzentriert.

Als Testobjekte wurden dazu Piktogramme verwendet, die zur Orientierung von Reisenden auf Flughäfen verwendet werden. Eine mögliche Anwendung der automatischen Erkennung dieser Piktogramme ist die Positionsbestimmung und Navigation mobiler Roboter. Da Piktogramme in vielen technischen Umgebungen in großer Zahl auftauchen, kann die Möglichkeit ihrer automatischen Erkennung die Installation spezieller Landmarken für die Navigation von Robotern überflüssig machen.

Planare Objekte wurden zur Erkennung ausgewählt, da die Deformation ihres 2D-Abbildes bei Translationen und Rotationen im dreidimensionalen Raum mathematisch recht einfach beschrieben werden kann (insbesondere bei der Vernachlässigung perspektivischer Effekte). Außerdem müssen bei ihrer Erkennung keine Selbstverdeckungen von Objekten berücksichtigt werden, da diese nicht vorkommen.

In Abschnitt 2 dieses Berichtes wird zunächst eine kurze Einführung in bisherige Arbeiten zur Erkennung planarer Objekte gegeben. Ein neu entwickelter Algorithmus zur Detektion von Bitangenten an zwei Bildregionen wird anschließend in Abschnitt 3 beschrieben.

Zur effizienten Erkennung von Objekten ist es sinnvoll, nur ausgewählte Modellmerkmale zur Erzeugung von Objekthypothesen zu verwenden. In Abschnitt 4 wird gezeigt, wie diese Auswahl durchgeführt und verteilt implementiert werden kann. Die Leistungsfähigkeit der so generierten Objekterkennungsprogramme wird in Abschnit 5 zusammengefaßt. In Abschnitt 6 wird schließlich auf laufende Arbeiten zur Erkennung nichtplanarer Objekte eingegangen.

2 Erkennung planarer Objekte

In den durchgeführten Arbeiten wurde vorausgesetzt, daß ein zu erkennendes planares Objekt (annähernd) beliebig im Raum verschoben und rotiert sein kann. Viele in der Vergangenheit durchgeführte Arbeiten machen stärkere Annahmen. Häufig sind nur Translationen und Rotationen parallel zur Bildebene zugelassen (euklidische Transformationen, 3 freie Parameter). In anderen Fällen sind auch Änderungen der Objektgröße möglich (Ähnlichkeitstransformationen, 4 freie Parameter). Bei beliebigen Rotationen im Raum existieren dagegen 6 freie Parameter, die die Lage eines Objektes bestimmen.

Die Beschreibung der Transformation eines Objektpunktes über Objekttranslation und -rotation und Kameraprojektion in die Bildebene läßt sich erheblich vereinfachen, wenn man perspektivische Effekte vernachlässigt. Dies ist dann ohne größere Fehler möglich, wenn die Differenzen in der Objekttiefe (= Abstand zur Kamera) zwischen einzelnen Objektpunkten klein im Vergleich zum mittleren Abstand zwischen Objekt und Kamera sind. Dies war bei den verwendeten Bildern von Piktogrammen der Fall. Die Transformation von Objektpunkten des Objektmodelles (z.B eine gegebenen Ansicht) zu einer Bildansicht läßt sich in diesem Fall durch eine affine Abbildung beschreiben.

$$\begin{pmatrix} x_{img} \\ y_{img} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_{mod} \\ y_{mod} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Dieser einfache Zusammenhang wurde in vielen bisherigen Arbeiten ausgenutzt, indem zu einer gegebenen Bildregion oder -kurve Merkmale berechnet wurden, die invariant gegenüber affinen Transformationen sind. Beispiele dafür sind Fourierdeskriptoren oder Momente höherer Ordnung. Diese Verfahren haben allerdings verschiedene Nachteile. Beispielsweise ist die Merkmalsberechnung häufig sehr anfällig gegenüber kleineren Unregelmäßigkeiten der Form der Objekte, die bei der Aufnahme und Vorverarbeitung realer Objekte zwangsläufig auftreten.

Aus diesen Gründen wurde ein anderer Erkennungsansatz gewählt, der bereits erfolgreich in anderen Arbeiten (z.B. [RZFM95]) eingesetzt wurde. Beim Erkennungsverfahren "Hypothese und Test" werden zur Hypothesenbildung Merkmalspunkte aus einem Objektmodell und dem vorliegenden Bild einander zugeordnet. Unter der Annahme, daß die Bildpunkte den zugeordneten Modellpunkten entsprechen, kann bei einer genügend großen Anzahl von solchen Korrespondenzen die Lage des Objektes in der Bildszene berechnet werden. Im vorliegenden Fall sind dazu 3 Korrespondenzen erforderlich. Aufgrund der berechneten Objekthypothese kann mit Hilfe des Objektmodells die Position weiterer Merkmale im Bild vorhergesagt werden. Indem im Bild nach diesen Merkmalen an der entsprechenden Position gesucht wird, kann eine Objekthypothese bestätigt oder verworfen werden.

Dieses Verfahren wird in Abbildung 1 veranschaulicht. Als Merkmalspunkte wurden lokale Maxima der Kurvenkrümmung verwendet.

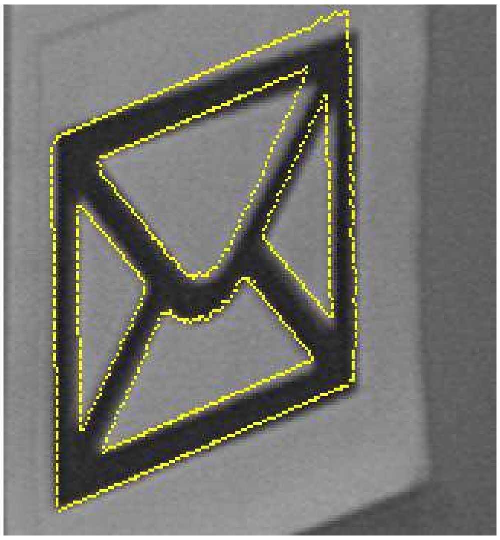


Abbildung 1: Beispiel für Hypothesengenerierung

Dieses Verfahren hat jedoch ebenfalls Nachteile. Falls beliebige Tripel von Modellmerkmalspunkten und Bildmerkmalspunkten verwendet werden, können sehr viele Objekthypothesen aufgestellt werden ($0(n^3m^3)$; n: # Objektmerkmalspunkte, m: # Bildmerkmalspunkte). Außerdem ist es auch recht schwierig, Merkmalspunkte zu finden, die unabhängig von der Objektlage stabil erkannt werden. Beispielsweise sind Krümmungsmaxima von Kurven i.A. nicht invariant gegenüber affinen Transformationen.

Diese Schwierigkeiten können zum größten Teil durch die Verwendung von sogenannten Bitangenten zur Berechnung von Merkmalspunkten vermieden werden [LSW88] [RZFM95]. Bitangenten berühren eine Kurve an zwei verschiedenen Punkten. Sie treten deshalb nur bei Kurven mit Konkavitäten auf. Da Tangentialität invariant gegenüber perspektivischer Projektion ist, sind auch Bitangenten und insbesondere die beiden Berührungspunkte einer Bitangente invariant. Falls die perspektivische Projektion durch eine affine Transformation angenähert werden kann, läßt sich auch noch ein dritter invarianter Merkmalspunkt finden. Dies ist der Punkt auf der Kurve zwischen den beiden Berührungspunkten mit maximalem Abstand zur Bitangente (siehe Abbildung 2). Diese Merkmalspunkte sind nicht nur theoretisch invariant, sondern in der Praxis auch recht unempfindlich gegenüber verrauschten Daten (z.B. durch Kurvenquantisierung). In [Bü96b] wurde ein Algorithmus zur Berechnung von Bitangenten beschrieben.

Ein weiterer wichtiger Vorteil bei der Verwendung von Bitangenten ist, daß sie Gruppen von 3 Merkmalspunkten liefern. Zur Generierung einer Objekthypothese ist es daher ausreichend, eine Bitangente (mit 3 Merkmalspunkten) aus dem Objektmodell mit einer Bitangente im Bild in Korrespondenz zu setzen. Die Komplexität der Hypothesengenerierung beträgt daher nur $O(nm)$ (n: # Bitangenten im Modell, m: # Bitangenten im Bild).

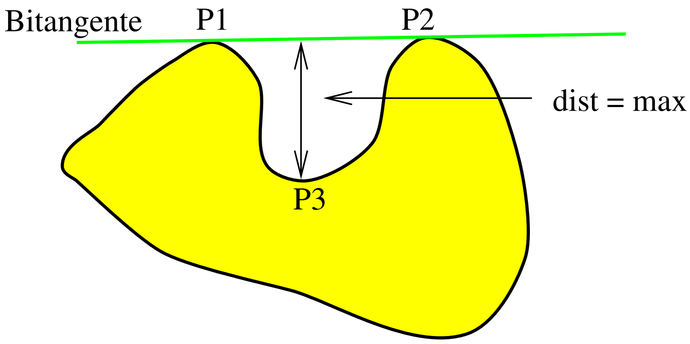


Abbildung 2: Bitangente mit Merkmalspunkten

3 Bitangenten an zwei Bildregionen

Leider finden sich Bitangenten nur an konkaven Bildregionen. Um ihre Vorteile auch für konvexe Bildregionen nutzen zu können, wurde im Rahmen der durchgeführten Arbeiten die Verwendung von Bitangenten an Paaren von Bildregionen untersucht. Insbesondere wurde ein effizienter Algorithmus zur Berechnung der Bitangenten entwickelt. Für ein Paar von nicht überlappenden konvexen Regionen existieren 4 Bitangenten (siehe Abbildung 3). Die Berührungspunkte sind wiederum invariant gegenüber perspektivischen (und affinen) Transformationen.

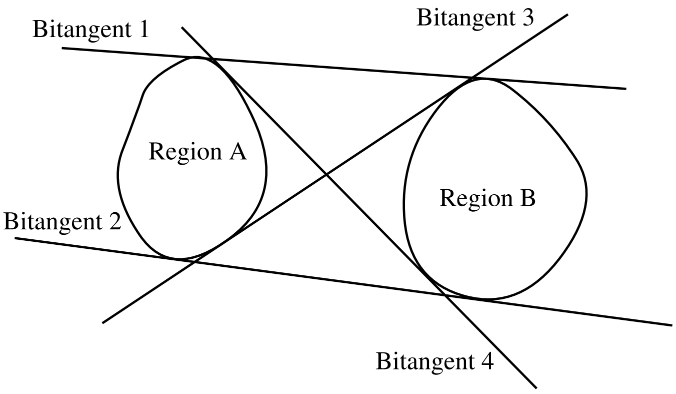


Abbildung 3: Ein Paar von konvexen Regionen und ihre 4 Bitangenten

Eine naheliegende Methode zur Bestimmung der Bitangenten 1 und 2 ist die Berechnung der gemeinsamen konvexen Hülle der Bildregionen A und B. Im Rahmen eines Algorithmus zur Berechnung der konvexen Hülle von beliebigen Punktmengen wurde in [PH77] gezeigt, wie dies mit einem Zeitaufwand von $O(p+q)$ (p,q: # Eckpunkte der zwei konvexen Polygone) möglich ist. Dieser Algorithmus kann in modifizierter Form auch zur Berechnung der Bitangenten 3 und 4 eingesetzt werden.

Der im Rahmen der durchgeführten Arbeiten entwickelte Algorithmus berechnet die 4 Bitangenten mit einem mittleren Zeitaufwand von $O(log^2(n))$. Dabei ist n die maximale Anzahl von Eckpunkten der konvexen Regionen A und B. Der Algorithmus ist iterativ:

1. Wenn A und B nicht konvex sind, ersetze sie durch ihre konvexe Hülle.

2. Wähle zufällig einen Punkt P_A aus A und einen Punkt P_B aus B (z.B. die Schwerpunkte).

3. Iteriere:

4. Berechne die Parameter einer Linie L, die die Punkte P_A und P_B verbindet. Finde neue Punkte P'_A und P'_B auf der konvexen Hülle von A bzw. B mit maximalem Abstand von L in der Richtung, die der zu findenden Bitangente entspricht. Diese Richtung ist relativ zur Richtung von L von A nach B: Bitangente 1: A links, B links; Bitangente 2: A rechts, B rechts; Bitangente 3: A rechts, B links; Bitangente 4: A links, B rechts.

5. Ersetze P_A durch P'_A und P_B durch P'_B. Falls $P_A \neq P'_A$ oder $P_B \neq P'_B$, starte eine neue Iteration.

Für Beweise der Korrektheit des Algorithmus und seiner Komplexität siehe [Bü96a]. Eine Iteration des Algorithmus ist in Abbildung 4 veranschaulicht.

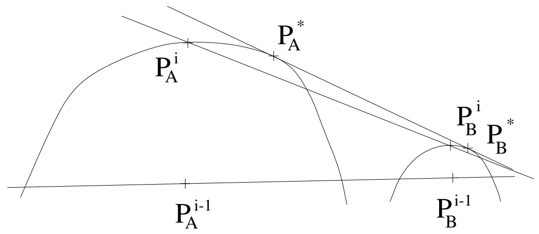


Abbildung 4: Eine Iteration des entwickelten Algorithmus

4 Verteilte Merkmalsauswahl

Mit Krümmungsmaxima, einfachen Bitangenten und Bitangenten an Paaren von Bildregionen steht ein relativ breites Repertoire von Merkmalspunkten zur Generierung von Objekthypothesen zur Verfügung. Im Abschnitt 2 wurde bereits dargelegt, daß die Anzahl der möglichen Objekthypothesen von der Größenordnung $O(nm)$ ist, wobei n und m die Anzahl der Tripel von Merkmalspunkten in Modell und Bild sind. Die Anzahl der Tripel von Krümmungsmaxima kann eingeschränkt werden, indem nur Tripel betrachtet werden, die auf einem Kurvenstück unmittelbar aufeinander folgen. Trotzdem bleibt die Anzahl möglicher Tripel hoch und nimmt beispielsweise für Bitangenten an Paaren von Modellregionen mit dem Quadrat der Anzahl der Modellregionen zu. Zur korrekten Erkennung reicht jedoch ein Tripel von Modellpunkten aus, das im Bild wiedergefunden wird. Um zur Hypothesengenerierung nur eines oder wenige Modellmerkmalstripel verwenden zu können, muß allerdings sichergestellt werden, daß diese Punkte stabil in allen in der Praxis auftauchenden Objektlagen detektiert werden können. Zumindest eines dieser Modelltripel muß in jeder Ansicht an der Position vorhanden sein, die durch die vorgegebene Objektlage vorhergesagt wird. Um solche stabilen Merkmale auswählen zu können, muß ihre Stabilität quantitativ bewertet werden. Dazu gibt es zwei grundsätzliche Möglichkeiten:

1. Heuristisch, z.B.: Bitangenten an flachen Konkavitäten verschwinden bei stärkeren Verkleinerungen oder bei entsprechenden Verzerrungen. Gute Objekthypothesen werden durch möglichst weit voneinander entfernte Merkmalspunkte gebildet.

2. Empirisch: In einer Menge von Trainingsbildern mit bekannten Objektlage werden Merkmale detektiert. Es werden die Merkmale ausgewählt, die in möglichst vielen Trainingsbildern zu finden sind.

Die heuristische Vorgehensweise hat den Nachteil, daß der Prozeß der Merkmalsdetektion komplex ist und aus mehreren Schritten besteht. Dazu gehört z.B. auch die Segmentierung des Bildes in Regionen. Es ist schwierig, das Resultat und die Stabilität dieses Prozesses mit einigen einfachen Regeln zu beschreiben. Daher wurde der empirische Ansatz gewählt. Um zu einer für die Erkennung relevanten Schätzung der Merkmalsstabilität zu gelangen, wird dabei analog zum Verfahren zur Verifikation von Hypothesen vorgegangen.

Dieses verläuft folgendermaßen: Position und Orientierung der Modellkantenelemente werden entsprechend der Objekthypothese in Bildkoordinaten transformiert. Zu jedem Kantenelement wird bestimmt, ob im Bild ein Kantenelement mit vergleichbarer Orientierung innerhalb eines maximalen Abstands vorliegt. Um diesen Vergleich effizient durchzuführen, wird wie in [RZFM95] eine Distanztransformation der Bildkanten verwendet. Die Qualität einer Objekthypothese wird als Anteil der gematchten Modellkantenelemente berechnet. Liegt dieser Wert über einer vorgegebenen Schwelle, so wird angenommen, daß die Objekthypothese korrekt ist. Die Qualität von zwei Objekthypothesen wird in Abbildung 5 veranschaulicht.

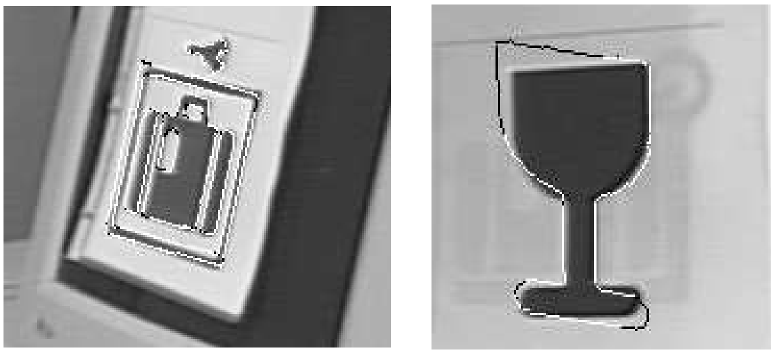


Abbildung 5: Die Qualität zweier Objekthypothesen (gefundene Modellkantenelemente: weiß): Anteil im linken Piktogramm (Bitangenten an zwei Regionen): 0,99; im rechten Piktogramm (Krümmungsmaxima): 0,73

Zur Auswahl von Modellmerkmalen wird für jede mögliche Gruppe von Modellmerkmalspunkten (einfache Bitangenten, Bitangenten an Paaren von Regionen, Punkte maximaler Krümmung) und jedes Trainingsbild eines Objektes die Qualität der korrekten Merkmalshypothese bestimmt. Auf der Grundlage dieser Informationen werden eine oder mehrere

# Prozessoren	1	2	4	8
Rechenzeit (s)	1109	563	295	222

Tabelle 1: Trainingszeiten für 16 Trainingsbilder und unterschiedlich Grade von Parallelität

Gruppen von Modellmerkmalspunkten - hier auch "Startmerkmale" genannt - ausgewählt, so daß die minimale Hypothesenqualität für das beste Startmerkmal für alle Trainingsbilder $M_{mingood} = \min_{im \in Im} \max_{f \in F} Hypothesenqualität(f, im)$ (Im: Trainingsbilder, F: ausgewählte Merkmale) einen vorgegebenen Schwellwert überschreitet. Dabei wird eine "Greedy"-Suchstrategie eingesetzt.

Anschließend wird für die ausgewählte Menge von Merkmalen ein Schwellwert bestimmt, bei dessen Überschreitung davon ausgegangen wird, daß eine Objekthypothese korrekt ist. Dazu wird das Objekt mit den ausgewählten Merkmalen in einer Reihe von Bildern gesucht, in denen das Objekt nicht auftaucht. Dabei wird die maximale Qualität der dabei erzeugten (falschen) Hypothesen berechnet. Der endgültige Schwellwert wird dann heuristisch zwischen dem zuletzt bestimmten Wert und der minimalen Hypothesenqualität $M_{mingood}$ so festgelegt, daß korrekte und falsche Hypothesen möglichst gut getrennt werden.

Die Mehrzahl der Berechnungen zur Merkmalsauswahl sind unabhängig voneinander. Sie können daher ohne Probleme verteilt ausgeführt werden. Zur Implementierung auf einer Gruppe von Workstations wurde eine Verteilung nach dem Master-Slave-Prinzip gewählt. Dabei sind die Berechnungen entsprechend der zu bearbeitenden Bilder verteilt. Für jedes Bild berechnet ein Arbeiterprozess die Qualität der Hypothesen, die mit allen möglichen Startmerkmalen erzielt werden kann. Falls die Anzahl der Trainingsbilder beträchtlich größer als die Anzahl der verwendeten Prozessoren ist, wird dadurch ein sehr gute Beschleuningung der Berechnungen möglich (siehe Tabelle 1).

5 Experimentelle Resultate

Zur Validierung der beschriebenen Methode zur Merkmalsauswahl wurden 24 Piktogramme verwendet. Als Trainingsbilder wurden künstlich transformierte Versionen einer einzelnen Modellansicht verwendet. Für jedes der 24 Piktogramme wurden Merkmale ausgewählt und ein Schwellwert für die Objektdetektion bestimmt. Zum Test wurden zu jedem Piktogramm 10 reale Testbilder verwendet, in denen das Piktogramm in verschiedenen Lagen zu sehen war.

Tabelle 2 zeigt die Ergebnisse der Erkennung in diesen Testbildern. Die Mehrzahl der Piktogramme wurde vollständig zuverlässig erkannt. Einige Objekte wurden in einzelnen Bildern nicht erkannt oder an Stellen erkannt, an denen sie nicht vorhanden waren (falsche Positive). Fast alle Probleme bei der Erkennung sind auf Instabilitäten bei der Segmentierung dünner Linien oder kleiner Bildregionen zurückzuführen.

Die Zeit für die Erkennung wurde auf einer HP K-260 Workstation (180 MHz) gemessen. Die Segmentierung in Bildregionen dauerte für die Testbilder der Größe 768×576 Bildelemente 3-4 Sekunden. Dieser Verarbeitungsschritt läßt sich durch eine Parallelisierung basierend auf der Arbeit des Mitkollegiaten M. Lückenhaus noch beschleunigen. Für

# gefunden (max. 10)	10	9	8-7	<7
# Piktog.	17	2	3	2

# falsche Positive	0	1-2	3-4	5-19	≥ 20
# Piktog.	16	2	2	3	1

Tabelle 2: Erkennungsergebnisse

die nachfolgende Phase der Hypothesengenerierung und -verifikation wurden 1-3 Sekunden benötigt. Für alle 24 Piktogramme wurden 37 Startmerkmale aus 996 möglichen Merkmalen ausgewählt. Ohne die Merkmalsauswahl hätte die Suche nach der korrekten Hypothese daher durchschnittlich 26,9 mal so lange gedauert.

Abbildung 6 zeigt die ausgewählten Merkmale für einige Piktogramme. Insgesamt war ein sehr großer Teil der ausgewählten Merkmale vom dem in den beschriebenen Arbeiten erstmals untersuchten Typ "Bitangente an einem Regionenpaar".

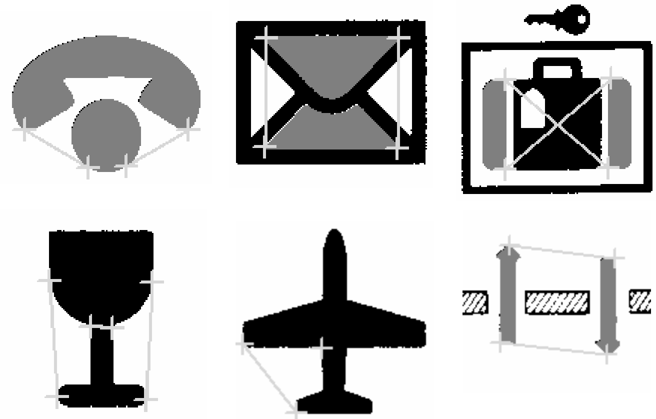


Abbildung 6: Beispiele für ausgewählte Merkmale: Bitangenten an Regionenpaaren, bis auf "Glas" und "Flugzeug" (einfache Bitangenten)

6 Laufende Arbeiten

Neuere Arbeiten konzentrieren sich auf die Erkennung von Objekten einer anderen Klasse. Dabei sollen nichtplanare Objekte anhand ihrer Kontur identifiziert werden. Die Objekte sind starre Gegenstände wie Flaschen oder Kannen. Da viele der Objekte eine gewölbte Form haben, sind sonst häufig verwendete dreidimensionale Kantenmodelle der Objekte für die Erkennung wenig hilfreich. Stattdessen wird eine ansichtenbasierte Repräsentation der Objekte (vgl. [Pop95]) verwendet. Hierbei wird die geometrische Information über die Form eines Objektes in einer Reihe von Ansichten des Objektes aus verschiedenen Blickrichtungen repräsentiert. Für die Erkennung eines Objektes wird die Bildinformation (hier: die Kontur) des Objektes mit den gespeicherten Ansichten aller bekannten Objekte verglichen.

Selbstverständlich gibt es zu jedem Objekt eine unendlich große Zahl von unterschiedlichen Ansichten. Für ein praktikables System muß daher ein ganzer Berich von Ansichten (=Blickwinkeln) durch einen einzigen Prototypen repräsentiert werden. Im geplanten Erkennungssystem soll in einem Prototyp die Position einer Reihe von Objektmerkmalen relativ zu einem Basismerkmal dargestellt werden. Damit der Prototyp einen ganzen Bereich von Blickwinkeln auf das Objekt repräsentieren kann, sind diese Positionen nicht einzelne Koordinaten sondern größere Positionsbereiche (siehe Abbildung 7).

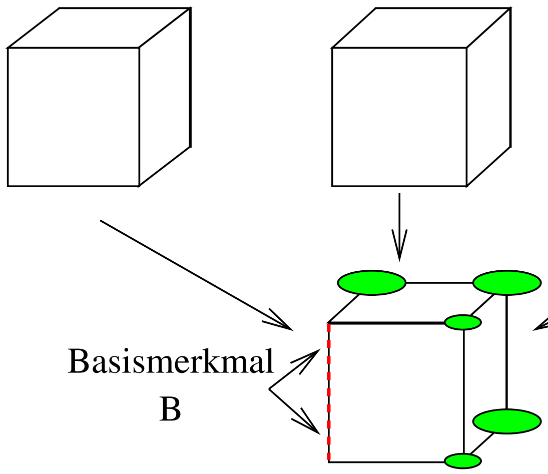


Abbildung 7: Zusammenfassung mehrerer Ansichten zu einem Prototypen; der Positionsbereich einiger Objektecken relativ zu einem Basismerkmal ist durch Ellipsen angedeutet

Bei der Generierung dieser Prototypen aus einer vorgegebenen Menge von Ansichten eines Objektes müssen zwei wesentliche Ziele beachtet werden. Zum einen soll die Anzahl der Prototypen möglichst gering sein, um die Erkennungszeit und den benötigten Speicheraufwand gering zu halten. Zum anderen dürfen die Positionsbereiche der Objektmerkmale nicht zu groß werden, damit die Objekte deutlich unterscheidbar bleiben. Beim entworfenen Verfahren zur Bildung der Prototypen ist ein schrittweises Wachstum des Bereichs von Blickwinkeln, die ein Prototyp repräsentiert, geplant. Dieses Wachstum wird durchgeführt, solange der Prototyp gut genung zwischen den repräsentierten Objektansichten und anderen Konturen unterscheiden kann. In jedem dieser Wachstumsschritte ist ein Vergleich des Prototyps mit einer größeren Anzahl von Konturen anderer Objekte erforderlich. Dieser beträchtliche Rechenaufwand wird nur in einer verteilten Implementierung mit einem annehmbaren Zeitaufwand möglich sein.

In den bisher durchgeführten Arbeiten wurde ein Ähnlichkeitsmaß für Konturen entwickelt. Wie in [Pop95] wird dabei Evidenz für die Ähnlichkeit von Objekten aufgrund der Bayesschen Regel gesammelt. Das entwickelte Verfahren ist allerdings einfacher und macht bisher einen recht leistungsfähigen Eindruck. Da der Algorithmus Schätzungen von Wahrscheinlichkeitsdichtefunktionen verwendet, können hier evtl. Ergebnisse der Arbeiten des Mitkollegiaten D. Ormoneit verwendet werden. Wie bei den Arbeiten zur Erkennung planarer Objekte werden auch bei diesem Erkennungsansatz Bitangenten verwendet. Abbildung 8 zeigt Beispiele für die Anwendung des Ähnlichkeitsmaßes auf eine Datenbank von 16 Objekten.

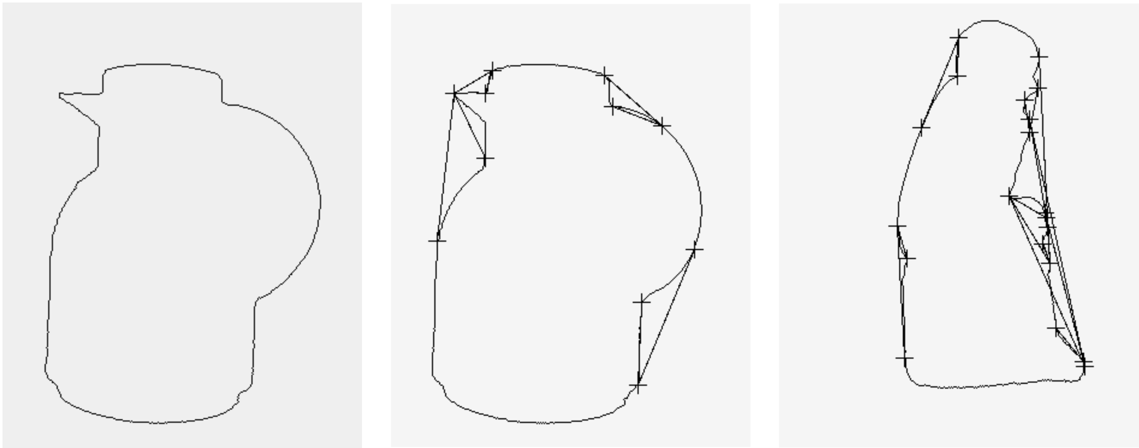


Abbildung 8: Links: Kontur einer Kanne; Mitte: Ähnliche Kontur - Ähnlichkeitswert 89,9 (mit Bitangenten); Rechts: Kontur mit höchstem Ähnlichkeitswert, die nicht von einem anderen Objekt stammt - Ähnlichkeitswert 26,0 (eine Statue einer sitzenden Person)

Literatur

[Bü96a] D. Büsching. Efficient pictograph detection using bitangents. Interner bericht, Technische Universität München, Forschungsgruppe Bildverstehen, 1996.

[Bü96b] D. Büsching. Efficiently finding bitangents. In *International Conference on Pattern Recognition*, S. 428–432, Wien, August 1996.

[LSW88] Y. Lamdan, J.T. Schwartz und H.J. Wolfson. Object recognition by affine invariant matching. In *Proc. CVPR*, S. 335–344, 1988.

[PH77] F.P. Preparata und S.J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, Februar 1977.

[Pop95] A.R. Pope. *Learning to recognize objects in images: acquiring and using probabilistic models of apperance.* Dissertation, University of British Columbia, December 1995.

[RZFM95] C.A. Rothwell, A. Zisserman, D.A. Forsyth und J.L. Mundy. Planar Object Recognition using Projective Shape Representation. *International Journal of Computer Vision*, 16:57–99, 1995.

1.3.3 Multi-Agenten Kooperation in Konkurrenzszenarien

Teilprojekt: Kooperierende Agenten und autonome Robotersysteme
Florian Fuchs

1 Einleitung

Multi-Agenten Systeme werden in verschiedensten Bereichen zunehmend als erfolgversprechende Lösungsansätze gesehen. Ihr Einsatz bietet sich oftmals natürlicherweise an, insbesondere wenn geographische Verteiltheit oder konträre Ziele der Teilkomponenten eines Systems dafür sprechen. Für die meisten dieser Systeme wird vorausgesetzt, daß sich die Agenten kooperativ verhalten und entsprechend ihren Fähigkeiten zu einem gemeinsamen Ziel beitragen. Diese Agenten besitzen ein dafür ausgerichtetes und aufeinander abgestimmtes Design.

Neben Agentengesellschaften, die einen überwiegend kooperativen Charakter haben, existieren auch solche, in denen die Agenten Aufgaben für vollkommen unabhängige Auftraggeber wahrnehmen. Es gibt Szenarios, in denen keine globale Zielfunktion existiert und in denen die Agenten ausschließlich ihre lokalen Ziele verfolgen. Kooperation findet hier nur dann statt, wenn sich alle an der Kooperation beteiligten Agenten davon Profit versprechen. Wir bezeichnen solche Szenarien als Konkurrenzszenarien. Auf diesen liegt der Fokus der Arbeit.

Das Hauptziel der Arbeit liegt in der algorithmischen Erfassung eines für Konkurrenzszenarien geeigneten Konfliklösungsmechanismus. Weiterhin soll untersucht werden, wie sich die lokale Strategiewahl eines Agenten auf seine Position in der Agentengesellschaft auswirkt und welches Verhalten des Gesamtsystems daraus resultiert, insbesondere wie sich gegebenenfalls Gleichgewichtszustände etablierern können. Entsprechende Untersuchungen werden anhand einer Anwendung aus dem Bereich der verteilten Ressourcenallokation durchgeführt.

Die anschließenden Abschnitte des Berichts gestalten sich wie folgt: Im Abschnitt 2 werden zunächst die Konsequenzen des Konkurrenzparadigmas diskutiert. Der Abschnitt 3 beschäftigt sich mit der Modellierung des Anwendungsszenarios und den Grundlagen für die Kompromißfindung, die in Abschnitt 4 beschrieben ist. Abschließend wird in Abschnitt 5 kurz auf die Implementierungsarbeiten eingegangen und in Abschnitt 6 erfolgt eine Zusammenfassung der Arbeit und eine Einordnung in die Thematik des Graduiertenkollegs.

2 Konkurrierende Agenten

Ein typisches Charakteristikum für Multi-Agenten Systeme ist, daß die Agenten aufgrund beschränkter Informationsausbreitung und unsicherer Perzeption von Information nur über eine eingeschränkte Sichtweise hinsichtlich des globales Systemzustandes verfügen. Desweiteren verfolgen die Agenten typischerweise ihre lokalen Ziele. Diese beiden Eigenheiten — die in zentral ausgerichteten Problemlösungsansätzen kaum eine Rolle spielen — haben zur Konsequenz, daß Zielkonflikte zwischen den Agenten auftreten, die gelöst werden müssen.

Dies ist insbesondere in Konkurrenzszenarien evident, da hier das Ausmaß an gemeinsamen Wissen besonders gering ist. Im allgemeinen kennen die Agenten nicht die Pläne, Ziele, Strategien usw. der anderen Agenten. Ein Agent stellt diese Art von Information nicht zur Verfügung, um zu verhindern, daß die anderen daraus Kapital schlagen können. Daher verbleibt nur die Möglichkeit, Informationen aus den für Kooperationen notwendigen Verhandlungen zu extrahieren und daraus Modelle der Konkurrenten zu bilden. Diese Modelle können dann nur unsicher und unvollständig sein.

Desweiteren ist in Konkurrenzszenarien auch nicht sichergestellt, daß Agenten Information wahrheitsgemäß weitergeben. Interessante Fragestellungen aus diesem Bereich werden in [PR94] untersucht.

Konkurrenzszenarien weisen einige Eigenheiten auf, die an die Konfliktlösungsmethode spezielle Anforderungen stellen. Methoden, die in rein kooperativen Szenarien überwiegend eingesetzt werden, sind i.d.R. ungeeignet. Nicht einsetzbar sind etwa Techniken, die Konfliktlösungsstrategien fest vorgeben oder eine gemeinsame Strategie für die Konfliktparteien über Verhandlung ermitteln (siehe z.B. [LLC91]). Es kann auch nicht vorausgesetzt werden, daß die Agenten identische Konfliktlösungskomponenten besitzen (siehe z.B. [KB91]), oder daß ein Agent eine spezielle Rolle im Konfliktlösungsprozess einnimmt, sei es als Agent mit bestimmender Funktion (siehe z.B. [PG92]) oder als Mittleragent (siehe z.B. [Wer90], [SRSF90]). Für eine nähere Beschreibung der dort eingesetzten Konfliktlösungstechniken sei auch auf [Fuc96] verwiesen.

Als wichtigste Anforderungen an einen Konfliktlösungsmechanismus, der für Konkurrenzszenarien geeignet ist, sind folgende zu nennen:

- **Symmetrie**
  Keine der Konfliktpartien nimmt eine spezielle Rolle im Konfliktlösungsprozeß ein. Die Konkurrenten agieren gleichberechtigt.
- **Private Strategien**
  Die Konfliktlösungsstrategien müssen privat sein in dem Sinne, daß sie von jeder Konfliktpartei frei gewählt werden können und vor dem Zugriff durch die anderen Agenten geschützt sind.
- **Keine Mittlerinstanzen**
  Die Agenten lösen Konflikte auf ihrer Ebene ohne die Hilfe von übergeordneten Instanzen.

Desweiteren existieren Anforderungen, die nicht spezifisch für Konkurrenzszenarien sind, sondern einen allgemeineren Charakter haben und für andere Szenarien ebenfalls eine Rolle spielen. Die wichtigsten sind folgende:

- **Stabilität**
  Der Konfliktlösungsmechanismus sollte kein instabiles oder chaotisches Verhalten des Gesamtsystems begünstigen, das aus verzögerter Wahrnehmung und Verarbeitung von Information resultieren kann (siehe [HH88]).
- **Adaptivität**
  Eine wesentliche Eigenart eines Marktes ist seine starke Dynamik. Die Ziele der

Agenten sind einer kontinuierlichen Veränderung unterworfen. Deshalb muß ein Agent die Fähigkeit haben, geeignete Strategien zu wählen, geprägt durch die eigenen Ziele, die Modelle, die er von den Konkurrenten besitzt, und den Anforderungen von Seiten seiner Auftraggeber.

- **Effizienz**
  Die Bewertung von Vertragsvorschlägen ist im allgemeinen nicht trivial und sehr komplex. Zudem ist die zur Verfügung stehende Zeit gerade auch aufgrund von Konkurrenzdruck begrenzt. Deshalb sind dafür effiziente Verfahren zu finden.

3 Anwendungsszenario

Die Umgebung, in der sich die Agenten befinden, ist ein Szenario aus dem Bereich der verteilten Ressourcenallokation. Aufträge, die von externen Auftraggebern in das betrachtete Agentensystem eingelastet werden, sollen unter Berücksichtigung von verschiedenen zeitlichen Aspekten den vorhandenen Ressourcen zugeordnet werden. Das Szenario beinhaltet somit ein verteiltes Scheduling Problem, das von Agenten gelöst werden soll, die in Konkurrenz zueinander stehen. Die beteiligten Agenten sind also bestrebt, den Teil eines globalen Zeitplanes, der im Zusammenhang zu den von ihnen zugeteilten Ressourcen steht, nach ihren lokalen Bewertungskriterien optimierend zu gestalten.

Wir betrachten dabei wertorientierte Domänen, in denen auch eine partielle Erfüllung von Zielen möglich ist und damit eine Kompromißbildung im Falle eines Zielkonfliktes. Im Gegensatz dazu stehen aufgaben- und zustandsorientierte Domänen, in denen im Falle eines Konflikts eine Partei auf die Erreichung ihres Zieles verzichten muß.

3.1 Scheduling-Modell

Das zugrundegelegte Scheduling-Modell ist ähnlich zu dem in [Win93] vorgeschlagenen Modell eines erweiterten Job Shop. Es umfaßt zum einen einen Menge von *Ressourcen*, die in *Ressourcengruppen* gegliedert sind, die jeweils zu einer bestimmten *Aktivität* korrespondieren. Zum anderen existieren *Tasks*, die über eine partielle Ordnung zu *Jobs* gruppiert werden, die jeweils eine Menge von Ressourcengruppen erfordern. Das Modell sieht vor, daß Jobs dynamisch in das System eingelastet werden und in den laufenden Scheduling-Prozeß eingebunden werden müssen (Online-Scheduling). Dieses Modell wird in [Fuc96] näher formalisiert.

Die Repräsentation der Zeitpläne erfolgt nicht über exakte Zeitpunkte oder Zeitintervalle. Stattdessen werden die Zeitpläne mittels Dichtefunktionen über der Zeit für jede Ressource repräsentiert. Diese Dichtefunktionen stellen eine grobgranulare Darstellungsform für die Kapazitätsausnutzung einer Ressource dar und bieten somit Flexibilität hinsichtlich der tatsächlichen Startzeitpunkte der Aktivitäten innerhalb eines vorgegebenen Rahmens. Sie stellen auch einen effizienten Weg zur Bewertung von Ressourcenanforderungen dar. Die Verwaltung von exakten Zeitplänen wäre anbetracht des frühen Planungszustandes und der damit verbundenen großen Planungsunsicherheit der hier betrachteten Planungsebene praktisch unmöglich.

3.2 Agentenmodell

Abb. 1 zeigt das konzeptionelle Agentenmodell, wie es für das Ressourcenallokationsszenario entworfen wurde und Verwendung findet. Der Agent ist eingebettet in seiner Umgebung, die in drei Teile untergliedert ist: Zum einen das *client environment*, in dem neue Aufträge erzeugt und in das System eingelastet werden, dann das *execution environment*, das durch die Ausführung[1] von Aufträgen für deren Entfernung aus dem System sorgt, und letztlich das *agent environment*, über das die Aufträge auf die Agenten verteilt bzw. umverteilt werden.

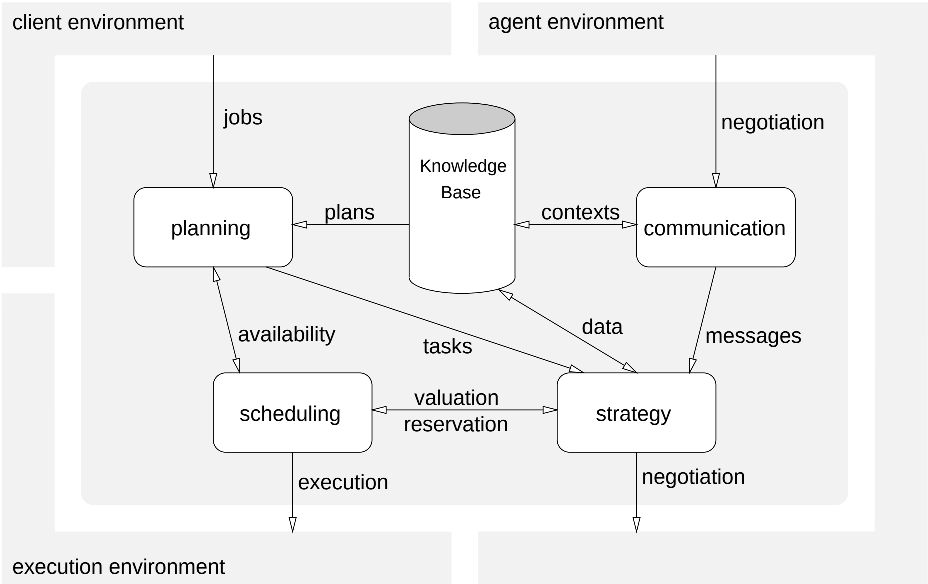


Abbildung 1: Agentenarchitektur

Ein Agent kapselt in sich das Wissen, das er zur Realisierung seiner Ziele benötigt, und außerdem die dafür notwendigen Wissensverarbeitungsmechanismen. Dieses Wissen liegt explizit repräsentiert in einer lokalen Wissensbasis vor, die in erster Linie Wissen über die anderen Agenten, den eigenen internen Zustand und Pläne als Instruktion zur Durchführung von Aufträgen enthält.

Zusätzlich zur Wissensbasis existieren vier interne Agentenkomponenten, die weitgehend unabhängig voneinander operieren und über interne Schnittstellen miteinander verbunden sind.

Die *Planungskomponente* verwaltet eine Menge von Jobs, die in unregelmäßigen Intervallen von Auftraggebern eintreffen. Sie erstellt einen Aktionsplan für jeden Job[2] und ermittelt die Scheduling Reihenfolge für die Einzelschritte eines Jobs.

Ein Agent ist in der Lage, über seine *Kommunikationskomponente* mit anderen Agenten zu kommunizieren. Aus eingehenden Nachrichten extrahiert er Information und übernimmt

[1] Ausführung muß hier nicht unbedingt die physikalische Ausführung bedeuten, sondern soll vielmehr auch als Weitergabe von Aufträgen an untergeordnete Planungsinstanzen verstanden werden.

[2] Derzeit findet in diesem Zusammenhang keine echte Planung statt, es wird vielmehr ein vorgefertigter Plan aus der Wissensbasis entnommen.

sie in die lokale Wissensbasis. Desweiteren wird die Nachricht gemäß ihres Typs und Inhalts in einen bestimmten Kontext eingeordnet. Der Agent übernimmt dann die für diesen Kontext bestimmte Rolle.

Die *Strategiekomponente* trägt den strategischen Anteil zum Konfliktmanagement bei. Diese Komponente generiert u.a. die Antworten auf die eingehenden Vertragsvorschläge (siehe 4).

Die *Schedulingkomponente* verwaltet die Zeitpläne, bewertet Vorschläge für Ressourcenbelegungen, und ist in der Lage, alternative Zeitpläne zu vorgegebenen zu finden. Der Scheduler ist verbunden mit dem execution environment und unternimmt die notwendigen Schritte für die Ausführung[3] der eingeplanten Arbeitsschritte.

3.3 Marktmechanismus als Kooperationsmodell

Die Basis des Kooperationsmodells wird von einem marktorientierten System gebildet (vgl. hierzu auch die Begriffe *Computational Ecology* [HH88], *Agoric Open System* [MD88] und *Marketoriented Programming* [Wel93]). Darunter verstehen wir ein System, das Markmechanismen wie z.B. Verhandlungsplattformen oder Zahlungsysteme zur Verfügung stellt, die von den Teilnehmern an diesem Markt genutzt werden.

Die Teilnehmer an diesem Markt sind Agenten. Diese besitzen Ressourcen, die sie zur Durchführung von Aufträgen benötigen. Im allgemeinen stehen nicht für alle Arbeitsschritte, die für einen Auftrag notwendig sind, die dazu erforderlichen Ressourcen lokal zur Verfügung. Deshalb ist eine geeignete Zerlegung und Verteilung der Aufträge unter den Agenten zu erreichen.

Dazu wird ein Zahlungssystem nach folgendem Muster eingerichtet. Jede Ressource verursacht Kosten. Dies gilt natürlich insbesondere für den Einsatz einer Ressource für die Durchführung eines bestimmten Arbeitsschrittes. Aber auch bei Nichtbenutzung fallen Kosten an. Jegliche Art von Kosten, die eine Ressource verursacht, werden vom Besitzer dieser Ressource getragen. Werden Aufträge von anderen Agenten übernommen, werden also Ressourcen anderen Agenten zur Verfügung gestellt, so werden die Kosten auf die Auftraggeber abgewältzt. Diese abgewältzten Kosten können zusätzlich Benutzungsgebühren enthalten, die für den Gebrauch der Ressourcen erhoben werden. Agenten, die einen Auftrag vergeben, geben ihrerseits alle entstehenden Kosten an ihren Auftraggeber weiter, wobei es sich dabei um einen anderen Teilnehmer am Markt handeln kann oder um einen externen Endverbraucher.

4 Konfliktlösung

Die aufgrund unterschiedlicher lokaler Interessen auftretenden Zielkonflikte werden durch eine beiderseitige Zielrelaxierung — eine Kompromißbildung — eliminiert. Auf diese Zielrelaxierung und die dafür notwendige Verhandlung wird im folgenden eingegangen.

[3] Bzw. deren Simulation.

4.1 Verhandlung

Die Zuordnung der eingelasteten Aufträge zu den Ressourcen erfolgt mittels Verhandlungen. Eine Verhandlung ist charakterisiert durch ein Verhandlungsprotokoll, das zum einen eine Menge von Verhandlungsprimitiven als kommunizierbare Nachrichten im Rahmen einer Verhandlung hinsichtlich Syntax und Sematik definiert und zum anderen die zulässigen Abläufe einer Verhandlung beschreibt.

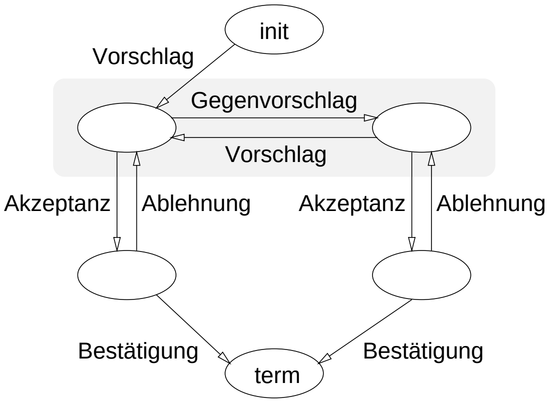


Abbildung 2: Protokoll

Das verwendete Verhandlungsprotokoll stellt eine Erweiterung des Contract Net Protokolls [Smi88] dar. Die starre Abfolge der dort definierten Primitive *request*, *offer*, *order* und *confirmation* wurde ersetzt duch eine Einbettung von abwechselnden *Vorschlag* und *Gegenvorschlag* (siehe Abb. 2) — ein Mechanismus, der für eine Kompromißfindung notwendig ist.

Der Inhalt dieser beiden Verhandlungsprimitive stellt jeweils einen *Vertragsvorschlag* dar, der sich aus zwei Teilen zusammensetzt. Er beinhaltet erstens eine Spezifizierung des Vertragsgegenstandes — im wesentlichen die in Frage stehende Ressource, die belegt werden soll — und zweitens verhandelbare *Vertragsbedingungen*, z.B. der Belegungszeitraum oder der für die Belegung erhobene Preis. Über diese Vertragsbedingungen wird im Verlauf der Verhandlung entweder eine Einigung erzielt und ein *Vertrag* kann zustande kommen, oder die Verhandlung scheitert, weil die unterschiedlichen lokalen Ziele unvereinbar sind.

An jeder Verhandlung gemäß diesem Protokoll nimmt jeweils ein *Auftraggeber* und ein *Auftragnehmer* teil. Auftraggeber und Auftragnehmer sind *Rollen*, die von Agenten temporär übernommen werden.

Multilaterale Verhandlung

Ein Agent kann bei einer Auftragsvergabe offensichtlich einen Vorteil erlangen, wenn es ihm möglich ist, mehrere Verhandlungen betreffend eines Verhandlungsgegenstandes mit unterschiedlichen Auftragnehmern simultan zu führen. Er kann dadurch Teilergebnisse, die er in einer dieser voneinander abhängigen Verhandlungen gewonnen hat, in anderen zu seinem Vorteil nutzen. Eine solche *multilaterale Verhandlung* erfordert Koordination zur Nutzbarmachung von Teilergebnissen und Synchronisation zur Sicherstellung eines eindeutigen Verhandlungsabschlusses.

4.2 Kompromißfindung

Die Agenten sind geprägt durch eine rationale Verhaltensweise. Rationalität ist genau dann gegeben, wenn ein Agent zum Ziel hat, seinen eigenen Profit zu steigern, der sich über seine lokale Nutzenfunktion definiert.

Der *Nutzen*, den er aus einem Vertragsschluß ziehen kann, wird hauptsächlich von zwei Einflußfaktoren bestimmt. Zum einen entstehen dem Agenten *Kosten* aus den Verpflichtungen, die er bei einem Vertragsabschluß übernimmt. Der Auftraggeber verpflichtet sich, den ausgehandelten Preis zu zahlen und dem Auftragnehmer entstehen Kosten durch die Belegung der ihm zugeordneten Ressourcen. Zum anderen besitzt jeder Vertrag jeweils einen bestimmten *Wert* für die Vertragspartner. Der Auftragnehmer erhält den ausgehandelten Preis und der Auftraggeber die für die termingerechte Erfüllung seiner Verpflichtungen notwendige Leistung. Der resultierende Nutzen ergibt sich aus der Differenz von Wert und Kosten.

Daneben existieren noch zwei weitere Punkte, die sich auf den Nutzen eines Vertrages für einen Agenten auswirken. Ressourcen verursachen im allgemeinen nicht nur zu Belegungszeiten Kosten sondern auch im Ruhezustand. Für die Überschreitung von in Verträgen festgelegten Terminen werden Geldstrafen erhoben. Beide Punkte wirken sich als Anreiz aus, an Verhandlungen teilzunehmen und Kompromißbereitschaft zu zeigen.

Kosten und Wert eines Vertrages sind abhängig vom aktuellen Zeitplan und der Qualität des resultierenden Zeitplans hinsichtlich der Präferenzen eines Agenten, d.h. in Hinblick auf Ressourcenauslastung, Durchlaufzeit, Termintreue, etc. Wäre ein Agent allein in einer Gesellschaft, so könnte er den Zeitplan so gestalten, daß er einen für ihn maximalen Wert hat und damit den größtmöglichen Nutzen bringt. Befindet er sich dagegen in einer Gesellschaft von Konkurrenten, so treten zwangsweise Interessenskonflikte auf und es muß ein Kompromiß in Bezug auf den Zeitplan gefunden werden. Das marktorientierte System dient als Werkzeug zur Steuerung dieses Kompromisses.

Zur Erzielung eines Kompromisses zwischen zwei Agenten erfolgt eine wechselseitige Modifikation eines Vertragsvorschlages[4]. Diese Modifikation ist gleichzusetzen mit einer Suche in einem Suchraum, der durch die Vertragsbedingungen aufgespannt wird. Jeder Punkt im Suchraum entspricht dabei genau einem möglichen Vertragsvorschlag. Die suche nach einem Gegenvorschlag wird gesteuert durch eine Bewertungsfunktion, die jedem möglichen Gegenvorschlag eine Bewertung zuordnet. Diese Bewertung ist abhängig vom

[4]Zu Beginn steht ein initialer Vertragsvorschlag des Auftraggebers.

aktuellen Zeitplan und Verhandlungsstand, d.h. insbesondere von den aktuellen Vertragsbedingungen.

In die Bewertungsfunktion fließen folgende drei Bewertungskriterien über eine geeignete Gewichtung ein:

- **Lokale Ziele**
  Die lokalen Ziele eines Agenten fließen über seine Nutzenfunktion ein.
- **Verhandlungshistorie**
  Ein Agent versucht, aus den Begegnungen mit Konkurrenten Informationen über deren Verhandlungsverhalten zu gewinnen und diese bei folgenden Begegnungen zu nutzen. Dazu ermittelt er aus der Abfolge von Vertragsvorschlägen im Rahmen einer Verhandlung die Flexiblität des Konkurrenten hinsichtlich jeder einzelnen Vertragsbedingung. Er mittelt die Flexibilitätswerte über eine begrenzte Verhandlungshistorie und benutzt sie derart zu Modifikation einer Vorschlagsbewertung, daß die Suche nach einem Gegenvorschlag in ein Richtung gelenkt wird, die für den Konkurrenten akzeptabel ist.
- **Zeitdruck**
  Ein Agent steht während einer Verhandlung in zweierlei Hinsicht unter Zeitdruck. Zum einen muß sich ein Auftragnehmer gegen Konkurrenten durchsetzen und ist damit gezwungen, innerhalb eines vernünftigen Zeitrahmens zu einem kompromißfähigen Vorschlag zu kommen. Zum anderen ergeben sich Zeitrestriktionen aus den einzuhaltenden Terminen. Deshalb erfolgt eine diesbezügliche Korrektur der Vertragsbewertung, die auch von der abgelaufenen Verhandlungszeit abhängig ist.

Im derart bewerteten Suchraum kann eines der Standardsuchverfahren angewandt werden — z.B. Simulated Annealing o.ä. —, um einen Gegenvorschlag zu ermitteln. Die Terminierung der Kompromißfindung ist durch Kostendruck gesichert, da der Nutzen eines Vertrages aufgrund der steigenden Wahrscheinlichkeit einer Terminüberschreitung mit zunehmender Verhandlungsdauer größer wird. Die Konvergenz des Verfahrens ist von den Agenten durch die Gewichtung des Zeitdruckaspektes in der Bewertungsfunktion steuerbar.

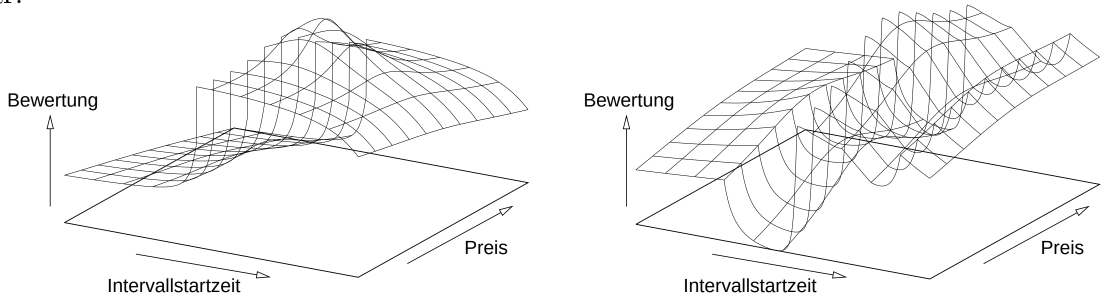


Abbildung 3: Bsp. von Bewertungen, links Auftraggeber, rechts Auftragnehmer

Abb. 3 zeigt ein Beispiel für typische Bewertungen, die aus einem Testlauf mit einem vorgegebenen Planungszustand gewonnen wurden. Aus Darstellungsgründen wurden dabei nur zwei Vertragsbedingungen betrachtet. Deutlich zu sehen ist dabei z.B. die konträre Auffassung hinsichtlich des Preises.

5 Implementierung

Die Implementierung der Agenten, die durch jeweils einen UNIX-Prozeß realisiert werden, erfolgt objektorientiert in der Sprache C++. Zur Realisierung der Kommunikation und Verteilung der Agentenprozesse auf einen Workstation Cluster wird auf das Programmierwerkzeug PVM (Parallel Virtual Machine) zurückgegriffen und für die Visualisierung von Verhandlungsverläufen und Systemzuständen kommt die Tcl/Tk Bibliothek zum Einsatz.

Gegenwärtig existiert eine prototypische Implementierung von Teilen des konzipierten Systems. Dabei handelt es sich um den Kern einer Verhandlung, der aus der Abarbeitung des Verhandlungsprotokolls und der Bewertung eines Vertragsvorschlages besteht. Mit dieser Teilrealisierung können bereits eingeschränkt Untersuchungen hinsichtlich der Kompromißfähigkeit der Agenten durchgeführt werden.

6 Schluß

Zusammenfassung

Diese Arbeit stellt ein Konzept zur algorithmischen Erfassung der Konfliktlösung in Multi-Agenten Systemen vor, das sich auf die Kompromißfindung abstützt. Die Neuerung und Abgrenzung gegenüber anderen Arbeiten auf dem Gebiet der Konfliktlösung in Multi-Agenten Systemen besteht in erster Linie in der strengen Befolgung des Konkurrenzparadigmas und der Einbeziehung der für Konkurrenzszenarien realistischen Annahmen von Zielrelaxierung, unsicherem Wissen, Verhandlungshistorien und multilateralen Verhandlungen. Die Konfliktlösungsstrategien eines Agenten sind vor dem Zugriff durch andere Agenten geschützt, was ihn in die Lage versetzt, seine lokalen Interessen zu wahren.

Stand der Arbeit und weiteres Vorgehen

Basierend auf den dargelegten Konzepten existiert eine prototypische Implementierung einer ersten Simulationsumgebung, die Analysen hinsichtlich des Konfliktlösungsvorganges zwischen zwei Agenten zuläßt. Im weiteren Verlauf der Arbeit wird diese Simulationsumgebung derart erweitert werden, daß die Beschränkung auf zwei Agenten entfällt. Damit sollen Untersuchungen hinsichtlich des Gewinnes eines Agenten bei Einsatz von unterschiedlichen lokalen Strategien und Betrachtungen des Gesamtsystems gemacht werden. Einhergehend findet eine weitergehende Formalisierung der Konfliktlösung statt.

Einordnung in das Graduiertenkolleg

Die Thematik dieser Arbeit weist einige Berührungspunkte mit anderen Teilprojekten des Graduiertenkollegs auf, die Anlaß zu Diskussionen und Möglichkeiten zum Erfahrungsaustausch bieten. Zum einen ist dabei die Thematik der Organisationsstrukturen und den darauf abgestimmten Konfliktlösungsmechanismen in Multi-Agenten Systemen zu nennen, die auch in den meisten anderen Teilprojekten des ersten Teilbereiches „Kooperierende Agenten in verteilten Anwendungen“ eine große Rolle spielen. Zum anderen existiert hinsichtlich des marktorientierten Ansatzes eine unmittelare Verbindung zu der von M. Backschat bearbeiteten Thematik eines agentenbasierten, wirtschaftlich orientierten Lastverteilungs- und

Ressourcen-Managementsystems für hierarchisch strukturierte numerische Algorithmen in verteilten Systemen. Marktsysteme stellen auch eine der Forschungsinteressen von D. Ormoneit im Rahmen seiner Arbeit im Kolleg dar, so daß sich hier ebenfalls Gespräche zum Erfahrungsaustausch ergaben. Nähere Details hierzu finden sich im Abschnitt 1.1.

Literatur

[Fuc96] F. Fuchs. Multi-Agent Collaboration in Competitive Scenarios. In *Proc. of the* 15^{th} *Workshop of the UK Planning and Scheduling Special Interest Group*, Liverpool, 1996.

[HH88] B. A. Huberman und T. Hogg. The Behaviour of Computational Ecologies. In B. A. Huberman, Hrsg., *The Ecology of Computation*, Nr. 2 aus Studies in Computer Science and Artificial Intelligence, S. 77–115. North-Holland, 1988.

[KB91] M. Klein und A. B. Baskin. A Computational Model for Conflict Resolution in Cooperative Design Systems. In S. M. Deen, Hrsg., *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems, October 1990, Univ. of Keele, UK*, S. 201–219. Springer, Berlin, 1991.

[LLC91] S. Lander, V. R. Lesser und M. E. Connell. Conflict Resolution Strategies for Cooperating Expert Agents. In S. M. Deen, Hrsg., *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems, October 1990, Univ. of Keele, UK*, S. 183–200. Springer, Berlin, 1991.

[MD88] M. S. Miller und K. E. Drexler. Markets and Computation: Agoric Open Systems. In B. A. Huberman, Hrsg., *The Ecology of Computation*, Nr. 2 aus Studies in Computer Science and Artificial Intelligence, S. 133–176. North-Holland, 1988.

[PG92] F. Polat und H. A. Guvenir. A Conflict Resolution Based Cooperative Distributed Problem Solving Model. In *Proc. of AAAI-92*, S. 106–115, 1992.

[PR94] M. Palatnik und J. S. Rosenschein. Long Term Constraints in Multiagent Negotiation. In 13^{th} *International DAI Workshop*, S. 265–279, Seattle, Washington, 1994.

[Smi88] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. In A. H. Bond und L. Gasser, Hrsg., *Readings in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[SRSF90] K. Sycara, S. Roth, N. Sadeh und M. Fox. Managing Resource Allocation in Multi-Agent Time-Constrained Domains. In *Proc. of a Workshop on Innovative Approaches to Planning, Scheduling, Control*. Kaufmann, 1990.

[Wel93] M. P. Wellman. A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[Wer90] K. Werkman. Design and fabrication problem solving through cooperative agents. Technical report, Lehigh University Bethlehem, 1990.

[Win93] A. Winkelhofer. *Zeitrepräsentation und merkmalsgesteuerte Suche zur Terminplanung*. Dissertation, Technische Universität München, 1993.

1.3.4 Konzepte zur agentenbasierten Parallelisierung in der Bildanalyse

Teilprojekt: Bildverstehen in verteilten Systemen
Maximilian Lückenhaus

1 Einleitung

Bildanalyseanwendungen stellen hohe Anforderungen an das verarbeitende System. Das bezieht sich einmal auf die Rechenleistung und zum zweiten auf die benötigten Ressourcen – insbesondere Speicherplatz. Wesentliche Gründe hierfür sind strenge Zeitvorgaben, die in der Bildanalyse häufig gelten (z.B. bei Echtzeitverarbeitung in der Robotik), sowie die großen Datenmengen, die zu verarbeiten sind (z.B.: ein 1024×1024 Bildpunkte umfassendes Grauwertebild mit 8 Bit Auflösung belegt bereits um 8 MByte). Es werden also Konzepte benötigt, die die Verarbeitungsdauer minimieren und die Ressourcenverwaltung gleichzeitig optimieren.

Zur Minimierung der Verarbeitungsdauer werden schon seit einiger Zeit erfolgreich Konzepte der Parallelverarbeitung in der Bildanalyse eingesetzt. Meist dienen sie der Parallelisierung einzelner Bildanalyseoperatoren ([FU94], [KRG94]) oder sie realisieren parallele Verfahren für ein eingeschränktes Anwendungsgebiet (z.B. Objektverfolgung [TPM95]). Häufig stehen solche Ansätze in starker Abhängigkeit zur verwendeten Hardware oder zu speziellen Rahmenbedingungen einer Anwendung, was eine Wiederverwendung von Verfahren und Programmcode stark erschwert. Im Gegensatz dazu existieren nur wenige Untersuchungen zu allgemein gehaltenen Ansätzen einer Parallelverarbeitung in der Bildanalyse (wie z.B. in [BBG94], [Web92]). Dieser Nachholbedarf bildet den Ausganspunkt der vorgestellten Arbeit. Es soll überprüft werden, wie sich verschiedene Konzepte der Parallelverarbeitung (Daten-/Taskparallelität, „Pipelining“) innerhalb **eines** Systems integriert zur Bildanalyse in verteilten Systemen einsetzen lassen. Dabei soll durch ein verbessertes Ressourcen-Management die Ressourcenauslastung optimiert und durch eine weitestgehend automatische Parallelisierung der Entwicklungsaufwand für parallele Bildverarbeitungsprogramme minimiert werden.

2 Agentenbasierte Parallelisierung von Bildanalyseanwendungen

Im folgenden wird der Ansatz einer agentenbasierten Parallelverarbeitung von Bildanalyseanwendungen dargestellt.

2.1 Konzepte zur Parallelverarbeitung in der Bildanalyse

In der Bildverarbeitung können verschiedene Parallelverarbeitungskonzepte angewendet werden. Da wäre einmal die *Taskparallelität*, wie sie von Betriebssystemanwendungen bekannt ist: Unabhängige (Teil-)aufgaben werden parallel bearbeitet (vgl. Abbildung 1). Ein anderes Konzept, das sich insbesondere für Bilddaten anbietet, ist die Datenparallelität: Die Eingabedaten (z.B. eine Bildmatrix) werden aufgeteilt und nebenläufig verarbeitet. Schließlich wäre noch das Pipelining-Konzept zu nennen, das sich besonders für Bildfolgeverarbeitung eignet. Jedes Element (Einzelbild) eines kontinuierlichen Stroms an Eingabe-

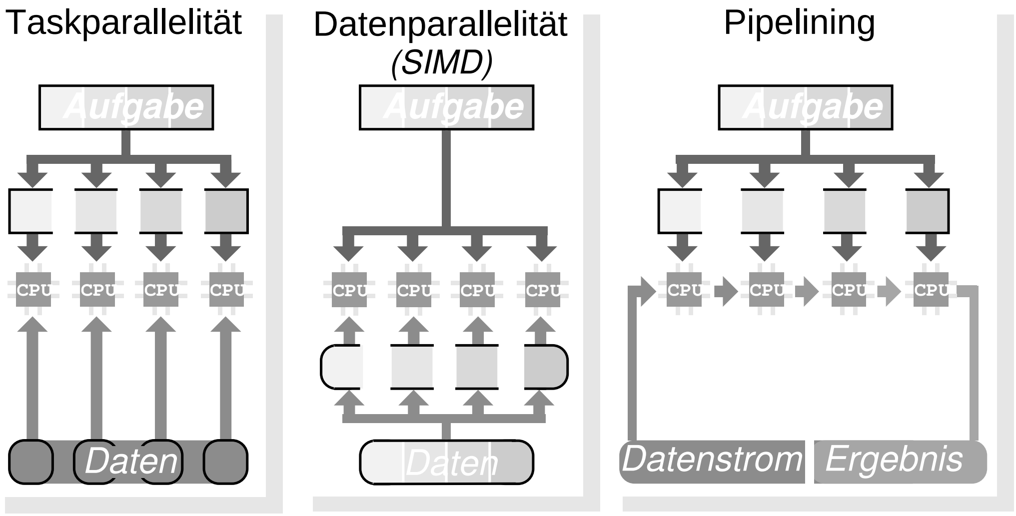


Abbildung 1: Konzepte zur Parallelverarbeitung

daten durchläuft dieselben Verarbeitungsschritte. Es liegt nahe, verschiedene Teilschritte für aufeinanderfolgende Bilder parallel auszuführen. Natürlich lassen sich die drei Parallelverarbeitungsmethoden auch miteinander kombinieren.

Eine agentenbasierte Bildanalyse soll nun den gleichzeitigen Einsatz der drei Methoden ermöglichen. Das Multiagentensystem wählt automatisch diejenige Methode bzw. Kombination aus, mit deren Hilfe ein Bildanalyseproblem auf der verfügbaren Hardware in minimaler Zeit bearbeitet werden kann. Hierfür muß der Anwender die Bildanalyseanwendung zuvor spezifiziert haben.

2.2 Modellierung von Bildanalyseprogrammen

Grundlage dieser Spezifikation bildet folgende Modellvorstellung: Die Bearbeitung einer Bildanalyseaufgabe geschieht mit Hilfe verschiedener Operatoren.

Bildanalyseoperator: Ein Operator ist die kleinste Einheit eines Bildanalyseprogramms und realisiert **einen** Arbeitsschritt. Die Komplexitäten verschiedener Operatoren variieren mit deren unterschiedlichen Funktionalitäten, die von einfachen Aufgaben (z.B. „Bild aus Datei lesen“) bis zu komplizierten Verarbeitungsschritten (z.B. „Zustandsschätzung mittels Kalman-Filterung“) gehen können. Der Operator selbst wird als *stromverarbeitende Funktion* modelliert, die einen Strom an Eingabedatenobjekten seiner Funktionalität entsprechend in Ausgabedatenobjekte transformiert (siehe Abbildung 2). Als Objektklassen für Daten kommen hierbei „Bilder“, „Bildregionen“ und „Steuerdaten“, die das Verhalten des Operators beeinflussen, in Betracht. *Operatoren* besitzen verschiedene *statische* bzw. *dynamische* Eigenschaften, deren Ausprägung bekannt sein muß, um eine parallele Bearbeitung möglichst gut planen und durchführen zu können. *Statische* Eigenschaften sind z.B. Name, Funktionalität, durchschnittliche Bearbeitungszeit/Ressourcenanforderung, wahrscheinlicher Nachfolgeoperator. *Dynamische* Eigenschaften sind z.B. „Aufenthaltsort“ (be-

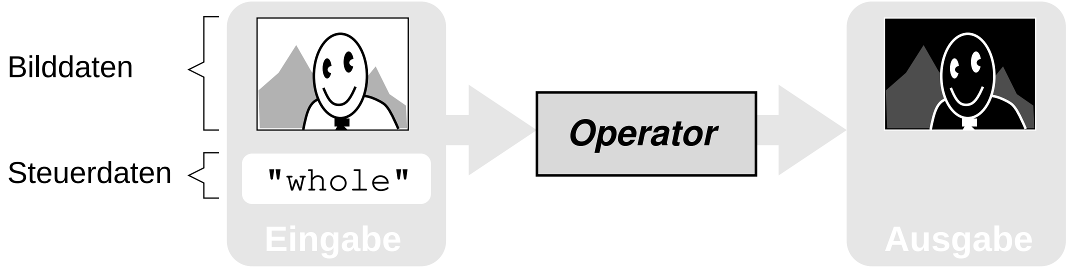


Abbildung 2: Modell eines Bildanalyseoperators (Im Bsp.: „Invertieren“)

arbeitende CPU), belegte Ressourcen, Bearbeitungszustand usw.

Eine Menge von Operatoren kann nun benutzt werden, um einen Bildanalyseauftrag zu bilden.

Bildanalyseauftrag: Ein Bildanalyseauftrag spezifiziert die programmtechnische Lösung einer Bildanalyseaufgabe. Er besteht aus einer Menge von Operatoren und den Abhängigkeiten zwischen diesen. Abhängigkeiten zwischen Operatoren werden als Relationen über *Ressourcen* modelliert (siehe Abbildung 3). Als *Ressource* wird in diesem Zusammenhang

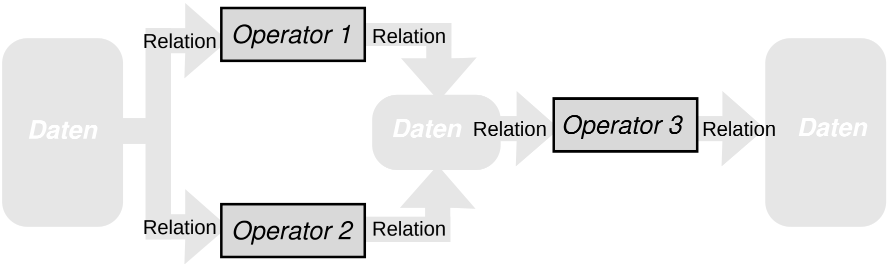


Abbildung 3: Modell eines Bildanalyseauftrags

ein Datenobjekt angesehen. Neben den drei Objektklassen „Bilder“, „Bildregionen“ und „Steuerdaten“ (siehe Datenobjekte für Operatoren) wird bei Bildanalyseaufträgen ein spezieller Typ „Synchronisationspunkt“ hinzugefügt, der es erlaubt, zeitliche Abhängigkeiten zu modellieren. Dies ist notwendig, um dem Anwender die explizite Sequentialisierung von Programm(-teilen) zu erlauben. Typische Relationen sind:

- Einfache Datenübergabe: Ein Ausgabeobjekt eines Operators ist das Eingabeobjekt eines anderen,
- Datendekomposition: Zwei Operatoren arbeiten auf verschiedenen Anteilen desselben Eingabedatenobjekts (*Datenparallelität*; im Beispiel die Operatoren 1 und 2),
- Datenkomposition: Die Ausgabeobjekte zweier Operatoren definieren verschiedene Anteile desselben Datenobjekts (im Beispiel die Operatoren 1 und 2),
- Pipelining: Zwei Operatoren sind im Sinne einer Pipeline hintereinandergeschaltet.

Das Bildanalysesystem benutzt die Spezifikation des Bildanalyseauftrags, um daraus automatisch ein paralleles Bildanalyseprogramm zu planen, das in einem verteilten System ausgeführt werden kann.

2.3 Motivation für den Einsatz eines Multiagentensystems

Zur verteilten Planung, sowie zur Steuerung der parallelen Bearbeitung wird ein Multiagentensystem verwendet. Hierfür spricht einmal die Eignung von Multiagentensystemen als Basis eines **verteilten** Planungsprozesses (siehe [Mar91], [AW91]). Für die **Steuerung** parallel ablaufender Bildanalyseprozesse benötigt man nebenläufig arbeitende Moduln, die miteinander kooperieren um ein globales Ziel zu erreichen. Auch diese Anforderung wird durch ein Multiagentensystem erfüllt. Die Lernfähigkeit der Agenten hilft dabei zusätzlich, das Verhalten des Bildanalysesystems langfristig zu optimieren bzw. ein Programm zur Bildfolgenanalyse schrittweise zu verbessern.

2.4 Agentenbasierte Planung und Durchführung von Bildanalyseanwendungen

Eine agentenbasierte parallele Planung und Bearbeitung kann nun nach folgender Methodik erfolgen (vgl. Abbildung 4):

Aufgabenspezifikation: Im ersten Schritt erstellt der Anwender eine Spezifikation des *Bildanalyseauftrags* durch Angabe der notwendigen Operatoren und deren Abhängigkeiten untereinander. Dadurch wird festgelegt, welche Bildanalyseaufgabe durch das Multiagentensystem bearbeitet werden soll. Wie dies geschieht, wird im wesentlichen durch das System selbst entschieden. Unterstützt wird es hierbei durch drei weitere „Informationsquellen“:

- Operatorwissensbasis, die die *statischen* und *dynamischen* Eigenschaften aller verfügbaren Operatoren beschreibt,
- Hardware-Spezifikation mit einer Beschreibung des bearbeitenden, verteilten Systems,
- Anbindung an die Ressourcenverwaltung des Betriebsystems, so daß Informationen zur aktuellen Verfügbarkeit von Ressourcen (Speicherplatz, Dateien usw.) eingeholt werden können.

Verteilte Planung und Ausführung: Aus der Spezifikation und unter Berücksichtigung des Zusatzwissens über Operatoren, Hardware und Systemauslastung, erstellt das Multiagentensystem einen parallelen Ausführungsplan, der unmittelbar durchgeführt wird. Die Verteilung von Planungsaufgaben an Agenten erfolgt hierarchisch gemäß einer schrittweisen Verfeinerung des Plans:

1. Entgegennahme der Aufgabenspezifikation und Aufteilung in nebenläufig zu bearbeitende Teilaufgaben.

2. Verfeinerung der Teilaufgaben bis auf Operator-Granularität (voneinander unabhängige Operatoren werden parallel bearbeitet).

3. Weiteres Verfeinern eines Operatoraufrufs bzgl. Datenparallelität, d.h., falls der Operatoraufruf dies zuläßt, wird er in mehrere Aufrufe desselben Operators auf Teildaten der ursprünglichen Eingabedaten abgebildet.

Zwischen jeder Verfeinerungsebene besteht die Möglichkeit, Teilpläne/ Teilaufgaben an andere Agenten zu delegieren. Diese können sich auch explizit um die Ausführung spezieller Teilaufgaben bewerben (siehe den späteren Abschnitt „Kooperation zwischen den Agenten“). Jeder mit einer Teilaufgabe betraute Agent ist einerseits für deren weitere Verfeinerung und andererseits für deren korrekte Bearbeitung verantwortlich. Die Ausführung erfolgt schon während der Aufteilung, d.h. Planung und Bearbeitung sind ineinander verzahnt. Nach erfolgreicher Ausführung wird das Ergebnis an die nächsthöhere Instanz zurückgegeben. Mit dieser Vorgehensweise ergibt sich eine verteilte Bearbeitung einer Bildanalyseaufgabe, flexibel genug, um auf Veränderungen (z.B. bzgl. Ressourcenverfügbarkeit, CPU-Last, oder auch veränderte Operatoreingabeparameter) lokal reagieren zu können, ohne den gesamten Planungsprozeß wieder neu aufnehmen zu müssen.

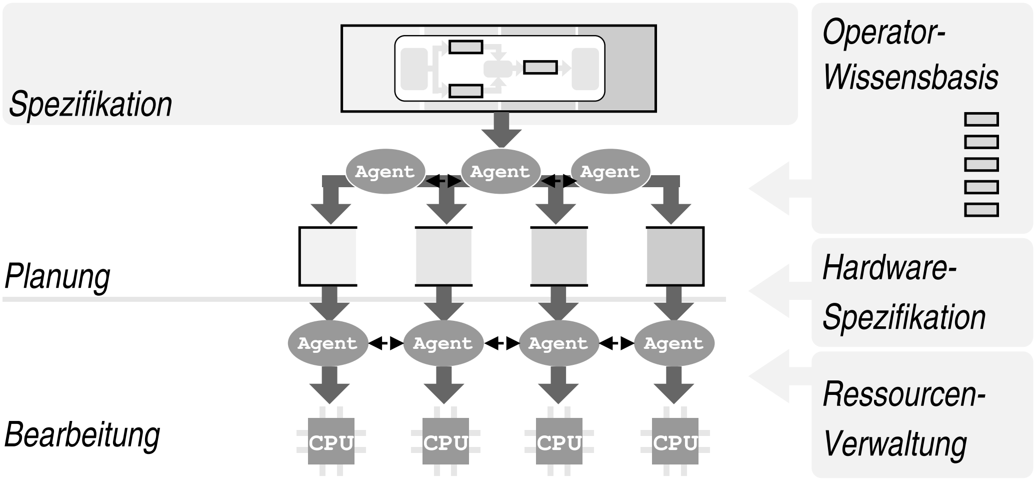


Abbildung 4: Agentenbasierte Planung and Bearbeitung

2.5 Entwurf eines Multiagentensystems für parallele Bildanalyse

Im folgenden wird das Design eines Multiagentensystems zur verteilten Planung und Ausführung von Bildanalyseaufgaben vorgestellt.

Eigenschaften des Multiagentensystems:

1. **Homogenität** – alle Agenten besitzen dieselben Fähigkeiten und sind flexibel in ihrem Rollenverhalten (z.B. „Plan-Entgegennahme“, „Koordination“, “Operatorausführung“ usw.). Diese Flexibilität ermöglicht schnelles Delegieren von Aufgaben an

beliebige Agenten ohne sich um eine spezielle Eignung des Kooperationspartners kümmern zu müssen.

2. **Agentencharakteristika**: Der einzelne Agent verfügt über folgende – im MAD-Arbeitskreis des Graduiertenkollegs als „typisch“ für einen Agenten postulierte – Eigenschaften:
   - ein *Agentenzyklus* definiert seine Arbeitsweise,
   - in einer privaten Wissensbasis liegt sein Wissen *explizit repräsentiert* vor,
   - er besitzt ein flexibles *Rollenverhalten*,
   - er besitzt die Fähigkeit zu *komplexer Kommunikation*, d.h. z.B. zu Rundrufen/Abstimmungen/Weitergabe von Daten usw. (diese Fähigkeiten sind auf ein Mindestmaß reduziert, siehe nächster Punkt).
3. **„Lean Agents“** – gemäß dem Aspekt der Rechenzeitminimierung sind die Agenten nur mit dem Mindestmaß an Fähigkeiten ausgestattet, das sie zur Bewältigung ihrer Aufgabe benötigen. Gleichzeitig sollen **komplizierte** Kommunikationsprotokolle und Kooperationsmethoden vermieden werden, um den durch die Agenten verursachten Overhead so gering wie möglich zu halten.

Letzterer Punkt ergab sich aus Anforderungen von Anwendern des Bildanalysesystems HORUS (eine Entwicklung des Lehrstuhls IX für Informatik, dessen künftige Weiterentwicklung die Firma *MVTec Software GmbH* übernimmt) – insbesondere mit dem Kollegiaten D. Büsching (siehe Bericht zu „Verteilte Generierung von Objekterkennungsprogrammen“). Um die Tragfähigkeit des vorgestellten Ansatzes in der Praxis überprüfen zu können, wird er als Erweiterung von HORUS implementiert. Da HORUS ein bereits ausgereiftes Produkt darstellt, das gleichermaßen in Forschung und Industrie verwendet wird, kann solch eine Erweiterung nur in enger Abstimmung mit den Anwendern sowie dem Entwicklerteam geschehen. Eine wichtige Anforderung ist hierbei die Effizienz. Der Geschwindigkeitsvorteil der Parallelisierung darf nicht durch einen zu großen Overhead der Agenten verlorengehen.

Wissensbasis des Agenten – „Datenkapselung von Wissen“: Unter demselben Gesichtspunkt ist auch die Agenten-Wissensbasis entworfen worden. Sie enthält einmal Wissen zum Agenten selbst, d.h. zu seinen Aufgaben und Zielen (z.B. ein Operatoraufruf, eine Bewerbung um eine Teilaufgabe usw.) und zum zweiten Wissen über die **Umwelt** des Agenten, also bzgl. Hardware und anderen Agenten (Kooperationsbereitschaft, Verfügbarkeit usw.). Nun bedarf es eines gewissen Aufwands all dieses Wissen konsistent zu halten. Die primäre Aufgabe des Agenten liegt aber in einer effizienten Ausführung von Bildanalyseanwendungen und weniger in der Bearbeitung seiner Wissensbasis. Daher wird hierin nur der Mindestsatz an Informationen abgespeichert, der ausschließlich den Agenten selbst betrifft (z.B. die durch **ihn** zu bearbeitetenden Aufgaben, **seine** Einschätzung der Kooperationsbereitschaft anderer Agenten). Jegliche weitere Information, die also für **mehrere** Agenten relevant ist, wird stattdessen als externes Wissen dargestellt und von den betreffenden Objekten verwaltet. Eigenschaften von Operatoren liegen somit in einer Operatorwissensbasis und Eigenschaften von Datenobjekten (Bilder, Regionen) in einer Bildobjektwissensbasis. Bearbeitet ein Agent ein Objekt, so kann er sich von dessen Wissensbasis alle benötigten Informationen holen, ohne sich um die Konsistenzhaltung kümmern zu müssen.

Kooperation zwischen den Agenten: Alle anstehenden Probleme werden durch die Agenten kooperativ gelöst, um den Geschwindigkeitsvorteil von Parallelverarbeitung zu nutzen. Wie viele Agenten hierbei kooperieren, hängt von Problemgröße und Systemlast ab. Da jeder Agent prinzipiell über alle Fähigkeiten verfügt, ein Problem auch allein zu lösen (siehe oben, Stichpunkt „Homogenität"), ist er nicht zwingend auf die Hilfe anderer Agenten angewiesen. Diese Tatsache markiert einen wesentlichen Unterschied zwischen dem vorliegenden Multiagenten-Ansatz für parallele Bildanalyse und anderen heterogenen Ansätzen, wie sie z.B. für Fertigungsanlagen existieren. Bei letzteren ist die Kooperation eine zwingende Notwendigkeit (z.B. weil eine Teilaufgabe ausschließlich durch einen speziell ausgerüsteteten Roboter zu erledigen ist) um die Gesamtaufgabe überhaupt lösen zu können. Im vorliegenden Ansatz hingegen wird sie nur dort verwendet, wo sie zu einer effizienteren Lösung eines Problems führt.

Die Form der Kooperation folgt einem „Contract-Net" ähnlichen Ansatz. Klassische „Contract-Net-Ansätze" entsprechen einem Schema von „Ausschreibung, Bewerbung und Zuschlag". Dieses wird nun um das Konzept der „Vorausbewerbung" erweitert. Ein Agent kann sich schon im voraus um die Ausführung einer speziellen Aufgabe bewerben. Umgekehrt delegiert ein auftraggebender Agent bevorzugt an einen Bewerber. Dies funktioniert nur, wenn spätere Aufgaben bereits im voraus für einen Agenten absehbar sind. Glücklicherweise ist das gerade in der Bildanalyse oftmals der Fall, da die Lösung eines Problems häufig in einer Standardfolge gewisser Operatoren besteht. D.h., mit der Ausführung eines Operators wird der baldige Aufruf eines typischen Nachfolgeoperators wahrscheinlich. Ein Agent kann dieses Wissen nutzen, um sich schon im voraus bei seinem Auftraggeber um die Ausführung des Nachfolgeoperators zu bewerben. Dadurch wird es möglich, eine Aufgabe dorthin zu delegieren, wo bereits Ressourcen vorliegen, die zur Bearbeitung benötigt werden. Bearbeitet z.B. ein Agent einen Operator und „weiß", daß die Ausgabebilddaten einem Nachfolgeoperator als Eingabe dienen, so gibt er eine entsprechende Bewerbung ab. Erhält er den Zuschlag für die Ausführung des Nachfolgeoperators, so liegen die Eingabedaten bereits vor. Dies ist insbesondere bei verteilter Bildanalyse auf einem Netzwerk von Vorteil, wo die Übertragung von Bilddaten von einem Knoten zum anderen extrem zeitintensiv ist.

3 Stand der Arbeit

Bis zum gegenwärtigen Zeitpunkt wurden die vorgestellten Konzepte zur agentenbasierten Parallelisierung in der Bildanalyse erstellt. Als Grundlage für eine Implementierung wurde ein Multiagentensystem entworfen. Bei dessen Design fanden auch Anregungen aus dem MAD-Arbeitskreis des Graduiertenkollegs Eingang (siehe auch Punkt „Agentencharakteristika" im vorhergehenden Abschnitt).

Die Implementierung des vorgestellten Ansatzes – insbesondere des Multiagentensystem-Prototyps – wurde begonnen. Hierfür wurde eine Bibliothek von effizienten Kommunikationsfunktionen implementiert, die auf die speziellen Rahmenbedingungen (z.B. verteilter, gemeinsamer Speicher) des Prototyps zugeschnitten sind. Weiterhin wurden Konzepte zur Umstrukturierung des bislang sequentiell arbeitenden Bildanalysesystems HORUS entwickelt, um die Voraussetzung für die Einbindung des Multiagentensystems zu schaffen. Diese betreffen u.a. die Verwaltung von Wissen über Operatoren bzw. Operatoraufrufe in einem verteilten System. Zum großen Teil sind diese Konzepte inzwischen implementiert, eine Arbeit, die in enger Kooperation mit dem HORUS-Entwicklerteam der Firma *MVTec Software GmbH* geschieht. Beim Implementierungsentwurf einer agentenbasierten Parallelisierung wurden die Wünsche und Anforderungen spezieller HORUS-Anwender (Graduiertenkolleg-Stipendiat D. Büsching, SFB-331-Teilprojekt L9 „Videobasierte Erfassung von Umweltinformationen durch ein autonomes, mobiles System für die schritthaltende Lokalisierung und schnelle Objekterkennung“ und „Forschungsgruppe Kognitive Systeme am Bayerischen Forschungszentrum für wissensbasierte Systeme – FORWiss“) eingebracht. Dies betrifft Fragen der Effizienz (insbesondere zeitliche Anforderungen), Eingriffsmöglichkeiten auf Anwenderseite (wie z.B. explizit sequentielle Durchführung von Teilaufgaben) und auch den Leistungsumfang der automatischen Parallelisierung, d.h., welche Parallelverarbeitungskonzepte möglich sind und wie der Anwender auf die Aufteilung von Daten und Aufgaben einwirken kann.

4 Ausblick

Zukünftige Arbeiten betreffen u.a. den konkreten Entwurf einer erweiterten Agentenplanungskomponente. Der gegenwärtige Multiagentensystem-Prototyp unterstützt lediglich die *Task-* und *Datenparallelisierung* einzelner Operatoren, jedoch noch nicht die einer Bildanalyseaufgabe. Hierfür wird eine erweiterte Planungskomponente benötigt, die den Agenten in die Lage versetzt, auch Operatormengen und deren Abhängigkeiten zu verarbeiten. Erst dann können *Pipeliningkonzepte* implementiert werden. Darüberhinaus ist eine Methode zur Spezifikation paralleler Bildanalyseprogramme zu definieren, die Syntax und Semantik der Planungseingabedaten festlegt.

Weitere Arbeiten betreffen eine Klasseneinteilung von Bildanalyseoperatoren bzgl. ihrer Parallelisierungsmöglichkeiten (Eignung zur Task-/Datenparallelität bzw. zum Pipelining) und die Fertigstellung des Multiagentensystem-Prototyps. Letzteres schafft die Voraussetzung, verschiedene Methoden der Agentenkooperation in der Praxis auf ihre Tauglichkeit zu untersuchen.

Literatur

[AW91] Gerhard Köstler Andreas Winklhofer. Temporale Planung in Multiagentenumgebungen. In W. Brauer und D. Hernandez, Hrsg., *Verteilte künstliche Intelligenz und kooperatives Arbeiten, 4. Internationaler GI-Kongreß Wissensbasierte Systeme, München*, S. 124–135, October 1991.

[BBG94] Martin Brunzema, Horst Burmeister und Dimitris Gerogiannis. Parallelization of an Image Analysis Application: Problems, Results and a Solution Framework. In *Vol. III of the 12th IAPR International Conference on Pattern Recognition, Jerusalem, Israel*, S. 406–411, October 1994.

[FU94] Afonso Ferreira und Stephane Ubeda. Ultra-Fast Contour Tracking, with Applications to Thinning. *Pattern Recognition*, 27(7):867–878, 1994.

[KRG94] Senthil Kumar, N. Ranganathan und Dimitry Goldgof. Parallel Algorithms for Circle Detection in Images. *Pattern Recognition*, 27(7):1019–1028, 1994.

[Mar91] Frank von Martial. Activity Coordination via Multiagent and Distributed Planning. In W. Brauer und D. Hernandez, Hrsg., *Verteilte künstliche Intelligenz und kooperatives Arbeiten, 4. Internationaler GI-Kongreß Wissensbasierte Systeme, München*, S. 90–101, October 1991.

[TPM95] Chew Lim Tan, Chiun Min Pang und Worthy N. Martin. Transputer implementation of a multiple agent model for object tracking. *Pattern Recognition Letters*, 16(11):1197–1203, 1995.

[Web92] Jon A. Webb. Steps Toward Architecture-Independent Image Processing. *IEEE Computer*, S. 21–31, February 1992.

1.3.5 Flexible Agenten im Netz- und Systemmanagement

Teilprojekt: Kooperierende Agenten im Netz- und Systemmanagement

Maria-Athina Mountzia

1 Einführung und Motivation

Die zunehmende Komplexität und Verteilung von Kommunikationsressourcen, Diensten und Anwendungen hat zur Folge, daß das Management einer solchen komplexen Umgebung immer schwieriger wird. Faktoren, die zu dieser Komplexität beitragen, sind [HA94]: die Anzahl und Vielfalt an Typen von zu managenden Komponenten, die Heterogenität der Systemsoftware, Schnittstellen und Protokolle sowie die Anzahl der angebotenen Dienste und der betroffenen Organisationsdomänen.

Die meisten Managementsysteme weisen nur ein zentralisiertes, plattformbasiertes, sehr aufwendiges Interaktionsparadigma auf. Der Einsatz von zentralisierten Managementansätzen ist bei solchen Umgebungen aus Leistungs- und Effektivitätsgründen problematisch. Das Auftreten von Problemen wie „single-point-of-failure“, hoher Netzverkehr aufgrund des Managements und Überlastung des zentralen Managers weisen darauf hin, daß verteilte Managementverfahren angewendet werden sollten. Außerdem reichen die bislang eingesetzten Managementverfahren nicht aus, um die Dynamik der heutigen verteilten Systemen zu bewältigen. Aus diesen Gründen ist ein Übergang auf verteilte, flexible Managementverfahren erforderlich.

Diese Probleme wurden schon früh erkannt und erlangten große Aufmerksamkeit sowohl im industriellen als auch im Forschungsbereich. Dabei entstanden interessante Ansätze, die einen ersten, wichtigen Schritt in diese Richtung machen. Durch diese ersten Ansätze wurde aber das Problem des verteilten Managements bei weitem noch nicht gelöst. Viele Aspekte des verteilten Managements wurden bis jetzt überhaupt nicht betrachtet, die für seinen breiteren Einsatz aber unerläßlich sind.

Die Feststellung und Analyse dieser Aspekte ist einer der Beiträge der in diesem Teilprojekt durchgeführten Arbeit. Dabei wird ein Rahmenwerk für die Delegierung festgelegt [MDR96], das als eine Art „road map“ zu verteiltem, flexiblen Management dient. Anhand dessen wird eine Agentenarchitektur vorgeschlagen, die eine Erweiterung des *Management by Delegation* Paradigmas [YGY91] durch Kontrolle und Kooperationsfähigkeiten ist. Die neue in diesem Rahmen entstandene Managementeinheit ist der *Flexible Agent*. Flexible Agenten erweitern die heutigen Managementmodelle, indem sie eine effektive und flexible Verteilung von Aufgaben und Funktionalität zwischen Managementeinheiten erlauben. Darüberhinaus wird ein auf flexiblen Agenten basierendes System vorgestellt, das auch mit Konzepten aus dem Bereich der künstlichen Intelligenz wie *Cooperative Distributed Problem Solving (CDPS)* und intelligente Agenten bereichert wird. Das führt zu der Entwicklung einer Methodik zur Realisierung von komplexen, verteilten Aufgaben im Bereich des Netz- und Systemmanagements.

2 Status Quo

Die existierenden verteilten Managementansätze können in zwei Gruppen unterteilt werden: *statische* und *dynamische* Ansätze. Die statischen Ansätze setzen das Vorhandensein von vordefinierten Funktionen auf dem Agenten voraus und erlauben keine Änderung dieser Funktionalität zur Laufzeit. Beispiele für solche statischen Managementansätze in der Internet- Management-Architektur sind Agenten, die die *Remote Monitoring (RMON)* und *Manager to Manager (M2M)* Management Information Base (MIB) realisieren ([Wal95], [CMRW93]). Analog dazu bietet die von der ISO im Rahmen des OSI-Referenzmodells definierte Managementarchitektur Agenten, die die *Systems Management Functions (SMFs)* [ISO94] realisieren. Aufgrund des Mangels an Flexibilität werden diese Ansätze nicht weiter verfolgt.

Die dynamischen Ansätze basieren auf dem Paradigma der *intelligenten Agenten*, das die dynamische Verlagerung von Funktionalität ermöglicht. In den letzten Jahren gab es allgemein großes Interesse an Agententechnologie. Agenten werden in unterschiedlichsten Bereichen benutzt, wie verteilten Systemen, rechnergestützter Gruppenarbeit, Fertigungssystemen, Robotertechnik usw ([GW94], [HCK95]). Was ein Agent eigentlich ist, ist seit vielen Jahren Gegenstand intensiver Forschung und wird kontrovers in verschiedenen Bereichen der Informatik diskutiert. Obwohl der Begriff bereits heute weit verbreitet ist, hat es sich als sehr schwer erwiesen, eine einheitliche, allgemein akzeptierte Definition zu geben. Bei den dynamischen Ansätzen geht es ausschließlich um den Aspekt der *Mobilität* von Agenten. Die transferierte Funktionalität ist in diesem Sinne ein mobiler Agent.

Die dynamischen Ansätze werden mit dem Begriff *Management by Delegation (MbD)* bezeichnet. Es gibt unterschiedliche Ausprägungen und Realisierungen dieses Konzepts. Sie reichen von *prozeßbasierten* Ansätzen wie z.B. elastic servers [Gol96] bis zu *sprachbasierten* Ansätzen wie z.B. Java [GM95], Tcl [Wel95] etc. In der Internet- und OSI- Welt gibt es auch ähnliche Bestrebungen, wie die Mid-Level-Manager MIB [CL94], Script MIB [LS96] und den Command Sequencer [ISO96].

Alle diese Methoden bieten nur Mechanismen an, die die dynamische Delegierung von Managementfunktionen ermöglichen. Zu diesem Zweck werden spezielle Protokolle entwickelt oder existierende Managementprotokolle verwendet (z.B. SNMP). Trotzdem reichen die Mechanismen nicht aus, um verteiltes Management betreiben zu können. Die existierenden Ansätze gehen nur auf *remote management* ein und beschäftigen sich nicht mit Aspekten was man eigentlich delegieren kann, wie man die delegierte Funktionen beschreibt, welche Kooperationsanforderungen sich nach der Delegierung ergeben, wie man das verteilte System kontrolliert usw.. Diese Fragestellungen stellen nur eine Teilmenge von wichtigen Fragestellungen dar, auf die noch keine Antwort gegeben wird. Außerdem beschränken sich die MbD Realisierungen bislang meist auf sehr einfachen Überwachungsaktivitäten als einziges Anwendungsszenario. Die in diesem Teilprojekt durchgeführte Arbeit betrachtet das Konzept globaler und verbindet es mit Konzepten aus dem Bereich der intelligenten Agenten, wo Aspekte der Verteilung und Kooperation zwischen Agenten ein wichtiger Faktor sind.

3 Rahmenwerk

Wie oben erwähnt gibt es bis jetzt nur Mechanismen zur Verteilung von Managementaufgaben. Eine nähere Betrachtung der vorhandenen Mechanismen und der Fragestellungen, die sich bei ihrem Einsatz ergeben, führt zu neuen Aspekten, mit denen man sich auseinandersetzen muß.

Die aus der durchgeführten Analyse resultierenden Aspekte werden in Abbildung 1 dargestellt und beinhalten:

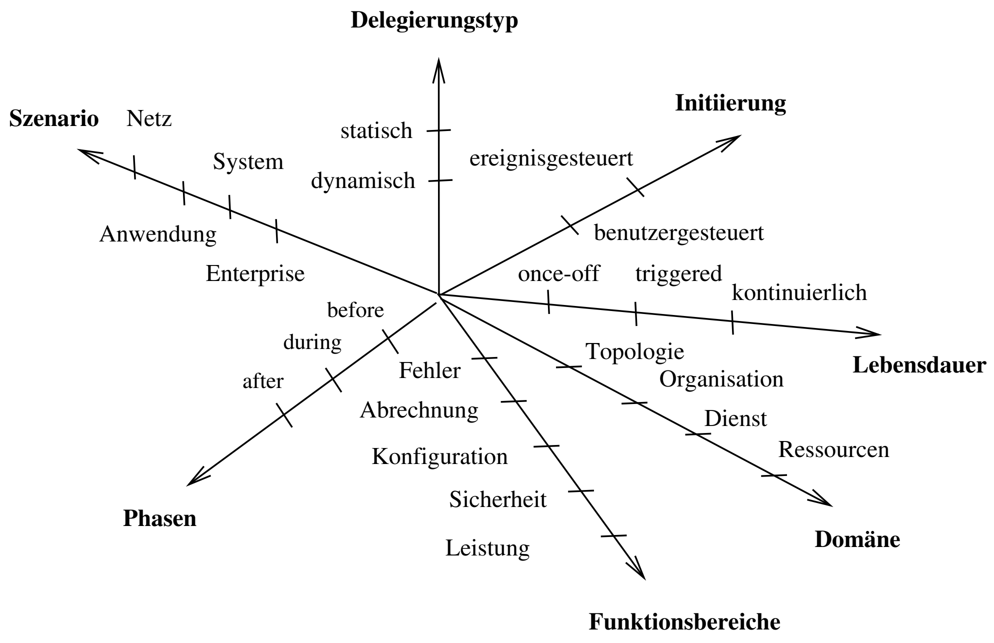


Abbildung 1: Aspekte der Delegierung

- *Delegierungstyp*

  Hiermit wird zwischen der dynamischen und statischen Delegierung unterschieden. Die Entscheidung wird anhand der bestehenden Anforderungen im verteilten System getroffen. Das ist meistens der Fall für Anforderungen, die sich in der Designphase der Anwendung ergeben. Für neue Anforderungen, die sich zur Laufzeit des Systems spezifiziert werden, soll dynamische Delegierung angesetzt werden.

- *Initiierung der Delegierung*

  Bei diesem Aspekt geht es um die Entscheidung, wann Funktionalität delegiert werden soll und ob die Initiierung benutzer- oder ereignisgesteuert ist. Die gleiche Frage stellt sich auch im Fall der *cascaded delegation*. Dabei wird Funktionalität in vielen Stufen einer logischen Hierarchie delegiert, was die Komplexität erhöht.

- *Lebensdauer von delegierten Funktionen*

Das bezieht sich auf das Verhalten einer Funktion, nachdem sie delegiert wurde. Es gibt Aufgaben, die delegiert werden und anschließend *kontinuierlich* im Agenten laufen. Es gibt aber auch eine andere Art von Aufgaben, die nicht unbedingt sofort nach der Delegierung ausgeführt werden und anschließend im Agenten in einem „sleeping mode" bleiben, bis sie wieder ein Ereignis oder der Benutzer ins Leben ruft (*triggered*). Eine andere Variante sind Aufgaben, die gleich nach der Ausführung vom Agenten entfernt werden (*once-off*).

- *Domänen*

  Dieser Aspekt bezieht sich auf die Frage, *wohin* im verteilten System delegiert werden soll. Das steht in Zusammenhang mit der Struktur der Umgebung und den vorhandenen Domänen. Das wiederum hängt von der *Topologie* des Systems und den im System geltenden *Politiken* (z.B. u.a. Sicherheitsanforderungen, Charakteristika der Endsysteme) ab.

- *Funktionsbereiche*

  Dieser Aspekt befaßt sich mit den Funktionsbereichen des Managements, die von der Delegierung betroffen sind. In jedem Funktionsbereich gibt es Funktionen, die gute Kandidaten für die Delegierung darstellen. Diese Funktionen können die Basis für generische, delegierbare Funktionen bilden. Die Anpassung dieser generischen Funktionen an die verschiedenen Funktionsbereiche soll anhand von spezifischen Parametern erfolgen, die dann die speziellen Anforderungen im jeweiligen Funktionsbereich widerspiegeln.

- *Managementszenario*

  Das Managementszenario, in dem die Delegierung angewendet wird, ist ebenfalls ein relevanter Aspekt.

- *Phasen der Delegierung*

  Eine chronologische Betrachtung des Delegierungsprozesses trägt zu der systematischen Analyse aller auftretenden Probleme bei. Die Hauptfragestellungen, die dadurch behandelt werden sollen, beziehen sich auf *was, wann, wohin, wie* delegiert wird. Drei Phasen werden identifiziert: die *Initiierung (before), Delegierung (during) und Betrieb (after)*. In der ersten Phase werden die Fragestellungen *was und wohin* delegiert werden soll, behandelt. Die zweite Phase beschäftigt sich mit den Fragestellungen, *wie* Funktionen delegiert werden. Das beinhaltet die Spezifikation von geeigneten Protokollen, Festlegung der Architektur von flexiblen Agenten und andere Anforderungen, wie Sicherheit des Systems. Die letzte Phase bezieht sich auf das Problem, wie das verteilte System kontrolliert werden kann und wie eine erfolgreiche Ausführung der Funktionen gewährleistet werden kann. Dabei tritt hauptsächlich das Problem der Kooperation zwischen den Agenten, der Koordination und Synchronisation auf. Außerdem sind Mechanismen erforderlich, die das Management der delegierten Funktionen von dem Delegator erlauben.

  Im Fall von kaskadierter Delegierung läßt sich die Betriebsphase wiederum in drei Phasen unterteilen (Abbildung 2).

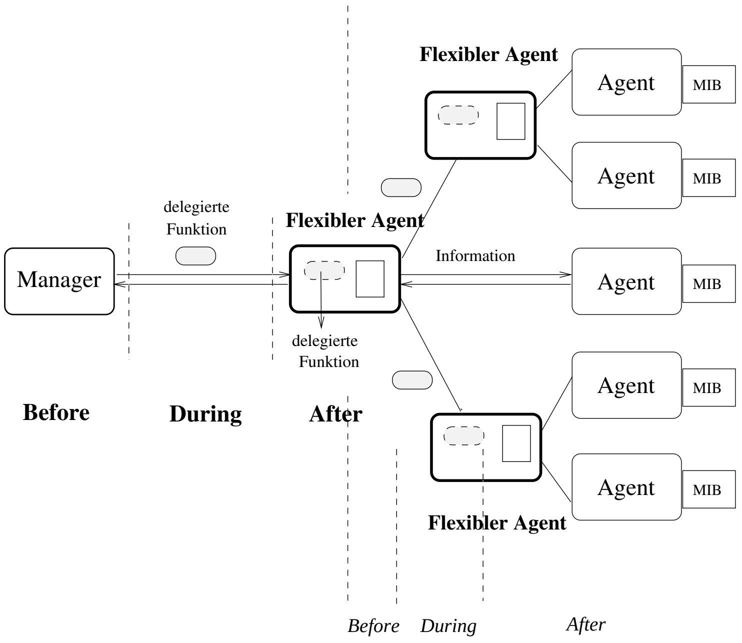


Abbildung 2: Phasen der Delegierung

4 Ein auf flexiblen Agenten basierendes Managementsystem

Um die Anforderungen, die sich aus den oben genannten Aspekten ergeben, zu erfüllen, wird ein flexibles, verteiltes Managementsystem vorgeschlagen. Es setzt eine Infrastruktur voraus, die aus einer Anzahl von flexiblen Agenten besteht. Flexible Agenten können beliebig in Gruppen organisiert werden, die kooperativ Managementanwendungen realisieren. Das Hauptmerkmal dieser Agenten ist ihre Kooperationsfähigkeit und Erweiterbarkeit.

Das System beinhaltet auch traditionelle Managementagenten (z.B. SNMP-Agenten) und Manager. Der Unterschied zwischen einem flexiblen Agenten und einem Manager ist nicht mehr sehr groß und basiert auf Kriterien wie dem Vorhandensein eines Displays oder dem Standort usw. Es gibt eine Anzahl von Managementfunktionen (z.B. alarm, topology, history), von denen einige in den flexiblen Agenten und einige im Manager durchgeführt werden. Flexible Agenten können sowohl als „Delegator“ als auch als „Delegee“ agieren (*dual role entities*). Die Delegierung erfolgt durch einen *pull* Modell. Zu diesem Zweck wird ein *Agenten-Ressourcen-Server* verwendet.

Die Architektur des flexiblen Agenten wird in Abbildung 3 dargestellt. Dabei stellen sich folgende Anforderungen:

- Generischer Ansatz (d.h. unabhängig von der unterliegenden managementplattform)
- Bereitstellung von speziellen Kommunikations-, Delegierungs- und Sicherheitsdiensten
- Portabilität

- Möglichkeit zur Kontrolle der delegierten Funktionen

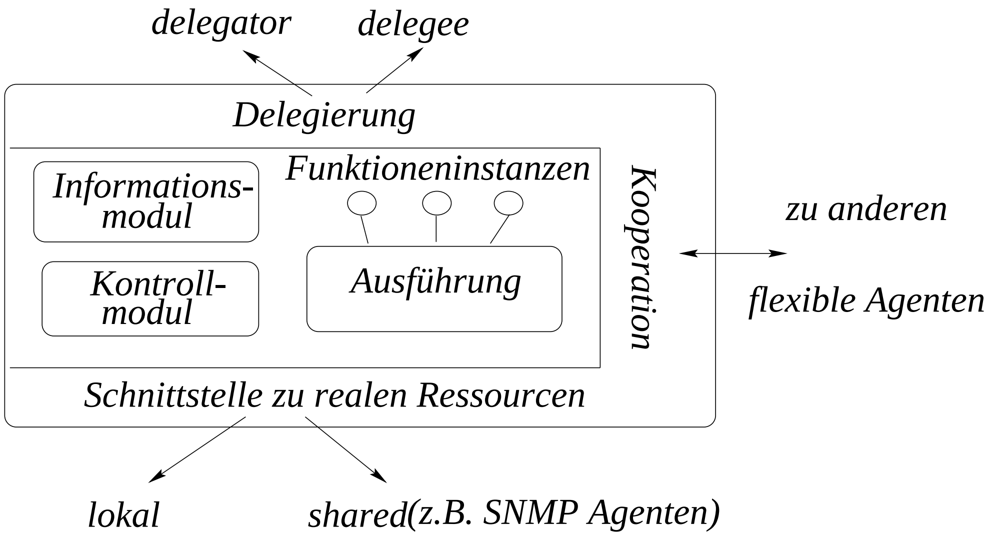


Abbildung 3: Architektur flexibler Agenten

Ein Trader wird für die Identifizierung der Kommunikationspartner und für das „Advertsising" der von den Agenten angebotenen Dienste verwendet. Bezüglich der Sicherheit werden Authenifizierungs- and Verschlüsselungsverfahren eingesetzt. Darüberhinaus werden Konzepte und Kriterien für die Bildung von Domanen und die Strukturierung von Agenten in den Domänen entworfen.

5 Methodik

Das oben beschriebene Managementmodell wird mit Konzepten aus dem Bereich von *Cooperative Distributed Problem Solving (CDPS)* kombiniert, um eine Methodik ([Mou96a], [Mou96b]) zu entwickeln. Diese Methodik dient zur flexiblen, verteilten Realisierung von komplexen Managementaufgaben und beinhaltet folgende Schritte: (i) Treffen der Entscheidung, ob sich die Managementanwendung für eine verteilte Realisierung eignet. (ii) Zerlegung der Aufgabe in Teilaufgaben gemäß des Konzepts von CDPS. (iii) Beschreibung der Teilaufgaben. (iv) Realisierung in einer verteilten Managementarchitektur.

Die Methodik wird in Abbildung 4 graphisch dargestellt.

1. *Identifizierung der Aufgabe*

   Als erstes soll herauskristallisiert werden, ob die Managementaufgabe für eine verteilte Realisierung geeignet ist. Für die meisten Managementanwendungen ist das normalerweise der Fall. Relevante Kriterien sind auch in [MEB+95] zu finden.

2. *Aufgabenzerlegung*

   Wenn eine verteilte Realisierung bevorzugt wird oder erforderlich ist, wird die Aufgabe in die entsprechenden Teil-/aufgaben zerlegt. Leider gibt es keinen Algorithmus

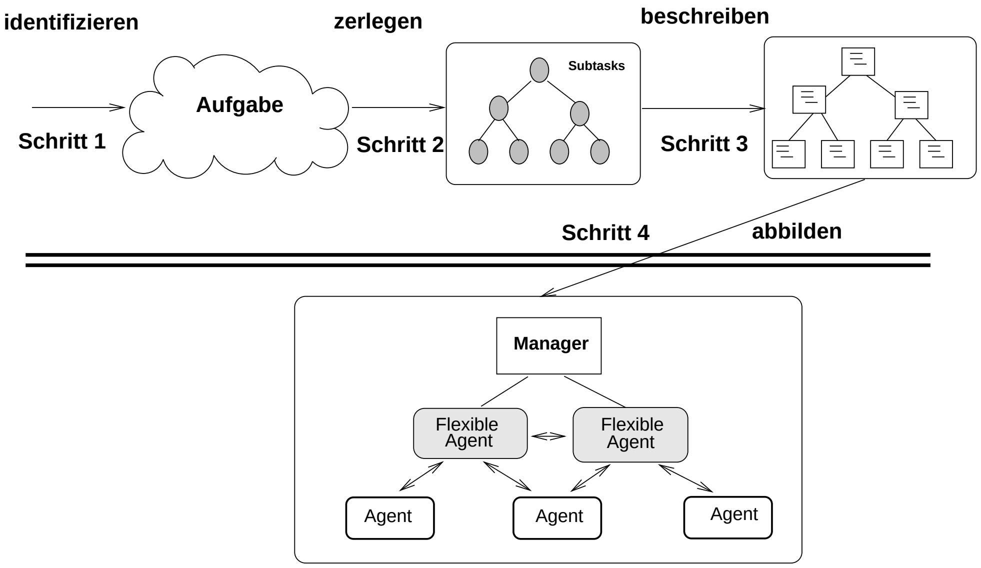


Abbildung 4: Methodik

zur Problemzerlegung. Die spezielle Art der Zerlegung basiert nur auf bestimmten Kriterien und ist natürlich anwendungs- und umgebungsspezifisch. Folgende allgemeine Kriterien können aber aufgezählt werden:

- parallele Ausführung der Teilaufgaben
- geringe Interaktion unter den Teilaufgaben
- effiziente Verwendung der Ressourcen
- Parametrisierung, Wiederverwendbarkeit
- verschiedene Informationstypen und -qualität
- verschiedene Problemlösungsstrategien

3. *Beschreibung des Multi-Agenten-Systems*

   Die aus der Zerlegung der globalen Aufgabe resultierenden Teilaufgaben müssen formal beschrieben werden. Zu diesem Zweck werden zwei Templates verwendet: (i) ein *Task Template* und (ii) ein *Agent Template*.

   Das Task Template bietet eine generische Beschreibung der Funktion, während das Agent Template die Instanziierung der Funktion in einer konkreten Umgebung beschreibt. Auf diese Art und Weise können die Beschreibung einer Funktion und ihre Realisierung für ein spezifisches Szenario einschließlich der Verteilungs- und Abhängigkeitsaspekten zwischen den Agenten voneinander getrennt erfolgen.

Task Template

Name: Name der Funktion
Input_Data: Daten
Input_Data_Type: Datentyp
Output_Data: Ergebnis
Output_Data_Type: Ergebnistyp
Function: Beschreibung der Funktion

Agent Template

Domain: Zuständigkeitsdomäne
Delegated_From: ID des Delegators
Delegated_To: id des Ausführungsmoduls
Agent_group: zugehörige Gruppe von Agenten in der konkreten Domäne
Agent_Group_Leader: Gruppenleiter
Consumers: Agenten, die Ergebnisse dieses Agenten verwenden
Producers: Agenten, die Input liefern
Partners: Agenten mit der gleichen Funktionalität in der Domäne
Execution_type: kontinuierlich, once-off oder triggered
Trigger_Mode: ereignis- oder benutzergesteuert
Privileges: zur Reihenfolgefestlegung und Synchronisierung
Security: Authorisierung, Zugriffsrechte
Capability: spezielle Anforderungen

4. *Abbildung auf die Managementarchitektur*

   Als letztes muß das Konzept auf die Managementarchitektur abgebildet werden, d.h. die Teilaufgaben müssen Managementagenten zugewiesen werden. Die Abbildung hat Auswirkungen auf alle vier Teilmodelle einer Managementarchitektur. Aufgrund der Verteilung gibt es neue Anforderungen an die im Agenten enthaltene Information. Es genügt nicht mehr, daß der Agent Information nur über sich selbst hat. Er soll auch über Information bezüglich der Fähigkeiten anderer Agenten und bezüglich des Kooperationsschemas verfügen. Normalerweise ist es nicht notwendig, Information über alle Agenten zu besitzen, sondern nur über eine relevante Teilmenge davon. Wegen der Kooperation unter den flexiblen Agenten braucht man neue Formen von Kommunikation. Das beinhaltet den Austausch von Nachrichten und Information unter Agenten und die Delegierung von neuer Funktionalität. *Knowledge Query Manipulation Language (KQML)* [FFMM93] ist ein Protokoll, das ein Message-Format und Message-Handling-Fähigkeiten anbietet, die auch für den hier vorgestellten Ansatz nötig sind.

6 Stand der Arbeit

Um die Anforderungen zu erfüllen, die sich aus dem hohen Grad der Verteilung und Dynamik in heutigen verteilten Systemen ergeben, ist der Einsatz von verteilten, flexiblen

Managementsystemen notwendig. Zu diesem Zweck werden Konzepte aus dem Bereich der intelligenten Agenten benutzt. In dieser Arbeit werden es ein Rahmenwerk vorgestellt, das die resultierenden Fragestellungen umfaßt und eine neue Agentenarchitektur, die diese Anforderungen erfüllt. Darüberhinaus wird eine Methodik entwickelt, die für die verteilte Realisierung von komplexen Managementaufgaben eingesetzt wird.

Eine prototypische Implementierung des Konzepts mit Hilfe des an der Carnegie Mellon Universität entwickelten *Java Agent Template* [Fro96] wird momentan für Fehlermanagementszenarien angestrebt. Weiterhin wird auch die Fragestellung der Indentifizierung von generischen delegierbaren Managementfunktionen verfolgt. Dabei werden existierende Zerlegungsansätze auf ihre Tauglichkeit für die Definition und Spezifizierung von delegierbaren Funktionen untersucht und bewertet. Abbildung des Kommunikations- und Kooperationsprotokolls zwischen flexiblen Agenten (auf der Basis von KQML) in existierende Transportprotokolle wird auch angestrebt.

7 Zusammenarbeit

Die Arbeiten, die im Rahmen des Teilprojekts durchgeführt wurden, führten zu einer Anzahl von Kooperationen. Die Zusammenarbeit im Rahmen des Graduiertenkollegs wird in dem entsprechenden Arbeitskreisbericht beschrieben. Hier wird nur auf die externe Zusammenarbeit näher eingegangen.

Industrie / Wirtschaft: Von großer Wichtigkeit ist die Kooperation mit den Hewlett Packard Laboratories in Bristol. Sie hat die Form des Austausches von Ideen und Erfahrungen im Bereich von intelligenten Agenten für Netz- und Systemmanagement. Dabei wurde uns die Teilnahme an der Erstellung von Konzepten bei Hewlett Packard über agentenbasiertes, verteiltes Management ermöglicht. Diplomarbeiten von Studenten der Technischen Universität München finden ebenfalls im Rahmen dieser Kooperation in Bristol statt. Im Graduiertenkolleg wurde von dieser Kooperation mehrmals berichtet. Im Rahmen der Arbeitskreistreffen wurde über die daraus gesammelten Erfahrungen diskutiert, damit die anderen im Agentenbereich tätigen Kollegiaten davon profitieren können.

Hochschulen und Forschungseinrichtungen:

Das Teilprojekt wird in Zusammenarbeit mit dem von Prof. Hegering geleiteten Münchner Netzmanagement Team (MNM-Team) durchgeführt. Das MNM-Team beschäftigt sich seit vielen Jahren mit Fragestellungen im Bereich von Netz- und Systemmanagement. In diesem Rahmen besteht intensive Kooperation mit vielen anderen Mitgliedern der Gruppe. Das vorhandene Wissen in allen Funktionsbereichen vom Netz-, System- und Anwendungsmanagement (nämlich Konfigurations-, Abrechnungs-, Leistungs-, Sicherheits- und Felhermanagement) bieten eine wichtige Hilfestellung beim Einsatz von kooperierenden Agenten in diesem Bereich. Dabei ergeben sich viele unterschiedliche Einsatzszenarien und das Konzept kann unter unterschiedlichsten Aspekten untersucht werden. Durch diese Zusammenarbeit und die fruchtbaren Diskussionen entstanden wissenschaftliche Papiere, die auf internationalen Konferenzen vorgestellt oder in wissenschaftlichen Zeitschriften veröffentlicht wurden (z.B. [CM96]).

Zusammenarbeit findet auch mit dem Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) statt. Diese Kooperation dient der Betrachtung des Konzepts des verteilten Managements unter dem Gesichtspunkt einer großen Produktionsumgebung und der Problemem, die sich im täglichen Betrieb ergeben. Durch diese Kooperation entstanden ebenfalls Veröffentlichungen (z.B. [MDR96]).

Die an der der Universität von Pretoria, Südafrika, durchgeführten Arbeiten im Umfeld von intelligenten Agenten und Management Policies waren ein Berührungspunkt zu den in diesem Teilprojekt durchgeführten Arbeiten. Aufgrund einer darauf basierender Zusammenarbeit entstand ein Papier über den Einsatz von intelligenten Agenten für die verteilte Implementierung von Management Policies. Ein Wissenschaftler aus der Universität von Pretoria verbrachte einen Forschungsaufenthalt in München und hielt während dieser Zeit auch einen Vortrag im Rahmen der Informatikolloquiums an der LMU München für das Graduiertenkolleg.

8 Forschungsaufenthalte

Im Rahmen der oben beschriebenen Kooperation mit Hewlett Packard fanden gegenseitige Besuche statt, die hauptsächlich zum Austausch von Forschungsergebnissen und zur gemeinsamen Durchführung von Analysen der untersuchten Aspekte dienten. Ein Beispiel ist die Zusammenarbeit für die Spezifikation eines Kommunikationsprotokolls zwischen den von Hewlett Packard entwickelten verteilten *Flipper* Agenten. Die diesbezüglich getroffenen Entscheidungen konnten natürlich auch in einem breiteren Kontext im verteilten Management verwendet werden. Außerdem konnte man bei der Spezifikation der Architektur der verteilten Agenten und ihrem Einsatz in der von Hewlett Packard entwickelten Netzmanagementplattform *HP OpenView* teilnehmen. Das trug wesentlich dazu bei, eine konkretere und realistische Vorstellung, über die aktuelle Probleme, die der Einsatz von kooperierenden Agenten im Management mit sich bringt, zu bekommen.

Die Forschungsaufenthalte fanden in Form von unregelmäßigen Besuchen bei Hewlett Packard mit einer jeweiligen maximalen Dauer von zwei Wochen.

Literatur

[CL94] J. Case und D. Levi. SNMP Mid-Level-Manager MIB. Internet draft, IAB, 1994. obsoleted by DISMAN drafts.

[CM96] L. Conradie und M.-A. Mountzia. A Relational Model for Distributed Systems Monitoring using Flexible Agents. In *Proceedings of the Third International Workshop on Services in Distributed and Networked Environments (SDNE 96), Macau*, 1996.

[CMRW93] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. Manager-to-Manager Management Information Base. RFC 1451, IAB, April 1993.

[FFMM93] T. Finin, R. Fritzson, D. McKay und R. McEntire. KQML: an Information and Knowledge Exchange Protocol. *Proceedings of International Conference on Building and Sharing of Very Large Scale Knowledge Bases*, Dezember 1993.

[Fro96] R. Frost. Java Agent Template. http://cdr.stanford.edu/ABE/JAVAAgent.html, 1996.

[GM95] J. Gosling und H. McGuilton. The JAVA Language Environment (White Paper). Technical report, Sun Microsystems, Inc., Oktober 1995.

[Gol96] G. Goldszmidt. *Distributed Management by Delegation*. Dissertation, Columbia University, 1996.

[GW94] C. Guilfoyle und E. Warner. Intelligent Agents: The New Revolution in Software. Technical report, OVUM, 1994.

[HA94] H.-G. Hegering und S. Abeck. *Integrated Network and System Management*. Addison-Wesley, 1994.

[HCK95] C. Harrison, D. Chess und A. Kershenbaum. Research Report, Mobile Agents: Are they a good idea? Technical report, IBM T.J. Watson Research Center, 1995.

[ISO94] ISO/IEC. Information Technology – Open Systems Interconnection – Systems Management – Management Functions. IS 10164-x, ISO/IEC, 1991-94.

[ISO96] ISO/IEC. Information Technology – Open Systems Interconnection – Systems Management – Part 21: Command Sequencer. DIS 10164-21, ISO/IEC, August 1996.

[LS96] D. Levi und J. Schönwälder. Script MIB: Definition of Managed Objects for the Delegation of Management Scripts. Internet draft, IAB, November 1996. Work in progress.

[MDR96] M. A. Mountzia und G. Dreo-Rodosek. Delegation of Functionality: Aspects and Requirements on Management Architectures. In *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations & Management, L'Aquilla, Italy*, Oktober 1996.

[MEB+95] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt und Y. Yemini. Decentralizing Control and Intelligence in Network Management. In Adarshpal S. Sethi, Yves Raynaud und Fabienne Faure-Vincent, Hrsg., *Proceedings of the 4th IFIP/IEEE International Symposium on Integrated Network management, Santa Barbara, CA*, Mai 1995.

[Mou96a] M. A. Mountzia. An Intelligent-Agent based framework for distributed systems management. In *Proceedings of the Third HP OVUA Workshop*, Marz 1996.

[Mou96b] M. A. Mountzia. Intelligent Agents in Integrated Network and Systems Management. In *Proceedings of the EUNICE'96 Summer School, Lausanne, Switzerland*, September 1996.

[Wal95] S. Waldbusser. Remote Network Monitoring Management Information Base. RFC 1757, IAB, Februar 1995.

[Wel95] B. B. Welch. *Practical Programming in Tcl and Tk*. Prentice Hell International, 1995.

[YGY91] Y. Yemini, G. Goldszmidt und S. Yemini. Network Management by Delegation. In I. Krishnan und W. Zimmer, Hrsg., *Proceedings of the 2nd International Symposium on Integrated Network Management*. Elsevier Science Publishers B. V. (North Holland), April 1991.

1.3.6 Agentengestützter Informationsaustausch in globalen Informationsräumen

Teilprojekt: Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit
Martina Nöhmeier

1 Einführung

1.1 Motivation

In den letzten Jahren war eine Vielzahl an technologischen sowie organisatorischen Entwicklungen zu beobachten, die dazu führten, daß sich sowohl die Bedeutung als auch der Fokus des Forschungsgebietes CSCW (Computer Supported Cooperative Work) deutlich vergrößert haben. Mit zunehmender Verbreitung des **distributed personal computing** [Dou96] sowie zahlreichen organisatorischen Umstrukturierungsmaßnahmen war häufig eine Dezentralisierung von Entscheidungs- und Kontrollfunktionen verbunden, sowie die zunehmend verteilte, kooperative Bearbeitung von Aufgabenstellungen in Industrie, Forschung und Lehre. Diese Vorgehensweise bringt jedoch mit sich, daß die zur Aufgabenplanung-, -durchführung, -kontrolle und -koordination nötigen Informationen in der Regel nicht in einer zentralen Instanz konzentriert, sondern über mehrere Wissensträger verteilt sind. Aufgrund der essentiellen Bedeutung von Information für Organisationen in Industrie, Forschung und Lehre [BG89], ist daher eine wichtige, im Zusammenhang von CSCW immer häufiger diskutierte Aufgabe die Unterstützung von Informationsaustausch. Eng gekoppelt an das Entstehen loser, z.T. virtueller Kooperationsstrukturen ist die Entwicklung eines vielfältigen Kontinuums von lokalen bis hin zu globalen Informationsräumen [5], die i.a. Verbindungen untereinander aufweisen und innerhalb derer Informationsaustausch stattfinden kann. Den theoretisch zahlreichen zusätzlichen Möglichkeiten zum Informationsaustausch steht jedoch die Problematik gegenüber, daß bisherige Werkzeuge für Informationsaustausch, die im klassischen Groupware-Bereich existieren, in globalen Informationsräumen nicht geeignet einsetzbar sind und andererseits trotz des Auftauchens von einer mittlerweile fast unüberschaubaren Anzahl an Indexierungs- und Suchwerkzeugen [WP96] nicht die Unterstützung geboten ist, die für Informationsaustausch in globalen Informationsräumen wünschenswert wäre.

1.2 Zielsetzung

Eine mögliche Alternative zu bisherigen Werkzeugkonzepten ist der Einsatz von Agenten. In Prognosen über die Entwicklung von Agententechnologie werden Aufgaben aus dem Bereich Informationsaustausch häufig als zukunftsträchtiges Einsatzgebiet von Agenten klassifiziert [GW94]. Agententechnologie ist jedoch kein homogenes, klar definiertes Hilfsmittel, das eine für jedes Aufgabenszenario eindeutige Strategie zur Entwicklung einer

[5] Unter dem Begriff **Informationsraum** sei dabei eine Menge von inhaltlich vernetzten Datenbestände und damit in Zusammenhang stehenden Informationsdienste zu verstehen, die für Informationsaustausch genutzt werden.

Problemlösung vorgibt. Vielmehr verbergen sich hinter diesem Konzept vielfältige Methoden, Prinzipien, Architekturen und Sprachen. Trotz oft euphorischer Prognosen, wird folglich oft nur sehr vage dokumentiert, warum sich Agenten für ein Aufgabengebiet wie Informationsaustausch eignen, welche nachvollziehbaren, wiederholbaren Vorgehensweisen zu positiven agentenbasierten Problemlösungen geführt haben oder welche Agentenmodelle die Anforderungen dieses Aufgabenbereiches zufriedenstellend abdecken.

Um tatsächlich entscheiden zu können, ob Agenten zu einer problemgerechten Unterstützung von Informationsaustausch in globalen Informationsräumen geeignet sind, sollen daher folgende 2 Teilziele in dieser Arbeit verfolgt werden:
1. Untersuchung der problematischen Aufgabencharakteristika und Anforderungen an die Unterstützung von Informationsaustausch in globalen Informationsräumen.
2. Analyse der Einsetzbarkeit von Agententechnologie für den aus 1. abgeleiteten Unterstützungsansatz.

Hierbei sind wichtige Teilfragestellungen die Identifikation von angestrebten Nutzeffekten von Agenten beim Informationsaustausch, die explizite Beschreibung einer gezielten Vorgehensweise, um die angestrebten Charakteristika der Lösung auch verwirklichen zu können, die Beschreibung des erforderlichen Agentenmodells (und dessen Abdeckung durch Bestandteile existierender Agentenarchitekturen), sowie eine Beschreibung wichtiger, für die Unterstützung von Informationsaustausch spezifischer Strategien und Beschreibungsformate, die kooperierende Agenten benötigen, um anfallende Teilaufgaben erfolgreich beschreiben und bearbeiten zu können.

2 Informationsaustausch in globalen Informationsräumen

2.1 Hintergrundentwicklungen

Die Analyse der Ist-Situation im Bereich des Informationsaustausches in globalen Informationsräumen kann aufgrund der der Entwicklungsgeschwindigkeit in diesem Bereich nur eine Momentaufnahme sein. Diese Momentaufnahme soll daher in die vorausgehenden und z.T. vorbereitenden Entwicklungen eingeordnet werden, um einen besseren Überblick für die mögliche weitere Entwicklungsrichtung zu erhalten.

Bei Betrachtung der Hintergrundentwicklungen lassen sich folgende Haupt-Etappen erkennen:

1. Schaffung bzw. Verbesserung der 'technischen' Voraussetzungen für elektronischen Informationsaustausch in globalen Netzen (Hardware-Vernetzung und Protokolle zum Zugriff auf entfernt liegende Daten)

2. Verbesserung der Zugriffsmöglichkeiten auf entfernte Information bzw. deren Anbieten für ein globales Publikum v.a. durch WWW (URL, Hypertextprinzip, Browser)

3. Aufgrund dieser Verbesserungen: Zunehmende Teilnahme am Aufbau eines zusammenhängenden Informationsraumes und darauf basierende neue Möglichkeiten einer sehr losen Form der Zusammenarbeit durch Informationsaustausch, die u.a. durch das Wegfallen räumlicher Grenzen, durch große Zahl aktiver und passiver Teilnehmer, sowie die Dominanz des Holprinzipes gekennzeichnet ist.

4. Zunehmende Variation der Inhalte des Informationsraumes, die im Rahmen des Informationsaustausches angeboten werden oder diesen unterstützen Es werden also nicht mehr nur passive Dokumente mit Rohinformationen angeboten, sondern auch Metainformationen über diese Rohinformation, Orientierungshilfen, Suchwerkzeuge und andere 'aktive' Schnittstellen zu lokalen Datenbeständen.
   Diese heterogenen Materialien werden dezentral angeboten und sind derzeit in interaktiver Weise über die jeweilige URL aus der Browserumgebung heraus ansprechbar (nutzbar).

2.2 Kontextanalyse für das Aufgabengebiet Informationsaustausch in globalen Informationsräumen

Neben den Hintergrundentwicklungen muß als zweiter Schritt der Anforderungsanalyse der Kontext des Aufgabenbereiches Informationsaustausch in globalen Informationsräumen betrachtet werden. Die Kontextstrukturierung erfolgt in Anlehnung an das NATURE-Weltenmodell [JPD+94].

Die explizite Zusammenstellung der relevanten Kontextbereiche soll dabei eine vollständigere, nicht zu einseitige Identifikation der kritischen Gegebenheiten und offenen Wünsche ermöglichen, die eine solidere Basis für das zu entwickelnde Unterstützungskonzept darstellt, als ein sporadisches Aufgreifen von isolierten, zusammenhangslosen Anforderungen.

Zum anderen dient das Weltenmodell dazu, die deutlich gewordenen Problemstellungen, offenen Wünsche und Herausforderungen zu gliedern, indem sie dem jeweiligen Kontextbereich zugeordnet werden. Durch diese Verbindung von Kontext und Anforderungen, die als Basis für das Unterstützungsmodell verwendbar ist, soll ein bruchloserer Übergang von Anforderungsanalyse und zu Problemlösungsentwicklung vollzogen werden.

Das Weltenmodell für Informationsaustausch enthält im wesentlichen vier Grundbestandteile:

1. Nutzer globaler Informationsräume wie z.B. WWW und deren Arbeitsumfeld
Diese Menge umfaßt alle Personen, die am Informationsaustausch in globalen Informationsräumen partizipieren, sei es als aktiver Anbieter oder passiver Nutzer des Informations-Angebotes.

2. Informationsangebot, das über globale Informationsräume wie z.B. WWW erreichbar ist.

3. Rechner-Systeme und deren Netzverbindungen, d.h. das technische Gerüst, das die Basis für die Verbindung zwischen Benutzerwelt, Informationsangebot und Informationsdiensten darstellt.

4. Informationswerkzeuge, die auf Basis des Informationsangebotes das Arbeiten mit diesen Informationen erleichtern oder sogar dynamisch aus schwer direkt nutzbaren Informationslagern geeignete Informationen zusammenstellen bzw. Eingabeinformation transformieren.
3 identifizierte wichtige Analysekriterien für diesen Kontextbereich sind die Aufgaben, die Inhalt angebotener Werkzeuge und Informationsdienste sind (z.B. domänenspezifische Suche nach Primärinformation oder nach Einstiegspunkten, Übersetzung), die vorgesehenen Varianten zur Nutzung dieser Informationsdienstleistungen, sowie die Alternativen, wie Informationsdienstleistungen zur Nutzung durch andere Teilnehmer angeboten werden können.

Die Gesamtmenge der Hilfsmittel weist prinzipiell eine große Leistungsvielfalt auf, für verschiedene Informationswünsche sind jedoch jeweils unterschiedliche Werkzeuge, bzw. Kombinationen daraus von Nöten. Die möglichen Angebots- und Nutzungsformen sind im Gegensatz zu den vertretenen Dienstinhalten noch sehr starr.

2.3 Kritische Gegebenheiten und offene Wünsche

Durch die Betrachtung der genannten Entwicklungstendenzen sowie des Kontexts und die vergleichende Betrachtung der Leistungsspektren und Nutzungsbedingungen der verschiedenen Werkzeuge treten eine Fülle verschiedener Probleme zutage, mit welchen die derzeitige Unterstützung von Informationsaustausch in globalen Informationsnetzen behaftet ist.

- Die erste Gruppe bilden hierbei kritische Gegebenheiten innerhalb der für Informationsaustausch zur Verfügung stehenden Informationsobjekte. Beispiele hierfür sind Größe, Heterogenität, extreme Dezentralität der Informationsmenge, hohe Dynamik und Unübersichtlichkeit des Informationsraumes.

- Die zweite Kategorie umfaßt kritische Gegebenheiten innerhalb der Benutzerwelt, die am Informationsaustausch beteiligt ist.

- Schließlich sind Mängel und offene Wünsche innerhalb der Werkzeugwelt für Informationsaustausch in globalen Netzen sichtbar, die sich in zwei Hauptgruppen gliedern: Mängel und offene Wünsche bezüglich des Leistungsspektrums, der Nutzungsformen und Angebotsmöglichkeiten und andererseits problematische Aspekte bezüglich des Werkzeugdesigns bzw. der Werkzeugarchitektur (Abgeschlossenheit, Ausrichtung auf isolierte, interaktive Nutzung)

Beispiele für einzelne Problemaspekte werden in [Nöh96] genannt.

2.4 Fazit

Die kritischen Eigenschaften von Benutzerwelt und Informationsraum sind eine der wesentlichen Ursachen dafür, daß bisherige Methoden der Werkzeugentwicklung und -realisierung noch nicht die wünschenswerte Unterstützungsqualität erzielt haben.

Die bisherigen Konzepte kranken i.a. nicht an der mangelnden Implementierbarkeit einer spezifizierten Aufgabe, sondern z.B. an unvorteilhafter oder mangelnder Strukturierung der zahlreichen zugrundeliegenden Aufgabenabläufe bzw. an einem zu starren und wenig anpaßbaren Einfrieren des bei der Lösungsentwicklung aktuellen Status Quo. Die ist oft verbunden mit einer Überspezifikation der Aufgabenabläufe. Es ist daher wenig sinnvoll, ein weiteres abgeschlossenes Werkzeug mit den vorgenannten Charakteristika zu realisieren, das möglichst viele Leistungsbestandteile anbietet.

Es muß vielmehr eine Methode entwickelt werden, wie bisherige dezentral vorhandene Werkzeugleistungen, Orientierungshilfen bzw. Nutzungsvarianten von Werkzeugen im Rahmen des Informationsaustausches situationsgerecht und in homogener, inhaltsorientierter und sogar kombinierter Form nutzbar gemacht werden können, so daß auf diese Weise das Arbeiten mit diesen Werkzeugen und Orientierungshilfen auf WWW-Basis wesentlich erleichtert werden kann.

Dies stellt im Grunde eine Fortsetzung der bereits in Abschnitt 2.1 angesprochenen Entwicklungskette dar, in der ja in den vorhergehenden Schritten zunächst dezentral liegende Daten und verschiedene Zugriffsprotokolle für **interaktiven**, inhaltsbasierten Zugriff auf die Informationen in homogener Weise nutzbar gemacht wurden (⇒ Browser) Eine vergleichbare Nutzungs-Homogenisierung soll nun für die nicht-interaktive Nutzung von Informationswerkzeug-Leistungen erreicht werden (⇒ Vermittler).

Das Ziel dieses Ansatzes ist also ein Konzept, das eine situationsgerechte Vermittlung von Informationsdiensten und Orientierungshilfen unter Verwendung dezentral existierender Werkzeuge durch Delegation von Anfragen an Agenten vorsieht.

3 *Agentenbasiertes Problemlösen*

Agententechnologie wird in der Literatur sehr kontrovers diskutiert. Ebenso vielfältig wie die Versuche zur Definition dieses Begriffes sind auch die Methoden zur Entwicklung von Problemlösungen auf der Basis eines dieser Agentenmodelle.

Das nachfolgend skizzierte Modellierungsvorgehen verwendet als Orientierung für die Notwendigkeit der einzelnen Arbeitsschritte abstrakte Problemcharakteristika, die aus der Anforderungsanalyse für den Bereich Informationsaustausch in globalen Informationsräumen abgeleitet wurden. Beispiele für solche abstrakten Problemcharakteristika sind in folgender Aufstellung beschrieben:

- Notwendigkeit von Integration heterogener Wissensquellen und Leistungserbringer
- Es existieren keine starren, gleichbleibenden Aktivitätsfolgen, vielmehr ist flexible, dynamische Erweiterbarkeit erforderlich
- Oft sind keine algorithmischen Verfahren, sondern Heuristiken die Basis für die Problemlösung, diese ändern sich erfahrungsgemäß häufig
- Das Anwendungsszenario umfaßt keine abgeschlossenen Systeme und Ressourcenquellen bzw. Datenrepositories
- Bei den einzelnen Systemkomponenten ist eine unterschiedliche Dynamik zu beobachten (es gibt eher statische und eher dynamische Komponenten)
- Häufig ist ein fließender Übergang zwischen Kooperation mit systeminternen und systemexternen Komponenten notwendig, hervorgerufen durch den natürlichen Verteiltheitscharakter der Aufgabenstellung (globale Informationsräume).

Es wird hierbei ein Mischvorgehen verwendet, bei welchem einerseits zunächst top-down eine Aufgabenzerlegung und -strukturierung durchgeführt wird, auf deren Basis schließlich ein strukturiertes Unterstützungsmodell für Informationsaustausch in globalen Informationsräumen erstellt wird. Hierauf erfolgt bottom-up der Aufbau eines geeigneten Agentenmodells, das bestmöglichst Standardisierungsversuche für Agententechnologie berücksichtigt, aber auch gezielt die Basiseigenschaften bereitstellt, die für den Aufgabenbereich

Informationsaustausch wesentlich sind, um die jeweiligen häufig benötigten Elemente nicht immer wieder neu entwerfen und aufeinander abstimmen zu müssen. In diese Basisschablonen können schließlich im letzten Modellierungsschritt aufgabenspezifische Teilfunktionalitäten integriert werden, die zur Teilautomatisierung der im 1. Modellierungsschritt entwickelten Unterstützungsworkflows erforderlich sind.

3.1 Strukturiertes Aufgabenmodell

Für das Aufgabenmodell wurde eine Reihe von Teilaufgaben abgeleitet, die sich aus den Anforderungen ergeben und die Workflows repräsentieren, die durch den Einsatz von Agenten teilautomatisiert bzw. integriert werden sollen. Die folgende Aufstellung listet einige wichtige Aktivitätsguppen und die zugrundeliegenden Anforderungen auf, eine vollständige detaillierte Beschreibung der darin enthaltenen, von Agenten zu automatisierenden Teilworkflows erfolgt mittels Spezifikationsmethoden aus dem Bereich des Workflow-Management. Dies würde jedoch den Rahmen dieses Berichtes sprengen.

- Ein Passiver Informationsnutzer hat Informationsbedarf ⇒
  WF: Spezifizieren des aktuellen Informationsbedarfs (z.B. deklarativ durch Einschränkung inhaltlicher oder organisatorischer Eigenschaften

- Informationsnutzer haben unterschiedliches Arbeitsumfeld z.B. bezüglich Hardwareausstattung, Mobilität, Arbeitsrhythmus und benötigen daher verschiedene Formen der Nutzung ⇒
  WF: Spezifizieren von gewünschten Nutzungsbedingungen für eine oder mehrere Informationsanfragen

- Der Informationsnutzer will (zumindest während eines Arbeitsvorganges) organisatorische Daten nicht immer wieder neu eingeben ⇒
  WF: Einrichtung und Verwaltung einer benutzerbezogenen, temporären Arbeitsumgebung innerhalb des Vermittlersystems

- Die Vielfalt existierender Informationswerkzeuge und Orientierungshilfen macht es für den Benutzer fast unmöglich, den Überblick über alle für ihn nützlichen Werkzeuge zu behalten. ⇒
  WF: Auswahl von geeigneten Werkzeugen für Informationsbedarf innerhalb verschiedener Informationsdomänen, verschiedener Reichweite, verschiedener Ausführlichkeit

- Ein Informationsnutzer will Anfrage nicht spezifisch formatiert und formuliert für jeden benötigten Informationsdienst einzeln stellen ⇒
  WF: Nachbearbeiten eines spezifizierten Auftrages (Anfrage + Nutzungsbedingungen) und Aufbereitung der Anfrage für reale Informationsdienste

- Das Leistungsspektrum der existierenden Informationsdienste und der Umfang des zugrundeliegenden Informationsangebotes ist so groß, daß nicht alle zur Unterstützung des Informationsaustausches wünschenswerten Funktionalitäten in einem Werkzeug nachgebildet werden können. ⇒
  WF: Delegation von atomaren Anfrageteilen an externe Informationsdienste

- Es gibt keine Zentrale, die alle existierenden Werkzeuge und deren genaue Schnittstellen kennt. Die Werkzeuge können aber nur für Aufgabendelegation durch Agenten genutzt werden, wenn ausreichende, geeignet strukturierte Information über ihre Leistungen und Schnittstellen zur Verfügung steht. Das Angebot an Werkzeugen ändert sich jedoch zu dynamisch, um eine statische Anbindung dieser Informationsdienste und Orientierungshilfen an das Vermittlungssystem nutzen zu können. ⇒
WF: Registrierung und dynamisches Einbinden von externen Informationsdiensten in das Vermittlungssystem, sowie dynamisches Aktualisieren und Entfernen von Registrierungsinformation

3.2 Agentenmodell

Bei der Auswahl des für den Aufgabenbereich Informationsaustausch geeigneten Agentenmodells wurden zahlreiche in der Literatur beschriebene Agentenmodelle (u.a. [DWS96], [EW95], [Hau94] auf ihre Verwendbarkeit für die Modellierung des zu entwickelnden Vermittlungssystems untersucht.

Bei dem für den Vermittlungsansatz erforderlichen Agentenmodell handelt es sich um ein Modell kooperierender Agenten. Grundsätzlich ergeben sich hieraus zwei Grundklassen von Agenteneigenschaften, die zu beschreiben sind, sog. Makroeigenschaften bzw. Mikroeigenschaften.

Für die flexible Kooperationsfähigkeit von Agenten sind v.a. drei Aspekte von tragender Bedeutung: Nebenläufiges Arbeiten, Kommunikation und Koordination der Agenten [Mül96]. Aus dem Bereich der Makroeigenschaften sind folglich für das Agentenmodell dieser Arbeit folgende Charakteristika relevant:

- Nebenläufiges Arbeiten:
Diese Eigenschaft stellt einen Teilaspekt der häufig mißverständlich geforderten 'Autonomie' dar. Agenten sollten nicht lediglich eine Strukturierung einer sequentiellen Aktivitätenfolge sein, sondern vielmehr nebenläufig ihre zugewiesenen Teilaufgaben erledigen, wobei Zusammenhänge zwischen Teilaufgaben und notwendige Synchronisierungen durch strukturierte Kommunikation auf der Basis von Sprechakten [Sea80] und durch nachfolgend beschriebene Koordinationsmechanismen berücksichtigt werden.

- Kommunikationsfähigkeit von Agenten untereinander:
Um einerseits im Inhalt der Nachricht beliebige Informationen integrierbar zu machen und andererseits zu gewährleisten, daß Agenten die für sie relevanten, enthaltenen Informationen unabhängig von einer starren, vorher bekannten Reihenfolge interpretieren können, wird komplexe Kommunikation auf der Basis strukturierter Nachrichten gefordert.
Da im Hinblick auf die Kommunikationsfähigkeit zwischen verschiedenen Agentengemeinschaften möglichst standardisierte Nachrichtenformate wünschenswert sind, wurden als Vorbild für das Nachrichtenformat, sowie die Strukturierung des Nachrichteninhalts Bestandteile einer vorgeschlagenen Standard-Agentenkommunikationssprache [FIP97] herangezogen.

- Die Koordination zwischen den Agenten wird durch den Einsatz folgender Methoden verwirklicht:
  - Konfliktvermeidung und Strukturierung der Agentengemeinschaft durch statische und dynamische Rollen
    Hierbei werden im Agentenmodell dieser Arbeit insgesamt drei verschiedene statische Rollen unterschieden: Ressourcenverwalter (für beschränkte oder anderweitig kritische SW-Ressourcen), Benutzeragenten (die als einzige in direkten Kontakt mit Benutzern treten) und Funktionsspezialisten (die einer zu erbringenden aufgabenspezifischen Teilfunktionalität zugeordnet sind und für die Kontrolle der dazu nötigen, z.T. hierarchisch delegierten Teilschritte verantwortlich sind.
  - Der 'Agenten-Server' (AS):
    Dieser ist im Grunde ein ausgezeichneter Verwalter, nämlich für die Agenten einer Agentengemeinschaft. Es darf pro Agentengemeinschaft nur ein einzige Instanz dieser Rolle existieren. Der AS ist der erste Agent einer Gemeinschaft, der erzeugt werden muß, bevor irgend ein anderer Agent aktiviert werden kann.
  - Delegation und Agentenerzeugung:
    Wie bereits angesprochen, können Agenten Teilaufgaben, die sie nicht selbst erledigen können per Auftragsnachricht an Spezialisten delegieren. Existiert der benötigte Spezialist noch nicht, oder wird eine zusätzliche Spezialisteninstanz benötigt, so kann ein Spezialist von dem Auftraggeber über den AS erzeugt werden.

Die Mikroeigenschaften werden v.a. durch folgende Charakteristika abgedeckt:

- Kommunikationsfähigkeit von Agenten zu seiner Umwelt: Hierzu müssen Agenten in der Lage sein, über Standardprotokolle mit externen Softwareeinheiten zu kommunizieren. Für die Agenten im Bereich Informationsaustausch erfolgt diese Kommunikation auf Basis des http-Protokolls, da hauptsächlich Informationsdienste betrachtet werden, die einen WWW-Schnittstelle besitzen.

- Wissensrepräsentation:
  Da innerhalb der zu lösenden Teilaufgaben kaum echte, komplexe Planungs-Aufgaben (z.B. Zeit-, Orts-, Reihenfolgeplanungen, Prognose- oder Diagnoseaufgaben) enthalten sind, sondern vielmehr situationsgerechtes Reagieren nötig ist, hat das Verhalten der beteiligten Agenten eher reaktiven Charakter.
  - Dauerhaft gespeichertes Faktenwissen wird im Agentenmodell dieser Arbeit in Form von strukturierten Dokumenten abgespeichert. Die Strukturierung erfolgt gemäß des SGML-Standards.
  - Dauerhaftes Verhaltenswissen wird in Form von einfachen Event-Aktions, bzw. Nachrichten-Aktions-Regeln im Agenten selbst repräsentiert. Dabei spielen zwei wesentliche reaktive Verhaltenselemente eine Rolle: Event-Behandlung und Reaktion auf die Nachrichten anderer Agenten.

3.3 Agenten-Spezialisten für Unterstützung von Informationsaustausch

Aus dem Unterstützungsmodell, das sich auf die Workflows des Aufgabenmodelles stützt, ergeben sich für jede der drei statischen Agentenrollen verschiedene Spezialisten, die jeweils Teilworkflows des Unterstützungsmodelles automatisieren bzw. koordinieren.
Beispiele für diese Spezialisten sind folgende Vertreter der 3 statischen Rollen:

Der *Anfragedialog-Benutzeragent* führt mit Benutzer auf der Basis dynamischer Formulare den Dialog zur Anfragespezifikation bzw. Ergebnispräsentation und generiert hierarchische Beschreibung von Anfrageaufträgen.

Der *Registrierungsdialog-Benutzeragent* führt mit Anbietern von Informationswerkzeugen Dialog zur erstmaligen Registrierung dieser Werkzeuge, sowie zur Modifikation der Registrierungsbeschreibung und veranlaßt Aktualisierung bzw. Generierung von SGML-Registrierungsdokumenten für das betroffene Werkzeug.

Der *Agenten-Verwalter (Agent-Manager)* verwaltet Information über die Agenten einer Agentengesellschaft und erzeugt bzw. terminiert Agenten.

Der *Registrierungsdokument-Verwalter* ist verantwortlich für SGML-Dokumente, in welchen Registrierungsinformation gespeichert ist und die für Anfrage-Formatierung bzw. Ergebnis-Bearbeitung, sowie für die Auswahl von geeigneten externen Werkzeugen für Anfrage-Delegation notwendiges Faktenwissen enthalten.

Der *Domänen-Verwalter* ist verantwortlich für das Bereitstellen von Faktenwissen, das die Zuordnung von Anfragedomänen und vermittlerinternen Attributen beschreibt und für den Anfragedialog benötigt wird. Zudem ist er für die Aktualisierung dieser Zuordnungen gemäß der vorliegenden Registrierungen zuständig.

Der *Anfrage-Verwalter* verwaltet bearbeitete, beim Vermittler für spätere Abholung über einen Benutzeragenten hinterlegte Anfragen und deren Ergebnisse.

Der *Anfrage-Scheduler* verwaltet unbearbeitete Anfrageliste mit zurückgestellten oder zu wiederholenden Anfragen und initiiert deren Ausführung.

Der *Anfrage-Akzeptor* empfängt vom Anfragedialog-Benutzeragenten eine Anfrage, delegiert in Abhängigkeit der Nutzungsbedingungen die Anfrage an den Anfrage-Aufbereiter bzw. übergibt sie dem Anfrage-Scheduler.

Der *Anfrage-Aufbereiter* identifiziert alle an externe Werkzeuge delegierbaren Anfragebestandteile, koordiniert die Delegation der Anfragebestandteile und veranlaßt jeweils die Erzeugung des für die weitere Bearbeitung zuständigen Spezialisten. Die Beschreibung von komplexen Anfragen inkl. ihrer Nutzungsbedingungen ist hierarchisch. Aus dieser Beschreibungshierarchie ergibt sich die Strategie zur jeweiligen Aufgabenzerlegung und damit zur Delegation von vorverarbeiteten Anfragen an geeignete Funktionsspezialisten oder Verwalter. Die Kooperationsbeziehungen innerhalb der Agentengesellschaft beschränken sich jedoch nicht auf Kooperation mit hierarchisch vertikal aufeinanderfolgenden Agenten (wie z.B. die Kooperation dynamischer Aufgabenspezialisten mit Verwaltern zeigt).

4 Zusammenfassung

Diese Arbeit untersucht die Einsatzmöglichkeiten für Agenten zur Unterstützung von Informationsaustausch in globalen Informationsräumen. Dabei wurde Wert darauf gelegt, nicht ein weiteres abgeschlossenes Werkzeug zur isolierten Bearbeitung von einzelnen Teilaufgaben oder zur Erprobung einzelner Agenteneigenschaften zu entwerfen, sondern basierend auf einer Analyse der Problemstellungen und Anforderungen dieses Aufgabenbereiches zu untersuchen, auf welche Weise Agententechnologie hier zur Unterstützungsentwicklung nutzbringend einsetzbar ist und welches gezielte schrittweise Modellierungsvorgehen zur Entwicklung der agentenbasierten Unterstützung angewendet werden sollte, um die angestrebten Anforderungen zu erfüllen.

4.1 Weiteres Vorgehen

Die weitere Arbeit wird sich mit einer Vervollständigung der Formate für die strukturierte Beschreibung des permanenten, aufgabenspezifischen Faktenwissens und Spezialistenfunktionalitäten (u.a. im Hinblick auf die Bearbeitung von Ergebnissen delegierter Anfragebestandteile) beschäftigen. Ergänzend werden Prototypen für die vorgeschlagenen Agentenschablonen, sowie Prototypen-Beispiele für deren Ergänzung zu aufgabenspezifischen Spezialisten für ausgewählte Aufgabenbeispiele erstellt.

4.2 Zusammenarbeit

Die für dieses Teilprojekt relevante kolleginterne Zusammenarbeit bezieht sich v.a. auf die Themenbereiche 'Kooperation und Organisationsstrukturen von Agenten', 'Vermittlung in offenen Architekturen', 'Spezifikation', sowie auf allgemeinere agentenbezogene Fragen, die innerhalb des Arbeitskreises MAD diskutiert wurden.

4.3 Externe Forschungsaufenthalte

Außerhalb des Graduiertenkolleg bestand Zusammenarbeit mit dem Rank Xerox Research Center RXRC Grenoble, Abt. Cooperation Technologies. Das RXRC-Projekt Constraint Based Knowledge Brokers beschäftigt sich mit der Entwicklung von leistungsfähigen Methoden für effizientes Information Retrieval und Kombination von Wissen. Dabei wird ein Ansatz verfolgt, der zur Anfragerepräsentation Feature Constraints [ABPS95] einsetzt. Die Zerlegung und Rekombination von Constraints für zusammengesetzte Anfragen erfolgt mittels hierarchischer Broker und Constraint Solving Techniken.

Im Rahmen eines viermonatigen Aufenthaltes bei RXRC konnte ich Erfahrungen aus dem Bereich des agentenbasierten Informationsaustausches bei der Weiterentwicklung des constraint-basierten Brokersystems einbringen [BPK+96]. Andererseits lieferte meine Tätigkeit in diesem Projekt für diese Arbeit wertvolle Erfahrungen über die Auswirkungen verschiedener Agenteneigenschaften und constraint-basierter Wissensrepräsentation bei der Realisierung von Brokersystemen.

Literatur

[ABPS95] J.-M. Andreoli, U. Borghoff, R. Pareschi und J. Schlichter. Constraint Agents for the Information Age. *J. Universal Computer Science*, 12, 1995.

[BG89] J. Burch und G. Grudnitski. *Information Systems*. John Wiley, 1989.

[BPK+96] Uwe M. Borghoff, Remo Pareschi, Harald Karch, Martina Nöhmeier und Johann Schlichter. Constraint-based Information Gathering for a Network Publication System. In *Proceedings of 2nd Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96)*, S. 45–59, April 1996.

[Dou96] Paul Dourish. *Open implementation and flexibility in CSCW toolkits*. Dissertation, University College London, 1996.

[DWS96] K. Decker, M. Williamson und K. Sycara. Modeling information aagents. Advertisements, organizational roles and dynamic behaviour. In *Proceedings of the AAAI-96 Workshop on Agent Modeling*, 1996.

[EW95] O. Etzioni und D. Weld. Intelligent Agents on the Internet: Fact, Fiction and Forecast. *IEEE Expert*, August 1995.

[FIP97] FIPA. Agent Communication Language. Technical Report FIPA'97 Specification Part 2, Revision 0.1, FIPA - Foundation for Intelligent Physical Agents, Januar 1997.

[GW94] C. Guilfoyle und E. Warner. Intelligent Agents: The New Revolution in Software. Technical report, OVUM, 1994.

[Hau94] Hans Haugeneder. IMAGINE Final Project Report. Technical Report Project 5362, ESPRIT, 1994.

[JPD+94] M. Jarke, K. Pohl, R. Dömges, St. Jacobs und H.W. Nissen. Requirements Information Management: The NATURE Approach. Technical Report 94-24, Aachener Informatik-Berichte, 1994.

[Mül96] H.J. Müller. Multi-Agent Systems Engineering. Technical report, Universität Bremen, AG KI, 1996.

[Nöh96] M. Nöhmeier. Agenteneinsatz in globalen Informationsräumen. In P.P. Spies, Hrsg., *Graduiertenkolleg: Kooperation und Ressourcenmanagement in verteilten Systemen (Zwischenbericht zum Frühjahr 1996)*, S. 22–26. Institut für Informatik – Technische Universität München, 1996.

[Sea80] J. Searle. *Speech act theory and pragmatics*. Reidel-Verlag, 1980.

[WP96] K. Webster und K. Paul. Beyond Surfing: Tools and Techniques for Searching the Web. *Information Technology*, Januar 1996.

1.3.7 Computational Economies, Wahrscheinlichkeitsdichten und Neuronale Netze zur Entscheidungsmodellierung in Multiagentensystemen

Teilprojekt: Schätzung von Wahscheinlichkeiten mit neuronalen Netzen
Dirk Ormoneit

1 Einführung

Von zentraler Bedeutung für die Entwicklung von Multiagentensystemen (MAS) ist die Modellierung sowohl des Entscheidungsverhaltens der einzelnen Agenten als auch ihres Kooperationsverhaltens in einer gemeinsamen Umwelt. Beide Problemstellungen lassen sich nicht zufriedenstellend mit Methoden der klassischen Künstlichen Intelligenz (KI) lösen. Im Hinblick auf den einzelnen Agenten gilt es, Konzepte wie etwa Autonomie, Lernfähigkeit sowie eine große Transparenz für den Benutzer zu realisieren. Eine besondere Bedeutung entfällt ebenfalls auf die Verarbeitung von unsicherem und vagem Wissen. Beim Zusammenwirken von mehreren Agenten innerhalb eines MAS ergeben sich Probleme aufgrund des Fehlens einer gemeinsamen, globalen Zielfunktion, sowie bei der Koordination der möglicherweise gegenläufigen Interessen einzelner Agenten.

Entsprechend den zwei genannten Problemstellungen ist diese Arbeit in zwei Teile untergliedert. Teil eins befasst sich mit der Verwendung von neuronalen Netzen zur Schätzung von Wahrscheinlichkeitsdichten. Wahschscheinlichkeitsmaße spielen eine zentrale Rolle bei der Beschreibung der Präferenzen eines einzelnen Agenten in einer unsicheren Umwelt. In Teil zwei untersuchen wir verschiedene entscheidungstheoretische Konzepte aus der Volkswirtschaftslehre in Hinblick auf ihre Anwendbarkeit in MAS.

2 Neuronale Netze zur Schätzung von Wahrscheinlichkeitsdichten

Ein entscheidungstheoretisches Modell für einen einzelnen Agenten besteht im wesentlichen aus zwei Komponenten: einer Nutzenfunktion, die allen möglichen Zuständen der Welt einen Nutzenwert zuordnet und einem (subjektiven) Wahrscheinlichkeitsmaß, dessen Modellierung Inhalt dieses Kapitels ist. Sind beide Komponenten verfügbar, so können sie mit Hilfe eines geeigneten Operators kombiniert und somit unsichere Alternativen miteinander verglichen werden.

Ein geeignetes Konzept zum Erlernen von Wahrscheinlichkeitsverteilungen aus Beispielen läßt sich mit Hilfen von künstlichen neuronalen Netzen (NN) ableiten. Wir diskutieren im folgenden entsprechende Netzstrukturen und Lernalgorithmen. Entsprechend Entscheidungssituationen mit und ohne Verfügbarkeit zusätzlicher Information unterscheiden wir hierbei die Schätzung von bedingten und unbedingten (= gemeinsamen) Wahrscheinlichkeitsdichten. Der folgende Abschnitt behandelt zunächst die Modellierung unbedingter Dichten mit Hilfe von sogenannten Gaussian-Mixture-Netzwerken.

2.1 Gaussian-Mixture-Netzwerke

Der Begriff "Gaussian Mixture" (GM) bezeichnet eine lineare Kombination von multivariaten Normalverteilungen (=Gaussians). Sofern bei der Wahl der Mischgewichtungen gewisse

Restriktionen beachtet werden, repräsentiert die entstehende Funktion selbst wieder eine Wahrscheinlichkeitsdichte. Man kann einsehen, daß das entsprechende Wahrscheinlichkeitsmodell in der Lage ist, eine große Klasse von praktisch relevanten Verteilungen zu approximieren. Ein bedeutender Vorteil von Gaussian Mixtures ist, daß sie sich auf natürliche Weise in eine Repräsentation als neuronales Netz überführen lassen ([Now91], [Orm93]). Entsprechend lassen sich zahlreiche Methoden der Modelloptimierung, insbesondere der populäre Backpropagationalgorithmus, problemlos auf das Modell anwenden.

Abbildung 1: *Wahre Dichte (links) und unregularisierte Dichteschätzung (rechts).*

Eine zweidimensionale Dichte und eine entsprechende Gaussian-Mixture-Dichteschätzung sind in Abbildung 1 dargestellt. Wie man sieht, ergibt sich ein Problem durch die geradezu dramatische Tendenz von Gaussian Mixtures zum "Overfitting". Overfitting bezeichnet das "Auswendiglernen" einzelner Datenpunkte der Trainingsmenge bei gleichzeitigem Verlust an Generalisierungsfähigkeit. Die Behebung derartiger Schwierigkeiten wird im Zusammenhang mit neuronalen Netzen als Regularisierung bezeichnet. Es gibt verschiedene gebräuchliche Verfahren, von denen wir drei auf das Dichteschätzungsproblem angewendet haben.

- **Bayessche Regularisierung**

  In einem Bayesschen Kontext werden gewissen Zuständen des Modells a priori (d.h. vor Eintreffen der Daten) höhere Wahrscheinlichkeiten zugeordnet als anderen. Mathematisch gesehen definiert man eine Prior-Wahrscheinlichkeitsverteilung auf dem Parameterraum des Modells. Anstelle der Likelihood der Daten betrachtet man dann die Posterior-Dichte der Parameter, welche sich mittels des Bayes-Theorems aus Prior und Likelihood ergibt. Das Problem ist, daß man darüberhinaus zur Anwendung bestimmter Algorithmen (EM Algorithmus, Gibbs-Sampling) den Posterior in einer analytisch geschlossenen Form zur Verfügung stellen muß. Dies läßt sich garantieren durch die Wahl des Priors aus der sogenannten "Conjugate Distribution Family" der Likelihood.

- **Bayessche Integration**

  Eine völlig konsequente Anwendung der Bayesschen Theorie erfordert, daß man nicht nur den Posterior maximiert, sondern außerdem mittels Integration die Modellparameter zur Berechnung der "Predictive Distribution" eliminiert. Das Integral kann stochastisch approximiert werden. Hiezu ist es allerdings nötig, vom Posterior der Parameter Samples zu generieren. Dies wiederum ist bei Verwendung von Conjugate Priors möglich durch Anwendung einer Variante des Gibbs-Sampling-Verfahrens ("Data-Augmentation").

- **Averaging Methoden**

  Eine weitere Möglichkeit der Regularisierung besteht in der Kombination mehrerer, unabhängig voneinander trainierter Netzwerke ([PC93], [Bre94]). Die einzelnen Lösungen lassen sich interpretieren als verschiedene lokale Minima der Fehlerfunktion, die vom Lernalgorithmus gefunden wurden. Es läßt sich zeigen, daß ein kombinierter Schätzer im Mittel zu einem kleineren Generalisierungsfehler führt als die einzelnen Prediktoren, sofern diese eine hinreichende Vielfältigkeit in ihren Vorhersagen aufweisen.

Experimente mit den drei vorgeschlagenen Regularisierungsmethoden zeigen ein uneinheitliches Ergebnis. Während für relativ kleine Eingangsdimensionen die Bayessche Regularisierung die besten Ergebnisse liefert, wird der Vorteil von Averagingmethoden erst bei höheren Dimensionen deutlich. Interessant ist ebenfalls, daß in unseren Experimenten ein vollständiger Bayeseanischer Ansatz kaum zu Verbesserungen gegenüber der Posterior-Maximierung führte. Alle vorgeschlagenen Regularisierungsmethoden verbessern das Ergebnis jedoch statistisch signifikant gegenüber der unregularisierten Dichteschätzung.

2.2 Bedingte Dichten

Zur Charakterisierung von Wahrscheinlichkeiten, deren Ausprägung durch zusätzliche Informationen (dem Wert zusätzlicher Variablen) dominiert wird, verwenden wir das Konzept der bedingten Wahrscheinlichkeitsdichte.[6] Bedingte Dichten lassen sich mit dem sogenannten Conditional Density Estimation Network (CDEN, [NHFO94]) approximieren, dessen schematischer Aufbau in Abbildung 2 dargestellt ist.

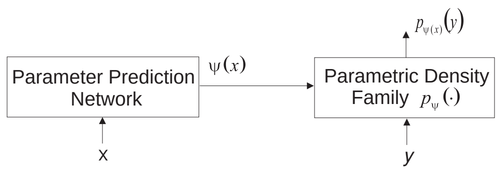


Abbildung 2: *Conditional Density Estimation Network (CDEN)*

Die grundlegende Idee ist, daß sich eine bedingte Dichte $p(y|x)$ durch eine parametrische Familie $p_\psi(y)$ darstellen läßt, wobei der Wert des Parametervektors $\psi = \psi(x)$ eine (möglicherweise nichtlineare) Funktion der unabhängigen Zufallsvariable x ist. Sofern man sowohl die parametrische Dichte $p_\psi(y)$ als auch $\psi(x)$ als neuronale Netze realisiert, erhält man das dargestellte Schema von zwei Netzen, in dem der Ausgang des ersten die Gewichte des zweiten determiniert. Diese Darstellung legt außerdem eine Variante des Backpropagation-Algorithmus sowie anderer NN-Lernalgorithmen zur Optimierung nahe.

Interessanterweise ergeben sich bei entsprechender Wahl von $p_\psi(\cdot)$ verschiedene bekannte Modelle als Spezialfälle. Verwendet man zum Beispiel Gaussian Mixtures, so erhält

[6] Im Prinzip läßt sich jede bedingte Dichte aus der gemeinsamen Dichte aller beteiligten Variablen herleiten. Ein derartiges Vorgehen ist jedoch in der Praxis nicht empfehlenswert. Der Grund hierfür ist, daß die gemeinsame Dichte ungleich mehr Information enthält als jede bedingte, so daß die Schätzung der gemeinsamen Dichte ein wesentlich schwierigeres Problem darstellt.

man das populäre "Mixture of Experts" Modell ([JJNH91]). Mit einer Normalverteilung mit rekurrentem (linearen) $\psi(x)$ ergibt sich eine Architektur zur Identifikation von ARCH/GARCH Prozessen. Diese bereits existierenden Dichtemodelle sind allerdings in Hinblick auf die in der realen Welt beobachteten Verteilungen häufig nicht anwendbar. So stellten wir zum Beispiel bei der Betrachtetung bestimmter Finanzzeitreihen fest, daß die dort vorhandenen Verteilungsmomente sich wesentlich von denen einer Normalverteilung unterscheiden. Um ein entsprechendes Dichtemodell im Rahmen des CDEN zu entwerfen, untersuchten wir die fogenden drei parametrische Dichten als mögliche Kandidaten für $p_\psi(y)$:

2.2.1 Maximum Entropy (ME) Density

Die Idee der ME-Dichte ist es, für gegebene Momente der Verteilung diejenige Dichte mit der maximalen Shannon-Entropie auszuwählen. Man wählt somit aus der Klasse von möglichen Dichten diejenige aus, die am wenigsten Information über die abhängige Zufallsvariable enthält, ihr Verhalten also am wenigsten genau spezifiziert. Ein weiterer Vorteil dieses Verfahrens ist, daß die Parameter von $\psi(x)$ nun mit der verallgemeinerten Momentenmethode (GMM) anstatt des sonst gebäuchlichen Maximum-Likelihood-Verfahrens geschätzt werden können, was sich als deutlich einfacher herausstellt. Figur 3 stellt die Entropie der ME-Dichte als Funktion des dritten und vierten Momentes dar. Ein hartnäckiges Problem ist die korrekte Parametrisierung der ME Dichte. In unserer momentanen Implementierung verwenden wir hierbei eine Erweiterung des Verfahrens von Zellner und Highfield ([ZH88]).

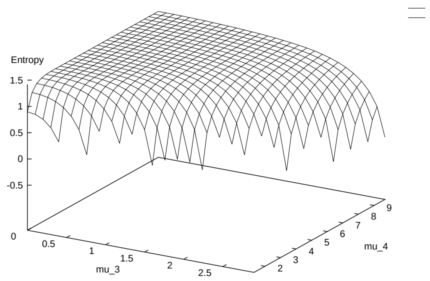


Abbildung 3: *Entropie der ME-Dichte als Funktion des dritten und vierten zentralen Momentes.*

2.2.2 Truncated Gram-Charlier (GC) Expansion

Um die numerischen Schwierigkeiten der ME-Dichte zu umgehen bietet sich die GC-Dichte an. Ausgangspunkt ist hierbei die sogenannte Gram-Charlier Expansion, mit deren Hilfe sich jede Wahrscheinlichkeitsdichte als unendliche Reihe um die Standardnormalverteilung darstellen läßt. Beschneidet man diese unendliche Reihe nach dem vierten Glied, so ergibt sich eine Wahrscheinlichkeitsdichte, die genau durch ihre ersten vier Momente parametrisiert wird. Das Problem hierbei ist, daß die beschnittene Funktion nicht mehr für alle theoretisch möglichen Momentenkombinationen positiv ist, was sich vor allem für die Maximum-Likelihood Schätzung als äußerst problematisch erweist. Eine Lösung hierfür ist eine geeignete Reparametrisierung der Dichte, die nur positive Funktionswerte zuläßt. Ein weiteres Problem ist, daß die Funktion in der von uns gewählten Parametrisierung nicht robust gegen lineare Transformationen ist. Figur 4 stellt diesen Sachverhalt anhalt der darstellbaren $\mu_3 - \mu_4$-Kombinationen für verschiedene Translationen dar. Wie man sieht, wird die zulässige Fläche bei einer Abweichung von $\mu_1 = 0, \mu_2 = 1$ immer kleiner und verschwindet schließlich ganz. Für den Einsatz der Dichte im CDEN ist dieser Umstand akzeptabel, wenn die bedingten Dichten sich nicht zu stark von der unbedingten Dichte unterscheiden und die Daten vor dem Training in angemessener Weise normiert werden.

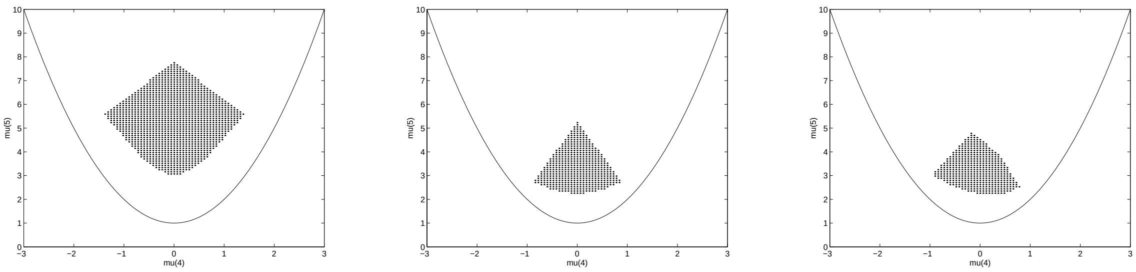


Abbildung 4: *Region der μ_3-μ_4-Kombinationen, die sich für veschiedene Werte von ν_1, ν_2 ergeben. Links: $\mu_1 = 0, \mu_2 = 1$, Mitte: $\mu_1 = 0, \mu_2 = 2$, Rechts: $\mu_1 = 0.2, \mu_2 = 2$. Durchgezogene Linie: theoretische Grenze für beliebige Dichten.*

2.2.3 Gaussian Basis Functions (GBF)

Ein völlig andersartiges Verfahren ergibt sich unter der Verwendung sogenannter Basisfunktionen. Basisfunktionen haben den Vorteil, daß sie sich durch ein besonders unproblematisches Trainingsverhalten auszeichnen. Der Nachteil ist eine zu große Flexibilität der sich ergebenen Funktion, was in dem bereits zuvor erwähnten "Overfitting" resultiert. Wir umgehen dieses Problem wiederum durch Verwendung einer Bayesschen Prior-Verteilung. Diese ist so konstruiert, daß Dichten mit hoher Shannon-Entropie, also, wie bereits ober erwähnt, niedriger Komplexität, bevorzugt ausgewählt werden. Figur 5 stellt einen Ausschnitt der in unserer Anwendung verwendeten Gaussschen Basisfunktionen dar.

Der Kernpunkt ist in allen Fällen die effiziente Berechnung der Ableitungen zur Anwendung des Backpropagation-Algorithmus. Entsprechende Verfahren wurden von uns detailliert untersucht. Bei der Anwendung auf die oben erwähnten Finanzdaten ist die Performanz aller vorgestellen Methoden deutlich besser als die der ARCH/GARCH oder Mixture of Experts Modelle.

3 Computational Economies und MAS

Während zahlreiche entscheidungstheoretische Konzepte bezüglich des Verhaltens eines einzelnen Agenten schon seit langem Einzug in die künstliche Intelligenz gehalten haben,

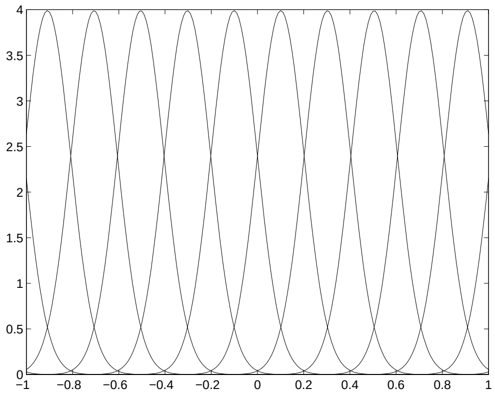


Abbildung 5: *Ausschnitt aus den Basisfunktionen der BF-Dichte.*

sind im Zusammenhang mit MAS vor allem Theorien über das Zusammenwirken von Agenten in einer gemeinsamen Umwelt von Interesse. Es existieren hierzu (mindestens) drei relevante Theorien aus den Sozialwissenschaften, die sich in Hinblick auf die ihnen zugrundeliegenden Annahmen, vor allem aber in der Verfügbarkeit von verteilten Algorithmen zur Berechnung von Gleichgewichtszuständen unterscheiden:

- **Partial Equilibrum Theory (PET)**

  PET beschäftigt sich mit dem Markt für ein einzelnes Gut in einer Ökonomie aus Konsumenten und Firmen unter der Annahme, daß die Preise und Umsätze in allen anderen Gütern von Veränderungen in diesem Markt unberührt bleiben. Diese Annahmen ermöglichen eine besonders einfache Analyse der Gleichgewichtszustände des Systems, sind aber für die Anwendung in MAS zu stark. Der Grund hierfür ist, daß in MAS praktisch immer komplementäre Güter existieren (z.B. CPU und Speicher) und demzufolge die Verfügbarkeit eines dieser Güter für einen Agenten mit Sicherheit seine Nachfrage nach dem Komplement beeinflußt.

- **General Equilibrum Theory (GET)**

  GET ist eine Verallgemeinerung der PET, in der auch Abhängigkeiten zwischen den Aktivitäten in Märkten für verschiedene Güter berücksichtigt werden. GET ist vor allem zur Formulierung von Allokationsproblemen in MAS geeignet. Konsumenten und Firmen (= Agenten) tauschen Güter (= Ressourcen) miteinander aus, bis ein Systemzustand erreicht ist, in dem alle Ressourcen effizient verwendet werden und keiner der Agenten eine Verbesserung seiner Situation durchsetzen kann (= Pareto-Optimalität im Competitive Equilibrum, siehe erstes Wohlfahrtstheorem). Der effiziente Austausch von Gütern wird determiniert durch einen Preisvektor. Die Suche nach diesem ist äquivalent zur Suche nach optimalen Allokationen mit Methoden der klassischen KI (und demzufolge im Allgemeinen auch genauso schwierig). Es existieren im Rahmen der GET verschiedene Preisfindungsmechanismen (z.B. Auktionen, Verhandlungen), die sich in Hinblick auf die zugrundeliegenden Annahmen und die Verteiltheit der durch sie implizierten Algorithmen unterscheiden. Ein besonders gut für eine verteilte Implementierung geeignetes Beispiel ist die Tâtonment Dynamik, mit deren Hilfe Wellman die optimale Allokation in einem Multicommodity-Flow-Problem berechnet ([Wel93]).

- **Game Theory (GT, Spieltheorie)**

  GT ist die allgemeinste der vorgestellten Theorien und beschäftigt sich mit Situationen, in denen Spieler (= Agenten) ihr eigenes Verhalten unter Berücksichtigung möglicher Verhaltensweisen der Gegenspieler optimieren. Offensichtlich umfaßt diese Definition praktisch alle in einem MAS denkbaren Situationen. Der Nachteil dieser Sichtweise liegt darin, daß ein Spieler nun nicht nur Wissen über seine eigene Nutzenfunktion, sondern auch über die aller Mitspieler besitzen und in seiner Entscheidung berücksichtigen muß. In der GET ist dies nicht der Fall, da mit Hilfe des Preismechanismus eine Art Seperierung der einzelnen Optimierungsprobleme erfolgt (für die Entscheidung eines Agenten ist bei gegebener eigener Nutzenfunktion und gegebenem Preisvektor die Nutzenfunktion der Mitspieler ohne Bedeutung). Die Situation in Spielen wird zusätzlich dadurch kompliziert, daß gewöhnlich das Wissen über die Präferenzen der Mitspieler nur unsicher ist und sich im Laufe der Zeit verändert, so daß Bayesche Konzepte zur Problemformalisierung verwendet werden müssen. Selbst nach erfolgreicher formalen Beschreibung des Problems sind Mechanismen zur Berechnung von Equilibra (= Lösungen) wesentlich komplizierter.

Aus der bisherigen Diskussion sollte klar geworden sein, daß sich die GET gut zur Modellierung von Allokationsproblemen in MAS eignet. Die Hauptschwierigkeit besteht in der Wahl eines geeigneten Mechanismus zur Bestimmung der Gleichgewichtspreise. Auktionen entsprechen globalen Optimierungsmethoden der KI, sind aber sehr kommunikationsintensiv. Ebenfalls vielversprechend erscheint in diesem Zusammenhang das Konzept der sogenannten Double-Auction, in der Händler sowohl Angebots- als auch Nachfragepreise angeben ([Wil85, Wil92]). Es erfolgt dann ein zentrales Matching. Der Vorteil gegenüber gewöhnlichen Auktionen besteht darin, daß eine Konvergenz zum Gleichgewichtszustand, der bei Double Auctions meistens mit dem walrasianischen Gleichgewicht uebereinstimmt, auch in wesentlich komplexeren Situationen als bei gewöhnlichen Auktionen erfolgt. Eine Alternative für eine gewisse, eingeschränkte Problemklasse ist die Tâtonment-Dynamik, mit deren Hilfe der Kommunikationsaufwand wesentlich reduziert werden kann.

4 Zusammenfassung

Das Verhalten eines einzelnen Agenten läßt sich als ein mathematisches Entscheidungsproblem unter Unsicherheit formalisieren. Neuronale Netze bilden eine vielversprechende Grundlage zur Schätzung entsprechender Wahrscheinlichkeitsmaße. Probleme aufgrund des Mangels an Wohldefiniertheit des Dichteschätzungsproblems lassen sich mit Hilfe von klassischen statistischen Methoden beheben.

In Hinblick auf das Zusammenwirken mehrerer Agenten in einem MAS stellen die Sozialwissenschaften verschiedene Theorien bereit. Besonders gut zur Formalisierung von Allokationsproblemen ist die GET geeignet, da die Optimierungsprobleme der einzelnen Agenten weitgehend seperiert voneinander gelöst werden und verteilte Algorithmen zur Bereichnung der Gleichgewichtszustände existieren. Eine Behandlung ensprechender Probleme im Rahmen der Spieltheorie erfordert einen wesentlich komplexeren Formalismus.

5 Ausblick & Zusammenarbeit mit anderen Teilprojekten

In Hinblick auf Teil I der dargestellten Arbeit werden wir uns zunächst weiterhin auf die Anwendung von Maximum-Entropy-Dichten zur Schätzung von bedingten Wahrscheinlichkeiten konzentrieren. Wie bereits erwähnt, ergeben sich verschiedene numerische Probleme bei der Schätzung der Modellparameter, sowie bei der Realisierung von Constraints auf den bedingten Momenten.

Bei den Computational Economies gilt unser Interesse hauptsächlich dem Preisfindungsmechanismus im Rahmen der GET. Als eine Verbesserung der dargestellten Methoden ist es möglich, das Verhandlungsproblem in einer Ökonomie selbst wieder im Rahmen der Spieltheorie zu betrachten. Ein bedeutender Vorteil dieser Sichtweise liegt darin, daß spieltheoretische Strategien rationalisierbar sind, d.h. daß Agenten Strategien befolgen, weil es zu ihrem einenem Besten ist und nicht aufgrund von vorgegebenen Regeln, deren Einhaltung sich unter Umständer nicht kontrollieren läßt (z.B. Bericht der wahren Nachfrage in einer Auktion). Ziel ist also die Definition eines Verhandlungsprotokolles (= Spiel), welches Gleichgewichtsstrategien besitzt, die zum gewünschten Equilibrum der Ökonomie (optimale Lösung) führen.

In den anderen Teilprojekten des Graduiertenkollegs bieten sich zahlreiche Anwendungs- und Testmöglichkeiten der in dieser Arbeit vorgestellten Ideen. So wurden von F. Fuchs (Teilprojekt: „Verteilte Inferenzsysteme“) verschiedene Verhandlungsmechanismen zur Konfliktlösung untersucht und hierbei interessante Ergebnisse erzielt. Für weitere theoretische Untersuchungen sind einerseits die von B. Quendt (Teilprojekt: „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen“) und M. Nöhmeier (Teilprojekt: „Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit“) im Rahmen ihrer praktischen Anwendungen gewonnenen Erkenntnisse und andererseits die Ergebnisse von R. Mayr (Teilprojekt: „Verteilte Algorithmen – Spezifikation, Modellierung, Korrekheit“) bezüglich der Entscheidbarkeit und Komplexität der Verifikation verschiedener Modellklassen für nebenläufige Systeme sicherlich äußerts hilfreich. Besonders interessant in Hinblick auf eine praktische Erprobung der vorgestellten Marktmechanismen ist natürlich das Teilprojekt von M. Backschat (Teilprojekt: „Verteilte numerische Algorithmen auf Bäumen“), in dem versucht wird, besagte Methoden zur Steuerung der Lastverteilung in Rechnernetzen anzuwenden. Die Erfahrungen aus dieser Anwendung sind ein wichtiger Bestandteil zur Beurteilung der praktischen Anwendbarkeit dieser Methodik.

Literatur

[Bre94] L. Breiman. Bagging Predictors. Technical report, UC Berkeley, 1994.

[JJNH91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan und G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation 3*, 1991.

[NHFO94] R. Neuneier, F. Hergert, W. Finnoff und D. Ormoneit. Estimation of Conditional Densities: A comparison of Neural Network Approaches. *Proceedings of the International Conference on Artificial Neural Networks*, 1:689–692, 1994.

[Now91] S. J. Nowlan. *Soft Competitive Adaption: Neural Network Learning Algorithms based on Fitting Statistical Mixtures.* Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[Orm93] D. Ormoneit. Estimation of Probability Densities using Neural Networks. Diplomarbeit, Technische Universität München, 1993.

[PC93] M. P. Perrone und L. N. Cooper. When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. In *Neural Networks for Speech and Image Processing.* Chapman Hall, 1993.

[Wel93] M.P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1, 1993.

[Wil85] R. Wilson. Incentive Efficiency of Double Auctions. *Econometrica*, 53(5):1101–1115, 1985.

[Wil92] R. Wilson. *Strategic Analysis of Auctions*, Bd. 1, Kapitel 8, S. 227–279. 1992.

[ZH88] A. Zellner und R.A. Highfield. Calculation of Maximum Entropy Distributions and Approximation of Marginal Posterior Distributions. *Journal of Econometrics*, 37:195–209, 1988.

1.3.8 Ein agentenbasiertes Signalisierungssystem für den offenen Telekommunikationsmarkt

Teilprojekt: Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen
Bernhard Quendt

Zusammenfassung

Die derzeitigen Deregulierungsbestrebungen lassen deutlich die beginnende Umgestaltung des Telekommunikationsmarktes (TK-Markt) nach marktwirtschaftlichen Gesichtspunkten erkennen. Einen plausiblen Zielzustand bildet der vorgestellte offene Telekommunikationsmarkt. Infolge seiner Ähnlichkeit zu herkömmlichen Märkten erscheint er als natürlicher Ansatz für die zukünftige Strukur des TK-Marktes. Auf dem offenen TK-Markt beteiligen sich mehrere Netzbetreiber mit ihren Ressourcen an der Bereitstellung eines Kommunikationsdienstes, erfolgt die Tarifierung der Ressourcen nachfrageorientiert und besitzt der Kommunikationsteilnehmer die Möglichkeit, die Auswahl der Netzbetreiber anhand seiner Preis- und Qualitätspräferenzen zu beeinflussen.
Der Kommunikationsteilnehmer zieht den größtmöglichen Nutzen aus den Eigenschaften des offenen TK-Marktes, wenn er bei der Netzbetreiberauswahl durch das Signalisierungssystem unterstützt wird. Ausgehend von einer modernen Signalisierungsarchitektur [RAC93] mit getrennter Ruf-, Ressourcen- und Verbindungssteuerung, wird eine erweiterte, agentenbasierte Ressourcensteuerung als Grundlage eines agentenbasierten Signalisierungssystems vorgestellt, welches obige Unterstützung ermöglicht. Die erweiterte Funktionalität der Ressourcensteuerung umfaßt dabei zum einen die Sichtung der Angebote, die die Netzbetreiber für ihre Ressourcen abgeben und zum anderen die Auswahl geeigneter Ressourcen im Sinne der Teilnehmerpräferenzen.
Die Funktionsweise der agentenbasierten Ressourcensteuerung wird abschließend anhand eines exemplarischen Rufaufbaus verdeutlicht.

1 Einführung: Der offene Telekommunikationsmarkt

Derzeitige Deregulierungsbestrebungen haben das Ziel, den seither monopolistisch geprägten Markt für Telekommunikationsdienste (TK-Dienste) gemäß den Prinzipien einer freien Marktwirtschaft umzugestalten. Obwohl der laufende Liberalisierungsprozeß noch nicht beendet ist, mehren sich bereits die Anzeichen für weitere Veränderungen in naher Zukunft [Soc95].
Die Komplexität bei der Bereitstellung von TK-Diensten steigt fortwährend:

- Multi-Media-Technik sorgt dafür, daß ein TK-Dienst nicht mehr nur eine einfache Verbindung zwischen zwei Kommunikationspartnern darstellt, sondern aus einer Vielzahl von Verbindungen mit unterschiedlichen und gegenseitig abhängigen Qualitätseigenschaften besteht.

- Konferenzdienste und die Notwendigkeit, heterogene TK-Endgeräte verbinden zu können, bedingen den Einsatz sogenannter Spezialressourcen [RAC93], z.B. Konferenzbrücken und Konverter.

Beides bietet neuen Netzbetreibern die Chance, mit ihren Ressourcen, die sie zu interessanten Preisen und Qualitätswerten anbieten, in den Wettbewerb zu treten.
Auf dem entstehenden deregulierten TK-Markt werden die für einen TK-Dienst benötigten Ressourcen nach wie vor weitgehend von einem Netzbetreiber bereitgestellt. Ein Betreiberübergang ist lediglich dann vorgesehen, wenn die Teilnehmeranschlüsse unterschiedlichen Betreibern zugeordnet sind. Der enstehende deregulierte TK-Markt ist daher ungeeignet, den freien Wettbewerb, der von anderen Märkten bekannt ist, zu ermöglichen. Folglich sind weitere Reorganisationsprozesse denkbar, die zu einem offenen TK-Markt führen. Die wesentlichen Kennzeichen des offenen TK-Marktes sind:

1. Beteiligung mehrerer Netzbetreiber mit ihren Ressourcen an der Bereitstellung eines TK-Dienstes.

2. Nachfrageorientierte Tarifierung der Ressourcen.

3. Freie Netzbetreiberauswahl durch den Dienstnutzer.

Die Folgen sind in Abbildung 1 anhand eines vereinfachten Konferenzrufes auf dem offenen TK-Markt dargestellt. Drei Netzbetreiber (NB) tragen mit ihren Ressourcen zur Bereit-

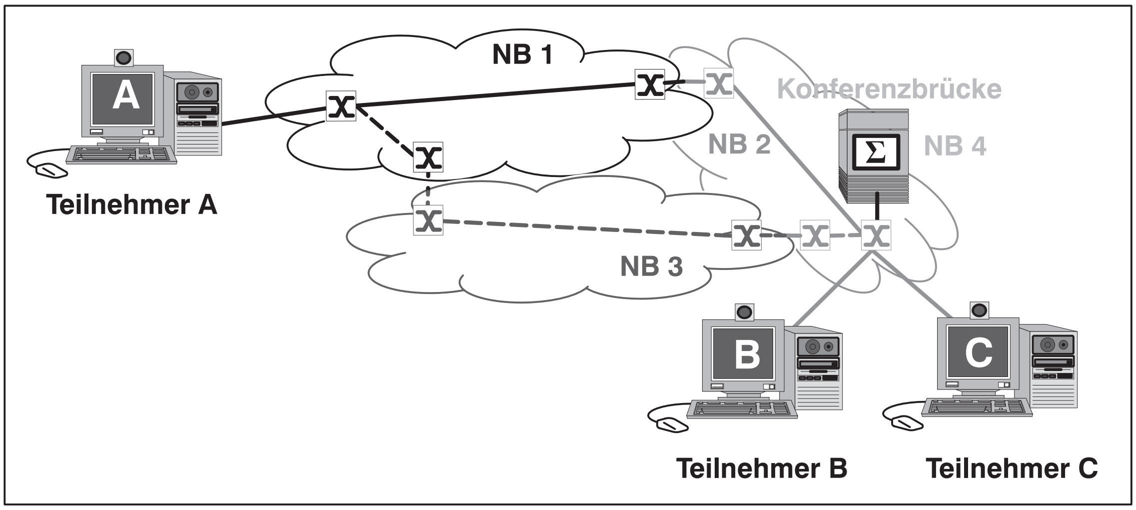


Abbildung 1: Ein Konferenzruf auf dem offenen TK-Markt.

stellung des Dienstes bei. Von NB 1 und NB 2 werden Übertragungswege genutzt, von NB 4[7] die Konferenzbrücke.
Auf dem offenen Auf dem offenen TK-Markt soll die Auswahl geeigneter Ressourcen im Ermessen des Dienstnutzers liegen, die Notwendigkeit langfristiger Verträge zwischen den einzelnen Netzbetreibern besteht daher nicht mehr.
Aus technischer Sicht besteht in der Integration von Ressourcen verschiedener Netzbetreiber keine prinzipielle Schwierigkeit, bei internationalen Verbindungen wird dies bereits

[7]Der Ressourcenbetreiber 4 wird im folgenden aus Gründen der Übersichtlichkeit ebenfalls als Netzbetreiber bezeichnet.

praktiziert [Sie93].
Das zweite Kennzeichen des offenen TK-Marktes betrifft die Tarifierung der Ressourcen. Eines der wesentlichen Prinzipien der freien Marktwirtschaft ist die Tatsache, daß der Preis von Angebot und Nachfrage abhängt. Auf dem offenen TK-Markt erfolgt die Ressourcentarifierung gemäß dieses Prinzips. W. Vickrey [Vic71] beschrieb als erster die positiven Folgen einer nachfrageorientierten Tarifierung bei TK-Diensten und anderen öffentlich bereitgestellten Diensten. Darauf aufbauend wurde in [Pou82] theoretisch nachgewiesen, daß dies sowohl wirtschaftlich sinnvoll für die Netzbetreiber, als auch profitabel für die Dienstnutzer ist.
Die gestrichelten Linien in Abbildung 1 beschreiben eine alternative Möglichkeit, den Konferenzruf durch Nutzung der Ressourcen von NB 3 einzurichten. Die Auswahl dieser Alternative wird von einer Vielzahl von Parametern abhängen, etwa vom Preis oder von Qualitätskriterien, wie Verzögerungszeiten, Jitter und Fehlerwahrscheinlichkeiten. Auf einem herkömmlichen Markt hat der Kunde die Freiheit, das passende Produkt aus den vorhandenen Alternativen auszuwählen. Ähnlich verhält es sich auf dem offenen TK-Markt. Dort besteht für den Dienstnutzer die Möglichkeit, den Weg durch das Netz mit zu bestimmen und auf diese Weise eigene Preis- und Qualitätsvorstellungen zu realisieren. Die dafür nötige Unterstützung durch das Signalisierungssystem wird im folgenden beschrieben.

2 Anforderungen und Ansätze für ein Signalisierungssystem

Die 'Produkte' des offenen TK-Marktes, die TK-Dienste, sind in ihrer Komplexität vergleichbar zu Produkten anderer Marktsegmente, z.B. Rechnersysteme oder Aktienpakete. Sie bestehen aus einer Vielzahl von Komponenten, die von verschiedenen Netzbetreibern zu unterschiedlichen Preisen und mit unterschiedlichen Qualitätsmerkmalen angeboten werden. Angenommen, ein Kunde möchte einen für seine Anwendung maßgeschneiderten Rechner erwerben oder ein Aktienpaket gemäß seinen finanziellen Mitteln und seiner Risikobereitschaft zusammenstellen. Einerseits werden Standardprodukte die Anforderungen des Kunden nur teilweise befriedigen, andererseits wird es dem Kunden an der Kenntnis aller Angebote und an der Expertise bei der Auswahl geeigneter Produktkomponenten fehlen. In diesem Fall wird der Kunde eine Art Vermittler beauftragen, etwa einen Fachhändler im Falle des Rechnersystems oder einen Makler im Falle des Aktienpakets.
Auf dem offenen TK-Markt ist die Existenz vergleichbarer Vermittlerdienste notwendig, um den Dienstnutzer bei der Auswahl geeigneter Ressourcen zu unterstützen. Sie werden im folgenden als Agentendienste bezeichnet, die zugehörige Vermittlereinheit als Agent. Da die Aufgabe des Signalisierungssystems in der Bereitstellung und Steuerung von TK-Diensten besteht, bilden die Agentendienste eine Basisfunktionalität des Signalisierungssystems für den offenen TK-Markt.
Um die Komplexität der Signalisierungsaufgaben für zukünftige TK-Dienste beherrschen zu können, wird in bereits bestehenden Forschungsaktivitäten eine Aufteilung in abgrenzbare Signalisierungsteilaufgaben vorgeschlagen [BMMM94, RAC93, Mul95]. Dieser Ansatz führt zu einer geschichteten Signalisierungsarchitektur. Im EU-Projekt RACE/MAGIC [RAC93] werden beispielsweise drei Steuerungebenen unterschieden: die Rufsteuerung, die Ressourcensteuerung und die Verbindungssteuerung. Ihre Aufgaben lassen sich am einfachsten anhand eines exemplarischen Rufaufbaus erklären.

Der Initiator eines TK-Dienstes beauftragt die Rufsteuerung mit dem gewünschten Ruf (z.B obige Konferenzschaltung). Diese bestimmt nun die für diesen Ruf geeignete Ressourcenkonfiguration unter Berücksichtigung der Präferenzen und Endgerätecharakteristika der weiteren Teilnehmer. Die Ressourcenkonfiguration enthält im allgemeinen verschiedene Verbindungen und Spezialressourcen (z.B Konverter und Konferenzbrücken). Sie stellt eine abstrakte, logische Beschreibung dar. Detaillierte Angaben, z.B Routinginformationen oder Standorte der Spezialressourcen sind nicht enthalten.
Die Rufsteuerung nutzt die Dienste der Ressourcensteuerung und gibt bei jener den Aufbau der Ressourcenkonfiguration in Auftrag. Im einzelnen hat die Ressourcensteuerung die folgenden Signalisierungsaufgaben zu verrichten:

- Ermittlung der verfügbaren Spezialressourcen (SR),
- Auswahl geeigneter SR (z.B anhand ihres Standortes),
- Reservierung der SR,
- Beauftragung der Verbindungssteuerung mit der Errichtung der Verbindungen zwischen Teilnehmern und SR.

Durch die Standardisierung der Schnittstellen zwischen den einzelnen Signalisierungsschichten kann ein offenes Signalisierungssystem realisiert werden [Mul95]. Es ermöglicht die Verrichtung der Signalisierungsaufgaben durch verschiedene Betreiber in Form von Signalisierungsdiensten. Diese können sich somit neben herkömmlichen Übertragungs- und Vermittlungsdiensten auch auf Dienste der Ressourcen- oder Rufsteuerung spezialisieren.
Im Hinblick auf den offenen TK-Markt sind Dienste der Ressourcensteuerung von großer Bedeutung. Insbesondere die oben angedeuteten Vermittlungs- bzw. Agentendienste bilden einen unerläßlichen Teil der Ressourcensteuerung.

3 Das Agenturmodell der Ressourcensteuerung

Abbildung 2 stellt ein erstes Modell der Ressourcensteuerung (ResS) dar. Es wird im folgenden als Agenturmodell bezeichnet und ergibt sich direkt aus den anschaulichen Überlegungen aus Abschnit 2. Die ResS besitzt eine mit einer Agentur vergleichbare Funktionsweise. Sie enthält die Angebote unterschiedlichen Netzbetreiber (NB 1 . . . NB 4) und als zentrales Element den Agenten, der folgende Agentendienste zur Bereitstellung eines TK-Dienstes verrichtet[8]:

Auftragsannahme: Ein Agentendienst bezieht sich auf die Auftragsannahme. Im Beispiel erhält der Agent über die Teilnehmerschnittstelle vom Initiator A den Auftrag, einen Konferenzruf zu den Teilnehmern B und C zu etablieren. Der Auftrag ist dabei eine abstrakte Beschreibung des gewünschten Rufes. Er umfaßt insbesondere Auftragsattribute, die die Initiatorpräferenzen hinsichtlich Preis und Qualität enthalten. Der Agent hat diese bei der späteren Ressourcenselektion zu berücksichtigen.

[8]Letztere sind im folgenden am Beispiel des Konferenzdienstes aus Abbildung 1 erläutert.

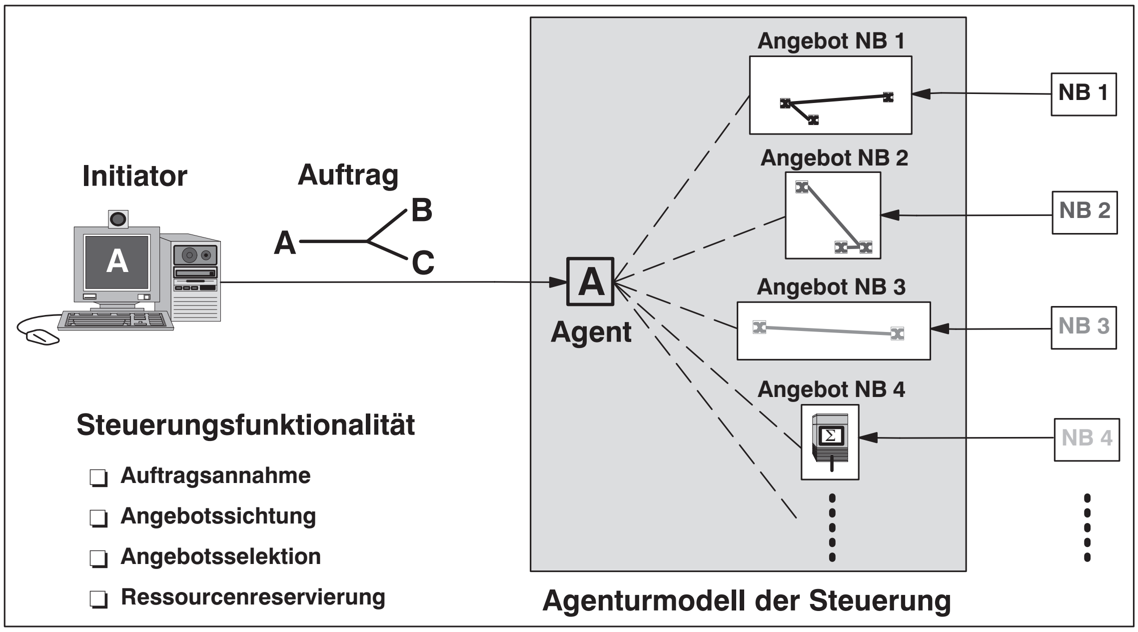


Abbildung 2: Das Agenturmodell der Ressourcensteuerung.

Angebotssichtung: Ein wesentliche Funktionalität der Ressourcensteuerung besteht darin, die Angebote der verschiedenen Netzbetreiber zu sichten. Erst auf dieser Grundlage kann eine effiziente Ressourcenselektion ausgeführt werden. Die Angebotssichtung kann auf zwei unterschiedliche Arten durchgeführt werden:

On-line Angebotssichtung: Der Agent startet den Prozeß der Angebotssichtung, nachdem er den Auftrag für einen Ruf erhalten hat. Die Vorgehensweise gleicht dabei dem in [Smi68] beschriebenen 'Kontrakt-Netz-Algorithmus': der Agent bestimmt die nötigen Ressourcen, kontaktiert geeignete Netzbetreiber und erwartet deren Angebote zurück.

Off-line Angebotssichtung: Der Prozeß der Angebotssichtung erfolgt vollständig getrennt von der Bearbeitung des Auftrags. Unabhängig von einem konkreten Aufrag führt der Agent fortlaufend, quasi autonom die Angebotssichtung durch. Dies hat den Vorteil, daß beim Eintreffen eines Auftrages die entsprechenden Netzbetreiberangebote bereits beim Agenten vorhanden sind.

Bezogen auf das Konferenzbeispiel besitzt der Agent am Ende der Angebotssichtung u.a. die Angebote der Netzbetreiber NB 1 bis NB 4.

Angebotsselektion: Die Angebote der verschiedenen Netzbetreiber sind durch Preis- und Qualitätsmerkmale gekennzeichnet, denen sich der Netzbetreiber verpflichtet. Unter Berücksichtigung der für den Auftrag benötigten Ressourcen und der Preis- und Qualitätspräferenzen des Initiators, reduziert der Makler die Anzahl möglicher Ressourcen und wählt schließlich die am besten geeigneten aus.
Angewandt auf das Konferenzbeispiel identifiziert der Agent zunächst zwei Alterna-

tiven, um den Konferenzdienst zu etablieren. Alternative 1 berücksichtigt die Ressourcen von NB 3, Alternative 2 nicht. Es sei nun angenommen, daß der Initiator eine preiswerte Konferenz einer qualitativ hochwertigen vorziehe und die Ressourcen von NB 3 zu einem günstigen Preis, aber nur mit durchschnittlicher Qualität angeboten werden. In diesem Fall wird der Agent die erste Alternative auswählen. Der Selektionsprozeß ist im allgemeinen wesentlich komplexer als im Beispiel dargestellt, insbesondere bei globalen, multimedialen Diensten. Ein vielversprechender Ansatz, auf der Basis eines hierarchischen Quellroutingalgorithmus, ist in Abschnitt 5.1.5 näher beschrieben.

Ressourcenreservierung: Nachdem der Agent die am besten geeigneten Ressourcen ausgewählt hat, erfolgt deren Reservierung bei den zugehörigen Netzbetreibern. Sie kann über das Netzmanagementsystem der Netzbetreiber oder direkt mit Hilfe eines geeigneten Signalisierungsprotokolles auf Verbindungsebene erfolgen.

4 Die Architektur der Ressourcensteuerung

Eine effiziente Funktionweise in einer realen Telekommunikationsumgebung stellt hohe Zeit-, Genauigkeits- und Verwaltbarkeitsforderungen an die Ressourcensteuerung (ResS). Obwohl das Agenturmodell geeignet ist, die prinzipielle Wirkungsweise der ResS zu illustrieren, kann daraus nicht direkt ein Architekturentwurf gewonnen werden, der diesen Anforderungen genügt. Die wichtigsten Anforderungen und die daraus resultierenden Designentscheidungen sind:

Performanz ⇒ Separate Angebotssichtung: Aufgrund technischer Neuerungen (Digitalisierung, Mehrtonwahl) erwarten TK-Dienstnutzer kurze Rufaufbauzeiten, die im Bereich einer Sekunde liegen. Unter diesen Performanzanforderungen erscheint eine On-line Angebotssichtung nicht machbar. Ein Entwurfsprinzip der ResS besteht daher in der Trennung von Angebotssichtung und Auftragsbearbeitung.

Verwaltbarkeit ⇒ Verteilte Steuerung: Hinsichtlich der Vielzahl von Angeboten, die zusätzlich, aufgrund nachfrageorientierter Tarifierung, regelmäßig aktualisiert werden müssen, erscheint nur eine verteilte Realisierung der ResS dem Verwaltungsaufwand gewachsen. Das Agenturmodell mit einem zentralen Agent wird dahingehend erweitert. Die ResS besteht demnach aus mehreren eigenständigen Agenten. Jeder Agent verwaltet dabei die Angebote eines geographisch begrenzten Verantwortungsbereichs (VB).

Effizienz ⇒ Hierachie und Kooperation: Ziel der ResS ist es, auf dem offenen TK-Markt die im Sinne des Dienstnutzers am besten geeigneten Ressourcen auszuwählen. Der Kenntnisstand des Agenten an Angeboten sollte daher möglichst hoch sein. Ideal wäre ein zentraler 'allwissender' Agent, der aber aus obigen Verwaltbarkeitsgründen ausscheidet. Ein Architekturentwurf auf der Grundlage hierarchisch organisierter, kooperierender Agenten stellt einen möglichen Kompromiß zwischen zentralem und verteilten Ansatz dar.

Hierarchische Organisation: Jeder einzelne Agent ist zunächst für die Angebote seines Verantwortungsbereichs (VB) zuständig. Zur Verwaltung eines globalen Netzes ist eine hierarchische Organisation vorgesehen. Die Agenten werden in Gruppen zusammengefaßt und jede Gruppe und die zugehörigen VB's durch einen (übergeordneten) Agenten verwaltet. Diese werden ebenfalls gruppiert, so daß letztendlich mehrere Hierarchiebenen enstehen. Je höher die Hierarchieebene ist, auf der der Agent angesiedelt ist, desto umfangreicher wird der zu verwaltende VB, aber desto weniger detailliert wird das Wissen, welches der Agent aus Gründen der Verwaltbarkeit über die einzelnen Angebote dieses Bereichs besitzen kann. Die notwendige Informationsreduktion erfolgt durch einen Zusammenfassungsprozeß (s. Abschnitt 5.1.4).

Agentenkooperation: Die Agenten einer Gruppe tauschen untereinander zusammengefaßtes Wissen, sogenannte zusammengefaßte Angebote, über die Angebote ihres VB's aus. Auf diese Weise besitzt jeder Agent über das detaillierte Angebotswissen innerhalb seines VB's hinaus weniger detaillierte Kenntnis über die Angebote anderer Agenten der Gruppe. Der Zusammenfassungs- und Kooperationsprozeß wird zum einen innerhalb jeder Hierarchieebene, aber auch zwischen den Hierarchieebenen durchgeführt. Durch die Kooperation mit übergeordneten Agenten baut der Agent schrittweise ein hierarchisch strukturiertes Angebotswissen über das gesamte Netz auf. Das wesentliche Merkmal besteht darin, daß je weiter ein fremder VB's von seinem eigenen VB entfernt ist, desto ungenauer wird das Wissen, welches der Agent über die Angebote des fremden VB's besitzt.
Das hierarchisch strukturierte Angebotswissen bildet die Grundlage für den Selektionsprozeß.

5 Die Funktionsweise der Ressourcensteuerung

Zur Veranschaulichung der Funktionsweise der Ressourcensteuerung sind die beiden wesentlichen Prozesse, die Off-line Angebotssichtung und die Ressourcenselektion während des Rufaufbaus, in den Abbildungen 3, 4 und 5 dargestellt.

5.1 Die Angebotssichtung

5.1.1 Physikalisches Netz und Angebote

Im unteren Teil von Abbildung 3 sind das physikalische Netz, das logische Netz der Angebote und die einzelnen Teilnetze der Netzbetreiber dargestellt. Die unterste Ebene bildet das physikalische Netz, bestehend aus den Übertragungsabschnitten und Vermittlungsknoten verschiedener Netzbetreiber. Auf der Basis seines Teilnetzes etabliert jeder Netzbetreiber logische Verbindungen und ermittelt ein Angebot für deren Nutzung. Dieses Angebot enthält Preis- und Qualitätsangaben, denen sich der Netzbetreiber verpflichtet. Die Angebote aller NB bilden zusammen das Angebotsnetz. Es ist in der Ebene über dem physikalischen Netz dargestellt.

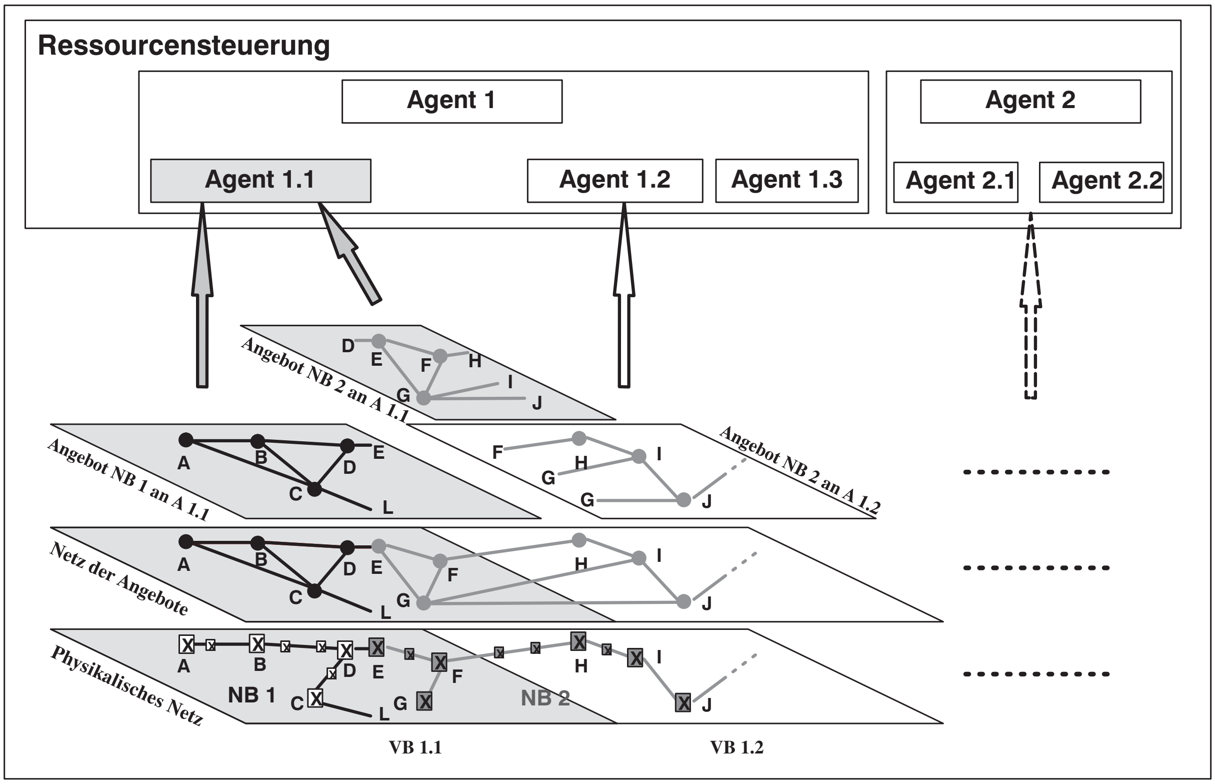


Abbildung 3: Verteilter und hierarchischer Aufbau der Ressourcensteuerung.

Bei der Nutzung von ATM als Vermittlungstechnik beziehen sich die Angebote auf virtuelle Pfade (VP's). Sie werden vom Netzbetreiber eingerichtet, weisen unterschiedliche Qualitätswerte auf (z.B. aufgrund von Routing- oder Prioritätsmechanismen) und erhalten nachfrageabhängig einen Nutzungspreis zugeordnet.

Die Erstellung sinnvoller Angebote kann durch die Netzmanagementsysteme der Netzbetreiber unterstützt, bzw. sogar vollständig automatisiert erfolgen.

5.1.2 *Hierarchischer Aufbau*

Wie in Abschnitt 4 beschrieben, wird das Angebotsnetz durch die hierarchisch organisierten, kooperierenden Agenten der ResS verwaltet. Die exemplarische ResS in Abbildung 3 enthält zwei Hierarchieebenen, mit insgesamt 5 Agenten und 2 übergeordneten Agenten[9]. Jeder Agent verwaltet die Angebote seines Verantwortungsbereichs (VB). Im Prinzip kann einerseits ein VB die Angebote unterschiedlicher NB umfassen, andererseits kann sich das Angebotsnetz eines NB über mehrere VB's erstrecken. So verwaltet in Abbildung 3 einerseits Agent 1.1 den grau unterlegten Bereich. Dieser enthält Angebote der NB 1 und 2. Andererseits ist das Angebotsnetz von NB 2 auf die VB's der Agenten 1.1 und 1.2 verteilt. Zwischen Netzbetreibern, genauer deren Netzmanagementsystemen und den Agenten der

[9]Bezüglich der Anzahl der Hierarchieebenen besteht dabei keine prinzipielle Einschränkung.

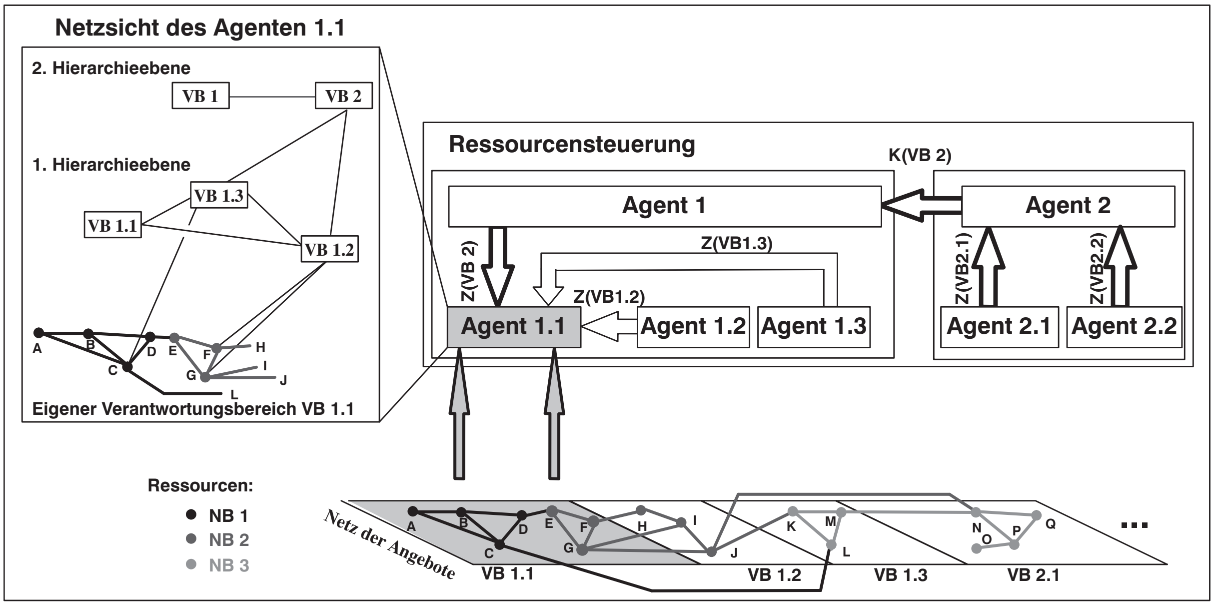


Abbildung 4: Die Off-line Angebotssichtung.

ResS findet ein Informationsfluß über die aktuellen Angebote statt.

5.1.3 Die Agentenkooperation zum Wissensaustausch

Die Pfeile in Abbildung 4 symbolisieren den relevanten Informationsfluß aus der Sicht des Agenten 1.1. Dieser konstruiert daraus ein hierarchisch strukturiertes Wissen über das Angebotsnetz, die 'Netzsicht' des Agenten. Aus den Angeboten der Netzbetreiber rekonstruiert der Agent den in seinen VB fallenden Teil des Angebotsnetzes. Über diesen 'lokalen' Bereich besitzt der Agent exakte Kenntnis. Zusätzlich erhält er die zusammengefaßten Angebote der seiner Gruppe angehörigen Agenten. Von Agent 1.2 und Agent 1.3 erhält er die zusammengefaßten Angebote Z(VB 1.2) und Z(VB 1.3). Aus ihnen baut er sein Wissen über die erste Hierarchieebene auf.
Der Informationsaustausch mit dem übergeordneten Agenten 1 ermöglicht Agent 1.1 Wissen über die Angebote der zweiten Hierarchieebene zu erwerben. Voraussetzung ist zunächst die Kooperation der übergeordneten Agenten untereinander. Sie hat zur Folge, daß im Beispiel Agent 1 die zusammengefaßte Information Z(VB 2) über die Angebote des gesamten Bereiches VB 2 ($VB = \bigcup_i VB2.i$) erhält, die er anschließend an Agent 1.1 weitergibt. Agent 2 berechnet Z(VB 2) aus Z(VB 2.1) und Z(VB 2.2) mit Hilfe des Zusammenfassungsprozesses.

5.1.4 Der Zusammenfassungsprozeß

Auf jeder Hierarchiestufe werden zusammengefaßte Angebote mit Hilfe eines Zusammenfassungsalgorithmus gebildet. Dies führt zu zusammengefaßten Preis- und Qualitätsinformationen. Die zusammengefaßte Preisinformation besteht aus einer Menge von Repräsentationspreisen. Für jedes Paar an Eingangsknoten des betrachteten VB's wird ein

Repräsentativpreis berechnet. Ein VB mit n Eingangsknoten führt so zu einem zusammengefaßten Angebot mit $\frac{n(n-1)}{2}$ Repräsentativpreisen. Der Repräsentativpreis für jedes Eingangspunktepaar wird aus den Preisen aller in diesem VB möglichen Routen unter Berücksichtigung der Zusammenfassungsstrategie gebildet. Einige einfache Zusammenfassungsstrategien sind:

Best-Case-Strategie: Der Preis der preiswertesten Route bildet den Repräsentativpreis.

Durchschnittspreis-Strategie: Der Repräsentativpreis ergibt sich als Mittelwert der Preise aller möglichen Routen.

Worst-Case-Strategie: Der Preis der teuersten Route bildet den Repräsentativpreis.

Weitere Strategien auf der Basis von statistischen Kennwerten über die Preisverteilung eines VB's werden derzeit untersucht. Für die Zusammenfassung von Qualitätswerten sind ähnliche Zusammenfassungsalgorithmen denkbar, die Zusammenfassung von Spezialressourcen gestaltet sich jedoch weitaus schwieriger.

5.1.5 Der Selektionsprozeß

Die Ressourcenauswahl erfolgt auf der Grundlage eines hierarchischen Quellroutingverfahrens. Anhand des hierarchisch strukturierten Wissens, welches der Agent im Laufe der Angebotssichtung über das Angebotsnetz aufgebaut hat, bestimmt er mit abnehmender Detailliertheit die beste Route zu den beteiligten Teilnehmern (s. Beispiel in Abschnitt 5). Er ermittelt in Frage kommende Alternativrouten und selektiert jene, deren Preis- und Qualitätscharakteristika den Anforderungen des Dienstauftrags am besten entspricht. Wenn beispielsweise der preiswerteste Ruf angefordert wird, wertet der Agent zunächst die zusammengefaßten Preise aus, um eine Preisvorhersage für die möglichen Alternativrouten zu treffen. Anschließend selektiert er die Route entsprechend der niedrigsten Preisvorhersage.

Der effektive Preis des etablierten Rufes wird von der Preisvorhersage mehr oder weniger stark abweichen. Verantwortlich dafür ist zum einen die nachfrageorientierte Tarifierung und die resultierende dynamische Preisänderung, zum anderen Fehler- oder Stausituationen, die lokale Umwege erzwingen.

Die Richtung der Abweichung und damit das Verhalten des Selektionsprozesses ist eng mit der gewählten Zusammenfassungsstrategie verbunden. So zeigt beispielsweise ein Selektionsprozeß auf der Basis der Best-Case Zusammenfassungsstrategie ein 'optimistisches' Verhalten, da die vorhergesagten Preise niedriger als die letztendlich beim Rufaufbau erreichten Preise sind.

5.2 Exemplarischer Rufaufbau

Abbildung 5 beschreibt den Aufbau eines Punkt-zu-Punkt Rufes zwischen zwei TK-Dienstnutzern A und B durch die agentenbasierte Ressourcensteuerung. Von zentraler Bedeutung auf dem offenen TK-Markt ist dabei der Selektionsprozeß. Hierfür wird ein hierarchischer Quellroutingalgorithmus verwendet. Voraussetzung ist das hierarchisch strukturierte

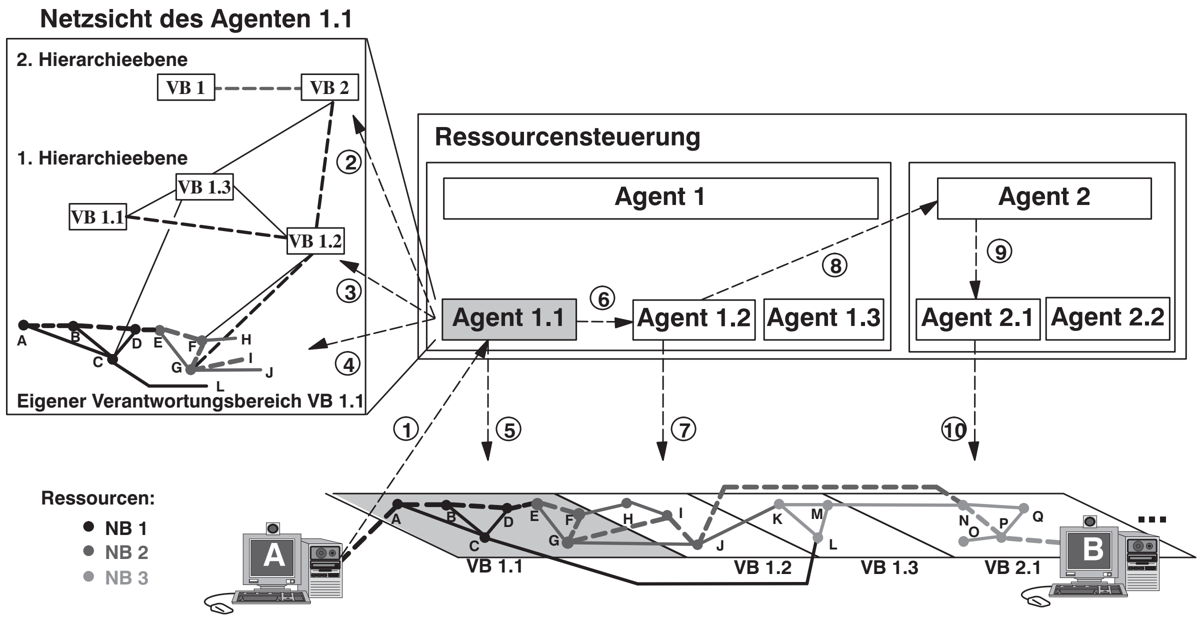


Abbildung 5: Exemplarischer Rufaufbau auf dem offenen TK-Markt.

Netzwissen, welches der Agent durch die Off-line Angebotssichtung erwirbt.
Zur besseren Darstellung des Selektionsprozesses wurde als Beispiel ein einfacher Punkt-zu-Punkt Ruf gewählt. Dies ist keine prinzipielle Einschränkung. Mit dem gleichen Algorithmus ist es möglich, komplexere Multi-Media-Rufe mit Punkt-zu-Mehrpunkt Verbindungen und Spezialressourcen aufzubauen. Die Betrachtung von Spezialressourcen erfordert allerdings das Wissen über deren Standort, Preis und Qualitätseigenschaften.
Die Verwendung eines hierarchischen Quellroutingalgorithmus weist einige Vorteile auf:

- Verglichen mit Routingsverfahren, die lediglich auf lokalem Wissen über einen VB basieren, führt das hierarchische Quellrouting infolge der globalen Netzsicht, wenn auch mit abnehmenden Detaillierungsgrad, zu besseren Routingergebnissen.

- Hierarchisches Quellrouting ermöglicht die gemeinsame Auswahl von Spezialressourcen und Verbindungen im Falle von Multi-Media-Rufen. Im Gegensatz zum Routingverfahren im EU-Projekt MAGIC [RAC93], das eine getrennte Auswahl, zunächst der SR und schließlich der benötigten Verbindungen vorsieht, läßt das hierarchische Quellrouting bessere Routingergebnisse auf dem offenen TK-Markt erwarten.

Der Aufbau des Beispielrufes durch die Ressourcensteuerung vollzieht sich in den folgenden Schritten:

1. Teilnehmer A möchte einen Ruf zu Teilnehmer B aufbauen. Da der Teilnehmeranschluß von A im VB von Agent 1.1 liegt, erhält letzterer den entsprechenden Auftrag.

2. Aus der Teilnehmeradresse von B und seiner Netzsicht erkennt Agent 1.1, daß der Teilnehmeranschluß von B im VB von Agent 2 liegt. Aus seiner Kenntnis über die 2.

Hierarchieebene leitet er desweiteren ab, daß lediglich eine mögliche Route vom VB des Agenten 1 zum VB des Agenten 2 existiert. Auf dieser Hierarchieebene ist daher kein Selektionsprozeß nötig.

3. Anders gestalten sich die Verhältnisse eine Hierarchieebene tiefer. Mit dem Ziel, ausgehend vom VB 1.1 den VB 2 zu erreichen, ermittelt der Agent 4 mögliche Alternativrouten (VB's 1.1-1.2-2, VB's 1.1-1.3-2, VB's 1.1-1.2-1.3-2, VB's 1.1-1.3-1.2-2)). Die Routen besitzen u.U unterschiedliche Preise und Qualitätseigenschaften, die der Agent 1.1 aus den zusammengefaßten Angeboten über VB 1.2 und 1.3 berechnet. Im folgenden sei angenommen, daß die erste Alternativroute (VB's 1.1-1.2-2) selektiert wird, da sie am besten die Auftragsanforderungen aus Schritt 1 erfüllt.

4. Im VB 1.1 läuft ein vergleichbarer Selektionsprozeß ab. Hier ist das Ziel, aus der Vielzahl möglicher Alternativrouten zum VB 1.2 die optimale Route auszuwählen. Die Selektion kann der Agent hierbei anhand der exakten Angebotskenntnis im VB 1.1 durchführen. Ihr Ergebnis sei die Route A-B-D-E-F-G-I.

5. Agent 1.1 leitet den Verbindungsaufbau zm VB 1.2 ein, indem er die entsprechenden Teilverbindungen bei NB 1 (A-B-D-E) und NB 2 (E-F-G-I) anfordert.

6. Nachdem Agent 1.1 alle Aufgaben durchgeführt hat, die ihm aufgrund seiner Netzsicht möglich sind, kontaktiert er Agent 1.2 und übermittet ihm einen Fortsetzungsauftrag. Dieser enthält neben dem Ziel-VB (VB 2) die Teilnehmerpräferenzen und die bisherigen Routingeintscheidungen (Schritte 2 und 3).

7. Auf der Grundlage seines hierarchischen Angebotswissens (Netzwissens) entscheidet sich Agent 1.2 für die Route I-J-N, die in den VB 2 führt und beauftragt die Verbindungen bei NB 2.

8. Agent 1.2 beauftragt Agent 2 mit der Fortsetzung des Rufaufbaus.

9. Agent 2 erkennt anhand der Adresse von Teilnehmer B, daß dessen Teilnehmeranschluß im VB von Agent 2.1 liegt und das die zuletzt etablierte Verbindung (J-N) bereits in diesem VB endet. Ein Selektionsprozeß ist daher nicht erforderlich, Agent 2 beauftragt direkt Agent 2.1 mit der weiteren Durchführung des Rufaufbaus.

10. Agent 2.1 führt das Routing im letzten VB durch und beauftragt schließlich NB 3 mit dem Verbindungsaufbau (via N-O) zum Teilnehmeranschluß von B.

6 Zusammenfassung

Der eingeführte offene Telekommunikationsmarkt sieht vor, daß die Betreiber für die einzelnen Ressourcen eines Telekommunikationsdienstes durch den Teilnehmer ausgewählt werden können. Dies stellt hohe Anforderungen an ein zukünftiges Signalisierungssystem. Ausgehend von einer Signalisierungsarchitektur [RAC93] mit getrennter Ruf-, Ressourcen- und Verbindungssteuerung, wurde eine erweiterte, agentenbasierte Ressourcensteuerung als Grundlage eines agentenbasierten Signalisierungssystems vorgestellt, welches obige Unterstützung ermöglicht. Die Hauptfunktionalität der Ressourcensteuerung besteht dabei in

der Angebotssichtung und Angebotsselektion. Im Verlauf der Angebotssichtung werden die Angebote der Netzbetreiber gesammelt, zusammengefaßt und verbreitet. Darauf aufbauend erfolgt im Rahmen eines Selektionsprozesses die Auswahl der am besten geeigneten Ressourcen im Sinne der Teilnehmerpräferenzen.
Um die Vielzahl der Angebote verwalten und daraus effizient Ressourcen auswählen zu können, wird die Realisierung der Ressourcensteuerung durch einen Verbund hierarchisch organisierter, kooperierender Agenten vorgeschlagen. Darüberhinaus erscheint eine Trennung von Angebotssichtung und Auftragsbearbeitung aus Performanzgründen sinnvoll.

Literatur

[BMMM94] H. Bussey, S. Minzer, P. Mouchtaris und S. L. Moyer. EXPANSE Software for Distributed Call and Connection Control. *Int. Journal of Communications Systems*, 7(2), April - Juin 1994.

[Mul95] H. Muller. A Signaling Framework for Multimedia Broadband Networks. In *Proceedings of 2nd IEEE MICC'95*, Langkawi, November 1995.

[Pou82] T. Pousette. Technology, Pricing and Investment in Telecommunications. *Working paper 71 of the Industrial Institute for Economic and Social Research, Stockholm, Sweden*, 1982.

[RAC93] RACE II MAGIC R2044. *Protocols and Concepts of B-ISDN Signaling. RACE II Multiservice Applications Governing Integrated Control (MAGIC), Deliverable 5*, November 1993.

[Sie93] G. Siegmund. *Grundlagen der Vermittlungstechnik*. R. v. Decker Verlag, Heidelberg, 1993.

[Smi68] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. In A. H. Bond und L. Gasser, Hrsg., *Readings in Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., 1968.

[Soc95] IEEE Communications Society. Regulatory Reform: Unleashing the Competitive Drive in the Information Age. *IEEE Communications Magazine*, 33(12), December 1995.

[Vic71] W. Vickrey. Responsive Pricing of Public Utility Services. *Bell Journal of Economics and Management Science*, 2, 1971.

1.3.9 Agenten-basierte, marktwirtschaftlich orientierte Lastverteilung für numerische Algorithmen

Teilprojekt: Verteilte numerische Algorithmen auf Bäumen
Martin Backschat

1 Einleitung

In zunehmendem Maße werden heutzutage Workstations als Arbeitsplatzrechner eingesetzt. Mit immer höherer Rechenleistung und größerem Speicherausbau reichen sie, zu einem Workstation-Netz zusammengefaßt, an die Leistungsfähigkeit moderner Supercomputer heran. Dabei sind sie zudem in der Anschaffung und Wartung billiger, als auch in der Handhabung einfacher.

Auf der anderen Seite entstehen heutzutage in Forschung und Technik Probleme, deren Rechen- und Speicheraufwand die Grenzen der lokal zur Verfügung stehenden Hochleistungsrechner sprengt. Der Trend geht deshalb dahin, die Probleme durch verteiltes Rechnen zu lösen, d.h. man greift auf eine Vielzahl von Rechnern zurück, die jeweils nur einen Teil des Problems bearbeiten. Häufig kann man dazu das Divide&Conquer-Prinzip anwenden. Hierbei wird das Gesamtproblem geeignet in Teilprobleme zerlegt, die im Idealfall unabhängig voneinander gelöst werden können. Das Divide&Conquer-Prinzip kann nun rekursiv solange angewendet werden, bis die entstehenden Probleme genügend klein für eine direkte Lösung sind. Am Ende werden die Teilergebnisse aufgesammelt und zum endgültigen Ergebnis zusammengefaßt. Durch eine ausreichend feine Aufteilung läßt sich der Speicherbedarf der Teilprobleme so weit reduzieren, daß sie auch auf Rechnern mit verhältnismäßig wenig Hauptspeicher Platz finden.

Viele neuartige numerische Algorithmen, wie sie etwa in den modernen Verfahren zur numerischen Simulation von ingenieurswissenschaftlichen Problemen [HHSZ94, GN95] eingesetzt werden, besitzen bereits eine Divide&Conquer-Struktur. Darunter fallen z.B. Mehrgittermethoden, Dünngitterverfahren und die sog. Kombinationstechnik. Ziel dieses Teilprojektes ist es, mit Hilfe von fortschrittlichen Werkzeugen der Informatik, diese Verfahren parallel zu implementieren. Darüber hinaus soll die Implementierung solcher numerischer Algorithmen auf verteilten Rechnernetzen konkret studiert und durch geeignete Maßnahmen optimiert werden.

Einige dieser numerischen Verfahren wurden im Rahmen dieses Teilprojekts durch meinen Vorgänger Anton Frank bereits auf ihr Potential hinsichtlich der Parallelisierung auf Workstation-Netzen untersucht [Fra96]. Es zeigt sich, daß sehr gute Ergebnisse erzielt werden können, wenn bestimmte Rahmenbedingungen erfüllt sind. Es müssen vorallem Methoden gefunden werden, die den speziellen Anforderungen von großen Workstation-Netzen gerecht werden, vorallem bzgl. der folgenden Aspekte:

- Workstation-Netze zeichnen sich durch Heterogenität in Bezug auf Betriebssystem, Hardware-Ausstattung, CPU-Leistung, verfügbare Software und Struktur im Dateisystem, sowie der Vernetzung aus.

- Workstations sind nicht exklusive Betriebsmittel, vielmehr laufen üblicherweise viele Applikationen mehrerer Benutzer gleichzeitig. Desweiteren ist die Nutzung und

Auslastung oft von der Tageszeit abhängig.

- Die Vernetzung zwischen den Workstations ist komplex und unterliegt einer unvorhersehbaren Dynamik hinsichtlich Netzlast und Konfiguration. Prozeßkommunikation ist demnach ein wesentlicher Engpaßfaktor im verteilten Rechnen.

Um die zur Verfügung stehenden Ressourcen möglichst optimal auszunutzen, ist demnach eine intelligente Verteilung der zu berechnenden Teilprobleme auf die vorhandenen Rechner ein entscheidender Punkt bei der Parallelisierung. Diese Managementaufgabe sollte nach Möglichkeit die Last automatisch so auf die Rechner verteilen, daß diese möglichst gut ausgelastet, aber keinesfalls überlastet sind. Bei Workstation-Netzen ist dabei insbesondere zu berücksichtigen, daß die Workstations vorallem als Arbeitsplatzrechner genutzt werden. Dieser Benutzerbetrieb darf durch die zusätzliche Last der Berechnungen nicht beeinträchtigt werden. Neben dem oben aufgeführten Charakter von Workstation-Netzen ist auch aufgrund des oft nicht exakt vorhersehbaren Laufzeitverhaltens der parallelen Applikationen statt manueller oder statischer Lastverteilung der Einsatz eines dynamischen, dezentral organiserten Systems von Vorteil.

In den letzten Jahren wurden diesbezüglich eine Vielzahl neuer Lastverteilungskonzepte entworfen, die sich an Analogien aus anderen Gebieten orientieren, Ein vielversprechender Ansatz ist dabei der Bezug zum Bereich der Marktwirtschaft [MD88, Fer89, Bog94]. Da die marktwirtschaftlichen Mechanismen das Ressourcenmanagement in unserer Gesellschaft sehr vereinfachen, könnte man erwarten, daß sie auch in verteilten Computersystemen ähnlich sinnvoll einsetzbar sind. Der Preismechanismus erlaubt Rechnern mit unterschiedlichen Leistungsprofilen, einen entsprechenden Nutzungspreis zu verlangen. Andererseits können Berechnungen entsprechend dem Guthaben ihres Auftraggebers auf denjenigen Rechnern stattfinden, die ihnen bezüglich ihrer Präferenzen (z.B. Preis/Leistung) am besten geeignet erscheinen. Im Rahmen dieses Teilprojekts sollten deshalb zunächst Marktmechanismen bzgl. ihrer Einsatzfähigkeit für die Lastverteilung in verteilten Systemen untersucht und in einem Lastverteilungssystem implementiert werden.

2 Marktwirtschaftliches orientiertes Ressourcenmanagement

2.1 Marktmechanismen in verteilten Systemen

Im ökonomischen Modell eines verteilten Systems koordinieren Marktmechanismen — Preise und Verhandlungen — die Aktivität von Objekten. Wie in der realen Marktwirtschaft sorgen diese Mechanismen für eine dezentrale Zuteilung der Ressourcen. Dies geschieht lediglich mittels eingeschränktem Wissen und lokalen Entscheidungen anstatt mit Hilfe einer zentralen Koordination.

2.1.1 Vorteile von Marktmechanismen

Im folgenden werden einige wichtige Aspekte und Vorteile des Einsatzes von Marktmechanismen in verteilten Systemen näher vorgestellt:

- *Ressourcenallokation:* Ein Großteil der mikroökonomischen Theorie beschäftigt sich mit dem Problem der optimalen Allokation von knappen Ressourcen. Auch in verteilten Systemen übersteigt die Nachfrage an Ressourcen üblicherweise das Angebot, wie etwa der Bedarf an Rechenzeit, Speicher, Kommunikationsbandbreite usw. In der realen Wirtschaft hat sich für die Lösung dieses Problems das Marktkonzept bewährt. In zahlreichen Forschungsarbeiten zeigt sich, daß die Marktmechanismen unter vielen Umständen eine optimale Allokation erreichen. Dazu ist keine zentrale Koordination nötig, vielmehr kommt das Marktgleichgewicht aufgrund des „egoistischen“ Handelns der einzelnen Teilnehmer zustande.

- *Effizienz:* Die Wohlfahrtsökonomie als Teil der mikroökonomischen Theorie liefert eine Definition von Effizienz, die in einigen Fällen präziser bzw. nützlicher als die in der Informatik übliche Bedeutung ist. Dort setzt man „effizient“ mit „schnell, relativ zu anderen möglichen Methoden“ gleich. Dies führt zu unpräzisen Angaben, da (schnellere) Alternativen zwar existieren, aber nicht bekannt sein können. Ökonome verallgemeinern den Effizienzbegriff nun insofern, als daß sie nicht nur die Laufzeit (effiziente Nutzung der Zeit), sondern die Nutzung aller verfügbaren Ressourcen betrachten. Ökonomische Effizienz bezieht sich auf den Tradeoff zwischen den angebotenen Ressourcen, sowie auf den Wettbewerb für ihre Nutzung und die Konditionen, die dafür sorgen, daß Ressourcen für den größtmöglichen Nutzen aller Beteiligten verwendet werden.

  Ein wichtiges Kriterium für Effizienz ist die *Pareto-Optimalität*. Eine Allokation ist pareto-optimal, falls sich der Nutzen keines Teilnehmers durch Umverteilung der Ressourcen erhöhen läßt, ohne daß dabei mindestens ein anderer eine Nutzeneinbuße hinnehmen muß. Diese verallgemeinerte Auffassung erlaubt nun auch die Bestimmung von rationellen Allokationen und Entscheidungen. Im Zustand der Pareto-Optimalität ist es nicht möglich, die Ressourcenbelegung bei gegebenen Präferenzen der Agenten unter jedem Aspekt zu verbessern. Somit lassen sich alle pareto-optimalen Zustände als rational bezeichnen, da Abweichungen nicht zu allgemein besseren Lösungen führen.

- *Dezentrale Entscheidungen:* Marktsysteme basieren auf dezentralen Entscheidungen und jeder Teilnehmer benötigt nur lokale Informationen, um seine Entscheidung mithilfe seiner Präferenzen und ggf. seiner Bewertungen von Ressourcen und Service vorzunehmen. In Situationen, in denen eine gewaltige Informationsflut zu verarbeiten ist, erscheint es unpraktisch oder sogar unmöglich, vernünftige Entscheidungen zentral zu fällen. Im Falle von unsicheren oder widersprüchlichen Informationen haben dezentrale Entscheidungsträger eine weitaus bessere Chance, eine sinnvolle Wahl zu treffen.

  Dezentralismus ermöglicht es auch, lokale Probleme durch speziell angepaßte Verfahren zu lösen. Interagierende Individuen können z.B. eigene, spezifischere und effizientere Methoden bzgl. Kommunikation und Informationsaustausch entwickeln. Eine zentrale Instanz müßte alle diese Austauschsprachen kennen, um auf jede Information zugreifen zu können.

- *Dynamische Anpassung:* Eine der größten Stärken eines Marktes ist seine Fähigkeit, sich schnell an plötzliche, unvorhersehbare Änderungen anzupassen. Jeder Marktteilnehmer muß nur seine eigenen Bewertungen an diese Änderungen anpassen. Ein neues Marktgleichgewicht reflektiert dann automatisch die neue optimale Ressourcenzuteilung. Dies ist auch ein signifikanter Vorteil in Bezug auf verteilte Systeme. Denn der Sytemzustand ändert sich laufend und ist im allgemeinen nicht vorhersehbar. Eine schnelle Anpassung an Änderungen ist deshalb sehr wichtig.

2.1.2 Marktorientierte Ressourcenzuteilung

Um die Zuteilung bzw. das Handeln von Ressourcen marktorientiert zu gestalten, wurden in der Literatur eine Reihe von Verfahren entwickelt. Die im folgenden kurz vorgestellten zwei Methoden eignen sich für eine Anwendung auf Ressourcen in verteilten Systemen besonders gut und werden deshalb im Rahmen dieses Teilprojekts näher untersucht bzw. geeignet erweitert.

- *Vickrey-Auktion:* Für die Zuteilung einer Ressource übernimmt ein Auktionator die Aufgabe, die Ressource an den Höchstbietenden zu versteigern. Sobald alle Gebote eingegangen sind, erhält der Agent mit dem höchsten Gebot den Zuschlag, wobei er nur den Geldbetrag des zweithöchsten Gebots zahlen muß. Die Vickrey-Auktion läßt sich auch dahingehend verallgemeinern, daß mehrere Ressourcen von verschiedenen Anbietern gleichzeitig (zu verschiedenen Preisen) zugeteilt werden [Var95]. Ein wichtiger Vorteil dieses Verfahren ist, daß die Bietenden keinen Anreiz zum Bluffen haben, d.h. die Höhe der Gebote entspricht der wahren Zahlungsbereitschaft der Agenten. Es sorgt somit für eine nutzen- und gewinnmaximierende, pareto-optimale Zuteilung. Ein weiteres interessantes Auktionsverfahren, die *Double Auction*, führt in der Regel ebenfalls zu pareto-optimalen Allokationen, basiert jedoch auf einem iterativen, kommunikationsintensiven Anpassungsprozeß.

- *Tatônnement-Verfahren:* Ziel dieser Verfahren ist es, ein Marktgleichgewicht bzgl. Angebot und Nachfrage nach allen Ressourcen herzustellen, d.h. Preise für alle Ressourcenarten zu finden, so daß Angebotsmenge gleich Nachfragemenge ist. Dazu wird iterativ der Preis(vektor) für die Ressourcen von einer verantwortlichen Instanz entsprechend dem jeweiligen Nachfrage- bzw. Angebotsüberschuß korrigiert und die angepaßten Angebote und Nachfragen eingeholt. Die allgemeine Gleichgewichtstheorie liefert Konditionen, bei denen solch ein Prozeß in endlich vielen Schritten zu einer pareto-optimalen Allokation führt. Für den Einsatz in verteilten Systemen eignet sich u.a. vorallem die dezentrale *WALRAS-Variante* [Wel93].

2.2 Evolution zu einem offenen Dienstmarkt

2.2.1 Kommunikation

Als Grundlage für die Interaktion von Agenten auf einem ausreichend hohem Niveau wurden vorallem im Bereich der Multiagentensysteme eine Vielzahl von Ansätzen verfolgt. Als besonders geeignet hat sich die Sprechakttheorie erwiesen, die die Interaktionen auf eine streng klassifizierte Sammlung von Äußerungen (sog. *Performatives*) zurückführt. Dieser Ansatz ist auch Basis von KQML („Knowledge Query and Manipulation Language", [MLF96]), einer textorientierten, standardisierten und erweiterbaren Kommunikationssprache für Multiagentensysteme. Dort gibt es drei Klassen von Performatives: Feststellungen (z.B. TELL, REPLY, ADVERTISE), Anweisungen (z.B. ASK-ONE, ASK-ALL, MONITOR) und Verbindlichkeiten (z.B. COMMIT, PROPOSE, AGREE). Sie soll auch in diesem Teilprojekt als Ausgangsbasis für eine eigene Kommunikationsprache dienen.

2.2.2 Dienste

Der über Rechnergrenzen hinweg angebotene Dienst zeichnet sich heutzutage als ein wichtiges Konstruktionsprinzip für offene, verteilte Systeme ab. Die Trennung zwischen dem Dienstanbieter und dem Dienstnutzer gibt beiden Seiten mehr Autonomie und ermöglicht die Schaffung von komplexeren ökonomischen Strukturen im elektronischen Bereich. Man spricht in diesem Zusammenhang von einem *offenen Markt von Diensten*, in dem Anbieter und Nutzer Kontrakte für die Dienstnutzung aushandeln. Die Tarifierung hängt dabei von Angebot und Nachfrage ab. Wie in realen Märkten können auf der Basis der bereits verfügbaren Dienste weitere, komplexere Dienste realisiert werden. Insgesamt wird also angestrebt, verteilte Systeme konstruktiv und modular aus exisitierenden Diensten zusammenzusetzen.

2.2.3 Dienstspezifikation

Als Grundlage für eine deklarative Dienstspezifikation und die Dienstvermittlung wurde im Rahmen dieses Teilprojekts das sehr allgemeine und leistungsfähige Konzept der *CRV-Formulare* entwickelt. CRV steht für „Capabilities, Requirements und Variables". Das bedeutet, daß eine Angebotsspezifikation und eine Nachfrage formularbasiert erfolgt. Die Einträge eines solchen Formulars sind in Abschnitten (Klassen) untergliedert. Abschnitte können Fähigkeiten (Fakten), Einschränkungen bzw. Anforderungen und verhandelbare Einträge in Form von benannten Attributen enthalten. Zum Beispiel kann ein Angebot spezifizieren, welche Rechenleistung zur Verfügung steht (Capability), welche Benutzer den Dienst in Anspruch nehmen dürfen (Requirement) und wielange die Berechnung maximal und zu welchem Preis dauert (verhandelbar). Jedem Eintrag läßt sich zudem noch eine Präferenz und eine Gewichtung zuordnen. Diese Angaben werden verwendet, um bei einer Vermittlung im Matching-Prozeß den Dienstangeboten Maßzahlen zuzuordnen, auf denen schließlich die Auswahl beruht. Fakten können auch als dynamisch gekennzeichnet sein, um darauf hinzuweisen, daß diese Information laufenden Änderungen unterliegt, wie etwa die Angabe des aktuellen Dienstnutzungspreises. Weiterhin lassen sich Formulare in Unterformulare gliedern oder durch Anhängen weiterer Attribute in den Abschnitten erweitern. Dies entspricht der Klassenschachtelung bzw. -vererbung in objektorientierten Sprachen.

Formulare können zur Laufzeit erzeugt und ausgefüllt werden. Viel flexibler und in Hinblick auf einen genormten Informationsaustausch interessanter sind jedoch *Formularschablonen*. Darin sind lediglich die Abschnitts- und Attributnamen, sowie deren Typ (z.B. Zahl, Liste, Zeichenkette) enthalten. Schablonen für die verschiedensten Spezifikationen werden in der Regel von den Dienstvermittlern zur Verfügung gestellt. Das Ausfüllen (Instantiierung) einer solchen Schablone kann nun schrittweise, sozusagen delegiert geschehen: Der Agent füllt alle für ihn relevanten Teile aus und gibt das Formular an einen im System tieferliegenden Teil weiter, der darin an den passenden Stellen systeminterne Daten einträgt, etwa die Rechnerleistung etc. Die Kopplung der in beliebiger Struktur in den Agenten gespeicherten Informationenen an Formularschablonen wird durch agentenspezifische Wissensdatenbanken automatisiert.

2.2.4 Verbreitung von Dienst- und Zustandsinformationen

Dienstvermittler nehmen Dienstspezifikationen in Form von CRV-Formularen entgegen und speichern diese in einer internen Datenbank. Einen Schritt weiter gehen Maklerorganisationen, wie sie in diesem Teilprojekt entwickelt werden. Dort liegt die Datenbank verteilt vor. Für erste Untersuchungen wurde eine hierarchische Organisationsform gewählt, bei der auf jedem Rechner ein lokaler Makler installiert ist. Dieser hält nur Kontakt zu seinem übergeordneten Cluster-Makler und übermittelt ihm laufend die lokal gesammelten Dienstspezifikationen. Dazu werden die vorhandenen CRV-Formulare mittels statistischer *Aggregation* zu einem CRV-Formular zusammengefaßt. Diese Informationsaggregation und -propagation wird zwischen allen Maklerebenen bis hin zur Maklerspitze fortgeführt.

Andersherum werden systemglobale Informationen von der Maklerspitze ebenenweise bis an die lokalen Makler weitergereicht. Somit hat jeder Makler ständig die nötigen Informationen über die verfügbaren Dienstangebote in seinem Verwaltungsgebiet, aber auch eine aggregierte Sicht auf den systemglobalen Zustand. Beide Informationen sind für eine effiziente, zuverlässige und in Bezug auf die Größe des verteilten Systems skalierbare Dienstvermittlung ausschlaggebend.

2.2.5 Dienstvermittlung und Angebotsrecherche durch Maklersysteme

Der (hierarchisch organisierten) Vermittlungsinstanz kommt desweiteren eine Optimierungsaufgabe zu: Sie muß für eine Anfrage einen (oder mehrere) Dienstanbieter aus der verteilten Angebotsdatenbank auswählen, so daß dabei die Anforderungen, Präferenzen und Spezifikationen beider Seiten möglichst optimal berücksichtigt sind. Dazu eignet sich die *Match-Operation*, bei der eine durch ein CRV-Formular spezifizierte Nachfrage mit einem Angebotsformular verglichen wird. Falls beide (bis auf die verhandelbaren Einträge) zur Deckung gebracht werden können, liefert die Operation eine Maßzahl zurück, die den Deckungsgrad angibt. Das Angebot mit der höchsten Maßzahl wird nun als potentieller Diensterbringer ausgewählt. Hierbei gilt es zu berücksichtigen, daß viele Angaben zeitlich variieren können (z.B. der Nutzungspreis). Desweiteren muß die Vermittlerinstanz im Anschluß daran die Aushandlung des Dienstnutzungsvertrages, d.h. das Fixieren der verhandelbaren Einträge in den CRV-Formularen, initiieren.

Auch während der Nutzungsphase müssen Dienstnutzer in der Lage sein, nach alternativen Dienstangeboten Ausschau zu halten und ggf. die bestehende Nutzung aufzukündigen, so daß sich die Dienstnutzung (Lastverteilung und -umverteilung) der Dynamik im verteilten System und dem Angebot und der Nachfrage auf dem offenen Markt anpassen kann.

3 Das Lastverteilungssystem Dynasty II

Als erster Schritt in Hinblick auf den Einsatz von Marktmechanismen und einem offenen Dienstmarkt wurde im Rahmen dieses Teilprojekts das Lastverteilungs- und Ressourcenmanagementsystem *Dynasty II* entwickelt. Es implementiert marktorientierte Mechanismen für die Zuteilung von Rechenaufträgen an Rechendienste in einem verteilten, heterogenen System, so daß es zu einer möglichst ausgewogenen Nutzung aller verfügbaren Dienste kommt. Darunter fallen etwa datenparallele SPMD-Applikationen. Hierbei startet die Anwendung auf mehreren Rechern denselben Berechnungsdienst, an die es anschließend, um die Berechnungsdauer minimal zu halten, Berechnungsaufträge so verteilen muß, daß alle Rechner ausgewogen an der Berechnung teilnehmen können.

3.1 Das Schichtenmodell

Basierend auf den erörterten Marktmodellen wird das Lastverteilungssystem konzeptionell in drei Schichten eingeteilt:

1. Die **Marktinfrastruktur** (Computational Economy) bietet ökonomisch effiziente verteilte Mechanismen für die Ressourcenallokation, wie Auktionen und Equilibrium-basierte Methoden (z.B. den Tatônnementprozeß). Grundlegende Ressourcen sind zunächst CPU- und Netzbandbreite, sowie Haupt- und temporärer Speicher. Diese Schicht ist desweiteren dafür verantwortlich, Preisstabilität und Allokationsfairneß sicherzustellen und Marktdefekte, wie etwa die Formung von Monopolen, zu verhindern.

2. Die **programmierbaren Marktteilnehmer** sind Agenten, d.h. kleine, aber intelligente, autonome und interagierende Programme. Sie bieten anderen Agenten Dienste an, wobei sie für mögliche Ressourcenbelegungen auf die Marktmechanismen der unterliegenden Schicht zurückgreifen. Gegebenenfalls formen sie dazu Organisationen.

   Die wichtigsten Agententypen sind *Transporter*, die die Kommunikationsinfrastruktur zur Verfügung stellen und anderen Agenten den Transport von Nachrichten und Daten ermöglichen. Spezialisierte *Interface*-Agenten interagieren mit den Frontends, die Bestandteil der nächst höheren Schicht sind, und bieten Berechnungsdienste an, die von den hinter den Frontends gekapselten SPMD-Applikationsinstanzen durchgeführt werden. *Faciliator*-Agenten führen transparente applikationsspezifische Vereinfachungen durch, indem sie beispielsweise komplexe Tasks aufteilen. Sie optimieren desweiteren die Ausführung von Berechnungen mittels Redundanz, indem sie im Falle eines ausreichenden Budgets selbständig eine Berechnung an mehrere Berechnungsdienste gleichzeitig schicken und das zuerst gelieferte Ergebnis verwenden.

   Agenten veröffentlichen ihre Dienstspezifikation (allg. Fähigkeiten), indem sie CRV-Formulare an *Makler*-Agenten schicken, die Teil einer (hierarchischen) Maklerorganisation sind, und die auch für eine effiziente systemweite Verbreitung des Gesamtangebots innerhalb des Maklersystems sorgen. Andererseits führen Makler auf Nachfrage auch Vermittlungen/Angebotsrecherchen durch und helfen damit anderen Agenten, gesuchte Dienste zu lokalisieren und auszuwählen. Durch ausgeglichene Verteilung von Anfragen an die verfügbaren Dienste können Makler Lastbalanzierung bzgl. Dienstnutzung erreichen.

Desweiteren bietet *Dynasty II* für diese Schicht eine Reihe von Mechanismen für die Interaktion zwischen Agenten an, sowie Methoden, die das Aushandeln von Nutzungsverträgen (QoS), die Geldflußsteuerung und die Kostenabrechnung automatisieren.

3. Die oberste Schicht bietet **Frontends zu Parallelen Umgebungen und zu Programmiersprachen** wie etwa PVM, MPI, C/C++, Java und Fortran. Dies erlaubt, *Dynasty II* in existierende parallele (SPMD-)Applikationen ohne größere Anpassungen zu integrieren. Durch Frontends für Programmiersprachen können parallele Anwendungen direkt auf die unteren Schichten zugreifen und haben somit etwa direkten Zugang zur Geldflußsteuerung, Kostenabrechnung und dem ökonomischen Haushalten.

3.2 Lastverteilung in parallelen numerischen Anwendungen

Um das Lastverteilungserhalten von *Dynasty II* in der Praxis zu studieren, wurde zunächst eine Implementation des ARESO-Algorithmus [HS94] gewählt und für *Dynasty II* angepaßt. ARESO dient der numerischen Simulation von struktur- und strömungsmechanischen Problemen. Es basiert auf einem adaptiven Finite-Elemente-Algorithmus, der mittels Gebietszerlegung mit rekursiver Substrukturierung gemäß dem Divide&Conquer-Prinzip das Ausgangsproblem in Teilprobleme zerlegt, die baumartig miteinander gekoppelt sind.

Die Tests wurden mit bis zu 96 Workstations, die über mehrere Cluster verteilt waren, durchgeführt. Die Messungen ergaben einen zufriedenstellenden Speedup, der die Skalierfähigkeit von *Dynasty II* erkennen läßt. Außerdem war die Nutzung der verfügbaren Rechner nahezu ausgeglichen. Das ARESO-Beispiel zeigte auch, daß sich eine einmal entworfene ökonomische Strategie für viele ähnliche Probleme und Netzwerkkonfigurationen wiederverwenden läßt, ohne daß weitere Änderungen in der Applikation und im Frontend durchzuführen sind.

4 Zusammenfassung und Ausblick

Die bisherigen Anstrengungen konzentrierten sich auf die Ausarbeitung des Konzepts für die agenten-basierte, marktwirtschaftlich orientierte Lastverteilungssystem als Grundlage für die Parallelisierung numerischer Anwendungen. Darauf aufbauend wurde das Ressourcenmanagement- und Lastverteilungssystem *Dynasty II* entwickelt, das als experimentelle Ausgangsbasis zur parallelen Implementierung numerischer Algorithmen dient. Die damit bisher erzielten Erfahrungen geben die Richtung zukünftiger Arbeiten vor.

Zunächst gilt es, das Lastverteilungssystem sowohl konzeptuell als auch in der Implementierung zu verfeinern. Konkret sollen die Strategien der Marktinfrastruktur in *Dynasty II* durch umfangreichere Untersuchungen von Marktmechanismen und Allokationsverfahren verbessert werden. Auf der Ebene des offenen Dienstmarkts kann mit Einführung eines ausgereifteren Vertragssystems durch Ressourcenplanung langfristig eine bessere Verteilung erreicht werden. Da numerische Programme oft mehrmals hintereinander gleichartig ablaufen (iterativ), können so die gewonnenen Lastverteilungsinformationen der ersten Läufe in das weitere Ressourcenmanagement einfließen. Ein weiterer zukünftiger Ansatzpunkt ist der Organisationsentwurf und die Organisationsstrukturierung. Denn vorallem in Hinblick auf den Einsatz in großen Workstation-Netzen gilt es, die Dienstvermittlung und Angebotsrecherche der Makler skalierfähig zu gestalten.

Begleitend zu der Ausarbeitung des Lastverteilungssystems werden in Zukunft auch verstärkt parallele numerische Algorithmen für das System implementiert und untersucht. Dies wird vorallem im Rahmen der im folgenden vorgestellten Zusammenarbeiten mit externen Projekten geschehen. Wichtige Aspekte sind dabei z.B., inwiefern sich durch geschickte Wahl der Netztopologie die Prozeßkommunikation beschleunigen läßt und welchen Einfluß die Budgetaufteilung in baumartig gekoppelten Berechnungen, der Marktmechanismus usw. auf den Laufzeitgewinn und die Kommunikationslokalität haben.

5 Zusammenarbeit

5.1 Externe Zusammenarbeit

Um die Entwicklung paralleler Programme und die Automatisierung der Verteilung und des Ressourcenmanagements zumindest für numerische Divide&Conquer-Algorithmen zu verschränken, besteht eine enge Kooperation zu dem Projekt B3 des Sonderforschungsbereichs 342 am Lehrstuhl Prof. Zenger. In diesem Projekt wird die funktionale Datenflußsprache FASAN entwickelt, die es dem Programmierer ermöglicht, unter Verwendung von elementaren Programmodulen (z. B. numerischen Grundalgorithmen) auf einfachere Weise parallele numerische Anwendungen zu erstellen [EPZ97]. Das Ziel der Zusammenarbeit ist, eine komplette Entwicklungs- und Laufzeitumgebung (Parallelisierung und Lastverteilung) zu erarbeiten, um darauf basierend gemeinsam das Verhalten von parallelen numerischen Anwendungen zu studieren und zu optimieren.

Im Rahmen der interdisziplinären Forschingsinitiative FORTWIHR arbeiten Informatiker, Numeriker und Ingenieure gemeinsam an der Simulation technischer Prozesse auf Supercomputern. Die Aktivitäten des Lehrstuhls liegen insbesondere im Bereich der Entwicklung von Verfahren zur Lösung partieller Differentialgleichungen sowie deren effizienter Implementierung auf modernen Parallelrechnern und Workstation-Netzen. Durch die inhaltliche Nähe zu diesen behandelten Themen ist eine Fülle von industrie-relevanten Problemstellungen [HHSZ94] als Kooperationsgrundlage vorhanden.

Mit dem RTB-Bayern (Regionales Testbed Bayern) bestand eine enge Zusammenarbeit auf dem Gebiet der Entwicklung einer verteilten Software-Plattform für große Workstationnetze (München und Erlangen) und moderne Parallelrechner. Das RTB stellte dazu eine große Rechner- und Netz-Infrastruktur zur Verfügung, um wichtige Aspekte des Ressourcenmanagements und der Lastverteilung für eine breite Palette von praxisrelevanten Anwendungen zu untersuchen. Gleichzeitig wurde das für diesen Rahmen angepaßte System für die zu lösenden Fragestellungen seitens der RTB-Projekte verwendet. Hierbei standen vorallem die beim Übergang von einem Local Area Network zu einem Wide Area Network entstehenden Probleme bzgl. Parallisierung und Lastverteilung im Vordergrund.

5.2 Zusammenarbeit im Graduiertenkolleg

Innerhalb des Graduiertenkollegs besteht eine enge Kooperation mit mehreren Teilprojekten. Hinsichtlich der Untersuchung von Marktmechanismen und der agenteninternen Entscheidungsfindung herrscht eine rege Zusammenarbeit mit D. Ormoneit (Teilprojekt: „Kooperierende Agenten in verteilte Systemen – Grundlagen"). Weiterhin sind vorallem die Untersuchungen von F. Fuchs (Teilprojekt: „Kooperierende Agenten und autonome Robotersysteme") bzgl. Verhandlungsmechanismen zur Konfliktlösung in Hinblick auf die in diesem Teilprojekt konzipierten CRV-Formulare und der Aushandlung von Nutzungsverträgen interessant. Die Ergebnisse, die M.-A. Mountzia (Teilprojekt: „Kooperierende Agenten im Netz- und Systemmanagement") aus dem Einsatz von Schablonen im Umgang mit flexiblen Agenten im Systemmanagement gewinnt, werden sicherlich hilfreich für die Weiterentwicklung der CRV-Formulare und der Dienstspezifikation sein. Mit Ch. Röder (Teilprojekt: „Programmentwickung für Parallelrechner und vernetzte Architekturen") besteht ein reger Austausch von Erfahrungen und Erkenntnissen bzgl. des Lastmanagements und der Realisation von Lastverteilungssystemen in Workstation-Netzen. Und schließlich dient die gemeinsame Untersuchung und Erprobung von skalierfähigen Maklerkonzepten und -organisationen zur Dienstvermittlung als Grundlage für eine Zusammenarbeit mit B. Quendt (Teilprojekt: „Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen"). Einzelheiten zu diesen und weiteren Zusammenarbeiten innerhalb des Graduiertenkollegs sind in Kapitel 1.1 im Rahmen der Diskussion der Querschnittsthemen ausgeführt.

Literatur

[Bog94] N. R. Bogan. Economic Allocation of Computation Time with Computation Markets. Technical Report MIT-LCS//MIT/LCS/TR-633, Massachusetts Institute of Technology, Laboratory for Computer Science, August 1994.

[EPZ97] R. Ebner, A. Pfaffinger und C. Zenger. FASAN — eine funktionale Agenten-Sprache zur Parallelisierung von Algorithmen in der Numerik. In W. Mackens und S. M. Rump, Hrsg., *Software Engineering im Scientific Computing, Workshop Hamburg, 1995*. Erscheint bei Vieweg, 1997.

[Fer89] D. F. Ferguson. The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms. *Ph.D. Thesis, Columbia Univ.*, 1989.

[Fra96] Anton Frank. Hierarchisch strukturierte numerische Algorithmen auf Workstation-Netzen. In P.P. Spies, Hrsg., *Graduiertenkolleg: Kooperation und Ressourcenmanagement in verteilten Systemen (Zwischenbericht zum Frühjahr 1996)*, S. 63–67. Institut für Informatik – Technische Universität München, 1996.

[GN95] M. Griebel und T. Neunhoeffer. Parallel Point- and Domain-Oriented Multilevel Methods for Ellipitic PDE's on Workstation Networks. *Erscheint in J. Comp. Appl. Math.*, 1995.

[HHSZ94] W. Huber, R. Hüttl, M. Schneider und C. Zenger. Distributed Numerical Simulation on Workstation Networks. In M. Griebel und C. Zenger, Hrsg., *Nume-*

rical Simulation in Science and Engieneering, Proceedings of the FORTWIHR Symposium oh High Performance Scientific Computing in Munich, June 17-18, 1993, Bd. 48 aus *Notes on Numerical Fluid Mechanics*, S. 67–82. Vieweg Verlag, Braunschweig, 1994.

[HS94] R. Hüttl und M. Schneider. Parallel Adaptive Numerical Simulation. SFB-Bericht 342/01/94 A, Institut für Informatik, TU München, 1994.

[MD88] Mark S. Miller und K. Eric Drexler. Markets and computation: agoric open systems. In B. A. Huberman, Hrsg., *The Ecology of Computation*, S. 133–176. North-Holland Publishing Company, Amsterdam, 1988.

[MLF96] J. Mayfield, Y. Labrou und T. Finin. Evaluation of KQML as an Agent Communication Language. *Lecture Notes in Computer Science*, 1037:347–360, 1996.

[Var95] Hal R. Varian. Economic Mechanism Design for Computerized Agents. In *First USENIX Workshop on Electronic Commerce*, S. 13–21, New York, Juli 1995. USENIX.

[Wel93] Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

1.3.10 Modellprüfung paralleler und verteilter Programmabläufe

Teilprojekt: Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme

Maximilian Frey

1 Einleitung

Modellprüfungsmethoden werden in letzter Zeit immer häufiger bei sicherheitsrelevanten industriellen Anwendung verwendet, um frühzeitig Fehler zu erkennen und damit die Glaubwürdigkeit des Systems zu erhöhen [CGL93, Cat96]. In den letzten 10 Jahren wurden einige Werkzeuge entwickelt, die automatisch überprüfen, ob ein gegebenes Modell eines verteilten und parallelen Systems die Anforderungen an dieses System erfüllen (u.a. [CGL93, Hol91]). Diese Werkzeuge basieren meistens auf einem globalen Modell des Systems bestehend aus allen globalen Zuständen und deren Verbindungen. Dabei können Systeme mit bis zu 10^{120} Zuständen überprüfen werden [CGL93]. Das reicht aus um Systeme in den frühen Phasen der Systementwicklung zu verifizieren, aber nicht zur Überprüfung von Implementierungen etwas größerer Systeme.

Im Gegensatz zur Modellprüfung von Systemen gibt es auf dem Gebiet der Modellprüfung von Abläufen bisher relativ wenige Arbeiten, obwohl dabei die Zustandsexplosion durch unterschiedliche Eingabewerte vermieden wird und durchaus Abläufe von Implementierungen größerer Systeme automatisch überprüft werden können. Modellprüfung von Abläufen kann zwar nicht zur Verifikation, wohl aber zum Testen und zur Fehlerlokalisierung verwendet werden. Die hier vorgestellte Modellprüfungsmethode kann für Systeme eingesetzt werden, die über gemeinsamen Speicher als auch über Nachrichten kommunizieren. Ein System besteht dabei aus mehreren Kontrollflüssen, die parallel ablaufen. Solche Kontrollflüsse werden im folgenden Threads genannt. Threads können in einem gemeinsamen Adreßraum oder in unterschiedlichen Adreßräumen ablaufen. Damit ist auch Verteilung von Threads möglich. Threads können über gemeinsame Variable oder über Nachrichten kommunizieren. Sie können statisch bei Programmstart oder dynamisch während des Programmablaufs erzeugt werden.

Der erste Teil des Berichts beschreibt in den Abschnitten 2, 3 und 4 die Arbeiten an einer auf Kausalnetzen und globalen Zuständen beruhenden Modellprüfungsmethode für parallele und verteilte Programmabläufe. Der zweite Teil des Berichts beschreibt im Abschnitt 5 wie die Modellprüfungsmethode zum Testen sowie zur Lokalisierung von Fehlern und Leistungsengpässen angewendet werden kann.

2 Modellierung von Programmabläufen

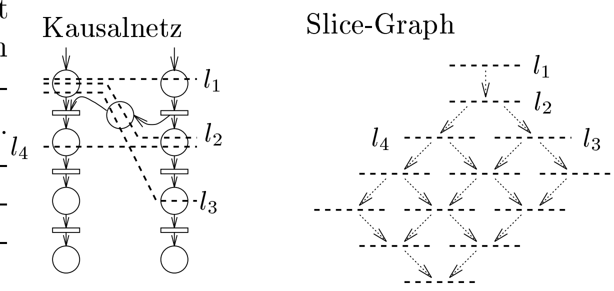
Zur Modellierung von Programmabläufen werden endliche Kausalnetze (u.a. [Rei86]) verwendet. Endliche Kausalnetze beschreiben bipartite Graphen (P, T, R), bestehend aus einer endlichen und nichtleeren Menge P von Plätzen und einer endlichen und nichtleeren Menge von Transitionen T. Der Graph wird durch die Relation R aufgespannt, ist azyklisch und jeder Platz im Graphen besitzt maximal einen Vorgänger und maximal einen Nachfolger.

Unterschiedliche Zustände von Threads eines parallelen oder verteilten Programms sind durch eine nichtleere und endliche Menge von Prädikaten *Pred* gegeben. Diese Prädikate sind spezifisch für je einen Thread. Auf *Pred* ist eine idempotente und bijektive Funktion $\mathfrak{Neg}$ definiert, die zu jedem Prädikat seine Negierung liefert. Zur Unterscheidung von Zuständen von Threads in einem Kausalnetz werden jedem Platz eine Menge von Prädikaten durch eine Funktion $\mathfrak{B}$ zugewiesen, die in diesem Platz gelten. Somit existiert für ein Prädikat p in einem globalen Zustand maximal ein Platz s mit $p \in \mathfrak{B}(s)$ oder $\mathfrak{Neg}(p) \in \mathfrak{B}(s)$.

Globale Zustände werden im Kausalnetz durch Slices dargestellt. Slices sind maximale Teilmengen von paarweise nicht kausal geordneten Plätzen [Rei88].

Der Slice-Graph $(\mathfrak{W}, \mathfrak{R})$ eines endlichen Kausalnetzes besteht aus einer Menge von Slices $\mathfrak{W}$ als Knotenmenge und einer Relation $\mathfrak{R}$, die die Kanten beschreibt. Ein Element aus $\mathfrak{R}$ beschreibt den Übergang von Slices bei der Fortschaltung einer Transition. Jeder Slice-Graph eines endlichen Kausalnetzes enthält genau einen initialen Knoten und genau einen terminalen Knoten und ist somit ein endlicher Verband.

Die nebenstehende Abbildung zeigt ein Beispiel für ein Kausalnetz und dessen Slice-Graph. Im Kausalnetz stellen Kreise Plätze und Rechtecke Transitionen dar. Gestrichelte Linien sind Slices des Kausalnetzes. Durchgezogene Pfeile beschreiben Elemente von R und gepunktete Pfeile stellen Elemente von $\mathfrak{R}$ dar.



3 Temporale Logik

Die zur Spezifikation von zu überprüfenden Eigenschaften verwendete Logik ist aus der Logik von [Rei88] entstanden. Die Syntax der Logik enthält die Prädikate aus *Pred* als atomare Formeln, sowie die Operatoren **and**, **not**, **next**, **sometime** und **until**. Die Logik ist dreiwertig. In einem Slice l kann eine Formel p erfüllt ($l \models p$), unerfüllt ($l \not\models p$) oder undefiniert sein ($l \mathrel{\models\!\!\!\!\!\!\sqcup} p$). Eine Formel kann undefiniert sein, da Prädikate p und Plätze s existieren können, in denen weder $p \in \mathfrak{B}(s)$ noch $\mathfrak{Neg}(p) \in \mathfrak{B}(s)$ gilt. Damit kann modelliert werden, daß eine Variable einer Formel in keinem Zustand eines Slices sichtbar ist. Die Semantik der Logik ist wie folgt definiert: Sei l ein Slice und p, q seien Formeln.

1. $l \models p \in Pred$ falls ein Platz $s \in l$ existiert, so daß $p \in \mathfrak{B}(s)$.
   $l \not\models p \in Pred$ falls ein Platz $s \in l$ existiert, so daß $\mathfrak{Neg}(p) \in \mathfrak{B}(s)$.
   $l \mathrel{\models\!\!\!\!\!\!\sqcup} p \in Pred$ falls für alle Plätze $s \in l$ gilt, daß $p \notin \mathfrak{B}(s)$ und $\mathfrak{Neg}(p) \notin \mathfrak{B}(s)$.
2. $l \models$ **not** (p) falls $l \not\models p$. $l \not\models$ **not** (p) falls $l \models p$. $l \mathrel{\models\!\!\!\!\!\!\sqcup}$ **not** (p) falls $l \mathrel{\models\!\!\!\!\!\!\sqcup} p$.
3. $l \models (p$ **and** $q)$ falls $l \models p$ und $l \models q$. $l \not\models (p$ **and** $q)$ falls $l \not\models p$ oder $l \not\models q$.
   $l \mathrel{\models\!\!\!\!\!\!\sqcup} (p$ **and** $q)$ falls ($l \mathrel{\models\!\!\!\!\!\!\sqcup} p$ und $l \models q$) oder ($l \models p$ und $l \mathrel{\models\!\!\!\!\!\!\sqcup} q$) oder ($l \mathrel{\models\!\!\!\!\!\!\sqcup} p$ und $l \mathrel{\models\!\!\!\!\!\!\sqcup} q$).
4. $l \models$ **next** p falls ein Slice l' existiert, so daß $(l, l') \in \mathfrak{R}$ und $l' \models p$.
   $l \not\models$ **next** p falls für alle Slices l' mit $(l, l') \in \mathfrak{R}$ nicht $l' \models p$ gilt. Nicht $l \mathrel{\models\!\!\!\!\!\!\sqcup}$ **next** p.

5. $l \models$ **sometime** p falls ein Slice l' existiert, so daß $(l, l') \in \mathfrak{R}^+$ und $l' \models p$.
   $l \not\models$ **sometime** p falls für alle Slices l' mit $(l, l') \in \mathfrak{R}^+$ nicht gilt, daß $l' \models p$.
   Nicht $l \models\!\!\!\!\!\uplus$ **sometime** p.

6. $l \models$ $(p$ **until** $q)$ falls ein Slice l' existiert, so daß $(l, l') \in \mathfrak{R}^+$ und $l' \models q$ und für alle Slices l'' mit $(l, l'') \in \mathfrak{R}^+$ und $(l'', l') \in \mathfrak{R}^+$ nicht gilt, daß $l'' \not\models p$.
   $l \not\models$ $(p$ **until** $q)$ falls für alle Slices l' mit $(l, l') \in \mathfrak{R}^+$ und $l' \models q$ gilt, daß ein Slice l'' existiert, so daß $(l, l'') \in \mathfrak{R}^+$ und $(l'', l') \in \mathfrak{R}^+$ und $l'' \not\models p$. Nicht $l \models\!\!\!\!\!\uplus$ $(p$ **until** $q)$

Eine Formel ist in einem Kausalnetz erfüllt, falls sie in allen Slices des Kausalnetzes erfüllt oder undefiniert ist. Sie ist unerfüllt, falls es einen Slice gibt, in dem sie unerfüllt ist.

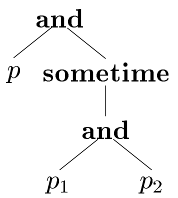
Die Formel (**not** p **or always not** $(p_1$ **and** $p_2)$) spezifiziert beispielsweise, daß von dem Zeitpunkt ab, in dem das erste mal p gilt, niemals p1 und p2 gleichzeitig gelten darf.

4 Modellprüfung

Die Grundidee der Modellprüfung ist die Markierung aller Teilformeln einer Formel mit den Slices, in denen diese Formel erfüllt ist. Das Ziel der Modellprüfung speziell im Hinblick auf Testen und Lokalisierung von Fehlern ist, festzustellen, warum eine Formel und damit eine Eigenschaft nicht erfüllt ist. Um diese Begründung liefern zu können, benötigt man die Slices und Plätze eines Kausalnetzes in denen Teilformeln nicht erfüllt sind. Somit muß im erster Schritt der Modellprüfung die Formel negiert werden.

Zur Berechnung der Slices, in denen eine Formel der Art **not** p erfüllt ist, benötigt man das Komplement der Slices, in denen p unerfüllt bzw. undefiniert ist. Wenn p ein Prädikat ist, kann **not** p durch $\mathfrak{Neg}(p)$ ersetzt werden. Wenn p eine Formel vom Typ **next** (p), **sometime** (p) oder $(p$ **until** $q)$ ist, gibt es keine Slices in denen p undefiniert ist. Um in allen anderen Fällen eine Berechnung der Slices, in denen p undefiniert ist, zu vermeiden, wird f so umgeformt, daß alle Negierungen vor Prädikate geschoben werden und die Komplementbildung nur bei temporallogischen Operationen stattfindet. Dazu wird eine neue logische Basis bestehend aus den Operatoren **and**, **or**, **next**, **allnext**, **sometime**, **always**, **until** und **before** verwendet.

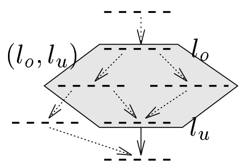
Zur Auswertung der Formel werden für Teilformeln alle Slices, in denen diese Teilformeln erfüllt sind, berechnet. Die Reihenfolge der Berechnung wird durch einen Formelbaum bestimmt, dessen Knoten die Teilformeln enthält. Ein Formelbaum ist ähnlich zum Syntaxbaum und enthält Prädikate in den Blattknoten. Die Söhne eines inneren Knotens im Formelbaum ergeben sich aus dem syntaktischen Aufbau der Formel des Knotens aus Teilformeln und einem Operator.



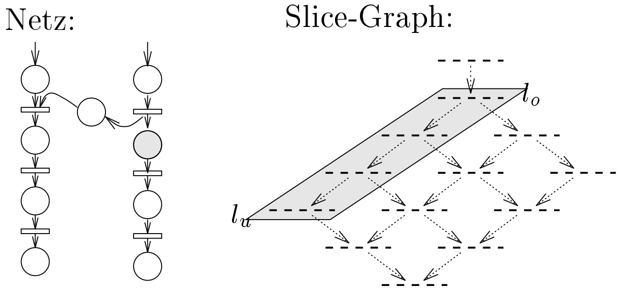
Formelbaum zur Überprüfung der Formel aus Abschnitt 3

Der Formelbaum wird bottom-up ausgewertet, indem die Slices des Knoten aus den Slices der Sohnknoten mit Hilfe einer Funktion abhängig von der Operation des Knotens berechnet werden.

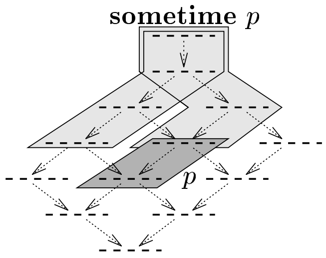
Die Berechnung der Slices kann dadurch effizienter gestaltet werden, daß nicht einzelne Slices berechnet werden, sondern Mengen von Slices gleichzeitig berechnet werden. Mengen von Slices werden durch zwei Slices l_o und l_u repräsentiert. Die Menge (l_o, l_u) besteht aus allen Slices l mit $(l_o, l) \in \mathfrak{R}^*$ und $(l, l_u) \in \mathfrak{R}^*$.



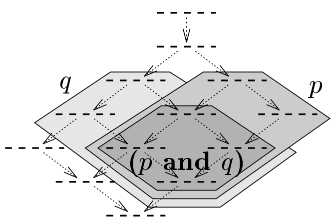
Dadurch können beispielsweise alle Slices die einen Platz s enthalten durch eine Slice-Menge $(lConc(s) \cup \{s\}, gConc(s) \cup \{s\})$ beschrieben werden, wobei $lConc(s)$ die Menge aller Plätze repräsentiert, die nicht kausal geordnet zu s sind und ihre Vorgänger bzgl. R sind kausal geordnet zu s. $gConc(s)$ repräsentiert die Menge aller Plätze, die nicht kausal geordnet zu s sind und ihre Nachfolger bzgl. R sind kausal geordnet zu s.



Die Slice-Mengen von Knoten mit der Formel **sometime** p werden berechnet, indem für alle Slice-Mengen (l_o, l_u) aus dem Sohnknoten und für alle Vorgänger l' von l_u bzgl. $\mathfrak{R}$ die Slice-Mengen (l_I, l') gebildet werden. l_I ist der initiale Slice des Slice-Graphen.



Die Slice-Mengen der Knoten mit Formeln $(p$ **and** $q)$ geschieht durch Schnitt aller Slice-Mengen aus einem Sohnknoten mit allen Slice-Mengen aus dem anderen Sohnknoten.



Falls während der Bottom-up-Auswertung des Formelbaums eine Slice-Menge im Wurzelknoten berechnet wurde, ist die Formel, die durch den Formelbaum überprüft wurde, im Kausalnetz nicht erfüllt. Mit Hilfe dieser Slice-Menge und den Slice-Mengen und Plätzen aus den Knoten des Formelbaums, aus denen die Slice-Menge im Wurzelknoten entstanden ist, kann durch ein Teilnetz dargestellt werden, warum diese Formel nicht erfüllt ist.

Im allgemeinen können Modellprüfungsmethoden in zwei Klassen eingeteilt werden. Tableau-basierte Methoden (u.a. [Win91, GW91]) und globale Methoden (u.a. [CGL93]). Dabei markieren Tableau-basierte Methoden alle Zustände mit Teilformeln, die in dem jeweiligen Zustand erfüllt sind, während globale Methoden Teilformeln einer Formel mit allen Zuständen markieren, in denen die Teilformeln erfüllt sind. Die hier vorgestellter Methode kann zur Klasse der globalen Modellprüfungsmethoden gerechnet werden, da die Teilformeln des Formelbaums mit allen Slices markiert werden, in denen die Teilformel erfüllt ist.

Am ähnlichsten ist die hier beschriebene Modellprüfungsmethode zu [Esp94]. Dort wird nur eine Menge von Slices betrachtet, die durch eine Menge von unteren Slices und eine

Menge von oberen Slices beschrieben wird. Ein Slice ist in dieser Menge, wenn er Vorgänger eines unteren Slices und nicht Vorgänger eines oberen Slices bzgl. $\mathfrak{R}^*$ ist. Diese Methode kann keine Formeln überprüfen, die **next** oder **until** enthalten.

5 Anwendung der Modellprüfungsmethode

Die in Abschnitt 4 dargestellte Methode kann zwar nicht zur Verifikation von Programmen, wohl aber zum Testen und zur Fehlerlokalisierung verwendet werden. Abbildung 1

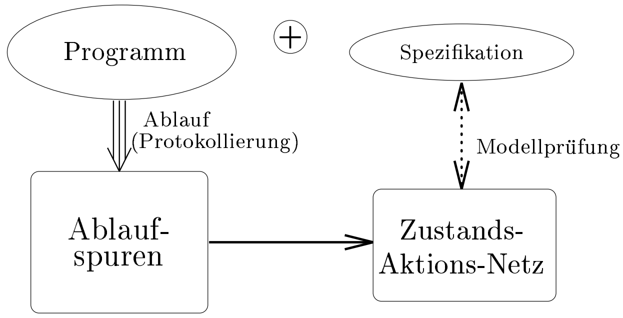


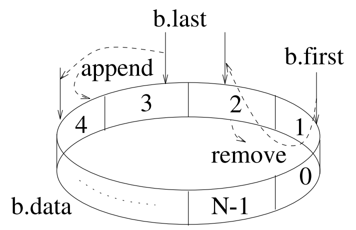
Abbildung 1: Überprüfung von Programmabläufen

beschreibt, wie unter Verwendung der Modellprüfungsmethode automatisch überprüft werden kann, ob die spezifizierten Eigenschaften während eines Programmablaufes erfüllt sind. Dazu sei ein paralleles Programm und die Spezifikation der Eigenschaften in temporaler Logik gegeben. Während eines Ablaufs des parallelen Programms werden Ereignisse, wie beispielsweise lesende und schreibende Zugriffe auf gemeinsame Variablen, Senden und Empfangen von Nachrichten, die Ausführung einer neuen Anweisung des Quellcodes oder das Betreten und Verlassen von Funktionen, in Ablaufspuren aufgezeichnet. Diese Ablaufspuren werden dazu verwendet, ein Kausalnetz zu erzeugen, das als Zustands-Aktions-Netz bezeichnet wird, da seine Knoten spezielle semantische Bedeutung haben. Schließlich wird mit Hilfe der Modellprüfungsmethode überprüft, ob die Spezifikation im Zustands-Aktions-Netz erfüllt ist.

Die Ablaufspuren beschreiben sequentiellen Abhängigkeiten innerhalb von Threads und kausale Abhängigkeiten zwischen Threads, wie beispielsweise zwischen einem lesenden Zugriff und dem letzten vorherigen schreibenden Zugriff auf dieselbe Variable. Dadurch beschreiben die Ablaufspuren eine partielle Ordnung auf den Ereignissen des Programmablaufs. Die Ablaufspuren beschreiben nicht nur den ursprünglichen Programmablauf, sondern eine Klasse von Programmabläufen, da die zufälligen Abhängigkeiten im Programmablauf bis zu einem gewissen Grad eliminiert werden. Aus diesem Grund können Fehler im Programm gefunden werden, die im ursprünglichen Programmablauf zufälligerweise nicht aufgetreten sind.

5.1 Testen und Lokalisierung von Fehlern

Um die Modellprüfungsmethode zum Testen und zur Lokalisierung von Fehlern verwenden zu können, müssen Kausalnetze, die temporale Logik und auch die Methode selbst



Definition der sequentiellen Funktionen remove und append

```
#define N 200
struct queue { int data[N]; int first; int last } b={{0},0,0};

put(int elem) {
  mutual exclusion w.r.t. put:
    wait until b not full;
    append(elem); }
get(int *elem) {
  mutual exclusion w.r.t. get:
    wait until b not empty;
    remove(elem); }
```

Abbildung 2: Ein Ringpuffer mit parallelem Zugriff

erweitert werden. Anhand des Beispiels in Abbildung 2 wird demonstriert wie, die Modellprüfungsmethode zum Testen und zur Fehlerlokalisierung verwendet werden kann. Das Beispielprogramm ist ein paralleles Programm, das einen Ringpuffer implementiert. Der Puffer besteht aus dem Datenbereich *data*, einem Zeiger *first*, der auf das Element im Datenbereich von dem ersten Element des Puffers zeigt und einem Zeiger *last*, der auf das letzte Element in Puffer zeigt. Die sequentielle Funktion *append* fügt nach dem Inkrementieren von *last* ein Element am Ende des Puffers ein. Die sequentielle Funktion *remove* holt nach dem Inkrementieren von *first* das erste Element aus dem Puffer. *put* und *get* sind Funktionen, die parallel ausgeführt werden, und, nachdem sie sich synchronisiert haben, *append* bzw. *remove* aufrufen. Das Programm zeigt in manchen Abläufen das fehlerhafte Verhalten, daß Werte, die von *get* geliefert werden, vorher nicht in den Puffer mit *put* eingefügt wurden. Dieses Verhalten läßt darauf schließen, daß eventuell eine der folgenden Arten von gegenseitiger Ausschluß nicht eingehalten wird:

- Gegenseitiger Ausschluß zwischen unterschiedlichen Threads, die *append* ausführen.
- Gegenseitiger Ausschluß zwischen unterschiedlichen Threads, die *remove* ausführen.
- Gegenseitiger Ausschluß zwischen *append* und *remove*, wenn maximal ein Element im Puffer ist.

Zur Beschreibung von parallelen und verteilten Programmabläufen wird festgelegt, daß jede Transition einer Ausführung einer Anweisung entspricht. Solche Transitionen werden als Aktionen bezeichnet. Andererseits enthalten Plätze nur Thread-lokale Information. So enthalten beispielsweise Plätze die Identifikation eines Threads, die Funktionen die von diesem Thread ausgeführt werden und Werte von Variablen auf die der Thread zugreifen kann. Plätze mit einer solchen Bedeutung werden als lokale Zustände bezeichnet. Ein Kausalnetz, das Aktionen und lokale Zustände enthält, wird als Zustands-Aktions-Netz bezeichnet. Abbildung 3 zeigt ein Zustands-Aktions-Netz, das aus einem Programmablauf des Programms aus Abbildung 2 entstanden ist. Während dieses Ablaufs wurden zwei Threads erzeugt, die *put* und *get* ausführen, wobei *put* den Wert 21 einfügte. Die lokalen Zustände enthalten Informationen über die Funktionen, die gerade ausgeführt werden. Dabei kann durch die Attribute START, IN und TERM unterschieden werden, ob ein Zustand der Anfangszustand der Ausführung, ein Zustand während der Ausführung oder der Endzustand der Ausführung einer Funktion ist. Außerdem enthalten die lokalen Zustände für jede Variable, die in diesem Zustand sichtbar ist, eine Menge von Werten. Eine Variable kann beispielsweise mehrere Werte in einem lokalen Zustand annehmen, wenn diese

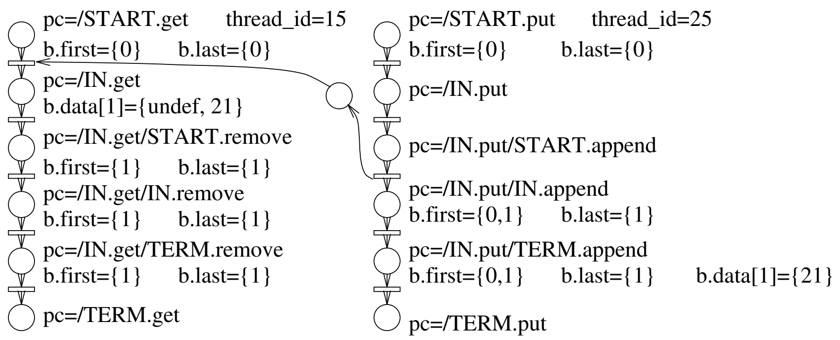


Abbildung 3: Zustands-Aktions-Netz eines Ablaufs des Puffer Beispiels

Variable in einem anderen Thread in einer nicht kausal zum lokalen Zustand geordneten Aktion beschrieben wird. Dann kann die Variable entweder den Wert vor diesem Zugriff oder den Wert nach diesem Zugriff annehmen.

Wenn die Eigenschaften bezüglich gegenseitigem Ausschluß näher betrachtet werden, sieht man, daß die temporale Logik aus Abschnitt 3 in zwei Richtungen erweitert werden muß: Zum einen müssen Prädikate durch Spezifikationen lokaler Zuständen ersetzt werden. Diese Spezifikationen müssen die Abarbeitung von Funktionen sowie Vergleiche von Variablenwerten beschreiben können. Zum anderen müssen Zustände unterschiedlicher Threads unterscheidbar sein. Bei Systemen mit dynamischer Erzeugung von Threads, können diese nicht mehr statisch vor dem Ablauf identifiziert werden. Die Syntax der Logik wird dazu um Platzhalter für Thread-Identifikatoren(TI), genannt TI-Variablen erweitert. Ein Prädikat p enthält dann zusätzlich eine TI-Variable t (t:p). Zur Beschreibung, daß Threads gleich ungleich oder in der Erzeugerrelation stehen, kann spezifiziert werden, daß eine Formel p nur gilt, wenn TI-Variablen t_1, t_2 in Relation zueinander stehen (p **if** $t_1 = t_2$, p **if** $t_1 \neq t_2$, p **if** $t_1 =$ **callerof(**t_2**)**). Die Semantik der Logik ist nicht mehr nur über Slices gegeben, sondern es wird zusätzlich eine Abbildung der TI-Variablen zu TIs des Programmablaufs (TI-Belegung) benötigt.

Zur Beschreibung von lokalen Zustände wird eine Aussagenlogik (lokale Logik) verwendet. Diese Logik enthält als atomare Prädikate Vergleiche von Variablen, **start.**f, **in.**f und **term.**f, wobei **in.**f die Ausführung einer Funktion f und **start.**f und **term.**f den Anfang bzw. das Ende der Ausführung beschreiben. Die Eigenschaften des gegenseitigen Ausschlusses können wie folgt in der Logik beschrieben werden:

- **not**(t1:(in.*append*) **and** t2:(in.*append*)) if t1 != t2
- **not**(t1:(in.*remove*) **and** t2:(in.*remove*)) if t1 != t2
- **not**(t1:(in.*append* and (*b.last-b.first*) modulo N<=1) **and**
  t2:(in.*remove* and (*b.last-b.first*) modulo N<= 1))

Die Modellprüfung ist nun dahingehend zu erweitern, daß zum einen die in den Blattknoten des Formelbaums vorkommenden Formeln der lokalen Logik anstatt der Prädikate in lokalen Zuständen zu überprüft sind. Durch die Erweiterung der Logik um TI-Variablen, wird zusätzlich zur Slice-Menge eine TI-Belegung erzeugt, die der TI-Variablen der Formel den TI des lokalen Zustands zuordnet. Der Formelbaum muß so geändert werden, daß die Mengen von Slice-Mengen in seinen Knoten bezüglich unterschiedlicher TI-Belegungen partitioniert werden. Bei der Berechnung der Slice-Mengen eines Knoten muß dann zuerst überprüft werden, ob die TI-Belegungen von Partitionen der Sohnknoten im Formelbaum zueinander passen. Wenn dies der Fall ist, wird im Knoten eine neue Partition mit der aus den Belegungen der Sohnknoten zusammengemischten Belegung erzeugt und alle Slice-Mengen dieser neuen Partition aus den Slice-Mengen der Partitionen der Sohnknoten wie in Abschnitt 4 beschrieben gebildet.

Zum Testen von parallelen und verteilten Programmen können Anforderungen aus der Programmentwicklung als Spezifikation verwendet werden. Außerdem sollte der Nichtdeterminismus insofern eliminiert werden, daß bei mehreren Abläufen des Programms dieselben Formeln der Spezifikation unerfüllt sind. Das bedeutet, daß Testfälle im Bezug auf Ihren Ausgang wiederholbar sein müssen. Durch die Verwendung von Zustands-Aktions-Netzen wird der Nichtdeterminismus zwar reduziert, aber noch nicht völlig eliminiert. So kann unterschiedliches Zugriffsverhalten auf Synchronisationsobjekte zu unterschiedlichen Zustand-Aktions-Netzen führen. Zur Eliminierung dieser Art des Nichtdeterminismus wird in Zusammenarbeit mit dem SFB 342 Teilprojekt A1 eine Testumgebung entwickelt [FO97].

Falls ein Fehler beim Testen aufgetreten ist, erleichtert die Beschreibung und mögliche Visualisierung von Gründen, warum eine Formel nicht erfüllt ist, die Formulierung von Hypothesen über die Ursache eines Fehlers. Insofern kann die Methode auch zu einer inkrementellen Annäherung an den Fehler genutzt werden. Am Ende dieser Annäherung an den Fehler steht dabei immer die eindeutige Lokalisierung des Fehlers, so daß eine Eliminierung möglich ist (siehe [Mye79, FW93, FW94b, FW94a, Fre96a]).

5.2 Lokalisierung von Leistungsengpässen

Zum Erkennen von Leistungsengpässen wurden in den letzten Jahren immer mehr Werkzeuge zur Visualisierung (u.a. [Abs90, BHL90, HJ91]) entwickelt. Diese Werkzeuge erlauben es sehr einfach und schnell Leistungsengpässen sowie deren Ursachen in parallelen und verteilten Systemen zu erkennen. Die Lokalisierung von Leistungsengpässen speziell bei einer relativ feingranularen Parallelität und die schrittweise Einkreisung der Ursachen des Leistungsengpässen im Quellcode einer Anwendung gestaltet sich dagegen oftmals schwierig. Die Einkreisung geschieht in denselben Schritten wie bei der Lokalisierung von Fehlern. Insofern kann bei geeigneter Anpassung die Modellprüfungsmethode auch zur Lokalisierung von Leistungsengpässen verwendet werden. Die Eigenschaften deren Nichterfüllung zu einem Leistungsengpaß führt, beschreiben oftmals obere bzw. untere Grenzen mit Hilfe von Meßgrößen für Leistungsmerkmale, wie die Ablaufzeit, die Anzahl der Kommunikation, Anzahl der Synchronisationen, usw. Beispielsweise kann beschrieben werden, daß zwischen zwei bestimmten Programmzuständen die Anzahl der Kommunikationen um das 3-fache höher ist als im Mittel. Gerade die Beschreibung der Programmzustände liefert dabei einen wichtigen Hinweis für die Lokalisierung eines Leistungsengpasses (siehe [Fre95b, Fre95a, And96, Fre96b]).

6 Abschließende Bemerkungen

In diesem Bericht wurde eine Modellprüfungsmethode für parallele und verteilte Programmabläufe beschrieben. Diese Methode verwendet Kausalnetze zur Beschreibung von

Programmabläufen. Die Logik zur Beschreibung von Eigenschaften basiert auf [Rei88]. Wichtige Eigenschaften der Modellprüfungsmethode sind, daß kein globales Modell gebildet werden muß, sondern es werden nur die globalen Zustände erzeugt, die zur Überprüfung der Eigenschaften unbedingt nötig sind. Eine weitere Verringerung des Aufwands ist die Zusammenfassung von globalen Zuständen zu Mengen, die durch einen obersten und untersten globalen Zustand in der Ablaufhierarchie gegeben sind. Dadurch kann erreicht werden, daß Formeln, die nur die Operatoren **and**, **or** und **always** enthalten, in einer Zeitkomplexität linear in der Größe des Kausalnetzes und exponentiell in der Länge der Formel überprüft werden können. Weiterhin wurde beschrieben, wie die Modellprüfungsmethode zum Testen und zur Lokalisierung von Fehlern und Leistungsengpässen eingesetzt werden kann. Die Implementierung der Modellprüfung ist noch nicht abgeschlossen. Die Komponenten zur automatischen Erzeugung des Kausalnetzes aus einem Programmablauf sind bereits implementiert. Die Überprüfungsmethode muß noch implementiert werden.

Zusammenarbeit innerhalb des Graduiertenkollegs ergab sich aufgrund der thematischen Nähe in den Querschnittsthemen „Verifikation“ bezüglich der Modellprüfungsmethode und ihrer Unterschiede zu anderen Methoden, „Spezifikation und Programmiersprachen“ bezüglich der Definition der Logik und „Charakteristika von Ressourcen“ bezüglich der Definition von Meßgrößen zur Beschreibung von Leistungsengpässen. Zusammenarbeit außerhalb der Graduiertenkollegs ergab sich am Lehrstuhl mit dem Forschungsprojekt „Parallele Programmierung in verteilten Systemen (ParMod)“ im Bereich Instrumentierung und Performance-Analyse von parallelen und verteilten Systemen, mit dem Bremer Institut für sichere Systeme, Technologiezentrum Informatik der Universität Bremen im Bereich Spezifikation von Eigenschaften mit temporaler Logik und im Bereich Modellprüfung verteilter und paralleler Systeme, und schließlich mit dem SFB 342 „Werkzeuge und Methoden zur Nutzung Paralleler Rechnerarchitekturen“ Teilprojekt A1: „Integration von Werkzeugumgebung und Rechnerarchitektur zur Parallelisierung“ im Bereich Testen paralleler und verteilter Systeme.

Literatur

[Abs90] F. Abstreiter. Visualizing and Analyzing The Runtime Behavior of Parallel Programs. In H. Burkhart, Hrsg., *CONPAR 90 – VAPP IV*, LNCS 457, S. 828–839. Springer, Berlin, 1990.

[And96] M. M. Andani. Profiling-Techniken für die Lastermittlung paralleler Programme. Diplomarbeit, Institut für Informatik, Technische Universität München, München, August 1996.

[BHL90] Th. Bemmerl, O. Hansen und Th. Ludwig. PATOP for Performance Tuning of Parallel Programs. In H. Burkhart, Hrsg., *CONPAR 90 – VAPP IV*, LNCS 457, S. 840–851. Springer, Berlin, 1990.

[Cat96] T. Cattel. Using concurrency and formal methods for the design of safe process control. In I. Jelly, I. Gorton und P. Croll, Hrsg., *Software Engineering for parallel and Distributed Systems*, S. 183–194, London, März 1996. IFIP, Chapman & Hall.

[CGL93] E. Clarke, O. Grumberg und D. Long. Verification Tools for Finite-State Concurrent Systems. In J.W. de Bakker, W.-P. de Roever und G. Rozenberg, Hrsg., *A Decade of Concurrency, REX School/Symposium, Noordwijkerhout, The Netherlands*, LNCS 457, S. 840–851. Springer, Berlin, Juni 1993.

[Esp94] J. Esparza. Model Checking Using Net Unfoldings. *Science of Computer Programming*, 23:151–195, 1994.

[FO97] M. Frey und M. Oberhuber. Testing Parallel and Distributed Programs with Temporal Logic Specification. to appear, 1997.

[Fre95a] M. Frey. Performance Debugging of Parallel Programs with Temporal Logic Specifications. In *Proceedings of the 21th EUROMICRO Conference, Como, Italy*, S. 170–178. Euromicro, IEEE, September 1995.

[Fre95b] M. Frey. A Temporal Logic Language for Performance Debugging of Parallel Programs. In R. Rodošek, Hrsg., *Proceedings of the 4th Int. Conf. on Verification In New Orientation, Maribor, Slowenia*, S. 80–103. Universität Maribor, Technischer Bericht 03/95, Juni 1995.

[Fre96a] M. Frey. Debugging Parallel Programs using Temporal Logic Specifications. In I. Jelly, I. Gorton und P. Croll, Hrsg., *Software Engineering for parallel and Distributed Systems*, S. 122–133, London, März 1996. IFIP, Chapman & Hall.

[Fre96b] M. Frey. Verwendung temporallogischer Spezifikationen zur Lokalisierung von Fehlern und Leistungsengpässen in parallelen Programmen. In P.P. Spies, Hrsg., *Graduiertenkolleg Kooperation und Ressourcenmanagement in verteilten Systemen, Zwischenbericht zum Frühjahr 1996*, S. 46–50, München, 1996. Institut für Informatik. Technische Universität München, Technischer Bericht TUM-I9611.

[FW93] M. Frey und A. Weininger. Using Temporal Logic Specifications to Debug Parallel Programs. *Microprocessing and Microprogramming*, 39:97–100, 1993.

[FW94a] M. Frey und A. Weininger. A Temporal Logic Language for Debugging Parallel Programs. In *Proceedings of the 20th EUROMICRO Conference, Liverpool, England*, S. 170–178. Euromicro, IEEE, September 1994.

[FW94b] M. Frey und A. Weininger. Using Specifications for Debugging Parallel Programs. In *Proceedings of the Int. Conf. on Applications in Parallel and Distributed Computing, Caracas, Venezuela*. IFIP WG 10.3, April 1994.

[GW91] P. Godefroid und P. Wolper. A Partial Approach to Model Checking. In *6th annual IEEE Symposium on Logic in Computer Science, Amsterdam*, S. 406–416, Los Alamitos, Juni 1991. IEEE Computer Society Press.

[HJ91] M.T. Heath und J.A.Etheridge. Visualizing the Performance of Parallel Programs. *IEEE Software*, 5(1):56–65, 1991.

[Hol91] G.J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[Mye79] G.J. Myers. *The Art of Software Testing*. Wiley, New York, 1979.

[Rei86] W. Reisig. *Petrinetze*. Springer, Berlin, 1986.

[Rei88] W. Reisig. Temporal Logic and Causality in Concurrent Systems. In *Concurrency 88*, LNCS 335, S. 121–139. Springer, Berlin, 1988.

[Win91] G. Winskel. A Note on Model Checking the Modal μ-Calculus. *Theoretical Computer Science.*, 83(1):157–167, 1991.

1.3.11 Steuerung kooperativer paralleler Theorembeweiser unter Ausnutzung von Ähnlichkeit

Teilprojekt: Verteilte Inferenzsysteme
Marc Fuchs

1 Zusammenfassung

Das Thema „Steuerung kooperativer paralleler Theorembeweiser unter Ausnutzung von Ähnlichkeit" ist dem Teilprojekt „Verteilte Inferenzsysteme" zugeordnet. Ziel des Themas ist die Entwicklung von Techniken zur Unterstützung paralleler, kooperativer Beweissysteme. Zwar sind durch die Entwicklung kooperativer Systeme, z.B. durch den Austausch von erzeugten Zwischenergebnissen (Fakten) zwischen den beteiligten Beweisern, synergetische Effekte und damit superlineare Speedups möglich. Jedoch ist zum Erreichen solcher Effekte ein kontrollierter Austausch *relevanter* Informationen nötig, da Fakten in riesiger Menge erzeugt werden. Somit sind Kriterien zu entwickeln, die es erlauben, die Relevanz erzeugter Fakten zur Lösung der Beweisaufgabe einzuschätzen. Insbesondere *Ähnlichkeitskriterien* zwischen noch offenen Beweiszielen und erzeugten Fakten können als Auswahlkriterien herangezogen werden und stehen im Mittelpunkt der Untersuchung dieses Projekts.

Das Thema wird erst seit September 1996 bearbeitet. In diesem Zeitraum wurde zunächst ein denkbarer a posteriori Ähnlichkeits- bzw. Relevanzbegriff einer Menge von Fakten zu einer Menge von Beweiszielen aufgestellt. Zusätzlich wurden Ansätze denkbarer Techniken zur a priori Ähnlichkeitsabschätzung entwickelt. Als erster konkreter Schritt wurde eine Auswahlkomponente von Fakten für die Zusammenarbeit des Beweisers SETHEO [LSBB92] mit dem Lemmagenerator DELTA [Sch94] konzipiert und implementiert. Durch erste Ergebnisse im Bereich des Gleichheitsbeweisens konnte nachgewiesen werden, daß die Möglichkeit der Auswahl relevanter Fakten aufgrund von Ähnlichkeitskriterien erfolgversprechend ist.

2 Einleitung

Automatische Deduktion im allgemeinen und automatisches Theorembeweisen im besonderen haben in den vergangenen Jahren zunehmende praktische Bedeutung erlangt. Automatisches Theorembeweisen dient der Lösung von Beweisproblemen (Ax, Th): Es soll festgestellt werden, ob eine gegebene Zielformel Th eine logische Folgerung einer gegebenen Menge von Formeln Ax ist. Eine oft verwendete Logik, in der Ax und Th formuliert sind, ist dabei die Prädikatenlogik 1.Stufe (PL1). Automatische Beweisverfahren finden inzwischen Einsatz in interaktiven Beweisumgebungen wie z.B. ILF [DGH$^+$94] und wurden auch schon in Anwendungen, z.B. zur Verifikation von Kommunikationsprotokollen erfolgreich eingesetzt (s. z.B. [Sch96]).

Trotz der großen Fortschritte im Bereich des automatischen Beweisens in den vergangenen Jahren und der daraus resultierenden verbesserten praktischen Einsetzbarkeit, verbleibt jedoch das grundlegende Problem der Beweissuche, nämlich die generelle Unentscheidbarkeit, die zur Bearbeitung eines in der Regel unendlich großen Suchraums führt. Aus diesem Grund sind Maßnahmen zur Behandlung dieser Suchraumproblematik von wesentlichem Interesse. Ein wichtiges Mittel zur Erhöhung der Leistungsfähigkeit existierender Beweiser ist die Konstruktion paralleler Versionen, die in der Vergangenheit erfolgreich durchgeführt wurde ([Sut95], [Sch95]). Eine wesentliche Bedeutung für die Lösung schwierigerer Probleme besitzt dabei die Konstruktion kooperativer Systeme, in der verschiedene Beweiser z.B. über den Austausch von Zwischenergebnissen (Fakten) zusammenarbeiten. Das grundlegende Problem einer solchen Zusammenarbeit stellt jedoch die große Zahl der erzeugten Fakten dar: ohne geeignete Filter- und Auswahlmechanismen relevanter Fakten ist eine sinnvolle Kooperation undenkbar. Hieraus motiviert sich das Ziel des Projekts, geeignete Konzepte zur Bestimmung der Relevanz von Fakten zu entwickeln und damit einen kooperativen Beweiser zu unterstützen. Zentral ist dabei der Begriff der *Ähnlichkeit* zwischen erzeugten Fakten und offenen Beweiszielen, der zur Abschätzung der Relevanz erzeugter Fakten verwendet werden soll.

Der vorliegende Bericht gliedert sich folgendermaßen: In Kapitel 3 wird Theorembeweisen mit Modellelimination näher vorgestellt, wobei insbesondere auf die Probleme der Entwicklung kooperativer paralleler Systeme eingegangen wird. In Kapitel 4 wird die zentrale Fragestellung dieser Arbeit eingeführt: Zum einen wird ein erster möglicher Ähnlichkeitsbegriff definiert, zum anderen werden allgemeine Konzepte zur Bewertung von Ähnlichkeit vorgestellt, die im weiteren Verlauf dieses Projekts realisiert werden sollen. Durch erste Experimente wird aufgezeigt, daß eine Faktenauswahl mittels Ähnlichkeitskriterien erfolgversprechend ist. In Kapitel 5 wird ein Ausblick auf die zukünftige Arbeit gegeben. Schließlich endet der Bericht mit Bemerkungen zur Kooperation mit externen Forschungseinrichtungen in Kapitel 6.

3 Paralleles Theorembeweisen mit Modellelimination

3.1 Modellelimination und Suche

Modellelimination (ME) ist ein vollständiger und korrekter Kalkül für PL1, der als Verfeinerung des Tableaukalküls aufgefaßt werden kann (s. auch [Lov78]). ME operiert mit Klauseln, also mit Disjunktionen von Literalen. Die Beweissuche mit ME ist üblicherweise als Aufzählungsverfahren für Klauseltableaux (also Bäume, deren Knoten mit Literalen markiert sind) realisiert. Wir beschränken uns im folgenden auf das Problem nachzuweisen, ob ein Literal Th eine logische Folgerung einer Klauselmenge Ax ist. Ausgehend von einem initialen Tableau T_0, das aus dem Negat von Th besteht, werden systematisch alle aus T_0 mit Ax konstruierbaren Tableaux aufgezählt. Zur Erzeugung eines Tableau ausgehend von einem gegebenen Tableau stehen dabei verschiedene Inferenzregeln (Extension, Reduktion) zur Verfügung (s. [Lov78]). Die Aufzählung möglicher Tableaux und damit möglicher Beweise wird so lange fortgesetzt, bis ein sogenanntes geschlossenes Tableau gefunden ist, d.h. ein Tableau, dessen sämtliche Äste geschlossen sind, also komplementäre Literalpaare enthalten. Im Verlauf der Tableaukonstruktion vorhandene Blätter von noch offenen Ästen eines Tableau werden auch als *Subziele* bezeichnet. Zur systematischen Erzeugung aller Tableaux muß also ein (üblicherweise unendlich großer) ODER-Suchbaum $\mathcal{T}$ abgearbeitet werden, der alle aus dem initialen Tableau durch unterschiedliche Reihenfolgen von Inferenzregelanwendungen konstruierbaren Tableaux enthält. Die Herleitung eines geschlossenen Tableau läßt sich auch als sukzessive Lösung der in einem Tableau vorhandenen

Subziele deuten.

Um den Suchbaum $\mathcal{T}$ abzuarbeiten, wird i.d.R. kein Breitensuchverfahren verwendet, da sowohl die Zahl, als auch die Größe der zu verwaltenden Objekte (Tableaux) zu stark ansteigt und damit eine explizite Speicherung impraktikabel ist. Stattdessen werden implizite Verfahren angewendet, die darauf beruhen, iterativ immer größer werdende, endliche Ausschnitte von $\mathcal{T}$ in einem Tiefensuchverfahren zu durchsuchen. Die endlichen Suchraumausschnitte werden dabei durch Struktureinschränkungen an die erzeugten Tableaux festgelegt, sogenannte *Ressourceschranken* $\mathcal{B}$, die bestimmen, ob noch Inferenzen mit einem erzeugten Tableau erlaubt sind, oder nicht.

Ein grundlegendes Problem der Beweissuche stellt der vorhandene Indeterminismus in jedem Knoten des Suchbaums $\mathcal{T}$ dar, der zu großen Suchräumen führt. Schwierigere Probleme können so i.d.R. mit herkömmlichen Aufzählungsverfahren nicht mehr in akzeptabler Zeit gelöst werden. Aus diesem Grund sind Verbesserungen der herkömmlichen Suchmethoden (Aufzählungsverfahren), u.a. in Form von Parallelisierungskonzepten, von großem praktischem Interesse im Bereich des automatischen Beweisens.

3.2 Paralleles Theorembeweisen

Parallele Theorembeweiser lassen sich grundsätzlich in zwei Klassen einteilen: *partitionierende* und *wettbewerbsparallele* Systeme. Partitionierende Systeme werden durch das Aufteilen des ODER-Suchbaums auf verschiedene Rechner realisiert. Jeder beteiligte Beweiser operiert auf einem anderen Teil des Suchbaums. Dadurch entstehen voneinander unabhängige Beweisaufgaben: falls einer der beteiligten Beweiser einen Beweis (geschlossenes Tableau) erzeugen kann, ist eine Lösung des Gesamtproblems erreicht. Alternativ dazu ist eine Realisierung wettbewerbsparalleler Systeme möglich: hierbei erhält jeder beteiligte Beweiser die komplette Beweisaufgabe, jedoch unterscheidet sich jeweils das verwendete Aufzählungsverfahren des Suchraums. Mit beiden dieser Verfahren sind superlineare Speedups möglich.

Parallelisierung bietet jedoch zusätzlich die Möglichkeit der *Kooperation*. So können sowohl partitionierende, als auch wettbewerbsparallele Systeme durch Austausch von Informationen *während* des Beweislaufs miteinander kooperieren. Kooperative Systeme bieten zusätzlich Möglichkeiten zur Effizienzverbesserung, indem durch Austausch von Daten Synergieeffekte entstehen.

Kooperation ist prinzipiell auf zwei Arten zu realisieren: Zum einen können *Steuerungsinformationen* ausgetauscht werden, zum anderen ist ein Austausch von *Subzielinformationen* möglich, also von Informationen über während der Beweissuche aufgetretene lösbare und unlösbare Subziele. Beschränken wir uns auf den Austausch von Informationen über Subziele, so lassen sich diese Informationen z.B. für *Lemmamechanismen* nutzen. Zur Realisierung von Lemmamechanismen werden positive (gültige) Fakten eines anderen im kooperativen Beweissystem beteiligten Beweisers in den eigenen Suchzustand integriert. Das wesentliche Ziel ist es dabei, die Beweislänge zu reduzieren, indem Fakten direkt zum Abschluß von Subzielen eines offenen Tableau genutzt werden können und der Beweis des Faktums nicht noch einmal (redundant) wiederholt werden muß. Das Problem solcher Lemmamechanismen besteht aber nun darin, daß mit zusätzlichen Fakten zwar Verkürzungen der Beweislänge möglich sind, andererseits aber die Verzweigungsrate des Suchraums bei

Hinzugabe vieler Fakten i.d.R. sehr stark zunimmt, da viele zusätzliche Inferenzen mit den Fakten möglich werden.

Grundsätzlich kann die Feststellung getroffen werden, daß Austausch von erzeugten Fakten ohne eine geeignete Filterung und Auswahl der Fakten kein sinnvoller Ansatz ist. Somit zielt das Projekt ganz wesentlich darauf ab, geeignete Auswahlkriterien für Fakten zu entwickeln. Mögliche Auswahlkriterien für Fakten sind eine geeignet erscheinende syntaktische Struktur (z.B. geringe Länge) oder ein großer Aufwand für die Herleitung. Eine wesentliche Steigerung der Auswahlrelevanz ist zu erwarten, wenn bei der Faktenauswahl das zu beweisende Ziel miteinzubezogen wird. Die Entwicklung von Ähnlichkeitskriterien zwischen dem Beweisziel und vorhandenen Fakten steht somit im Vordergrund der Realisierung von Lemmamechanismen. Im folgenden wird die dazu bereits erfolgte Arbeit beschrieben.

4 Ähnlichkeit zwischen Teilzielen und Fakten

Wie zuvor beschrieben sollen Ähnlichkeitskonzepte zur Bewertung von generierten Fakten eingesetzt werden. Im folgenden sei folgende Ausgangssituation gegeben: Das Beweisproblem sei (Ax, Th) und zusätzlich sei eine Menge von Fakten F gegeben (logische Folgerungen aus Ax), aus der eine geeignete Teilmenge $L \subseteq F$ ausgewählt werden soll, die zusätzlich zum Beweis herangezogen wird. Die Objekte, zwischen denen Ähnlichkeit festgestellt werden soll, sind also eine Menge von Subzielen (Literalen) und eine Menge von Fakten.

4.1 Begriffsbestimmung

Analog zu anderen Gebieten der KI, in denen Ähnlichkeit eine große Rolle spielt (z.B. fallbasiertes Schließen), läßt sich auch in der hier betrachteten Domäne eine eindeutig definierte a posteriori Ähnlichkeit angeben, die a priori abgeschätzt werden soll. So sind z.B. bei Problemlösungsaufgaben im Rahmen des fallbasierten Schließens Probleme ähnlich, die eine gleiche Lösung besitzen, was durch Ähnlichkeit der Problembeschreibungen abgeschätzt wird. Analog dazu ist eine Ähnlichkeit zwischen einer Zielklausel Th und einer Faktenmenge L gegeben, wenn die Verwendung von L zu einer Lösung von Th mit möglichst wenig Inferenzen führt, was alleine durch Betrachtung von Th und L abgeschätzt werden soll.

Insgesamt läßt sich eine (i.d.R. asymmetrische) Ähnlichkeit zwischen einer Literalmenge (Klausel) Th und einer Menge von Fakten L in folgender Weise definieren (zur Vereinfachung im Fall von Hornklauseln): Sei ein (endlicher) Suchbaum $\mathcal{T}$ (von Tableaux) gegeben. Th und L heißen ähnlich ($Sim_{\mathcal{T}}(Th, L)$) genau dann, wenn ein Beweis (geschlossenes Tableau) für Th mit $L' \subseteq L$, $L' \neq \emptyset$ existiert. Im Rahmen von konkreten Anwendungen bestimmt sich, wie ein solcher Suchbaum $\mathcal{T}$ gegeben ist. So ist für Lemmamechanismen i.d.R. interessant, ob für eine gegebene Zielformel Th innerhalb eines (durch eine Ressource $\mathcal{B}$ und der Axiomatisierung Ax und L bestimmten) endlichen Suchraumausschnitts $\tilde{\mathcal{T}}$ die Beziehung $Sim_{\tilde{\mathcal{T}}}(Th, L)$ gilt. Eine üblicherweise verwendete Ähnlicher-Beziehung zwischen Objekten, also z.B. Faktenmenge L_1 ist ähnlicher zu Th als L_2, kann durch die Anzahl A_1 bzw. A_2 der bzgl. eines gegebenen Suchverfahrens $\mathcal{S}$ bis zum Auffinden eines Beweises von Th mit L_1 bzw. L_2 aufgezählten Tableaux definiert werden. So ist L_1 ähnlicher zu S als L_2, falls $A_1 < A_2$ gilt. Da eine Abschätzung von A_1 bzw. A_2 nicht praktikabel ist, kann als mögliche Abschwächung die Ähnlicher-Beziehung aufbauend auf den für die jeweiligen Beweise mindestens benötigten Suchressourcen abgeschätzt werden. Aufgrund des üblicherweise exponentiell wachsenden Suchraums bei Erhöhung eines Ressourcewerts ist eine Annäherung obiger Ähnlicher-Relation denkbar, indem zum Vergleich der Eignung zweier Faktenmengen L_1 und L_2 nur noch untersucht wird, mit welcher Faktenmenge ein Beweis auch mit einer kleineren Ressourceschranke gefunden werden kann.

4.2 Lemmamechanismen und Ähnlichkeit

Um Lemmamechanismen aufbauend auf einem solchen Ähnlicher-Begriff realisieren zu können, muß eine Faktenmenge $L \subseteq F$ gefunden werden, die zum einen ähnlicher zum Ziel als andere Faktenmengen $L' \subseteq F$ ist und zum anderen nach Möglichkeit eine minimale Menge mit dieser Eigenschaft ist. Obwohl also prinzipiell alle Teilmengen von F auf Ähnlichkeit zu Th zu untersuchen sind, wurde bisher der Versuch unternommen, sich auf die isolierte Betrachtung der Ähnlichkeit jeweils eines $\lambda \in F$ zu Th zu beschränken. Die Grundidee besteht dabei darin abzuschätzen, ob ein $\lambda \in F$ bzgl. einer gegebenen Ressourceschranke $\mathcal{B}$ auf einem Ast eines geschlossenen Tableau für Th liegen kann, wobei in dem Tableau auch andere Fakten aus F vorkommen dürfen. Ausgehend von solchen Bewertungen kann dann ein Lemmamechanismus z.B. einen gewissen Prozentsatz k der Fakten aus F mit den günstigsten Bewertungen auswählen und diese Fakten zur Lösung des Beweisproblems hinzuziehen.

Im folgenden soll eine ungefähre Idee der Verfahren gegeben werden, die bisher verwendet wurden: Es wurde bisher der Ansatz gewählt mit einer vorgegebenen (kleinen) Ressourceschranke das Ziel Th auf genau diejenigen Tableaux $T_1, \ldots, T_n$ zurückzuführen, die jeweils nur noch ein offenes Subziel S_i, $1 \leq i \leq n$, besitzen. Diese mit wenigen Ressourcen und damit wenig Aufwand erstellten Tableau können üblicherweise nicht mit einem Faktum aus F abgeschlossen werden. Dennoch geben sie in Form der offenen Subziele einen Anhaltspunkt für die Gestalt der Ziele (von strukturell komplexeren) Tableaux, die mit einem Faktum aus F abgeschlossen werden können. Zur Bestimmung der Ähnlichkeit eines $\lambda \in F$ zu Th wird deshalb bislang ein Maß aufbauend auf „Unifikationsabständen" zwischen λ und den Zielen $S_1, \ldots, S_n$ verwendet. Dazu wird im wesentlichen die „Verschiedenheit" der einer Unifikation eines Ziels S_i und λ im Wege stehenden Teilterme bewertet. Ausgehend von unterschiedlichen Bewertungen dieser Teilterme wurden zwei verschiedene Ähnlichkeitsmaße entwickelt.

Tabelle 1 zeigt einige mit solchen Techniken erzielten Ergebnisse. Dabei wurde der ME-Beweiser SETHEO und der Lemmagenerator DELTA verwendet, mit dessen Hilfe für jedes Problem eine Faktenmenge F erstellt wurde.[10] Spalte 1 zeigt den Namen des Problems aus der Problembibliothek TPTP [SSY94], einer standardisierten Sammlung von Beweisaufgaben im Bereich der PL1. In den Spalten 2 und 3 finden sich die Laufzeiten von SETHEO ohne Zugabe der logischen Folgerungen F aus der Axiomatisierung (Fakten), die DELTA erzeugt hat, bzw. mit Zugabe aller von DELTA erzeugten Fakten. (Alle Laufzeiten auf

[10] SETHEO und DELTA wurden am Lehrstuhl von Prof. Jessen, also im Umfeld des vorliegenden Projekts entwickelt.

Bsp.	Setheo	Setheo/Delta	Setheo-Sim1	Setheo-Sim2	Setheo-Synt
GRP010-4	33s	52s	5s	4s	139s
GRP168-1	184s	17s	23s	>1000s	$> 1000s$
HEN003-5	>1000s	691s	>1000s	11s	11s
BOO003-1	35s	19s	<1s	<1s	<1s
BOO003-2	344s	>1000s	28s	129s	377s
BOO003-4	10s	36s	2s	6s	12s
BOO004-2	497s	>1000s	105s	22s	628s
BOO004-4	16s	39s	2s	5s	20s
BOO005-2	629s	>1000s	34s	49s	110s
BOO005-4	23s	56s	5s	5s	5s
BOO006-2	387s	>1000s	36s	47s	42s
BOO006-4	141s	503s	5s	5s	5s

Tabelle 1: Experimente mit Lemmamechanismen im Bereich PL1 (mit Gleichheit)

einer Sun Ultra II.) Die Spalten 4 und 5 zeigen die Ergebnisse von zweien nach obigen Techniken entwickelten Auswahlverfahren (Setheo-Sim1, Setheo-Sim2). Es wurden jeweils 25% der von DELTA erzeugten Fakten ausgewählt und der ursprünglichen Beweisaufgabe hinzugegeben. Schließlich finden sich in Spalte 6 die Ergebnisse mit einer auf syntaktischen Kriterien beruhenden Auswahl von ebenfalls 25% der erzeugten Fakten. Insgesamt kann also ein deutlicher Effizienzgewinn bei Verwendung von Ähnlichkeitskriterien im Vergleich zu dem Standardbeweiser und den anderen Lemmamethoden beobachtet werden. Insbesondere zeigt sich, daß eine ungefilterte Zugabe von Fakten (Setheo/Delta) nur in wenigen Fällen (z.B. GRP168-1) zu Erfolgen führt, da der Suchraum zu stark anwächst. Auch eine Faktenfilterung beruhend auf einfachen syntaktischen Kriterien ist wenig erfolgversprechend, da die Relevanz von Fakten i.d.R. nicht korrekt eingeschätzt werden kann. Eine verbesserte Realisierung von Lemmamechanismen ist durch die Verwendung wettbewerbsparalleler Systeme mit unterschiedlichen Filtermechanismen zu erreichen.

5 Ausblick

Für die Zukunft ist eine Weiterarbeit im Bereich von Verfahren, die auf dem Messen von Unifikationsabständen beruhen, geplant. So sollen die bisher eingesetzten heuristischen Kriterien um theoretisch fundiertere Methoden erweitert werden, die ausgehend von Ax und gegebenen Differenzen zwischen Zielen und Fakten Aussagen erlauben, ob eine solche Differenz behoben werden kann. Zum anderen sind Verfahren zu entwickeln, die überprüfen, ob ein Subziel und ein Faktum auf einem Ast eines geschlossenen Tableau liegen können.

Die entwickelten Verfahren sind zu implementieren und Komponenten zum Messen von Ähnlichkeit zu entwickeln. Die danach anstehende Arbeit betrifft den Bereich der Anwendung der Verfahren in parallelen Beweisern, wie z.B. dem im Rahmen des SFB 342 entwickelten kooperativen parallelen Beweiser CPTHEO. Anwendungen sind für Faktenauswahl (Lemmamechanismen), als auch zur Suchsteuerung oder Task-/Beweiszielauswahl denkbar.

6 Externe Zusammenarbeit

Externe Zusammenarbeit besteht mit dem SFB 342 an der TU München, als auch mit der Universität Kaiserslautern. Im Rahmen des SFB 342 besteht eine enge Zusammenarbeit mit dem Teilprojekt A5. Die entwickelten Verfahren zum Messen von Ähnlichkeit können, wie oben erwähnt, innerhalb der im SFB entwickelten Beweiser eingesetzt werden. Zudem besteht die Möglichkeit des Erfahrungsaustauschs über die Entwicklung paralleler Beweiser. Eine ähnliche Zusammenarbeit besteht auch mit dem DFG-Projekt „Cooperation in Heterogeneous Theorem Prover Networks" in Kaiserslautern. Auch hier ist ein Einsatz der entwickelten Ähnlichkeitskriterien in dem in Kaiserslautern entwickelten Beweisernetz geplant.

Literatur

[DGH+94] B. I. Dahn, J. Gehne, Th. Honigmann, L. Walther und A. Wolf. *Integrating Logical Functions with ILF.* Preprint, Humboldt University Berlin, Department of Mathematics, 1994.

[Lov78] D. W. Loveland. *Automated Theorem Proving: a Logical Basis.* North-Holland, 1978.

[LSBB92] R. Letz, J. Schumann, S. Bayerl und W. Bibel. SETHEO: A High Performance Theorem Prover. *Journal of Automated Reasoning*, (8), 1992.

[Sch94] J. Schumann. DELTA - A Bottom-Up Preprocessor for Top-Down Theorem Provers. System Abstract. In *Proceedings of CADE-12.* Springer, 1994.

[Sch95] J. Schumann. *SiCoTHEO - Simple Competitive parallel Theorem Provers based on SETHEO.* Technical Report, Department of Computer Science, Munich University of Technology, 1995.

[Sch96] J. Schumann. *Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study.* Technical Report, Department of Computer Science, Munich University of Technology, 1996.

[SSY94] C. B. Suttner, G. Sutcliffe und T. Yemenis. The TPTP Problem Library. In *Proceedings of CADE-12.* Springer, 1994.

[Sut95] C. B. Suttner. *Static Partitioning with Slackness.* PhD thesis, Department of Computer Science, Munich University of Technology. 1995.

1.3.12 Ressourcenmanagement in verteilten Systemen

Teilprojekt: Konstruktion heteromorph paralleler Systeme
Sascha Groh

1 Verteiltes Ressourcenmanagement

Vernetzte Workstations bieten ein großes Potential an Ressourcen, wie zum Beispiel eine hohe Rechenleistung, um Anforderungen komplexer paralleler Anwendungen (Simulationen, numerische Probleme etc.) zu erfüllen. Es ist Aufgabe verteilter Betriebssysteme, diese Ressourcen transparent und effizient verfügbar zu machen. Die Vorteile von Workstationclustern liegen in ihrer relativen Preiswertigkeit im Vergleich zu Parallelrechnern, ihrer Geschwindigkeit, ihrer höheren Verfügbarkeit und nahezu beliebiger Skalierbarkeit. Diese Vorteile sind aber nur relevant, wenn es möglich ist, Workstationcluster auch einfach für parallele Anwendungen nutzen zu können. Hierzu ist eine effiziente und für den Benutzer transparente Ressourcenverwaltung notwendig.

Bekannte verteilte Betriebssysteme (z.B. [DCM+90, BCE+94, RAB+90, RDH+96]) sind größtenteils nach dem Client-Server-Modell aufgebaut, welches auch immer häufiger in den Betriebssystemen von Einzelplatzrechnern zu finden ist. Die Grundlage bildet ein Mikrokern, der die Hardware veredelt (und von Hardwaredetails abstrahiert), sowie mehrere Serverprozesse, die entweder Teilaufgabe des Ressourcenmanagement übernehmen, oder besondere Anwendungsdienste anbieten. Hierbei werden jedem Server bestimmte Ressourcen zur Verwaltung übergeben, normalerweise nach Ressourcenklassen aufgeteilt, wie z.B. Speicherverwaltung durch einen Pager. Bei der Erweiterung auf verteilte Systeme werden die Fähigkeiten der Server um Funktionalitäten zur Überbrückung der physikalischen Verteiltheit erweitert, wie z.B. einem Distributed-Shared-Memory- (DSM-) Pager. Die Funktionalität des Servers ändert sich hierdurch zum Teil nur marginal. In dem genannten Beispiel stehen zusätzlich zum Paging auf den Hintergrundspeicher auch Funktionen für das Pagen auf andere Rechner zur Verfügung. Der größere Aufwand bei diesen Erweiterungen ist bei der Verwaltung der Information über die aktuelle Rechnerzuordnung einer Speicherseite zu leisten. Diese Information muß auf jeder Workstation vorhanden sein, um auf die Speicherseite zugreifen zu können.

Der Nachteil dieser Vorgehensweise liegt in der Inflexibilität der einzelnen Verwaltungseinheiten [CKK95, KPA+93]. Jeder Server ist entweder nur mit grobgranularen, statischen Verwaltungsstrategien ausgestattet, die es ihm erlauben, Anforderungen aller Anwendungen innerhalb des Systems zu bearbeiten, oder aber er ist auf eine Anwendung spezialisiert und reagiert nicht oder nur wenig auf Einflüsse anderer Anwendungen. Für eine effiziente Ressourcenverwaltung in verteilten Systemen ist aber sowohl die Anwendungsanpassung, d.h. das Eingehen der Ressourcenverwaltung auf spezifische Bedürfnisse der Anwendung, wie zum Beispiel Kooperationsbeziehungen zwischen Anwendungsteilen, als auch eine kooperative, anwendungsübergreifende, globale Ressourcenverwaltungsstrategie notwendig. Ohne Anwendungsanpaßbarkeit ist es der Ressourcenverwaltung unmöglich, auf Abhängigkeiten innerhalb einer Anwendung Rücksicht zu nehmen, und auf diese Weise die Effizienz zu steigern. Ohne die Kooperation ist es möglich, daß die Ressourceverwaltungsserver widersprüchliche Optimierungsentscheidungen treffen, die zwar jeweils aus ihrer lokalen Sicht optimal und richtig sind, aber auf das gesamte System bezogen zu einer schlechten Ressourcenausnutzung führt.

Diese bestehenden Probleme sollen mit dem in diesem Teilprojekt verfolgten Ansatz gelöst werden, indem jeder Anwendung, die sich in dem Workstationcluster in Ausführung befindet, durch eine Vielzahl von unterschiedlichen abstrakten Manageragenten verwaltet wird. Jeder einzelne Manageragent ist auf Anwendungsteile zugeschnitten und somit in der Lage für die Anwendung die geeigneten Ressourcen zu finden und zu belegen. Durch die Kooperationsfähigkeit der Manageragenten ist es möglich, auch globale Optimierungsziele bei den Verwaltungsentscheidungen zu berücksichtigen.

2 Anwendungsorientiertes Ressourcenmanagement

In dem in diesem Teilprojekt verfolgten Ansatz wird das verteilte System in seiner Gesamtheit betrachtet. Das bedeutet, daß unter dem abstrakten Begriff **verteiltes System** jedes Objekt innerhalb des System, unabhängig davon, ob es sich um reales oder ein abstraktes Objekt handelt, verstanden wird. Bei den realen Objekten handelt es sich um die einzelnen Hardwareobjekte, welche innerhalb des Workstationclusters vorhanden sind. Beispiele hierfür sind die Speicherzellen oder die Prozessoren der einzelnen Workstations, sowie die Verbindungsnetze. Die betrachteten Hardwareobjekte können in unterschiedliche Abstraktionsebenen eingeteilt werden. So ist es möglich eine Speicherzelle zu betrachten (dies entspricht der feinsten Abstraktionsebene, *in der Abbildung Ebene* 1) oder eine Speicherseite, die ihrerseits aus einer Menge von Speicherzellen besteht. Eine Speicherseite betrachtet als Objekt des verteilten Systems befindet sich auf einer höheren Abstraktionsebene als die Speicherzellen, durch die sie realisiert wird. Dieser Abstraktionsschritt läßt sich fortsetzen, bis alle Hardwareobjekte zu einem einzigen Objekt zusammengefaßt sind, der verteilten Hardware als Ganzes (*in der Abbildung Ebene* m).

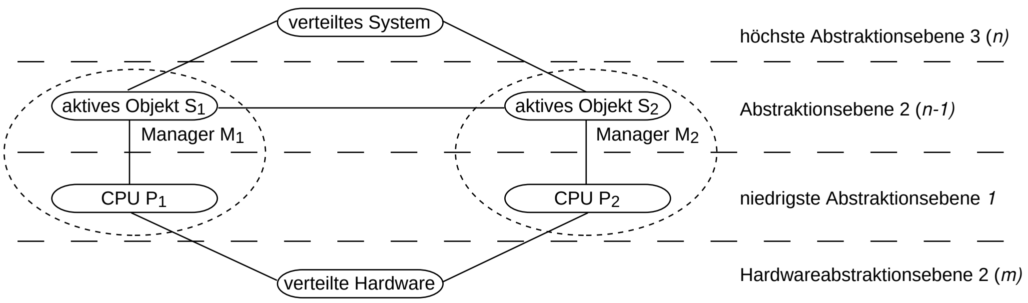


Abbildung: *Abstraktionsebenen, vertikale und horizontale Ressourcen, Managerzuordnung*

Ein entsprechende Vorgehensweise kann auf die abstrakten Objekte der im verteilten System laufenden Anwendungen angewendet werden. Top-Down betrachtet stellt das verteilte System die höchste Abstraktionsebene dar (*in der Abbildung Ebene* n). Es beinhaltet alle abstrakten Anwendungsobjekte und alle Objekte des Ressourcenmanagements incl. der Hardwareobjekte. Auf tieferen Abstraktionsebenen befinden sich Objekte, die einzelne Anwendungen repräsentieren, oder auf noch tieferen Ebenen Objekte, die einzelne Teile der Anwendung repräsentieren, wobei es sich sowohl um aktive Objekte handeln kann, d.h. um Objekte, die Berechnungen durchführen, als auch um passive Objekte, d.h. um Objekte, die Daten aufnehmen können. Die tiefste Abstraktionsebene bilden die feingranularen

Hardwareressourcen, die von den abstrakten Objekten benötigt werden, um die Aufgaben, die an sie gestellt werden, erfüllen zu können.

Die Aufgaben des Ressourcenmanagements besteht nun darin, die verteilten System-Objekte unter Berücksichtigung der vorhandenen Hardwareressourcen zu konkretisieren. Dabei hat das Management auf die Eigenschaften und Anforderungen der einzelnen Objekte einzugehen und diese zu erfüllen. Diese Aufgabe wird in dem hier vorzustellenden Ansatz in mehreren Schritten durchgeführt. Der erste Schritt ist ein Modell zu entwickeln, das es ermöglicht, die Aufgabe exakter zu beschreiben. Dazu müssen die Eigenschaften der Objekte und die Anforderungen, die sie an andere (abstrakte oder reale) Objekte stellen, formal beschrieben werden (Abschnitt 3). Aufbauend auf diese formale Beschreibung der Anforderungen besteht der nächste Schritt darin, diese Anforderungen mithilfe von abstrakten Verwaltungseinheiten durchzusetzen (Abschnitt 4). Der letzte Schritt ist die Realisierung dieser abstrakten Verwaltungseinheiten (Abschnitt 5).

3 Modellierung des Ressourcenmanagements

Wie in Abschnitt 1 erläutert wurde, ist es in einem verteilten System notwendig, die Ressourcenverwaltung an die Anwendung anzupassen. Im folgenden wird unter einer Anwendung ein paralleles Programm in Ausführung verstanden, welches aus vielen parallelen Teilberechnungen (Aktivitäten genannt) besteht. Der Grad der Parallelität kann dabei während der Laufzeit schwanken. Die einzelnen Aktivitäten sind in der Lage, auf vielfältige Arten einen Informationsaustausch vorzunehmen. Daraus entstehen unterschiedlichste Abhängigkeitsbeziehungen zwischen den einzelnen Aktivitäten, die es bei der Ressourcenverwaltung zu berücksichtigen gilt. Diese Abhängigkeiten formal in einem Modell zu beschreiben und für die Verwaltung nutzbar zu machen, ist Inhalt dieses Abschnittes.

3.1 Charakteristika von Ressourcen

Die Grundlage für den verfolgten Ansatz bildet eine differenzierte Betrachtung von Ressourcen. Ressourcen werden in vielen Ansätzen gleichgesetzt mit Hardwareressourcen, wie z.B. Prozessoren oder Hauptspeicherzellen. Den Managementsystemen stehen dementsprechend nur wenig, rein hardware-orientierte Information zur Verfügung, was zu einer ineffizienten Verwaltung führt (z.B. [BS93]). Bei anwendungsorientierten Ansätzen ist das Problem ähnlich gelagert. Es steht zwar zusätzlich Information von der Anwendung zur Verfügung, die Verwaltung beschränkt sich aber auf Hardwareressourcen, die der Anwendung zugeteilt worden sind. Diese eingeengte Sicht der Ressourcen wird im folgenden aufgebrochen. Einem Top-Down-Ansatz folgend können verschiedene Realisierungsebenen in ein verteiltes System eingezogen werden. Die höchste Realisierungebene bildet das abstrakte Gesamtsystem (*in der Abbildung Ebene n*). Seine Aufgabe ist es, die an das System gestellten Aufgaben zu lösen. Hierzu bedient es sich anderer Objekte, im Normalfall aktiver Objekte (Aktivitäten). Die Aufgabe dieser Objekte (Ressourcen genannt), die somit wiederum eine neue Abstraktionsebene (*in der Abbildung Ebene $n-1$*) bilden, ist es, Teilaufgaben zu lösen, wobei sich die einzelnen Teilaufgaben stark voneinander unterscheiden können. Aber allen Objekten auf diesen Abstraktionsebenen ist gemein, daß sie wiederum auf andere Objekte angewiesen sind, um ihre Aufgaben erfolgreich erledigen zu können. Eine allgemeine

Definition für Ressourcen, auf ein Objekt S bezogen, lautet wie folgt:

Definition (Ressourcen) Die Ressourcen eines Objekts S sind alle Objekte, die als Hilfmittel zur Durchführung der Aufgaben von S benötigt werden.

Die benötigten Objekte können in zwei Ressourcenkategorien unterteilt werden. Auf der einen Seite kann ein Objekt S_2 eine **horizontale Ressource** darstellen, wenn es sich auf derselben Abstraktionsebene befinden, wie Objekt S_1, welches S_2 benötigt (*in der Abbildung ist S_2 horizontale Ressource von S_1*). Ein Beispiel für eine horizontale Ressource ist eine Aktivität, die als Kommunikationspartner für einen Nachrichtenaustausch dient. Auf der anderen Seite kann S_2 eine **vertikale Ressource** darstellen, wenn es sich auf einer tieferen Abstraktionsebene als das Objekt S_1 befindet, in diesem Fall spricht man von einer (Teil-) Konkretisierung oder Realisierung des Objektes S_1 durch das Objekt S_2 (*in der Abbildung ist z.B. P_1 eine vertikale Ressource von S_1*). Dabei kann es sich um beliebige andere Objekte des Systems handeln, einschließlich Hardwareressourcen. Alle vertikalen Ressourcen eines abstrakten Objektes beschreiben das Objekt vollständig auf einer tieferen Abstraktionsebene. Wiederholt man den Konkretisierungsschritt rekursiv, passiert man unterschiedlichste Abstraktionsebenen, bis die Rekursion schließlich bei den Hardwareressourcen endet, die die tiefste Abstraktionsebene darstellen.

Mit der gegebenen Definition ist es möglich, von jedem Objekt, das als Ressource betrachtet wird, spezielle Eigenschaften herauszuarbeiten. Diese Eigenschaften können nach folgenden Kriterien unterschieden werden. Zum ersten gibt es Eigenschaften, die ein Objekt von anderen Objekten auf derselben Abstraktionebene unterscheidet. Zum zweiten gibt es Eigenschaften, die das Objekt anderen Objekten auf höheren Abstraktionsebenen zusichert und die somit bei seiner Realisierung erhalten bleiben müssen. Und zum dritten gibt es die Eigenschaft, die Aussagen darüber macht, ob das Objekt für andere Objekte zur Realisierung zur Verfügung steht. Anhand dieser Eigenschaften wird es möglich, Ressourcen in Klassen zu unterteilen. Die dabei relevanten Eigenschaften, die zu der Klassenbildung führen, sind von dynamischer Natur und kontextabhängig. Im nächsten Abschnitt wird ein Beschreibungsmodell vorgestellt, mit dem der relative Ressourcenkategorienbezug dargestellt wird, und das als Grundlage für die Klassenbildung dient.

3.2 Formales Beschreibungsmodell für ein Ressourcenmanagement

Bei dem verwendeten Beschreibungsmodell handelt es sich um ein Graphersetzungssystem [van90, SW92]. Wie bereits in Abschnitt 3.1 erwähnt, ist es von entscheidender Bedeutung, daß das Beschreibungsmodell in der Lage ist, alle für das Management relevanten Eigenschaften (u.a. Abhängigkeitseigenschaften und Realisierungseigenschaften) vollständig zu erfassen. In einem Graphersetzungssystem steht hierzu die Beschreibungsvielfalt der Graphen zur Verfügung. Im einzelnen handelt es sich dabei um die Knoten von Graphen, die die Objekte des Systems beschreiben, und um die Kanten, die die Abhängigkeits- und Teile der Realisierungseigenschaften widerspiegeln. Alle anderen Eigenschaften können durch Attribute repräsentiert werden, die den Kanten und den Knoten beigestellt werden. Somit beschreibt ein Graph den Zustand des Systems zu einem gewissen Zeitpunkt; er ist ein „Schnappschuß“ des dynamischen Systems.

Welche Eigenschaften für das Management relevant sind, wird durch die Graphersetzungsregeln festgelegt. Die Regeln geben wieder, welche Bedingungen vorliegen müssen, um eine Managementfunktionalität einsetzen zu können, und wie sich der Systemzustand durch diese Maßnahme verändert. Beide Seiten einer Ersetzungsregel beschreiben eine Ressourcenklasse. Auf der einen Seite ist die Ressourcenklasse beschrieben, welche vorliegen muß um eine Regel einsetzen zu können. Die Grundlage für das Erkennen einer Ressourcenklasse und somit dem Erkennen der Regelanwendung ist durch die Vorbedingungen beschrieben, die mit der Regel geknüpft sind. Eine Vorbedingung ist zum Beispiel, daß ein Objekt, welches eine auslagerbare Speicherseite darstellt, die Eigenschaft der Nichtbenutzung in einer gewissen Periode erfüllen muß. Wenn eine Speicherseite diese Eigenschaft hat, so bildet sie mit anderen Speicherseiten, die diese Eigenschaft ebenfalls besitzen, eine Ressourcenklasse. Auf alle Objekt in dieser Klasse ist eine Regel anwendbar, die diese Klasse als Vorbedingung fordert. Die Zugehörigkeit zu dieser Ressourcenklasse ändert sich bei dem ersten Zugriff auf diese Speicherseite, weswegen von einer dynamischen Klassenbildung gesprochen werden muß. Dieses Beispiel macht auch ersichtlich, daß neben der Vorbedingung noch eine zweite Voraussetzung für eine Regelanwendung im Beschreibungsmodell vorgesehen werden müssen, da meist mehrere Objekte eine Klasseneigenschaft erfüllen. Für die Auswahl zwischen anwendbaren Regeln und damit Managementfunktionalitäten sind zu jeder Graphersetzungsregel Strategien zu spezifizieren. Eine ausführlichere Darstellung des zugrundeliegenden Graphersetzungssystem findet sich in [Gro96a, Gro96b].

4 Verteiltes Ressourcenmanagement mit Manageragenten

Mit den Vorarbeiten aus Abschnitt 3.2 ist es möglich, ein anwendungsorientiertes Ressourcenmanagement zu konstruieren. Die Grundlage bilden die Abstraktionsebenen und die Ressourceneigenschaften. In diesem Abschnitt wird ein Realisierungsmodell vorgestellt, das in der Lage ist, die Ressourcen verteilt zu verwalten und dabei die Eigenschaften der Ressourcen, die für die Klassenzuordnung notwendig sind, zu berücksichtigen. Im nächsten Abschnitt wird aufgezeigt wie das Realisierungsmodell verwirklicht und dabei der Overhead für das Management so gering wie möglich gehalten werden kann.

4.1 Manageragenten

Das in Abschnitt 3.2 skizzierte Graphersetzungssystem beschreibt formal die Managementvorgänge in einem verteilten System. Bei der Realisierung des Ressourcenmanagements müssen die beschriebenen Aufgaben auf Ausführungseinheiten übertragen werden. Dabei ist auf die physikalische Verteiltheit Rücksicht zu nehmen. Demzufolge müssen die Ausführungseinheiten, sowie die Information für diese verteilt und die Zuständigkeiten gegeneinander abgegrenzt werden.

Jede dieser Einheiten muß in der Lage sein, autonom Entscheidungen treffen zu können, um einem potentiellen Flaschenhals bei der Verwaltung entgegenzuwirken. Dies ist nur möglich, wenn die Aufgaben im System verteilt werden. In diesem Realisierungsmodell wird zur Entflechtung der Aufgaben ein Ansatz gewählt, bei dem jeder Aktivität ein Manager beigestellt wird (*in der Abbildung wird z.B. Manager M_1 dem Objekt S_1 beigestellt*).

Dieser abstrakte Manager hat die Aufgabe, der Aktivität alle für die Berechnung notwendigen Ressourcen bereitzustellen und zu verwalten. Diese Aufgabe kann in zwei wesentliche Bereiche aufgegliedert werden. Zum einen besteht für den Manager eine Verantwortung für alle Objekte, die von der Aktivität zur Konkretisierung verwendet werden, die also vertikale Ressourcen für die Aktivität sind (*in der Abbildung ist z.B.* M_1 *für* P_1 *verantwortlich*). Zum anderen muß er dafür Sorge tragen, daß die Aktivität in der Lage ist, auf Objekte zuzugreifen, die von anderen Aktivitäten konkretisiert wurden, als auch mit anderen Aktivitäten zu kommunizieren (*in der Abbildung ist* M_1 *für die Kommunikation mit* S_2 *verantwortlich*). Innerhalb des Graphersetzungssystem bedeutet dies, daß jeder abstrakte Manager für einen Untergraphen verantwortlich ist. Dabei ist in jedem Untergraphen genau eine abstrakte Aktivität enthalten, womit jeder Manager genau ein Objekt auf seiner höchsten Abstraktionsebene enthält und seine Aufgabe darin besteht, dieses Objekt mit vertikalen Ressourcen zu realisieren (vertikale Ressourcenverwaltung) und die Zugriffe auf dieses Objekt, oder seine Konkretisierungsobjekte zu ermöglichen, sowie Zugriffe dieser Objekte auf andere Objekte zu ermöglichen (horizontale Ressourcenverwaltung).

Um diesen Aufgaben gewachsen zu sein, wird der Manager mit Fähigkeiten ausgerüstet, die aus Agentensystemen [WJ95, HCK96] bekannt sind, weswegen in diesem Zusammenhang auch der Begriff **Manageragenten** verwendet wird. Jeder Manageragent ist in der Lage, autonom Entscheidungen für sein Aufgabengebiet zu fällen. Da er aber, wie schon erwähnt, nicht für alle Ressourcen in einem System verantwortlich ist, sondern nur für die von seiner Aktivität konkretisierten, muß er mit anderen Manageragenten kooperieren, um seiner zweiten Aufgabe, die Zugriffe auf gemeinsame Ressourcen (incl. Hardwareressourcen) zu ermöglichen, zu erfüllen. Die Grundlage für die Kooperationsfähigkeit stellt die Kommunikationsfähigkeit dar. Jeder Manageragent ist in der Lage, mit anderen Manageragenten zu kommunizieren und sich von diesen Information über das System zu besorgen. Diese prinzipielle Kommunikationsfähigkeit kann aber aus Effizienzgründen bei der Realisierung eingeschränkt werden, wie in Abschnitt 5 beschrieben wird.

Eine weitere Fähigkeiten der Manageragenten ist die Adaptionsfähigkeit, zusammen mit der Mobilität. Jeder Manageragent ist in der Lage, sich den wechselnden Bedingungen seiner Umwelt anzupassen. In einem laufenden System bedeutet dies, daß jeder Manageragent nicht mit starren Konzepten und Verfahren an seine Umgebung herantritt, wie zum Beispiel Standard- (fixen) Speicheranforderungen, sondern sich den Gegebenheiten, also den zur Verfügung stehenden Ressourcen, anpaßt. Auf diese Weise wird es möglich, das Flexibilitätspotential des Graphersetzungssystems auf die Realisierung zu übertragen.

Eine weitere Eigenschaft des Manageragentensystem ist seine beliebige Skalierbarkeit. Die Anzahl der Manageragenten ist nach oben nicht beschränkt. Für jede neue Aktivität in dem verteilten System wird ein neuer Manageragent generiert. Beendet eine Aktivität ihre Berechnung und liegen alle Voraussetzungen für deren Auflösung vor, so wird auch der Manageragent aufgelöst. Auf diese Weise schwankt die Zahl der Manageragenten, die die Ressourcenverwaltung in dem verteilten System durchführen, stark über die Lebenszeit des Systems.

4.2 Verteiltes Management

Das Ressourcenmanagement wird in dem verteilten System durch das kooperierende Manageragentensystem durchgeführt. Möglichen Konflikten wird durch eine Hierarchie begegnet. Diese Hierarchie entwickelt sich entsprechend der Dynamik des Systems. Jede Aktivität wird von einer bereits existierenden Aktivität (Elternaktivität) generiert. Demzufolge wird auch jeder Manageragent von einem existierenden Manageragent (Elternmanager) generiert. In Konfliktsituationen wird diese Hierarchie dazu verwendet, um die Konflikte zu lösen.

Die Hardwareressourcen werden in diesem System ebenfalls durch die Manageragenten verwaltet. Am Anfang liegen diese Ressourcen gebündelt in der Hand weniger Anfangsmanageragenten. Über die Laufzeit ändert sich dies mit steigender Anzahl von Manageragenten. Jeder Manageragent verwaltet gemäß dem Modell alle Hardwareressourcen, die zu seiner Aktivität vertikale Ressourcen sind. Dabei handelt es sich um die Ressourcen, die seine Aktivität benötigen, um ihre Aufgaben erledigen zu können (an dieser Stelle wird von Ausnahmen, die durch ein Delegierungskonzept entstehen und von den Ressourcen des Managers, abgesehen). Darüber hinaus kann er auch für die Verwaltung von freien potentiellen vertikalen Ressourcen, die für die Konkretisierung von Objekten verwendet werden können, verantwortlich sein. Den Überblick über weitere vorhandene Ressourcen gewinnt er durch Kommunikation und Kooperation mit anderen Manageragenten. Dieser Überblick muß nicht vollkommen sein, sondern es reicht, wenn jeder Manageragent gerade die zur Erfüllung seiner Aufgaben notwendige Information hat. Hierbei ist es für jeden Manageragent nützlich, daß er die Information über seine Aktivität besitzt. Jeder Manageragent wird bei der Generierung auf die Bedürfnisse seiner Aktivität zugeschnitten. Auf diese Weise wird erreicht, daß er sein erstes Ziel, die Ressourcenanforderungen seiner Aktivität zu erfüllen, gerecht wird.

Aber neben den Anforderungen seiner Aktivität muß sich der Manageragent auch übergeordneten Optimierungs- und Ressourcenverteilungskriterien unterordnen. Ohne die Hilfe von anderen Manageragenten wird es ihm nicht möglich sein, seiner Aktivität Zugriffe auf horizontale Ressourcen bereitzustellen oder freie Ressourcen von anderen Manageragenten zu erhalten, wenn er diese als vertikale Ressourcen benötigt. Ein Zugriff auf ein entferntes Objekt läuft über den Manageragenten, dessen Aktivität das Objekt als vertikale Ressource enthält. Durch Verhandlungen zwischen den beiden Manageragenten wird ein Weg gesucht, dem Anforderer den Zugriff, unter Berücksichtigung des Systemzustandes, zu ermöglichen. Auf demselben Weg kann ein Manageragent freie Ressourcen, die er zur Realisierung seiner Objekte benötigt, von anderen Manageragenten erhalten. Eine Einigung zwischen zwei Manageragenten wird immer erzielt, entweder auf direktem Weg, da ein Manageragent in der Hierarchie höher (z.B. der Elternmanageragent) steht und damit die Entscheidung fällt, oder indem ein Manageragent angerufen wird, der in der Hierarchie höher steht als beide verhandelnden Manageragenten und er somit Entscheidungen über beide Manageragenten fällen kann. Vorteilhaft ist bei diesen Verhandlungen, daß jeder Manageragent Information, die als Attribute und Kanten in dem Graphen enthalten sind, über seine Aktivität besitzt, und es auf diese Weise möglich wird, die Verhandlungen an den Anwendungsanforderungen auszurichten. Mithilfe dieser Verhandlungen wird es einem Manageragenten möglich für das System günstige Entscheidungen zu treffen und somit seinen Beitrag zum

Ressourcenmanagement des verteilten Systems beizusteuern.

5 Die Realisierung der Manageragenten

Im letzten Abschnitt wurde das Manageragentenmodell vorgestellt. Es erlaubt eine anwendungsangepaßte Ressourcenverwaltung unter Berücksichtigung der Systemressourcen. Um dieses Modell sinnvoll einsetzen zu können, ist eine effiziente Realisierung notwendig. Dabei besteht die Hauptaufgabe in der Realisierung der Manageragenten. Durch eine differenziertere Betrachtung der Aufgaben der Manageragenten wird es möglich, unterschiedlichste Realisierungsformen für diese zu verwenden und auf diese Weise den Overhead für das Ressourcenmanagement so klein wie möglich zu halten.

Jeder Manageragent hat abstrakt dieselben Aufgaben. Betrachtet man diese aber etwas genauer, so öffnet sich ein weites Spektrum an Realisierungsalternativen für die Manageragenten. Dieser Vorgang ist vergleichbar mit der Realisierung von Anwendungsobjekten. Auch hier besteht ein weites Spektrum an Alternativen. Bei kleinen, kurzlebigen Aktivitäten ist es möglich, auf gewisse Funktionalitäten zu verzichten, wie zum Beispiel die Fähigkeit, physikalisch mobil zu sein. Der Overhead, der nötig wäre, um eine kurzlebige Aktivität von einem Rechner zu einem anderen zu befördern, ist in vielen Fällen höher, als die Aktivität zu Ende rechnen zu lassen. In dem Graphersetzungssystem bedeutet dies, daß ein Manageragent nur einen Teil der Ersetzungsregeln kennt, eben den Teil, der für seine Ressourcen relevant ist. Die dadurch entstehenden Unterschiede bei den Manageragenten erlauben es, auch den Manageragenten, analog zu den Anwendungsobjekten, unterschiedlich zu realisieren. In einigen Fällen wird es notwendig sein, den Manageragent aktiv zu realisieren, in anderen Fällen kann eine passive Realisierung ausreichend sein.

Stellt ein Manageragent während der Laufzeit fest, daß seine Fähigkeiten, d.h. sein vorhandenes Regelwerk, nicht ausreichen, um seinen Aufgaben gerecht werden, zum Beispiel, weil sich die Aktivität anders verhält, als am Anfang erwartet, so besteht die Möglichkeit, ihn in eine andere Realisierung zu transformieren und ihm somit neue Fähigkeiten zu verleihen. Es ist klar, daß diese Transformation ihrerseits Ressourcen benötigt, aber auf der anderen Seite werden, während der Manageragent noch in der kleineren Fassung vorliegt, Ressourcen gespart, die dadurch anderen Objekten zur Verfügung stehen. Dieselben Möglichkeiten bestehen für die Verkleinerung eines Manageragenten. Wird sein voller Funktionsumfang nicht mehr benötigt, kann er in eine reduzierte Fassung überführt werden und auf diese Weise Ressourcen freigeben.

6 Stand der Arbeit

Nach Abschluß der allgemeinen Vorarbeiten und der Erarbeitung des Beschreibungsmodells sowie des Manageragentenmodells konzentrieren sich die aktuellen Arbeiten auf die Implementierung des Ansatzes. Hierzu wurde ein Prototyp entwickelt, der in erster Linie passive vertikale Ressourcen in das Management einbezieht. Aufbauend auf diesen Prototypen finden zur Zeit Implementierungsarbeiten statt, um den Prototypen zu einem lauffähigen System zu erweitern. Diese Arbeiten können in drei Teilbereiche unterteilt werden. Die Arbeiten im ersten Bereich konzentrieren sich auf die Weiterentwicklung der Behandlung der passiven Ressourcen in dem Prototypen. Dies umfaßt die Implementierung von neuen Strategien und deren Einsatzfelder.

Die Arbeiten im zweiten Bereich beziehen sich auf die stärkere Integration des Managements aktiver Ressourcen. Hierzu wird ein Modell entwickelt, welches für das Lastmanagement, aufbauend auf das formale Beschreibungsmodell alle Einflußfaktoren, wie sie sich aus den Ressourceneigenschaften ergeben, wie zum Beispiel gemeinsam benutzte Objekte, Kommunikationsbeziehungen, berücksichtigt. Der nächste Schritt in diesem Gebiet wird sich mit der Implementierung und Integration dieses Modells in den dann weiterentwickelten Prototyp befassen.

Das dritte Arbeitsgebiet stellt die Integration des Prototypen in die Arbeiten des Sonderforschungsbereiches 342, Teilprojekt A8 dar. Die Ergebnisse des Multiagenten-Systems ergänzt in geeigneter Weise die im Rahmen von dem Teilprojekt A8 durchgeführten Arbeiten. Als Grundlage für die Implementierung dient ein Cluster von Sun UltraSparc-Rechnern unter Solaris 2.5, der auch für die Arbeiten des Sonderforschungsbereiches verwendet wird.

7 Zusammenfassung

In diesem Bericht wurde ein neues Konzept für ein verteiltes Ressourcenmanagement vorgestellt. Die Managementaufgaben werden von einem Multi-Agenten-System durchgeführt, das sich bei seinen Entscheidungen auf ein Graphersetzungssystem abstützt. Jeder Aktivität wird hierbei ein Manageragent zur Seite gestellt. Mithilfe der Kooperation unter den Manageragenten werden die Ressourcen in dem gesamten System verteilt verwaltet. Durch die enge Beziehung zwischen dem Manageragenten und der assoziierte Aktivität wird eine Anwendungsangepaßtheit erreicht, die es ermöglicht, auf die spezifischen Bedürfnisse der Anwendung einzugehen. Die Kooperation unter den Manageragenten ist die Grundlage für eine faire und effiziente Ressourcenverteilung zwischen den einzelnen Anwendung in dem System. Ein möglichst kleiner Managementoverhead wird durch unterschiedlich mächtige Manageragenten verwirklicht, sowie der Möglichkeit, Manageragenten während der Laufzeit in unterschiedliche Realisierungsformen zu transformieren. Die Grundlage für diese Transformationen bildet ein Graphersetzungssystem.

8 Zusammenarbeit mit anderen Partnern

Bei der Entwicklung des Manageragentensystems bestand eine sehr enge Zusammenarbeit mit dem Sonderforschungsbereich 342 „Werkzeuge und Methoden zur Nutzung paralleler Rechnerarchitekturen“ Teilprojekt A8 „Konzepte und Verfahren zur Konstruktion heteromorph paralleler Systeme“. Die aktuellen Arbeiten im Teilprojekts A8 konzentrieren sich zum einen auf die Entwicklung eines neuen Mikrokerns, der auch als Grundlage für das Manageragenten-System geeignet ist, um die Kontrolle über alle Ressourcen in einem verteilten System zu erlangen (z.Z. beschränkt Solaris diese Kontrolle). Zum anderen wurde ein Compiler für eine parallele, objektbasierte Programmiersprache entwickelt. Der in dieser Arbeit entwickelte Prototyp kann als Ausführungsgrundlage für die parallelen Programme dienen, die von dem neuen Compiler erzeugt werden. Hierbei wird sehr viel Wert auf den Informationsaustausch zwischen Compiler und Ressourcenmanagement gelegt [PE97].

Die Zusammenarbeit mit anderen Teilprojekten des Graduiertenkollegs hatte ebenfalls sehr starken Einfluß auf die Arbeiten in diesem Teilprojekt. Es stehen/standen kompetente Ansprechpartner bezüglich Multi-Agenten und Graphersetzungssystemen zur Verfügung, mit denen auftretende Probleme oder Fragestellungen gelöst werden können/konnten. Anforderungen an ein verteiltes Ressourcenmanagement können/konnten mit Kollegiaten aus Teilprojekten diskutiert werden, die sich verstärkt mit parallelen und verteilten Anwendungen befassen. Eine ausführlichere Beschreibung der Kooperationen innerhalb des Graduiertenkollegs findet sich in Abschnitt 1.1.

Literatur

[BCE+94] B. N. Bershad, C. Chambers, S. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage und E. G. Sirer. SPIN - An extensible microkernel for application-specific operating system services. In *Proceedings of the 1994 European SIGOPS Workshop*, Schloš Dagstuhl, Germany, September 1994.

[BS93] W. J. Bolosky und Michael L. Scott. False Sharing and its Effect on Shared Memory Performance. *Proc., Fourth Symp. on Experiences with Distributed and Multiprocessor Systems (SEDMS)*, September 1993.

[CKK95] John B. Charter, Dilip Khandekar und Linus Kamb, Hrsg. *Distributed Shared Memory: Where We Are and Where We Should Be Headed*, May 1995.

[DCM+90] P. Dasgupta, R.C. Chen, S. Menon, M.P. Pearson, R. Ananthanarayanan, U. Ramachandran, M. Ahamad, R.J. LeBlanc, W.F. Appelbe, J.M. Bernabeéu-Aubán, P.W. Hutto, M.Y.A. Khalidi und C.J. Wilkenloh. The Design and Implementation of the *Clouds* Distributed Operating System. *Usenix Computing Systems*, 3(1), 1990.

[Gro96a] Sascha Groh. Designing an efficient resource management for parallel distributed systems by the use of a graph replacement system. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'96)*, S. 215–225, August 1996.

[Gro96b] Sascha Groh. Entwicklung eines Ressourcenmanagements für Verteilt-Parallel-Kooperative-Systeme mit einem Graphersetzungssystem. In *Graduiertenkolleg Kooperation und Ressourcenmanagement in verteilten Systemen - Zwischenbericht zum Frühjahr 1996*, Nr. TUM-I9611. Technische Universität München, März 1996.

[HCK96] C. G. Harrison, D. M. Chess und A. Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM Research Division, 1996.

[KPA+93] K.Murray, P.E.Osmon, A.Valsamidis, A.Whitcroft und T.Wilkinson. Experiences with Distributed Shared Memory. Technical Report TCU/SARC/1993/3, City University, September 1993.

[PE97] M. Pizka und C. Eckert. A Language-Based Approach to Construct Structured and Efficient Object-Based Distributed Systems. In *Proc. of the 30th Hawaii*

Int. Conf. on System Sciences, Bd. 1, S. 130–139, Maui, Hawai, January 1997. IEEE CS Press.

[RAB+90] M. Rozier, V. Abrossimov, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard und W. Neuhauser. Overview of the CHORUS Distributed Operating Systems. Technischer Bericht CS/TR-90-25, Chorus Systèmes, 1990.

[RDH+96] John Rosenberg, Alan Dearle, David Hulse, Anders Lindström und Stephen Norris. Operating system support for persistant and recoverable computations. *Communications of the ACM*, 39(9):62–69, September 1996.

[SW92] Andy Schürr und Bernhard Westfechtel. Graphgrammatiken und Graphersetzungssysteme. Technical Report 92-15, Technical University of Aachen (RWTH Aachen), 1992.

[van90] J. Leeuwen van. *Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics Handbook of Theoretical.* Elsevier Science Publishers B. V., Amsterdam, 1990.

[WJ95] M. Wooldridge und N. R. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

1.3.13 Ein Kalkül für verteilte Programmierung, basierend auf Graph-Ersetzung

Teilprojekt: Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für verteilte Programmierung
Barbara König

1 Einleitung

Für die Entwicklung verteilter Systeme benötigt man Programmiersprachen und Programmierwerkzeuge, deren Entwicklung den Möglichkeiten der Hardware bisher noch ein gutes Stück hinterherhinkt.

Um verteilte Programmierung wirklich verstehen zu können, benötigt man geeignete Modelle und Kalküle. Als günstig erscheinen *deklarative* Programmiersprachen, die nicht auf Anweisungsfolgen beruhen, sondern Aufbau und Verhalten eines verteilten Systems beschreiben (Stichwort „Programmieren mit Spezifikationen"). Es liegt dann am Compiler oder Interpreter die Reihenfolge der zu bearbeitenden Aufträge und die Verteilung der Daten und des Codes festzulegen. Parallelität ist also *implizit* im Programm enthalten.

Der im Teilprojekt entwickelte Kalkül SPIDER ist eine einfache deklarative Sprache. Er beschreibt Prozesse, die durch asynchronen, unidirektionalen Nachrichtenaustausch miteinander kommunizieren, wobei sie Nachrichten über Eingabe-Ports empfangen und über Ausgabe-Ports versenden. Es können sowohl Prozesse, als auch Verweise auf Ports verschickt werden. Von den meisten anderen Kalkülen, die verteilte Systeme beschreiben, unterscheidet sich SPIDER in folgenden Punkten:

- SPIDER-Ausdrücke sind **hierarchische Hypergraphen**, d.h. Graphen, bei denen jede Kante beliebig viele Quell- und Zielknoten haben darf. Verteilte Systeme können im allgemeinen durch Graphen in natürlicher Weise beschrieben werden.
  Die operationelle Semantik von SPIDER wird durch Graphersetzung beschrieben. Damit ist man einer verteilten Implementierung schon sehr nah, da Graphersetzung inhärent parallel ist, d.h. disjunkte Graphteile können unabhängig voneinander ersetzt werden (siehe auch Abschnitt 6).
- Es werden **keine Port-Namen** benötigt. Ein Prozeß spricht seine Ports nicht über Namen, sondern über deren Position an. Der Graph beschreibt dann, wie diese Ports miteinander verbunden werden und so eine Kommunikationsstruktur entsteht. Daher kann ein Ausdruck des Kalküls als Black Box betrachtet werden, der mit einem anderen Ausdruck ohne Kenntnis interner Namen kommunizieren kann.
  In anderen Kalkülen [MPW89a, Tho89, Mil80] muß man Namen „innerhalb" eines Prozesses kennen, um mit diesem kommunizieren zu können. Weil diese Namen festgelegt sind, ist es schwierig, Prozesse wiederzuverwenden.
- Es ist möglich sowohl **Prozesse**, als auch **Port-Adressen zu versenden**. Die Eigenschaft, Ports zu versenden und damit die Struktur des verteilten Systems zur Laufzeit zu ändern, nennt man *Mobilität* [MPW89a].
  Die Tatsache, daß man ganze Ausdrücke (Prozesse) als Inhalt einer Nachricht versenden kann, macht SPIDER zu einem Higher-Order-Kalkül. Beim Versenden von

Ausdrücken wird die Frage, ob die globalen Namen des Ausdrucks statisch oder dynamisch [Tho89] gebunden werden sollen, überflüssig. Ein SPIDER-Prozeß mit m Eingabe- und n Ausgabe-Ports kann in jeden Kontext eingesetzt werden, in dem ein solcher Prozeß benötigt wird.

Im nächsten Abschnitt wird der Begrifff des Prozeßgraphen—des Grundbausteins der Syntax von SPIDER—eingeführt. Anschließend werden anhand eines Beispiel (Remote Procedure Call) Syntax und Semantik von SPIDER eingeführt und später in einem eigenen Abschnitt zusammengefaßt. Ein Vergleich mit verwandten Arbeiten und ein Ausblick schließen den Bericht ab.

2 Prozeßgraphen

Die Grundlage des Kalküls SPIDER sind sogenannte Prozeßgraphen, die verwendet werden, um die Struktur eines verteilten Systems zu beschreiben.

Der Begriff Prozeßgraph entspricht im wesentlichen dem Begriff des Multi-pointed Hypergraph aus [Hab92], d.h. eines Graphen, bei dem jede Kante (= Hyperkante) mehrere Quell- und Zielknoten haben kann. Die Hyperkanten heißen entweder *Prozeß* oder *Nachricht*. Die Knoten eines Hypergraphen stellen die *Ports* eines verteilten Systems dar. Eine Funktion *send* ordnet einer Nachricht einen Port zu (auf dem sie gesendet wird).

Definition 1 (Prozeßgraph) Sei L eine Menge von Labels.
Ein Prozeßgraph H ist ein Tupel $H = (V, P, M, s, t, l, send, in, out)$ mit:

- V ist eine endliche Menge von Ports
- $E := P \cup M$ ist eine disjunkte Vereinigung von Prozessen und Nachrichten, auch *Hyperkanten* genannt
- $s, t \colon E \to V^*$ ordnen jeder Kante eine Sequenz von *Quell-* bzw. *Zielports* zu.
- $l \colon E \to L$ ordnet jeder Kante ein Label zu
- $send \colon M \to V$ ordnet jeder Nachricht den Port zu, an den sie geschickt wird
- $in, out \in V^*$ sind Sequenzen von Ein- bzw. Ausgabe-Ports oder auch *externen Ports*, die zur Einbettung eines Hypergraphen verwendet werden. EXT sei die Menge aller externen Ports. Alle anderen Ports heißen *interne Ports*.
  Die Sequenz $in \circ out$[11] enthält keinen Port mehrfach.

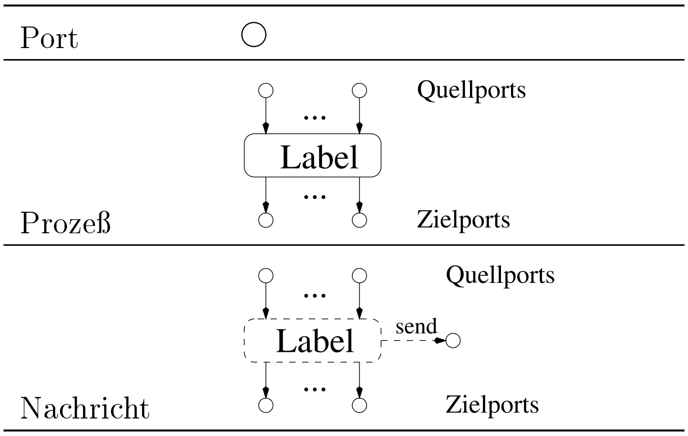
Es wird gefordert, daß kein Port in der Sequenz in als Zielport oder als Ziel einer Nachricht verwendet wird, und daß kein Port in der Sequenz out als Quellport vorkommt. D.h. Eingabe-Ports dürfen nicht als Ausgabe-Ports benutzt werden und umgekehrt. □

Die Elemente eines Prozeßgraphen H werden mit V_H, P_H, M_H, E_H, s_H, t_H, l_H, $send_H$, in_H, out_H, EXT_H, etc. bezeichnet.

Die *Kardinalität* einer Kante $e \in E_H$ ist definiert durch[12]: $card(e) := (|s_H(e)|, |t_H(e)|)$. Die Kardinalität eines Prozeßgraphen H ist $card(H) := (|in_H|, |out_H|)$.

[11] $\circ$ bezeichnet die Konkatenation von Sequenzen.
[12] $|s|$ bezeichnet die Länge einer Sequenz s.

Den oben definierten Graphen kann man durch folgende Notation darstellen:



Da die graphische Darstellung von Prozessen und Nachrichten an Spinnen erinnert und ein Prozeßgraph dann mit einem Spinnennetz verglichen werden kann, hat der Kalkül den Namen Spider erhalten.

Um die Dynamik eines verteilten Systems beschreiben zu können, ist es notwendig, Graph-Transformationen durchzuführen. Eine der am häufigsten verwendeten Transformationen auf Prozeßgraphen ist die *Hyperkanten-Ersetzung*, die an die Definition in [Hab92] angelehnt ist.

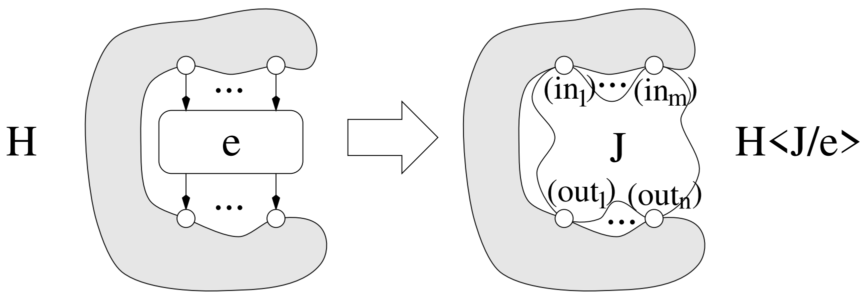


Abbildung 1: Ersetzung von Hyperkanten

Es wird eine Hyperkante e aus einem Prozeßraph H entfernt und durch einen Prozeßgraphen J ersetzt. Die Kante e und der Graph J haben dieselbe Kardinalität (m, n). Dabei werden die Quell- und Zielknoten der ersetzten Kante mit den externen Knoten des Hypergraphen identifiziert (siehe Abbildung 1). D.h. der erste Quell-Port von e wird mit dem ersten Eingabe-Port von J, der mit (in_1) bezeichnet wird, verschmolzen, usw. Die graue Fläche repräsentiert den restlichen Graph H, ohne die Kante e.

Der entstehende Prozeßgraph heißt $H\langle J/e\rangle$.

3 Motivation: Remote Procedure Call

Um die Syntax und Semantik von Spider einzuführen, wird ein Remote Procedure Call beschrieben. Die Situation ist folgende: Ein Client-Prozeß möchte eine Datenbank-Abfrage an einen entfernten Server stellen. Dazu wird die Abfrage als Nachricht an den Server verschickt. Damit der Server dem Client auch antworten kann, wird der Eingabe-Port des Clients an die Nachricht gehängt.

Als Prozeßgraph läßt sich die Situation darstellen wie in Abbildung 2. Dieser Prozeßgraph ist bereits das Gerüst für ein SPIDER-Programm.

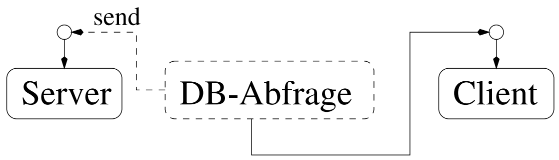


Abbildung 2: Beispiel: Client schickt Datenbank-Abfrage an entfernten Server

Die Funktionalität der einzelnen Komponenten kann wiederum mit Prozeßgraphen beschrieben werden. Die Datenbank-Abfrage ist in diesem Fall ein Prozeß (ein mobiler Agent), der über einen Eingabe-Port den Inhalt einer Datenbank empfängt und über seinen Ausgabe-Port das errechnete Abfrage-Ergebnis ausgibt. D.h. die Datenbank-Abfrage hat das in Abbildung 3 gezeigte Aussehen.

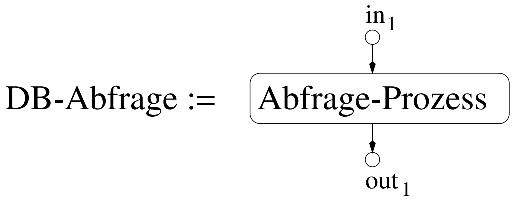


Abbildung 3: Datenbank-Abfrageprozeß

In der Beschreibung des Servers muß angegeben werden, daß der Server auf eine Nachricht wartet, auf welchem Port diese Nachricht ankommen soll und was mit der empfangenen Nachricht und dem daran befestigten Port geschehen soll.

SPIDER wurde durch Anpassung von Ideen aus dem λ-Kalkül entwickelt. Daher werden die obigen Informationen durch eine Prozeßabstraktion beschrieben, die der Funktionsabstraktion (λ-Abstraktion) im λ-Kalkül ähnelt. Das bedeutet vor allem, daß in der Prozeßabstraktion eine Variable angegeben wird, für die der Inhalt der Nachricht eingesetzt werden soll. Abbildung 4 (links) zeigt einen ersten Versuch, den Server zu beschreiben.

Der Zustand des Servers *nach* Empfang der Nachricht wird durch einen Prozeßgraphen „innerhalb“ der Prozeßabstraktion beschrieben. Der Prozeßgraph gibt an, in welche Prozesse sich der Server aufspaltet und welche Nachrichten er verschickt. Dieser Nachfolgezustand des Servers wird nach Empfang der Nachricht in den Gesamtgraphen eingebettet, wie in Abbildung 1 dargestellt.

Die Bestandteile der Prozeßabstraktion haben dabei folgende Bedeutung:

Variable T: gibt an, wo der Inhalt der ankommenden Nachricht eingesetzt werden soll. In diesem Fall wird der ankommende Abfrage-Prozeß an eine Stelle gesetzt, an der er den Inhalt einer Datenbank als Nachricht empfangen kann.

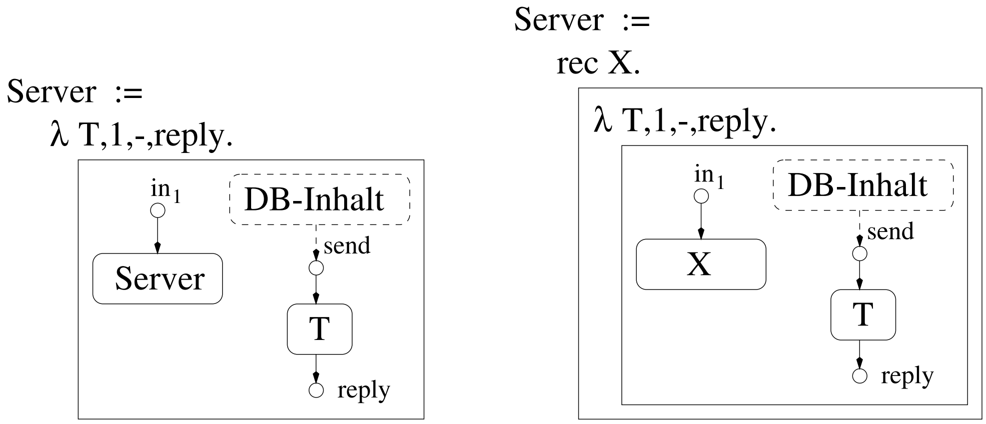


Abbildung 4: Server als SPIDER-Ausdruck: mit Fixpunktgleichung (links) oder mit Fixpunktoperator (rechts)

Portnummer 1: Gibt die Nummer des Ports an, über den die Nachricht empfangen werden soll. (In diesem Fall gibt es nur einen Port.)

Quellport-Sequenz –: Gibt eine Sequenz von Ports des inneren Graphen an, die mit den ankommenden Quellports verschmolzen werden sollen. – steht für die leere Sequenz.

Zielport-Sequenz *reply*: Gibt eine Sequenz von Ports des inneren Graphen an, die mit den ankommenden Zielports verschmolzen werden sollen. *reply* ist dabei eine einelementige Sequenz, die angibt, daß der mit *reply* markierte Port des inneren Graphen mit dem von der Nachricht mitgebrachten Port verschmolzen werden soll. D.h. die Antwort-Adresse wird mit einem Ausgabe-Port des Abfrage-Prozesses verbunden, so daß dessen Ausgabe direkt an den Client geschickt wird.

Der Name Server taucht auf der linken und rechten Seite der Definition in Abbildung 4 (links) auf, um zu zeigen, daß derselbe Server auch nach Empfang und Verarbeitung einer Nachricht noch verhanden sein soll. Solche Fixpunkt-Gleichungen kann man in SPIDER mit Hilfe eines Fixpunktoperators *rec X* auflösen (siehe Abbildung 4, rechts).

Das Verhalten des Clients wird hier nicht näher beschrieben.

Um die angegebenen Spezifikationen besser verstehen zu können wird in Abbildung 5 der Empfang einer Datenbank-Abfrage dargestellt. Durch eine Fixpunktexpansion wird der Server zunächst in die Lage versetzt, eine Nachricht zu empfangen, eine Kopie des Servers wird erzeugt. Danach wird die Abfrage vom Server empfangen und eingesetzt. Der Abfrage-Prozeß wird eingebettet und wird dadurch mit der Nachricht, die den Datenbank-Inhalt mitbringt und mit dem Client verbunden. Der Server kann jetzt unabhängig von der laufenden Datenbank-Abfrage weiterarbeiten, der Abfrage-Prozeß schickt das Ergebnis direkt an den Client.

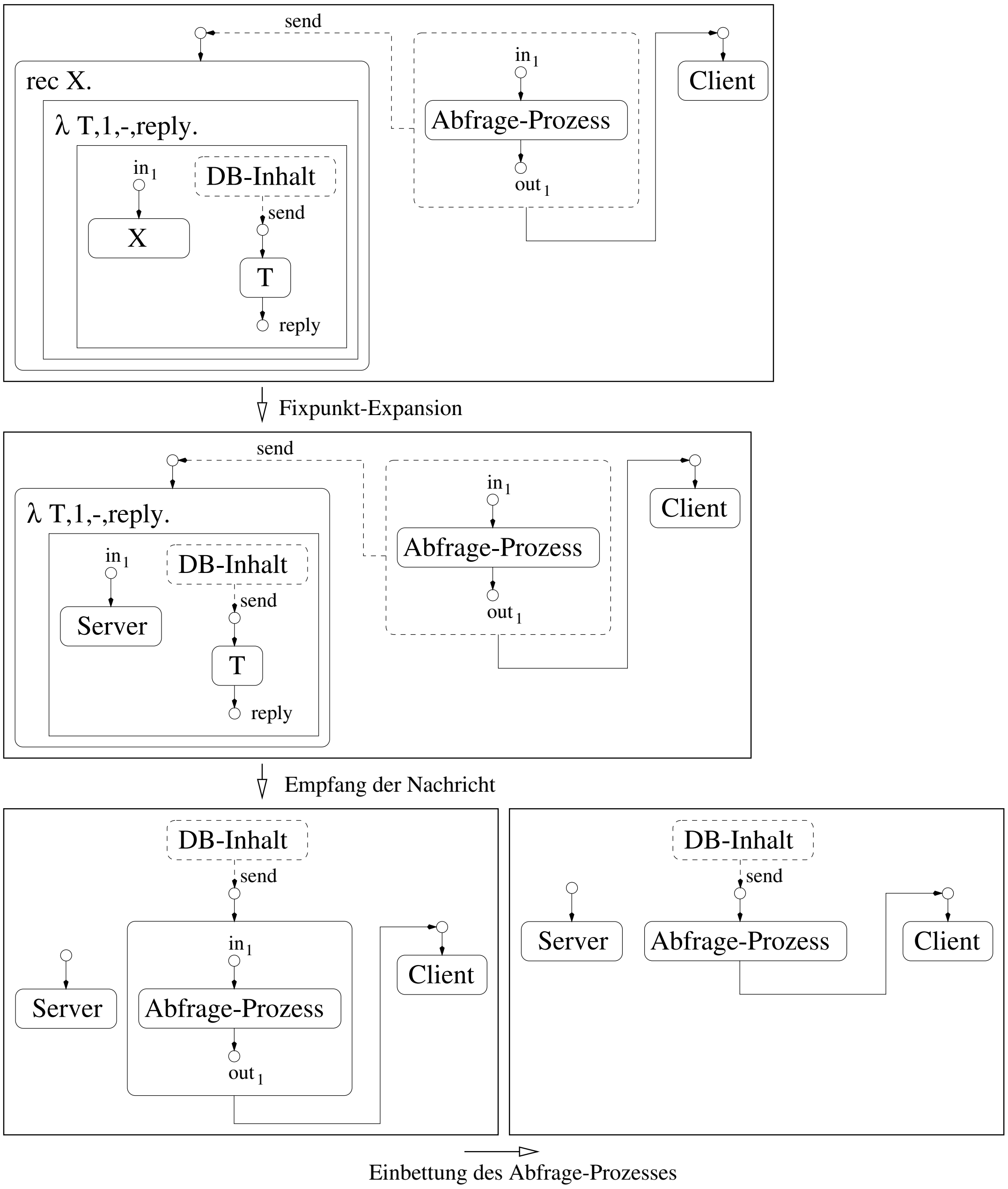


Abbildung 5: Empfang einer Datenbank-Abfrage

4 Syntax und Semantik von SPIDER

Im vorigen Abschnitt wurden alle syntaktischen Konstrukte und alle Reduktionen der operationellen Semantik von SPIDER bereits informell eingeführt. Dieser Abschnitt soll dazu dienen, alle Konstrukte noch einmal zusammenzufassen.

Auf der Basis von Prozeßgraphen (Definition 1) wird im folgenden definiert, wie ein SPIDER-Ausdruck auszusehen hat.

Definition 2 (SPIDER-Ausdruck) Sei Id eine Menge von *Identifiern*. Die Menge der Ausdrücke $\mathcal{E}$ wird induktiv definiert. Ein Ausdruck ist von einer der folgenden vier Formen:

Variable: $x \in Id$

Prozeßgraph: $H \in \mathcal{P}$, wobei $\mathcal{P}$ die Menge aller Prozeßgraphen ist, deren Kanten mit Ausdrücken aus $\mathcal{E}$ markiert sind.

Prozeßabstraktion: $\lambda x, p, a, b.H$ mit $x \in Id,\ p \in \mathbb{N},\ a, b \in V_H^*,\ H \in \mathcal{P}$

Dabei kommt kein Ports in der Sequenz a als Zielport oder als Ziel einer Nachricht und kein Port in der Sequenz b als Quellport vor. Außerdem enthalte die Sequenz $a \circ b$ keinen externen Port und keinen Port mehrfach.

Fixpunktoperator: $rec\ x.expr$ mit $x \in Id,\ expr \in \mathcal{E}$

□

Variable, die weder durch eine Prozeßabstraktion, noch durch einen Fixpunktoperator gebunden sind, heißen im folgenden *freie Variable*. $expr[expr'/x]$ bezeichne den Ausdruck, der entsteht, wenn alle freien Variablen im Ausdruck $expr$ durch den Ausdruck $expr'$ ersetzt werden. Dabei müssen u.U. gebundene Variablen umbenannt werden, um zu verhindern, daß freie Variablen versehentlich gebunden werden (analog zum λ-Kalkül).

In SPIDER gibt es drei Arten von Reduktionen, die bereits alle in Abbildung 5 vorkamen:

Einbettung: Seien H, J Prozeßgraphen aus $\mathcal{P}$.

$e \in P_H$ sei ein Prozeß von H, der mit J markiert ist, d.h. $l_H(e) = J$. Die Kardinalitäten von J und e seien gleich.

Dann reduziert H zu $H\langle J/e\rangle$ (siehe Abbildung 1).

Diese Reduktion bringt verschiedenen Hierarchiebenen eines Prozeßgraphen miteinander in Kontakt.

Fixpunktexpansion: $rec\ x.expr$ reduziert zu $expr[rec\ x.expr/x]$

Empfang einer Nachricht: Sei H ein Prozeßgraph, der einen Prozeß r enthält, der mit $\lambda x, p, a, b.J$ markiert ist und eine Nachricht n, die mit einem beliebigen Ausdruck $expr$ markiert ist. Die Nachricht werde an den p-ten Quellport des Empfängers geschickt.

Die Reduktion erfolgt in mehreren Schritten:

- Im Prozeßgraph J werden alle freien Vorkommen von x durch $expr$ ersetzt.
- Der Prozeß r und die Nachricht n werden aus dem Prozeßgraph H entfernt und durch $J[expr/x]$ ersetzt
- Die bisherigen Quell- und Zielports von r werden mit den externen Ports von J verschmolzen.
- Die bisherigen Quell- und Zielports von n werden mit den Ports von J verschmolzen, die durch a und b bezeichnet werden.

5 Vergleich mit verwandten Arbeiten

Spider wurde entwickelt, indem Ideen aus dem **λ-Kalkül** [Bar90, HS86] auf Graphen übertragen wurden. Der Empfang einer Nachricht in Spider hat starke Ähnlichkeiten zur β-Reduktion im λ-Kalkül. Aufgrund der Übereinstimmungen zwischen beiden Kalkülen, kann der λ-Kalkül relativ einfach in Spider simuliert werden.

Wie bereits in der Einleitung erwähnt wurde, unterscheidet sich Spider von den meisten anderen verteilten Kalkülen durch die Darstellung von Ausdrücken als Graphen und durch die Vermeidung globaler Port-Namen.

Wie auch der **π-Kalkül** von Robin Milner [MPW89a, MPW89b, Mil92] bietet Spider die Möglichkeit, Adressen zur versenden (Mobilität). Zusätzlich ist man aber auch in der Lage, ganze Ausdrücke zu verschicken. Der Kalkül **CHOCS** von Brent Thomsen [Tho89] bietet dagegen die Möglichkeit, Ausdrücke, aber keine Adressen, zu versenden.

Die asynchrone Kommunikation in Spider zeigt Ähnlichkeiten mit dem **Aktor-Modell** von Agha und Hewitt [Agh86, AMST92, Agh89]. Aktoren sind jedoch nur Behälter für Programme eines anderen Programmier-Paradigmas, wohingegen das Verhalten von Spider-Prozessen auf homogene Weise in einem einzigen Paradigma beschrieben werden kann.

Eine mit Spider verwandte Möglichkeit der Beschreibung verteilter Systeme ist die Spezifikation mit Hilfe von **Graph-Grammatiken** oder Graph-Ersetzungssystemen, wie z.B. die Δ-Grammatiken von J.P. Loyall [Loy92] oder die Concurrent Combinators von Nabuko Yoshida [Yos94]. Hier gibt es jedoch keine programmiersprachlichen Konzepte, wie Abstraktion oder Higher-Order-Kommunikation.

6 Ausblick und Zusammenarbeit

Es wurde ein graphischer Kalkül Spider zur Beschreibung und Programmierung von verteilten Systemen vorgestellt. Er soll zu einer graphischen Programmiersprache erweitert werden, mit der auch die Beschreibung großer Systeme möglich ist.

Dazu wurden bereits folgende Erweiterungen vorgenommen:

- Entwicklung eines **Typsystems**, mit dem es möglich ist, Spider-Programme auf Laufzeitfehler zu untersuchen
- Verbinden von Spider mit einer **funktionalen Sprache**, um den Funktionsumfang zu erweitern

Folgendes ist noch zu tun:

- Implementierung von SPIDER mit Hilfe eines **verteilten Graphersetzungs-Systems**. Damit kann man ein SPIDER-Programm verteilt abarbeiten lassen, ohne daß sich der Programmierer explizit Gedanken über die Verteilung von Code und Daten machen muß. Zu diesem Zweck soll das verteilte Graphersetzungs-System von Boris Reichel (siehe Abschnitt „Verteilte attributierte Graphersetzung"), das in demselben Teilprojekt entwickelt wird, eingesetzt werden.
  SPIDER-Ausdrücke können relativ einfach in Graphgrammatik-Produktionen übersetzt und dann von einem Graphersetzungs-System bearbeitet werden.
- Erstellung einer graphischen Bedienoberfläche als **Entwicklungsumgebung** für SPIDER. Die Erstellung von Bedienoberflächen für verteilte Systeme soll im Teilprojekt verstärkt weitergeführt werden (siehe auch den Fortsetzungsantrag). Dabei könnten verteilte Bedienoberflächen mit Hilfe von SPIDER beschrieben werden.
- Einbindung **externer Prozeß-Beschreibungen**
- **Denotationelle Semantik** für SPIDER

Die Arbeit wird in enger Kooperation mit Boris Reichel, der im selben Teilprojekt innerhalb des Graduiertenkollegs arbeitet, durchgeführt. Beide Teilbereiche beschäftigen sich mit Graphersetzung als deklarativem Formalismus zur Beschreibung verteilter Systeme. Während dieser Teilbereich sich in diesem Zusammenhang hauptsächlich damit beschäftigt, Graphersetzung als Grundlage für visuelle Programmiersprachen einzusetzen, arbeitet Boris Reichel vor allem an der verteilten Implementierung von Graphersetzungs-Systemen und realisiert damit den in SPIDER implizit enthaltenen Parallelismus.

Innerhalb des Graduiertenkollegs gibt es Beziehungen zu den Querschnittsthemen „Graphgrammatiken/Graphersetzungssysteme", „Spezifikation und Programmiersprachen" und „Verifikation", denn ein Teil dieses Themas dreht sich um Bisimulations- und Verhaltensäquivalenzen für Prozesse, wie sie auch für SPIDER-Ausdrücke verwendet werden.

Außerdem besteht eine Zusammenarbeit mit anderen Mitgliedern des Lehrstuhls Eickel, wie z.B. dem LOPEX-Projekt („Logikgestützte Programmierumgebungen"). Dabei ergeben sich Anknüpfungspunkte in den Bereichen „verteilte objekt-orientierte Programmierung" und „Entwicklungsumgebungen für Programmiersprachen".

Literatur

[Agh86] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, Massachusetts, 1986.

[Agh89] Gul Agha. Supporting Multiparadigm Programming on Actor Architectures. In *Proceedings of PARLE '89*, S. 1–19. Springer-Verlag, 1989. LNCS 366.

[AMST92] Gul Agha, Ian Mason, Scott Smith und Carolyn Talcott. Towards a Theory of Actor Computation. In R. Cleaveland, Hrsg., *The Third International Conference on Concurrency Theory (CONCUR '92).* Springer-Verlag, 1992. LNCS 630.

[Bar90] H.P. Barendregt. Functional Programming and Lambda Calculus. In Jan van Leeuwen, Hrsg., *Formal Models and Semantics, Handbook of Theoretical Computer Science*, Bd. B, S. 321–364. Elsevier, 1990.

[Hab92] Annegret Habel. *Hyperedge Replacement: Grammars and Languages.* Springer, 1992. LNCS 643.

[HS86] J. Roger Hindley und Jonathan P. Seldin. *Introduction to Combinators and λ-Calculus.* London Mathematical Society, 1986. Student Texts 1.

[Loy92] Joseph Patrick Loyall. Specification of Concurrent Systems Using Graph Grammars. Report No. UIUCDCS-R-92-1752, University of Illinois at Urbana-Champaign, Department of Computer Science, May 1992.

[Mil80] Robin Milner. *A Calculus of Communicating Systems.* Springer-Verlag, 1980. LNCS 92.

[Mil92] Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.

[MPW89a] R. Milner, J. Parrow und D. Walker. A Calculus of Mobile Processes, Part I. Tech. Rep. ECS-LFCS-89-85, University of Edinburgh, Laboratory for Foundations of Computer Science, 1989.

[MPW89b] R. Milner, J. Parrow und D. Walker. A Calculus of Mobile Processes, Part II. Tech. Rep. ECS-LFCS-89-86, University of Edinburgh, Laboratory for Foundations of Computer Science, 1989.

[Tho89] Brent Thomsen. A Calculus of Higher Order Communicating Systems. *Proceedings 16th Annual Symposium on Principles of Programming Languages*, S. 143–154, 1989.

[Yos94] Nobuko Yoshida. Graph Notation for Concurrent Combinators. In Takayasu Ito und Akinori Yonezawa, Hrsg., *Theory and Practice of Parallel Programming, Intternational Workshop TPPP '94*, S. 393–412. Springer-Verlag, 1994. LNCS 907.

1.3.14 Model Checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen

Teilprojekt: Verteilte Algorithmen: Spezifikation, Modellierung, Verifikation, Korrektheit
Richard Mayr

1 Einleitung

Mit zunehmender Komplexität moderner Software– und Computersysteme gewinnen formale Methoden zum Sicherstellen von Korrektheit zunehmend an Bedeutung. Im Wesentlichen lassen sich diese Methoden in drei Klassen unterteilen:

- Nicht-algorithmische Methoden: Formale Spezifikation und Beweistechniken.
- Semi-algorithmische Methoden: Theorembeweiser mit mit mehr oder weniger großer menschlicher Interaktion.
- Automatische Methoden: Entscheidungsverfahren und Model checking.

Obwohl die formalen Grundlagen von sequentiellen Programmen relativ gut bekannt sind finden formale Methoden erst in neuerer Zeit Eingang in praktische Anwendungen. Bei parallelen Programmen ist die Situation bislang noch erheblich schwieriger. Die formalen Grundlagen der Semantik nebenläufiger Systeme sind bei weitem noch nicht so gut bekannt wie für sequentielle Systeme, da sie erheblich komplexer sind. Ein wichtiger Forschungsgegenstand ist daher die Semantik nebenläufiger Systeme, die Entwicklung von Methoden zur Verifikation, sowie die Integration von algorithmischen und semi-algorithmischen Methoden.

In meiner Diplomarbeit habe ich mich mit Termersetzungssystemen höherer Ordnung befaßt, die eine wichtige Grundlage für Theorembeweisersysteme darstellen (siehe auch [MN94]). Jetzt befasse ich mich mit den formalen Grundlagen der Semantik nebenläufiger Systeme mit unendlichen Zustandsräumen. Hier gilt mein Interesse speziell den automatischen Methoden wie Model checking und der Komplexität von Problemen, die im Zusammenhang mit der Verifikation nebenläufiger Systeme stehen.

2 Bisimulation

Die wichtigsten Methoden zur Beschreibung von nebenläufigen Systemen sind Petri Netze und Prozessalgebren. Nebenläufige Systeme mit endlichen Zustandsräumen wurden schon relativ ausgiebig untersucht, wobei häufig Ergebnisse aus der Theorie der formalen Sprachen verwendet wurden. Für diese Art von Systemen sind alle verhaltensbasierten Äquivalenzen entscheidbar, und viele automatische Systeme sind zur Analyse ihres Verhaltens erstellt worden.

Neuere Forschungen befassen sich vor allem mit der Entscheidbarkeit von semantischen Äquivalenzen für bestimmte Klassen von nebenläufigen Systemen mit unendlichen Zustandsräumen. Aus einem bekannten Theorem aus der Theorie der formalen Sprachen folgt, daß Sprachgleichheit für die Klasse der kontextfreien Prozesse unentscheidbar ist

[HU79]. Deswegen werden stärkere Äquivalenzbegriffe betrachtet. Diese stärkeren Äquivalenzen berücksichtigen die Begriffe livelock, deadlock und Kausalität. Der wichtigste dieser neuen Äquivalenzbegriffe ist die Bisimulation. Da diese im Gegensatz zu allen anderen verhaltensbasierten Äquivalenzen für kontextfreie Prozesse entscheidbar ist ist sie zu einem zentralen Begriff im Bereich der Theorie der nebenläufigen Systeme geworden [GH91, HT90] [CHM93].

Die Frage ist nun, für welche Klassen von Prozessen es entscheidbar ist, ob zwei Prozesse bisimular sind, und ob es einen effizienten Algorithmus gibt, der das Problem löst. In Prozessalgebren werden Prozesse durch Prozessterme beschrieben. Es gibt eine formale Syntax nach der diese Terme aus atomaren Aktionen und Operatoren aufgebaut sind. Das dynamische Verhalten der Prozesse wird durch Regeln der Form $a \stackrel{\alpha}{\to} b$, beschrieben, wobei a und b Prozesse und α eine atomare Aktion ist. Dies bedeutet, daß Prozess a eine Aktion α ausführt und dadurch zu Prozess b wird. Somit definiert ein Prozess ein beschriftetes Transitionssystem (LTS). Ein LTS ist ein (möglicherweise unendlicher) gerichteter Graph, dessen Knoten mit Prozesstermen–, und dessen Kanten mit atomaren Aktionen beschriftet sind. Die Semantik der Prozesse wird dann durch eine Äquivalenzrelation über der Termalgebra definiert, wobei die Äquivalenzklassen Prozesse repräsentieren.

Definition 1.1 Sei A eine Menge von Konten, Act eine Menge atomarer Aktionen und (A, θ) ein LTS, wobei $\theta : A \to pow(Act \times A)$ die Transitionen bestimmt. Es sei

$$\theta_\alpha(a) := \{x \mid (\alpha, x) \in \theta(a)\}$$

Man schreibt $a \stackrel{\alpha}{\to} a'$ für $a' \in \theta_\alpha(a)$. Eine binäre Relation $\sim$ auf A ist eine Bisimulation, wenn

$$\begin{aligned} \forall a, b \in A \text{ s.t.} a \sim b\ \forall \alpha \in Act. \quad & (a \stackrel{\alpha}{\to} a' \ \Rightarrow \exists b \stackrel{\alpha}{\to} b'.\ a' \sim b') \wedge \\ & (b \stackrel{\alpha}{\to} b' \ \Rightarrow \exists a \stackrel{\alpha}{\to} a'.\ a' \sim b') \end{aligned}$$

Es gibt immer eine größte Bisimulation, die eine Äquivalenzrelation ist und **starke Bisimulation** genannt wird.

Eine weitere elegante Charakterisierung ist folgende: Eine Äquivalenzrelation $\sim$ ist eine **Bisimulation**, wenn

$$\forall a \in A\ \forall \alpha \in Act.\ \theta_\alpha([a]_\sim) \subset [\theta_\alpha(a)]_\sim$$

Zwei Knoten a, b in einem LTS werden bisimular genannt, wenn es eine Bisimulation $\sim$ gibt mit $a \sim b$.

Starke Bisimulation ist manchmal zu strikt. Daher wird eine neue Äquivalenz definiert, die berücksichtigt, daß Prozessinterne Aktionen nicht nach außen hin sichtbar sein sollten.

Sei $\stackrel{a}{\Rightarrow} := (\stackrel{\tau}{\to})^* \stackrel{a}{\to} (\stackrel{\tau}{\to})^*$ für $a \in Act$ und

$$\stackrel{\hat{a}}{\Rightarrow} := \begin{cases} \stackrel{a}{\Rightarrow}, & \text{für } a \neq \tau \\ (\stackrel{\tau}{\to})^*, & \text{für } a = \tau \end{cases}$$

Eine binäre Relation R über den Zuständen eines LTS ist eine *schwache Bisimulation*, wenn

$$\begin{aligned} \forall s_1, s_2 \in S \text{ s.t.} s_1 R s_2\ \forall a \in Act. \quad & (s_1 \stackrel{a}{\to} s_1' \ \Rightarrow \exists s_2 \stackrel{\hat{a}}{\Rightarrow} s_2'.\ s_1' R s_2') \wedge \\ & (s_2 \stackrel{a}{\to} s_2' \ \Rightarrow \exists s_1 \stackrel{\hat{a}}{\Rightarrow} s_1'.\ s_1' R s_2') \end{aligned}$$

Es gibt eine größte schwache Bisimulation, die als $\approx$ geschrieben wird. Trivialerweise gilt $\sim\subseteq\approx$ für jedes LTS.

Es folgt, daß die Dynamikregeln nur dann eindeutig die Dynamik der Quotientenalgebra beschreiben, wenn die gewählte Äquivalenz eine Bisimulation ist. Dies ist einer der Hauptgründe dafür, daß Bisimulation bei Prozessalgebren eine so wichtige Rolle spielt.

Eine der wichtigsten Sprachen zur Beschreibung von parallelen Prozessen ist der Calculus of Communicating systems (CCS) von Milner [Mil89]. CCS-Prozesse werden gebildet aus atomaren Aktionen, sequentieller Komposition, paralleler Komposition, nichtdeterministischer Auswahl und Prozessynchronisation. Eine Untermenge von CCS sind die Basic Parallel Processes (BBP) die durch atomare Aktionen, nichtdeterministische Auswahl und parallele Komposition gebildet werden. Die Algebra der kontextfreien Prozesse wird mit BPA bezeichnet. Es gibt einen engen Zusammenhang zwischen BPPs und communication free Petri nets, der Klasse von markierten Petri Netzen, bei denen jede Transition genau eine Stelle im Vorbereich hat. Normierte BPP/BPA-Prozesse sind solche, die in jedem erreichbaren Zustand mindestens eine terminierende Berechnung haben.

Die folgende Tabelle zeigt einige Ergebnisse zur Entscheidbarkeit von starker Bisimulation. Mein Beitrag dazu betrifft den Fall der deterministischen BPP [May96d]. Die anderen Ergebnisse sind von Milner [Mil89], Jančar [Jan94], Taubner [Tau89] Hirshfeld, Jerrum und Moller [HJM94].

	nichtdeterministisch	deterministisch
Petri Netze	unentscheidbar	entscheidbar, EXPSPACE-hart
BPA	entscheidbar, Komplex. ?	polynomiell
normed BPA	polynomiell	polynomiell
BPP	entscheidbar, Komplex. ?	**polynomiell**
normed BPP	polynomiell	polynomiell

Eine andere Frage ist, ob ein Prozess mit unendlichem Zustandsraum und ein endliches System semantische gleich sind (bez. starker/schwacher Bisimulation). Mein Beitrag dazu ist zu den BPP [May96f, May96d]. Die Ergebnisse zu den Petri Netzen stammen von Jančar [Jan94].

	starke Bisimulation	schwache Bisimulation
Petri Nets	entscheidbar (nicht elementar)	unentscheidbar
BPP	**entscheidbar (elementar)**	**entscheidbar**

3 Semantische Erreichbarkeit

Wenn mit Klassen von semantisch äquivalenten Prozessen gearbeitet wird stellen sich neue Probleme im Bereich der Verifikation. Die Eigenschaft, daß ein bestimmter Zustand (z.B. ein Verklemmungszustand) nicht erreichbar ist, ist nicht mehr hinreichend für die Verifikation eines Systems. Es könnte sein, daß ein semantisch äquivalenter (ebenso schlechter) Zustand erreichbar ist.

Somit stellt sich das Problem der semantischen Erreichbarkeit. Das ist das Problem, ob ein Zustand erreichbar ist der zu einem gegebenen Zustand semantisch äquivalent ist. Als semantische Äquivalenz wird die starke Bisimulation verwendet. Dieses Problem wurde für verschiedene Klassen von Systemen mit unendlichen Zustandsräumen untersucht

[May96c]. Während es für allgemeine Petri Netze unentscheidbar ist, ist es entscheidbar für kontextfreie Prozesse und NP-vollständig für normierte BPP.

4 Model checking

Model checking ist eine verbreitete Technik um Eigenschaften nebenläufiger Systeme zu verifizieren. Die bekannten Algorithmen lassen sich in zwei Klassen einteilen: iterative und Tableau-basierte. Die iterativen Algorithmen berechnen alle Zustände eines Systems die eine bestimmte Eigenschaft haben, während die Tableau-basierten entscheiden, ob ein Zustand eine Eigenschaft hat.

4.1 Communication free nets und branching time Logik

Dieser Algorithmus ist Tableau-basiert und entscheidet ob ein Communication free net eine Eigenschaft erfüllt, die mit folgender branching time Logik beschrieben werden kann. Diese Netze beschreiben nebenläufige Systeme mit unendlichen Zustandsräumen. Es wurde gezeigt, daß es dennoch ausreicht nur endlich viele Zustände zu untersuchen um das Model checking Problem zu entscheiden.

Formeln der branching time Logik EF haben folgende Syntax:

$$\Phi ::= s \geq k \ \mid \ s \leq k \ \mid \ \neg\Psi \ \mid \ \Phi_1 \wedge \Phi_2 \ \mid \ \Phi_1 \vee \Phi_2 \ \mid \ \Box\Psi \ \mid \ \Diamond\Psi$$

wobei $s \geq k (s \leq k)$ bedeutet, daß auf der Stelle s mindestens (höchstens) k Marken liegen.

Sei Ω die Menge aller Markierungen. Die Denotation einer EF-Formel ist wie folgt:

$$\begin{array}{lcl} \|s \geq k\| & = & \{\Sigma \mid \Sigma(s) \geq k\} \\ \|s \leq k\| & = & \{\Sigma \mid \Sigma(s) \leq k\} \\ \|\neg\Phi\| & = & \Omega - \|\Phi\| \\ \|\Phi_1 \wedge \Phi_2\| & = & \|\Phi_1\| \cap \|\Phi_2\| \\ \|\Phi_1 \vee \Phi_2\| & = & \|\Phi_1\| \cup \|\Phi_2\| \\ \|\Diamond\Phi\| & = & \{t \mid \exists t \xrightarrow{\sigma} t'.\ t' \in \|\Phi\|\} \\ \|\Phi_1 \vee \Phi_2\| & = & \|\Phi_1\| \cup \|\Phi_2\| \\ \|\Box\Phi\| & = & \{t \mid \forall t'.t \xrightarrow{\sigma} t' \Rightarrow t' \in \|\Phi\|\} \end{array}$$

Die Eigenschaft $t \in \|\Phi\|$ wird auch als $t \models \Phi$ geschrieben. Eine Instanz des Model Checking Problems ist ein communication-free net N, eine Anfangsmarkierung Σ und eine EF-Formel Φ. Das Problem ist, zu entscheiden ob $\Sigma \models \Phi$. Dieses Problem ist PSPACE-complete [May96f, May96d]. Es genügt dabei die Zustände zu betrachten die in exponentiell vielen Schritten erreicht werden können. Subprobleme dieses Problems sind vollständig für die k-te Stufe der der polynomiellen Hierarchie. Im Gegensatz dazu ist dieses Problem für allgemeine Petri Netze unentscheidbar.

4.2 PA-Prozesse

PA-Prozesse stellen eine gemeinsame Obermenge von Basic Parallel Processes (BPP) und kontextfreien Prozessen (BPA) dar. In dieser Prozessalgebra gibt es Operationen für parallele Komposition, sequentielle Komposition, nichtdeterministische Auswahl und Rekursion. Die Syntax ist wie folgt:

Sei $Act = \{a, b, c, \ldots\}$ eine abzählbar unendliche Menge von atomaren Aktionen und $Var = \{X, Y, Z, \ldots\}$ eine unendliche Menge von Variablen. Die Menge der PA Ausdrücke ist definiert durch folgende Syntax:

$$E ::= \epsilon \mid X \mid aE \mid E + E \mid E \| E \mid E.E$$

Ein PA-Prozess wird durch eine Menge von rekursiven Gleichungen $\{X_i := E_i \mid 1 \leq i \leq n\}$ beschrieben, wobei die X_i verschieden sind und die E_i PA-Ausdrücke sind die höchstens die Variablen $\{X_1, \ldots, X_n\}$. enthalten. Es wird die Bedingung gestellt, daß die alle Vorkommen von Variablen in den E_i bewacht sind, d.h. im Bereich eines action prefix vorkommen. Das stellt sicher, daß PA-Prozesse Transitionsgraphen mit endlichem Verzweigungsgrad erzeugen. Bei unbewachten Ausdrücken ist das nicht der Fall wie das Beispiel $X := a + a \| X$ zeigt.

Für jedes $a \in Act$ sei die Relation $\xrightarrow{a}$ die kleinste Relation die folgenden Inferenzregeln genügt.

$$aE \xrightarrow{a} E \qquad \frac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'} \qquad \frac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'} \qquad \frac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'}(X := E)$$

$$\frac{E \xrightarrow{a} E'}{E \| F \xrightarrow{a} E' \| F} \qquad \frac{F \xrightarrow{a} F'}{E \| F \xrightarrow{a} E \| F'} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F}$$

Eine andere Möglichkeit PA-Prozesse zu beschreiben ist es, Zustände durch einen Term der Form

$$G ::= \epsilon \mid X \mid G_1.G_2 \mid G_1 \| G_2$$

zu beschreiben und die Dynamik des Systems durch eine Menge von von Regeln Δ der Form $X \xrightarrow{a} G$ zu beschreiben.

Das wird durch folgende Regeln definiert:

$$X \xrightarrow{a} G \quad \text{if } (X \xrightarrow{a} G) \in \Delta$$

$$\frac{E \xrightarrow{a} E'}{E \| F \xrightarrow{a} E' \| F} \qquad \frac{F \xrightarrow{a} F'}{E \| F \xrightarrow{a} E \| F'} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F}$$

Eine Variante der Logik EF wird verwendet um Eigenschaften von PA-Prozessen zu beschreiben.

$$\Phi \stackrel{\text{def}}{=} a \mid [a]\Phi \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2 \mid \Diamond\Phi$$

wobei $a \in Act$ atomare Aktionen sind.

Sei Ω die Menge aller Prozesse in der Prozessalgebra. Die Denotation $\|\Phi\|$ einer Formel Φ ist wie folgt definiert:

$$\begin{array}{lcl}
\|a\| & = & \{t \mid \exists t \xrightarrow{a} t'\} \\
\|\neg\Phi\| & = & \Omega - \|\Phi\| \\
\|\Phi_1 \wedge \Phi_2\| & = & \|\Phi_1\| \cap \|\Phi_2\| \\
\|\Diamond\Phi\| & = & \{t \mid \exists t \xrightarrow{\sigma} t'.\ t' \in \|\Phi\|\}
\end{array}$$

Die Eigenschaft $t \in \|\Phi\|$ wird auch als $t \models \Phi$ geschrieben. Eine Instanz des Model Checking Problems ist ein PA-Prozess t und eine EF-Formel Φ. Das Problem ist, zu entscheiden ob $t \models \Phi$ gilt.

Es wurde gezeigt das dieses Problem entscheidbar ist [May96b]. Interessanterweise ist EF die einzige Temporallogik (außer Hennessy-Milner Logik) mit einem entscheidbaren Model Checking Problem für PA-Prozesse. Alle anderen gebräuchlichen Logiken (EG, CTL, LTL, linearer μ-Kalkül, modaler μ-Kalkül) sind für PA-Prozesse unentscheidbar [Esp, Esp96, BH96, EK95]. Außerdem wurde gezeigt, daß das Erreichbarkeitsproblem für PA-Prozesse NP-vollständig ist [May96c, May97], und daß sich einige einfachere Verifikationsprobleme für PA-Prozesse in polynomieller Zeit lösen lassen [May97].

4.3 μ-Kalkül

Im Gegensatz zu der im letzten Abschnitt beschriebenen branching time Logik ist Model checking mit dem modalen μ-Kalkül unentscheidbar für BPP [Esp]. Für den linearen μ-Kalkül ist das Model checking Problem sogar für allgemeine Petri Netze entscheidbar. Jedoch ist der bekannte Algorithmus dafür nicht primitiv rekursiv und nicht als Beweisverfahren geeignet. Es wurde daher ein Tableauverfahren entwickelt das eine gute Intuition und bessere Einsicht in das Problem ermöglicht. Dieses Tableauverfahren ist zwar vollständig und kann daher als Entscheidungsverfahren verwendet werden, doch ist dies nicht das Hauptziel. Der Nutzen des Tableauverfahren liegt darin, daß es deutlich macht, warum eine Eigenschaft gilt und als Beweisverfahren verwendet werden kann. Aus Platzgründen muß auf eine weitere Darstellung hier verzichtet werden. Ich verweise daher auf [May96e] für eine ausführliche Darstellung.

5 PAN-Prozesse

Da Petri Netze und PA-Prozesse in ihrer Ausdrucksmächtigkeit unvergleichbar sind stellt sich die Frage nach der kleinsten gemeinsamen Verallgemeinerung. Es wurde gezeigt, daß sich diese Modell von nebenläufigen Systemen in einem sehr allgemeinen Formalismus für Termersetzungssysteme einbetten lassen. Die kleinste gemeinsame Verallgemeinerung, PAN-Prozesse, folgt daraus unmittelbar. PAN-Prozesse sind echt mächtiger als Petri Netze und Model checking mit jeder Logik (außer Hennessy-Milner Logik) ist dafür unentscheidbar. Es wurde jedoch gezeigt, daß das Erreichbarkeitsproblem für PAN-Prozesse immer noch entscheidbar ist [May96a].

6 Ausblick

Geplant sind weitere Forschungen in folgenden Bereichen:

- Weiterentwicklung von Tableauverfahren für den linearen und modalen μ-Kalkül.
- Untersuchungen zur Komplexität von Model Checking Problemen für spezielle Klassen von Petri Netzen (insbesondere für reversible Netze).
- Integration von Entscheidungsverfahren und Beweisverfahren mit Theorembeweisern.

- Untersuchungen zur Entscheidbarkeit und Komplexität von semantischen Äquivalenzen (z.B. starker und schwacher Bisimulation) für verschiedene Modelle von nebenläufigen Systemen mit unendlichen Zustandsräumen.

7 Kooperationen innerhalb und außerhalb des Graduiertenkollegs

Das Graduiertenkolleg „Kooperation und Ressourcenmanagement in verteilten Systemen" ist in zwei Teilbereiche unterteilt. Der erste Teil „Kooperierende Agenten in verteilten Anwendungen" befaßt sich mit verteilten Multiagentensystemen. Der zweite Teilbereich „Programmiermodelle und Ressourcenmanagement für verteilte Systeme" hat ein weiteres Spektrum an Themengebieten. Mein Forschungsprojekt gehört zum zweiten Teil des Graduiertenkollegs. Es gibt hier eine intensive und fruchtbare Kooperation in den Bereichen der verteilten Inferenzsysteme und verteilter Verarbeitung von Graphstrukturen.

Ein weiteres Feld der Zusammenarbeit ist der Arbeitskreis „Ressourcen in verteilten Systemen" (RivS). In diesem Arbeitskreis werden Themen erörtert, die im Zusammenhang mit Problemen des Ressourcenmanagements in nebenläufigen und verteilten Systemen stehen. Ein besonderer Vorteil hierbei besteht darin, daß die Themen von den einzelnen Teilnehmern aus verschiedenen Blickwinkeln beleuchtet werden können. Daraus ergeben sich interessante neue Aspekte und ein tieferes Verständnis dieses Forschungsbereichs.

Außerdem gibt es eine enge Zusammenarbeit mit dem Teilbereich A3 des Sonderforschungsbereichs SFB-342, hier insbesondere in den Bereichen Petri Netze, Komplexitätstheorie, Model checking und Semantik nebenläufiger Systeme .

Literatur

[BEM96] J. Bradfield, J. Esparza und A. Mader. An effective tableau system for the linear time μ-calculus. In F. Meyer auf der Heide und B. Monien, Hrsg., *Proceedings of ICALP'96*, Nr. 1099 aus LNCS. Springer Verlag, 1996.

[BH96] A. Bouajjani und P. Habermehl. Constrained Properties, Semilinear Systems, and Petri Nets. In Ugo Montanari und Vladimiro Sassone, Hrsg., *Proceedings of CONCUR'96*, Nr. 1119 aus LNCS. Springer Verlag, 1996.

[CHM93] S. Christensen, Y. Hirshfeld und F. Moller. Bisimulation Equivalence is Decidable for Basic Parallel Processes. In E. Best, Hrsg., *Proceedings of CONCUR 93*, Nr. 715 aus LNCS. Springer Verlag, 1993.

[EK95] J. Esparza und A. Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *CAV'95*, Nr. 939 aus LNCS, S. 353–366. Springer Verlag, 1995.

[Esp] J. Esparza. Decidability of Model Checking for Infinite-State Concurrent Systems. To appear in Acta Informatica.

[Esp95] Javier Esparza. Petri Nets, Commutative Context-Free Grammars and Basic Parallel Processes. In Horst Reichel, Hrsg., *Fundamentals of Computation Theory*, Nr. 965 aus LNCS. Springer Verlag, 1995.

[Esp96] J. Esparza. More Infinite Results. In B. Steffen und T. Margaria, Hrsg., *Proceedings of INFINITY'96*, Nr. MIP-9614 aus Technical report series of the University of Passau. University of Passau, 1996.

[GH91] J. F. Groote und H. Hüttel. Undecidable equivelences for basic process algebra. Technical Report ECS-LFCS-91-169, University of Edinburgh, August 1991.

[HJM94] Y. Hirshfeld, M. Jerrum und F. Moller. A polynomial algorithm for deciding bisimulation of normed context free processes. Technical report, LFCS report series 94-286, Edinburgh University, 1994.

[HT90] D.T. Huynh und L. Tian. On deciding readiness and failures equivelences for processes. Technical Report DTDC-31-90, University of Texas at Dallas, 1990.

[HU79] J.E. Hopcroft und J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 1979.

[Jan93] P. Jančar. Decidability Questions for Bisimilarity of Petri Nets and some related problems. Technical Report ECS-LFCS-93-261, Edinburgh University, April 1993.

[Jan94] P. Jančar. Decidability questions for bisimilarity of Petri nets and some related problems. In *Proceedings of STACS'94*, Bd. 775 aus *LNCS*. Springer Verlag, 1994.

[Jan95] P. Jančar. Undecidability of Bisimilarity for Petri Nets and some related problems. *Theoretical Computer Science*, 1995.

[May96a] Richard Mayr. Combining Petri Nets and PA-Processes. TU-München, 1996.

[May96b] Richard Mayr. Model Checking PA-Processes. Technical Report TUM-I9640, TU-München, December 1996.

[May96c] Richard Mayr. Semantic Reachability for Simple Process Algebras. In B. Steffen und T. Margaria, Hrsg., *Proceedings of INFINITY'96*, Nr. MIP-9614 aus Technical report series of the University of Passau. University of Passau, 1996.

[May96d] Richard Mayr. Some Results on Basic Parallel Processes. Technical Report TUM-I9616, TU-München, March 1996.

[May96e] Richard Mayr. A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ-Calculus. Technical Report TUM-I9634, TU-München, October 1996.

[May96f] Richard Mayr. Weak Bisimulation and Model Checking for Basic Parallel Processes. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'96)*, Nr. 1180 aus LNCS. Springer Verlag, 1996.

[May97] Richard Mayr. Tableau Methods for PA-Processes. In Didier Galmiche, Hrsg., *Analytic Tableaux and Related Methods (TABLEAUX'97)*, Nr. ? aus LNAI. Springer Verlag, 1997.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[MN94] Richard Mayr und Tobias Nipkow. Higher-Order Rewrite Systems and their Confluence. Technical Report TUM-I9433, TU-München, August 1994. To appear in TCS.

[Tau89] D. Taubner. Finite representation of CCS and TCSP programs by automata and Petri nets. In *LNCS 369*. Springer Verlag, 1989.

1.3.15 Objektorientierte Spezifikation verteilter Systeme

Teilprojekt: Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme
Markus Podolsky

1 Einleitung

Verteilte Systeme spielen in der Anwendungsentwicklung eine immer wichtiger werdende Rolle. Dazu haben Technologien wie z.B. Common Object Request Broker Archtitecture (CORBA), Distributed Component Object Model (DCOM), Distributed Computing Environment (DCE) und das Aufkommen des World Wide Web (WWW) oder Standardisierungen wie z.B. Open Distributed Processing (ODP) entscheidend beigetragen. Aber neben der Bereitstellung einer Ablaufumgebung braucht der Anwender insbesondere Unterstützung bei der Software-Entwicklung.

Verteilte Systeme sind von Natur aus komplex. So können mehrere Kontrollflüsse gleichzeitig vorhanden sein, wodurch beispielsweise Synchronisationsmechanismen erforderlich werden. Dies gilt es zu erkennen und zu modellieren. Des weiteren müssen die Topologie des Systems und die damit in Beziehung stehenden Aspekte in den Entwurf aufgenommen werden. Wichtig ist, daß der Entwurf sämtliche Eigenschaften und Gegebenheiten eines verteilten Systems beinhaltet.

Die Objektorientierung hat sich als geeignetes Modell für verteilte Systeme erwiesen. Objekte bzw. Objektgruppen entsprechen Verteilungseinheiten, die ihrer Umgebung eine Schnittstelle zur Verfügung stellen. Die Kommunikation wird auf Methodenaufrufe der Objekte untereinander abgebildet. Im folgenden wird eine objektorientierte Spezifikationsmethode für den Entwurf verteilter Systeme vorgestellt. Dabei spielt die Anwendungsnähe eine große Rolle.

Im nächsten Kapitel werden gängige Entwurfsmethoden erläutert und ihre Eignung für die Anwendungsentwicklung bewertet. Aus dieser Beurteilung wird ein Ansatz zur objektorientierten Spezifikation verteilter Systeme hergeleitet. Abschließend werden im Ausblick zukünftige Arbeitsgebiete dargestellt.

2 Motivation

Im folgenden wird ein kurzer Überblick über gängige Vorgehensweisen beim Software-Entwurf gegeben. Zudem wird deren Bedeutung und Einsatz bei der Anwendungsentwicklung erläutert. Dabei wird eine Klassifizierung in objektorientierte Entwurfsmethoden und formale Spezifikationsmethoden vorgenommen.

In den letzten Jahren haben sich objektorientierte Entwurfsmethoden, wie beispielsweise OMT [RBP+91], UML [BRJ97], OOSE [JCJO92], Booch-Method [Boo94], Fusion [CAB+94], Martin/Odell [MO92], Shlaer/Mellor [SM92] oder Wirfs-Brocks [WBWW90] in der Anwendungsentwicklung etabliert. Sie verfolgen den Ansatz die reale Welt als System von interagierenden Objekten zu beschreiben. Jedes Objekt stellt der Umgebung eine bestimmte Schnittstelle bereit. Die Objektinteraktion erfolgt durch die Möglichkeit, Methoden von Schnittstellen anderer Objekte aufzurufen.

Um die Systembeschreibung übersichtlich zu halten, modellieren objektorientierte Entwurfsmethoden die reale Welt aus drei verschiedenen Sichtweisen:

- Sicht auf die Datenstrukturen (Objektmodell)
  Diese Sichtweise beeinhaltet die Deklarationen der Objekte, die Vererbungshierarchien und sonstige Beziehungen zwischen den Objekten, wie z.B. Aggregation. In der Regel erfolgt die Modellierung dieser Sichtweise mittels erweiterter Entity-Relationship-Diagramme.

- Sicht auf das Verhalten (Verhaltensmodell)
  Neben dem Verhalten jedes einzelnen Objekts ist insbesondere die Interaktion der Objekte untereinander von Interesse. Für die Modellierung dieser Aspekte stehen beispielsweise Zustandsübergangsdiagrame, Message Sequence Charts, Objektinteraktionsdiagramme usw. zur Verfügung.

- Sicht auf die Architektur (Architekturmodel)
  Diese Sichtweise betrachtet die Architektur des Systems. Sie kann einerseits funktional, z.B. mittels Datenflußdiagrammen, beschrieben werden. Andererseits läßt sich die Architektur des Systems auch rekursiv durch die Komposition der Architekturen der beteiligten Komponenten beschreiben, welche letztendlich aus einzelnen Objekten bzw. Objektgruppen bestehen. Hierbei spricht man von einer dekompositionalen Beschreibung der Architektur.

Im folgenden werden prinzipielle Vor- und Nachteile objektorientierter Entwurfsmethoden erläutert:

+ Objektorientierte Entwurfsmethoden verwenden größtenteils graphische Notationen und ermöglichen dadurch übersichtliche und intuitiv verständliche Modellierungen.

+ Für die Umsetzung der Anforderungen in ein Analysemodell stellen sie Vorgehensweisen bereit. Dies vereinfacht das Auffinden und Beschreiben von Objekten und deren Interaktion.

+ Sie sind relativ leicht verständlich und erlernbar.

– Wesentliche Eigenschaften des Systems bzw. der Abläufe darin können mit ihnen nur informell beschrieben werden. Daher verfügen sie nicht über die notwendige formale Grundlage, um den erstellten Entwurf zu validieren und zu verifizieren.

– Für die Modellierung verteilter Systeme bieten sie zwar zum Teil Konstrukte an, um beispielsweise die Allokation der Objekte oder die Aufrufsemantik (synchron, asynchron) festzulegen. Aber Eigenschaften wie etwa gegenseitiger Ausschluß sind damit noch nicht spezifizierbar.

Für die Spezifikation von verteilten Systemen stehen auf der anderen Seite formale Methoden zur Verfügung. Sie ermöglichen eine eindeutige Beschreibung des Systems. Als Beispiele seien Focus [BDD+93], SDL [IT93], Lotos [Tur93], Estelle [Tur93], Petri-Netze [Rei86], algebraische Spezifikation und temporale Logik erwähnt. Je nach Mächtigkeit dieser Methoden lassen sich damit sämtliche Systemaspekte oder nur Ausschnitte davon beschreiben. Beispielsweise läßt sich mit temporaler Logik lediglich das Verhalten eines Systems, nicht aber dessen Architektur beschreiben. Aufgrund der jeweiligen Modellierungstechnik erleichtern die Methoden entweder die Modellierung der operationellen oder die Modellierung der denotationellen Aspekte des Systemverhaltens. Die jeweils anderen Eigenschaften des Systems lassen sich in der Regel nur umständlich beschreiben. Als Beispiele seien die Modellierung des gegenseitigen Ausschlusses in SDL bzw. die Modellierung operationeller Aspekte mittels temporale Logik erwähnt.

Als Vor- und Nachteile formaler Spezifikationsmethoden lassen sich folgende Punkte aufführen:

+ Es existieren Methoden und Werkzeuge, um die Korrektheit der Spezifikation zu überprüfen.

+ Die Spezifikation kann als Grundlage für Tests verwendet werden.

– In der Regel sind formale Methoden nicht leicht erlernbar und zudem sind die mit ihnen erstellten Modellierungen meist unübersichtlich und kaum intuitiv verständlich. Das ist wohl der Grund, weshalb sie bei der Anwendungsentwicklung kaum verwendet werden.

Aufbauend auf diese Beurteilungen wird im nächsten Kapitel eine anwendungsorientierte Spezifikationstechnik für verteilte System charakterisiert und dargestellt.

3 Objektorientierte Spezifikation

Betrachtet man die Bedürfnisse der Anwendungsentwickler, so ergibt sich die Nachfrage nach einer Entwicklungsmethode, welche den Entwurf intuitiv verständlicher Modelle erlaubt, die zudem die notwendige formale Grundlage besitzen, um die Überprüfung des Programms anhand seiner Spezifikation zu ermöglichen. Aus der vorherigen Gegenüberstellung von objektorientierten Entwurfsmethoden und formalen Spezifikationsmethoden wurden die Stärken und Schwächen beider Ansätze aufgezeigt. Im Prinzip müßte man die Stärken dieser Ansätze kombinieren, um eine Entwicklungsmethode zu erhalten, welche den Bedürfnissen der Entwickler entspricht.

Im vorliegenden Bericht wird eine Methode zur objektorientierten Spezifikation verteilter Systeme vorgestellt, welche den oben genannten Bedürfnissen gerecht werden soll. Dabei liegt der Schwerpunkt auf einer graphischen Notation, welche eine formale Beschreibung des verteilten Systems erlaubt.

Bei der Entwicklung einer Spezifikationsmethode ergeben sich im wesentlichen folgende Schwerpunkte:

- Ermittlung der Anforderungen an verteilte Systeme

- Wahl einer formalen Modellierung

Diese Punkte werden in den nächsten Abschnitten noch genauer erläutert.

3.1 Anforderungen an verteilte Systeme

Bei der objektorientierten Modellierung betrachtet man das zu erstellende verteilte System als Menge von eigenständigen, interagierenden Objekten. Bei der Entwicklung eines verteilten Systems spielt neben der Verteilung der Objekte und deren Aufbau, also der Architektur des Systems, insbesondere die Beschreibung deren Interaktion eine große Rolle. Im Unterschied zu sequentiellen Systemen sollen die drei Sichtweisen beim objektorientierten Entwurf beispielsweise Beschreibungsmöglichkeiten für folgende Aspekte bieten:

- Objektmodell
  Das Objektmodell soll die Kennzeichnung aktiver Objekte, also Objekte mit eigenem Kontrollfluß erlauben. Zusätzlich soll für jede Klasse angegeben werden können, wie es eintreffende Anfragen bearbeitet. Beispielsweise könnte man hierfür eine FIFO-Semantik zugrunde legen oder Prioritäten für Anfragen einführen.

- Verhaltensmodell
  Das Verhaltensmodell beschreibt das Verhalten der Objekte. Dabei betrachtet man sowohl deren Interaktion als auch deren internes Verhalten, also die Aktionen, die durch einen Methodenaufruf angestoßen werden. Es muß mächtig genug sein, um die möglichen Kontrollflüsse in verteilten System beschreiben zu können.
  Ein wesentlicher Bestandteil des Verhaltensmodells bei verteilten Systemen ist die Nebenläufigkeitskontrolle. Hier soll beschrieben werden können, wie die Synchronisation zwischen den interagierenden Objekten stattfindet. Dabei lassen sich zwei Arten der Synchronisation aufführen. Zum einen kann die Synchronisation innerhalb eines Objekts erforderlich sein, wenn z.B. mehrere Anfragen bei einem gemeinsam benutzten Objekt vorliegen. Dabei soll z.B. angegeben werden können, daß bestimmte Methoden bzw. Teile von Methoden unter gegenseitigem Ausschluß ablaufen sollen. Zudem können Methoden bzw. Teile von Methoden in lesend und schreibend klassifiziert werden, wodurch die Synchronisation im Sinne eines Leser-Schreiber-Problems erfolgen soll. Brauchen andererseits mehrere Objekte zugleich mehrere gemeinsam benutzte Ressourcen, so muß die Synchronisation nach einem Protokoll erfolgen. Dieses Protokoll muß zum einen spezifiziert werden und zum anderen muß definiert werden, welche Objekte sich wann danach zu richten haben.
  Des weiteren soll beispielsweise die Aufrufsemantik bei Methodenaufrufen spezifizierbar sein oder die Ausführung von Programmteilen als Transaktion gekennzeichnet werden können.

- Architekturmodell
  Auch das Architekturmodell wird grundlegend zu erweitern sein. Es muß die Zuordnung von Objekten zu Verteilungseinheiten erlauben, wobei ggf. die Verteilungseinheiten physischen Rechnerknoten zuordnet werden. Neben der Allokation und Replikation von Objekten soll es auch eine Möglichkeit bieten, die Migration von Objekten zu beschreiben. Die Replikation von Objekten kann Auswirkungen auf deren internes Verhalten haben. Bei einem schreibenden Zugriff auf ein Replikat muß die Konsistenz der Replikate untereinander erhalten bleiben, wofür wiederum die nötigen Mechanismen zu beschreiben sind.

Nach der Angabe von Anforderungen an verteilte Systeme wird im nächsten Abschnitt ein Konzept vorgestellt, wie diese Aspekte formal modelliert werden können.

3.2 Formale Modellierung

Mittels einer formalen Spezifikation können die oben erwähnten Anforderungen an verteilte Systeme eindeutig beschrieben werden. Um einen übersichtlicheren Entwurf zu erhalten, sollen diese Anforderungen explizit in der Modellierung auftreten und sich nicht implizit aus Teilen des Entwurfs ergeben. Betrachtet man oben genannte Anforderungen genauer, so erkennt man, daß viele davon einfacher mit denotationellen als mit operationellen Beschreibungsmitteln zu formulieren sind. Andererseits lassen sich die Interaktionen der Objekte untereinander und die Aktionen, die beim Aufruf von Methoden ausgeführt werden, einfacher mit operationellen Beschreibungsmitteln formulieren. Dies legt nahe beim Entwurf eine Kombination von operationellen und denotationellen Beschreibungstechniken zu verwenden.

Im folgenden wird eine derartige, objektorientierte Beschreibungstechnik für verteilte Systeme vorgestellt. Dabei wird das zu erstellende System aus den drei oben angegebenen Sichtweisen betrachtet, welche nun der Reihe nach dargestellt werden.

Für die Beschreibung des Objektmodells, welches die Deklarationen der Klassen und deren Beziehungen untereinander enthält, werden erweiterte Entity-Relationship-Diagramme benutzt, wie sie z.B. bei OMT verwendet werden. Zusätzlich werden oben genannte Erweiterungen des Objektmodells, wie etwa die Beschreibung der Service Invocation Semantic, ins ER-Modell aufgenommen.

Mit dem Verhaltensmodell wird die Interaktion der verteilten Objekte und das Verhalten jedes einzelnen Objektes beschrieben. Das Verhalten eines Objekts wird durch seine Methoden bestimmt. Die Interaktion der Objekte erfolgt durch die Kommunikation der Objekte untereinander, welche durch den Aufruf von Methoden anderer Objekte erreicht wird. Für das Verhaltensmodell wurde eine Beschreibungstechnik entwickelt, die sich an eine Kombination von SDL und MSC [IT94] anlehnt, sich aber dennoch stark davon unterscheidet. Zuerst wird die Beschreibungstechnik vorgestellt und anschließend werden die Unterschiede zu SDL und MSC erläutert.

Für jedes Objekt werden seine Methoden in einem Object Behaviour Chart (OBC) beschrieben. In Anlehnung an MSC wird das Verhalten einer Methode an einer „Zeitachse" angegeben. Die Zeitachse verläuft von oben nach unten. Ist die Zeitachse unterbrochen, so bedeutet dies, daß darin auftretende Aktionen zeitlich nicht geordnet sind, sie können parallel ausgeführt werden. Für die Beschreibung des Verhaltens stehen gebräuchliche operationelle Aspekte zur Verfügung, wie beispielsweise bedingte Verzweigung, Iteration, Aufruf einer Objektmethode, Erzeugen bzw. Löschen eines Objekts.

In dieser Form können mit den OBCs die operationellen Aspekte modelliert werden. Die Beschreibung denotationeller Systemeigenschaften wird wie auch bei [FP97] durch die Erteilung von Eigenschaften an Klassen bzw. Objekte erreicht. Dabei wird der Weg eingeschlagen, daß beim Entwurf eine Menge von vordefinierten Eigenschaften zur Verfügung stehen. In Abbildung 1 sind die Sprachkonstrukte von Object Behaviour Charts dargestellt.

Aus diesen Eigenschaften soll evtl. Code generiert werden können, so daß der Entwickler lediglich die Eigenschaften ins Modell aufzunehmen braucht und er sich nicht darum

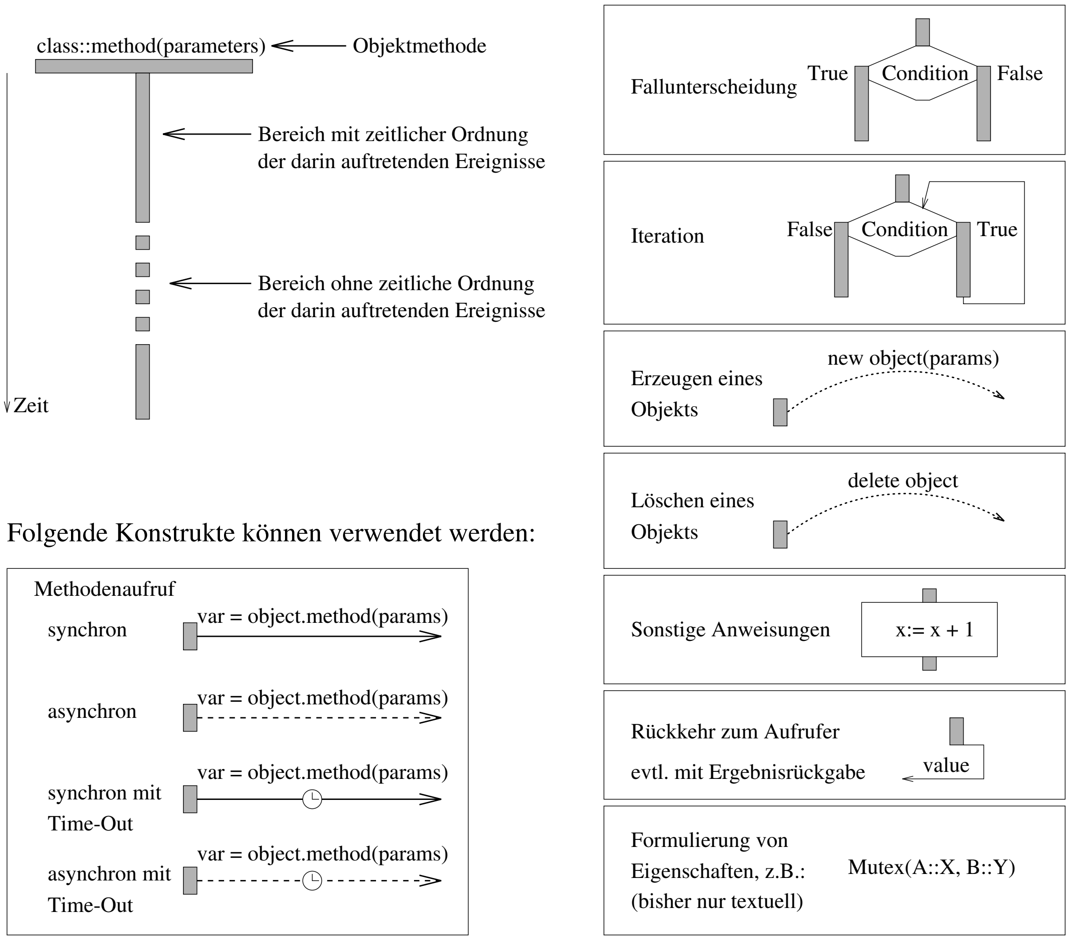


Abbildung 1: Sprachkonstrukte von Object Behaviour Charts

kümmern muß, wie er diese Eigenschaft tatsächlich zu implementieren hat. Er gibt also nur an, was das System leisten soll, und nicht wie es erreicht werden soll. Beispielsweise kann er an eine Auswahl von Objektmethoden die Eigenschaft „gegenseitiger Ausschluß" erteilen, d.h. diese Methoden dürfen nur unter gegenseitigem Ausschluß ausgeführt werden. Insbesondere braucht er nicht mehr angeben, wie der gegenseitige Ausschluß erreicht wird. Letztendlich sollen dem Entwickler eine bestimmte Menge von Basiseigenschaften für die Modellierung zur Verfügung stehen, die er bei Bedarf in den Entwurf aufnehmen kann.

Object Behaviour Charts können auch zusammengesetzt werden. Dabei werden die OBCs der aufgerufenen Objektmethoden an die Aufrufpfeile gelegt. In Abbildung 2 wird ein kleines Beispiel dargestellt.

Im folgenden werden die Unterschiede dieser Beschreibungstechnik zur Kombination von SDL und MSC aufgezeigt. Bei SDL und MSC, welche ursprünglich zur Spezifikation von Telekommunikationssystemen entwickelt wurden, liegt der Schwerpunkt der Verhaltensmodellierung auf der Beschreibung von Prozessen, welche Nachrichten austauschen. SDL-Prozesse können während ihres Ablaufs Nachrichten senden und empfangen. Das ist ein wesentlicher Unterschied zur objektorientierten Sichtweise. Bei der objektorientierten Modellierung entsprechen Nachrichten dem Aufruf einer Objektmethode. Eine Objektmethode kann während ihres Ablaufs nur andere Objektmethoden aufrufen, was der Sende-

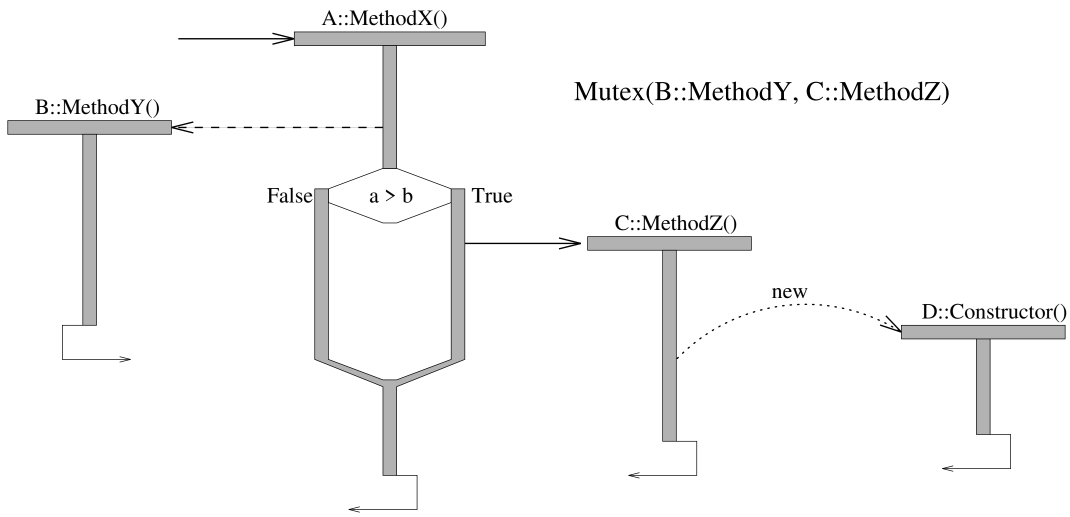


Abbildung 2: Beispielszenario

operation in SDL entspricht. Für die Empfangsoperation in SDL gibt es beim objektorientierten Entwurf keine Entsprechung. Während mittels MSC und SDL nur operationelle Aspekte modellierbar sind, können mit der hier vorgestellten Methode zusätzlich denotationelle Aspekte, wie etwa der gegenseitige Ausschluß, als Eigenschaften direkt in die Modellierung aufgenommen werden. Die Eigenschaft tritt somit explizit im Entwurf auf. Bei Verwendung von SDL und MSC muß man selbst eine operationelle Beschreibung dieser Eigenschaft angeben, was umständlich und fehleranfällig ist. Zudem geht der Bezug dieser Eigenschaft zum Entwurf verloren, weil sie nicht explizit aus dem Entwurf ablesbar ist, sondern sich implizit aus Teilen der SDL-Spezifikation ergibt.

Im nächsten Abschnitt wird neben der Zusammenfassung ein kurzer Ausblick über zukünftige Arbeitsgebiete gegeben.

4 Zusammenfassung und Ausblick

In diesem Bericht wurden eine Möglichkeit zur objektorientierten, formalen und für die Anwendungsentwicklung geeigneten Modellierung verteilter Systeme vorgestellt. Die dabei eingeführten Object Behaviour Charts können zur Spezifikation des Verhaltens von verteilten Objekten verwendet werden. Neben der operationellen Modellierung des Verhaltens erlauben sie auch die Erteilung von Eigenschaften an Objekte bzw. Objektmethoden. Für die angesprochenen Eigenschaften sind noch deren jeweilige Umsetzungen festzulegen und anzugeben. Zusätzlich sind noch die Konstrukte zur Beschreibung des Objekt- und des Architekturmodells zu vervollständigen und deren graphische Repräsentation festzulegen. Aufbauend auf diese Modelle soll eine Vorgehensweise für den objektorientierten Entwurf verteilter Systeme erarbeitet werden, um den Entwickler bei typischen Problemen im Zusammenhang mit dem Entwurf verteilter Systemen zu unterstützen. Dabei sind Entwurfskriterien für wichtige Aspekte von verteilten Systemen anzugeben, wie etwa die Wahl der Verteilungseinheiten, das Auffinden von Objekten, bei denen synchronisiert werden muß oder die Wahl eines geeigneten Synchronisationsverfahrens. Zusätzlich sind die Vorgehensweise und die Techniken zur Modellierung anhand praxisrelevanter verteilter Anwendungen zu erproben und zu bewerten.

Literatur

[BDD+93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. Gritzner und Weber R. The Design of Distributed Systems - An Introduction to FOCUS. Technical report, Technische Universität München, Institut für Informatik, 1993.

[Boo94] G. Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings, Menlo Park, CA, 2nd. Auflage, 1994.

[BRJ97] G Booch, J. Rumbaugh und I. Jacobson. *The Unified Modeling Language, Version 1.0.* 1997.

[CAB+94] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes und P. Jeremaes. *Object-Oriented Development: THE FUSION METHOD.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[FP97] M. Frey und M. Pucko. Formal Specification of CSCW Applications with Concurrent Abstract Data Types. *to appear in Journal of Systems Architecture*, 1997.

[IT93] ITU-T. *Recommendation Z.100, Specification and Description Language (SDL).* ITU, 1993.

[IT94] ITU-T. *Recommendation Z.120, Message Sequence Charts (MSC).* ITU, 1994.

[JCJO92] I. Jacobson, M. Christerson, P. Jonsson und G. Overgaard. *Object Oriented Software Engineering.* Addison-Wesley, Reading, MA, 1992.

[MO92] J. Martin und J. Odell. *Object-oriented Analysis and Design.* Prentice Hall, Englewood Cliffs, NJ, 1992.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen. *Object-Oriented Modelling and Design.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[Rei86] W. Reisig. *Petrinetze.* Springer, Berlin, 1986.

[SM92] S. Shlaer und S. Mellor. *Object lifecycles: modelling the world in states.* Yourdon Press computing series, Englewood Cliffs, N, 1992.

[Tur93] K. J. Turner. *Using Formal Description Techniques - An Introduction to Estelle, Lotos and SDL.* John Wiley and Sons, 1993.

[WBWW90] R. Wirfs-Brock, B. Wilkerson und L. Wiener. *Designing Object-Oriented Software.* Prentice Hall, Englewood Cliffs, NJ, 1990.

1.3.16 Verteilte attributierte Graphersetzungssysteme

Teilprojekt: *Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für die verteilte Programmierung*

Boris Reichel

1 Motivation

Zur Beschleunigung der Berechnung massiver Problemstellungen werden seit einiger Zeit parallele und verteilte Rechnerarchitekturen eingesetzt. Die hierbei erzielbaren Effizienzsteigerungen hängen stark von der Struktur der zugrundeliegenden Problemstellung ab. Generell gilt: Je besser sich ein Problem in voneinander unabhängig berechenbare Teilprobleme zerlegen läßt, um so höher die erwartbare Beschleunigung.

Graphersetzungssysteme werden schon seit längerem zur deklarativen Beschreibung komplexer strukturierter Systeme verwendet. Die dieser Spezifikationsform **inheränte Parallelität** kann bei der Berechnung massiver Problemstellungen durch verteilte Ausführung entsprechender Teilberechnungen zur Effizienzsteigerung ausgenutzt werden.

Die Berechnung eines Graphersetzungssystems besteht aus einer Folge von Graphersetzungen. Sie entsprechen den parallel auszuführenden Teilberechnungen. In massiven Graphen existieren im allgemeinen eine Menge verschiedener Untergraphen die zu einem gegebenen Zeitpunkt potentiell ersetzt werden können. Da die Reihenfolge der Ersetzungen im allgemeinen nicht durch das Graphersetzungssystem vorgegeben wird können alle unabhängigen Ersetzungen parallel durchgeführt werden. Die verteilte Realisierung eines allgemeinen Graphersetzungssystems bietet somit eine **universelle verteilte Plattform** für beliebige durch Graphersetzungssysteme spezifizierbare Probleme.

Im folgenden werden einige exemplarische Beispiele für die vielfältigen Einsatzmöglichkeiten von Graphersetzungssystemen aufgezählt. In [BA78] werden Graphersetzungssysteme zur Spezifikation der Transformationen während des **Entwurfprozesses von relationalen Datenbanken** verwendet. Das Ersetzungssystem beschreibt hier die *operationelle Semantik* der entsprechenden Entwurfsschritte. Der Entwurfsprozess kann somit in einem formalen Rahmen betrachtet werden. Die **Codeoptimierung** bei der Übersetzung von Programmiersprachen beruht im wesentlichen auf Baumtransformationen. In [Nag80] wird die Baumdarstellung des Zwischencodes durch eine graphartige verallgemeinert und die Optimierungsschritte durch entsprechende Graphersetzungen beschrieben. [Fu82] beschäftigt sich mit **syntaktischer Musterverarbeitung**. Graphgrammatiken werden hier für die Spezifikation der hierarchischen Struktur von komplexen Mustern, wie zum Beispiel chinesische Schriftzeichen oder Gitterdarstellungen von dreidimensionalen Objekten verwendet. Die Eignung zur **Spezifikation von objektorientierten Systemen** wird in [JR91] untersucht. Die Spezifikation des verwendeten Objektmodells durch Graphersetzungsregeln bietet den Vorteil, daß Untersuchungen des dynamischen Verhaltens der Objekte auf Untersuchungen des zugrundeliegenden Graphersetzungssystems zurückgeführt werden können.

2 Klassifikation von Graphersetzungssystemen

Graphersetzungssysteme werden durch ein Produktionensystem und eine Ersetzungssemantik (Ableitungsbegriff) definiert. Für die Ersetzungssemantik existieren folgende Ausprägungen:

Einzelersetzung : Der Ableitungsbegriff ist so definiert, daß sich der zu verarbeitende Graph immer nur durch Anwendung **genau einer** Ersetzungsregel verändert. Hierbei wird ein zum Referenzgraph isomorpher Untergraph durch einen neuen Graphen ersetzt. Der Rest bleibt unverändert.

Überdeckungsersetzung : Im Gegensatz zu Einzelersetzung ist der Ableitungsbegriff derart definiert, daß der zu bearbeitende Graph in jedem Ersetzungsschritt vollständig überschrieben wird. Hierzu muß der Graph durch isomorphe Bilder der Referenzgraphen vollständig überdeckt werden, die **alle gleichzeitig** durch den jeweils zugehörigen Graphen ersetzt werden.

Beide Ersetzungssemantiken eignen sich zur Spezifikation komplexer strukturierter verteilter Systeme. Sie führen allerdings zu völlig unterschiedlichen Sichtweisen. Die *Überdeckungsersetzung* beschreibt ähnlich zu zellularen Automaten *synchron getaktete verteilte Systeme*, *Einzelersetzung* dagegen *ungetakteter verteilter Systeme*. Da in diesem Teilprojekt verteilte Systeme in einer heterogenen Umgebung spezifiziert werden sollen, wurden die Untersuchungen auf Graphersetzungssysteme mit Einzelersetzungssemantik eingeschränkt.

Die für die Spezifikation von Graphersetzungssystemen verwendeten Grammatiken können in ihrer Form und Ausdrucksmächtigkeit erheblich variieren. Sie unterscheiden sich prinzipiell in folgende Punkte.

Graphen : In einigen Arbeiten werden die verarbeitbaren Graphen - d.h. die linken Seiten der Produktionen - zu Gunsten einer effizienten Mustererkennung eingeschränkt. In [Fu95] werden zum Beispiel *gerichtete Graphen mit Wurzelknoten* betrachtet.

Einbettungsbeschreibung : Wenn jeder Produktion eine individuelle Einbettungsbeschreibung zugeordnet wird, wird sie als *explizit* bezeichnet. Im Gegensatz hierzu verwenden bei der *impliziten* alle Produktionen die gleiche Einbettungsbeschreibung. Wenn die Einbettung durch einen Kontextgraphen angegeben wird, wird sie als *statisch* bezeichnet. Da der Kontextgraph für die Einbettung des neuen Graphen benötigt wird, kann er bei der Mustererkennung zusammen mit dem Referenzmuster gesucht werden. Der Kontextgraph bekommt dann implizit die Bedeutung einer Kontextbedingung. *Dynamische* Einbettungsbeschreibungen sind Terme die angeben, wie der neue Graph mit seiner Umgebung verbunden werden muß. Sie müssen explizit nach Erkennen eines Referenzmusters ausgewertet werden. Im allgemeinen sind sie *tiefenbeschränkt*, d.h. es wird nur eine bestimmte Umgebung des Referenzmusters untersucht.

Kontextbedingungen : Sie müssen erfüllt sein, wenn eine Produktion für einen Ersetzungsschritt verwendet werden soll. Wenn keine Kontextbedingungen angegeben sind hängt die Produktionsanwendung nur vom Erkennen des Referenzgraphen ab. Wenn nur die Struktur des Kontextes relevant ist so spricht man von einer *strukturellen* Kontextbedingung. Die oben beschriebene statische Einbettung kann als eine solche betrachtet werden. In attributierten Graphersetzungssystemen hat man die Möglichkeit einzelne Ableitungsschritte abhängig von Attributwerten durchzuführen. Derartige Kontextbedingungen werden *attributabhängig* genannt.

Die in der Literatur verwendeten Graphersetzungssysteme lassen sich unter Vernachlässigung von Einbettungsbeschreibung und Kontextbedingungen im wesentlichen auf folgende "Grundformen" zurückführen.

Node Label Controlled (NLC): Die Grammatik beschreibt die Ersetzung einzelner Knoten (abhängig von deren Typ). Sie werden häufig auch als *kontextfrei* bezeichnet.

Edge Label Controlled (ELC): Die Grammatik beschreibt die Ersetzung einzelner Kanten (abhängig von deren Typ). Die Kanten können hier mehrere Start- und/oder Zielknoten besitzen (*Hyperkanten*).

Handle Controlled (HC) : Die Grammatik beschreibt die Ersetzung ganzer, zu einem gegebenen Referenzgraph isomorpher Untergraphen.

Programmed: Neben der Grammatik werden zusätzlich Abhängigkeiten zwischen den einzelnen Produktionen angegeben. Sie schränken die Reihenfolge der Anwendung von Produktionen ein.

Im Rahmen des Teilprojekts werden *Handle-Ersetzungssysteme* untersucht. Durch ihre starke Ausdrucksmächtigkeit (verallgemeinerte Chomsky-Systeme) eignen sie sich als Spezifikationsgrundlage für eine große Klasse von Systemen. Knoten werden hier im allgemeinen zur Modellierung von Systemkomponenten verwendet. Abhängigkeiten zwischen diesen Komponenten werden durch entsprechend markierte Kanten ausgedrückt. Die Folge der durch das Graphersetzungssystem erzeugten Graphen entspricht dann der Folge von Systemzuständen, die bei der Systemberechnung durchlaufen wird.

Graphgrammatiken beschreiben somit im Gegensatz zu imperativen Paradigmen nicht die Verarbeitungsschritte die zur Problemlösung führen, sondern die für die Problemlösung benötigten Systemkomponenten - d.h. Daten, Prozesse,... - und deren Abhängigkeit, sowie deren Veränderungen während der Berechnung. Sie bieten somit die *deklarative Spezifikation* von Problemen auf hohem Abstraktionsniveau. Die in ihnen enthaltenen Datenabhängigkeiten können für eine effiziente Verteilung a priori analysiert werden.

In vielen Systemen ist Berechnung gleichbedeutend mit der Veränderung der Beziehungen zwischen einer invarianten Menge von Komponenten. So bleiben die Atome von Makromolekülen unverändert, obwohl sich ihre Bindungen untereinander durch erwärmen oder abkühlen verändern. Damit die Graphgrammatik dieses Verhalten modellieren kann, wird das Graphersetzungssystem als eine *Mischform aus Knoten- und Kantenersetzung* definiert.

Die durch die Grammatik spezifizierten lokalen Zustandsänderungen des Systems hängen im allgemeinen von der Struktur und den Attributen des Kontextes des zu ersetzenden Subgraphen ab. Es hat sich gezeigt, daß der zu berücksichtigende Kontext in den meisten

Problemstellungen beschränkt ist. Für eine effiziente Realisierung der Kontext- und Attributauswertung werden Grammatiken mit *statischer Einbettungsbeschreibung* verwendet. Die Einschränkung hat den Vorteil, daß die Auswertung des Kontextes in die Mustererkennungsphase des Ersetzungssystems integriert werden kann.

3 Beispiel

Im folgenden wird das verwendete Graphersetzungssystem anhand des klassischen verteilten Problems der "dinierenden Philosophen" informell dargestellt. Für eine ausführlichere Diskussion der Systemeigenschaften sei auf [Loy92, Tan95] verwiesen.

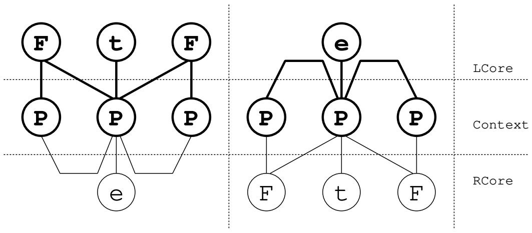


Abbildung 1: Grammatik der dinierenden Philosphen

Abbildung 1 zeigt die, aus zwei Produktionen bestehende, das Problem spezifizierende Grammatik. Die Knotenmarkierungen haben folgende Bedeutungen: `P` ≅ Philosoph, `F` ≅ Gabel, `e` ≅ Zustand "essend" und `t` ≅ Zustand "denkend".

Jeder Philosoph muß mit genau einem Zustand assoziiert sein. Er kann abhängig von seinem aktuellen Zustand die Aktionen *starte essen* und *beende essen* durchführen. Die linke Produktion beschreibt die Aktion *starte essen*. Sie besteht aus den drei miteinander verbundenen Untergraphen `LCore`, `Context` und `RCore`. Wobei `LCore` den zu ersetzenden Graphen, `Context` seinen minimalen Kontextgraphen und `RCore` den ersetzenden Graphen beschreiben. Die in Graphgrammatiken benötigte Einbettungsvorschrift ist implizit durch die Kanten zwischen `LCore` und `Context` bzw. `RCore` und `Context` definiert. Damit ein Philosoph die Aktion *starte essen* durchführen kann, muß er sich im Zustand "denkend" befinden und auf die Gabeln zu seiner Linken und Rechten zugreifen können. Nach der Durchführung befindet sich der Philosoph im Zustand "essend" und die Gabeln sind für die anderen Philosophen nicht mehr zugreifbar. In der Produktion wird dieses Verhalten durch das Löschen und Erzeugen der entsprechenden Knoten und Kanten modelliert. Die zweite Produktion modelliert die Aktion *beende essen*. Vor Durchführung der Aktion muß sich der Philosoph im Zustand "essend" befinden. Nachdem der Philosph seine Gabeln wieder zwischen sich und seine beiden Nachbarn gelegt hat kehrt er wieder in den Zustand "denkend" zurück.

Abbildung 2 zeigt eine lokale Zustandsänderung des Systems "dinierende Philosophen". Sie beschreibt die Durchführung der Aktion *starte essen* durch den Philosophen in der linken oberen Ecke des Zustandsgraphen. Der im linken Graphen fett gedruckte Bereich entspricht dem erkannten, zur linken Seite der Produktion isomorphen, Untergraphen der entsprechend der Produktion verändert wird.

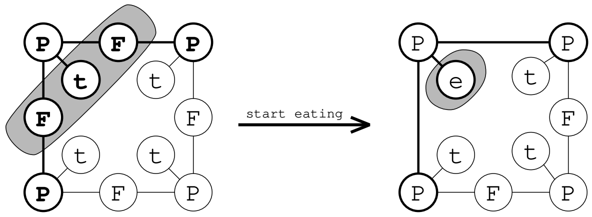


Abbildung 2: Eine Aktion des Systems "dinierende Philosophen"

4 Problemstellung

Im folgenden werden die Problemstellungen, die im Rahmen des Teiprojekts behandelt werden genauer definiert. Ein Graphersetzungssystem Γ ist definiert durch ein Produktionensystem P und einen Ableitungsbegriff $\rightarrow$. Die Produktionen spezifiziert hierbei das Systemverhalten eines speziellen Systems. Der Ableitungsbegriff gibt hingegen an, wie die Produktionen zur Veränderung des Systemzustands herangezogen werden. $g \xrightarrow{p} g'$ steht für die Aussage, daß der Graph g durch Anwenden der Produktion p in den Graphen g' überführt werden kann.

Definition 1 (Graphersetzungsproblem)
Das *Graphersetzungsproblem* ist definiert durch das Tupel (P, g), wobei $P = \{p_1, p_2, ..., p_n\}$ die Menge aller *Produktionen (Ersetzungsregeln)* und g der zu verarbeitende *initiale Eingabegraph (initialer Systemzustands)* ist.

Seine Lösung ist die u.U. unendliche Folge von *Zustandsgraphen* $L(P, g) =_{def} (g_i)_{i=1,2,...}$ mit $g_1 = g$ und $g_i \xrightarrow{p} g_{i+1}$ mit $p \in P$.

Vor Durchführung eines Ersetzungsschritts $g \xrightarrow{p} g'$ muß erst das isomorphe Bild der linke Seite der Produktion p, d.h. eine *Instanz*, im Zustandsgraphen gefunden werden. Diese Aufgabe wird als das *induzierte Grapherkennungsproblem* bezeichnet. Die Menge der zu erkennenden Graphen (Referenzgraphen) entspricht der Menge der linken Seiten der Produktionen des Graphersetzungsproblems.

Definition 2 (Grapherkennungsproblem)
Das *Grapherkennungsproblem* ist definiert duch das Tupel (R, g), wobei $R = \{r_1, r_2, ..., r_n\}$ die Menge aller *Referenzgraphen* und g der zu verarbeitende *Eingabegraph* ist.

Seine *Lösung* ist die Menge
$L(R, g) =_{def} \{(r, g') | r \in R, g'$ ist Subgraph von $g : r, g'$ sind isomproph $\}$.

Einer der Vorteile von Graphgrammatiken zur Spezifikation verteilter Berechnungen ist ihre deklarative Form. Sie macht keine Aussagen über die Reihenfolge der einzelnen Produktionsanwendungen. Im allgemeinen besitzt ein Zustandsgraph eine ganze Menge von potentiell ersetzbaren Untergraphen. Da jeder Ersetzungsschritt mit einer mehr oder weniger aufwendigen Attributberechnung verknüpft ist, sollten sie in einer verteilten Umgebung so weit möglich parallel ausgeführt werden. Hierbei muß allerdings darauf geachtet werden, daß die ursprüngliche Semantik des Graphersetzungssystems erhalten bleibt. Sie besagt, daß immer ein Ersetzungsschritt nach dem anderen durchgeführt wird - die Reihenfolge ist dabei unerheblich. Für parallele Ersetzungsschritte bedeutet dies, daß sie durch sequentielle simuliert werden können müssen. Diese Eigenschaft ist die Basis für die Definition von unabhängigen Ersetzungsschritten. Da hierbei die konkreten Anwendungsstellen der Ersetzung von Interesse sind muß der Ableitungsbegriff entsprechend angepaßt werden. $g \xrightarrow{i_p} g'$ beschreibt, wie der Graph g durch die Produktionsinstanz i_p in g' überführt wird. Die Produktionsinstanz i_p gibt hierbei die von der Veränderung betroffenen Komponenten von g und g' an.

Da die Lösung des Graphersetzungsproblems wenigstens eine teilweise Lösung des Grapherkennungsproblems voraussetzt, sollte sie aus Effizienzgründen so weit möglich ebenfalls verteilt erfolgen.

5 Verteilte Grapherkennung

Üblicherweise benutzen Grapherkennungsalgorithmen einen "Suchplan", der die Erkennung der Referenzgraphen aus einer Menge atomarer Einzelaktionen - `lese Knoten A, lese Kante B, ...` - beschreibt [Kau83, Vol92, Fu95, RS95]. Dies ist hinreichend für sequentielle Umgebungen, in welchen nur eine Verarbeitungseinheit für die Erkennung zuständig ist. Da Teile der Referenzmuster in einer verteilten Umgebung im allgemeinen parallel erkannt werden können, kann dieses Konzept nur bedingt für die verteilte Verarbeitung übernommen werden. Im Rahmen des Teilprojektes wurde deshalb ein Algorithmus zur verteilten Grapherkennung entwickelt. Er wurde seinerseits durch eine Graphgrammatik spezifiziert. Diese Vorgehensweise bietet den Vorteil, daß keine explizite Kommunikation modelliert werden muß.

Es wird davon ausgegangen, daß die einzelnen Verarbeitungseinheiten einen *gemeinsamen Graphen* analysieren und verändern. Die einzelnen Produktionen der Grapherkennungsgrammatik spezifizieren eine Menge atomarer Aktionen deren Anwendung zur Erkennung aller Referenzmuster führen. Die Synchronisation der Verarbeitung wird vom zugrundeliegenden verteilten Graphersetzungssystem erledigt.

Der Algorithmus basiert auf der Tatsache, daß jeder mehrknotige zusammenhängende Graph in zwei, durch mindestens eine Kante verbundene, Untergraphen zerlegt werden kann. Sie beschreibt die rekursive Zerlegung der Referenzmuster. Der so erzeugte "Zerlegungsplan" beschreibt, analog zu den sequentiellen "Suchplänen", ebenfalls die rekursive Erzeugung der entsprechenden Referenzmuster.

Zur Erkennung eines Musters vereinigt der Algorithmus entsprechend des vorher erzeugten "Zerlegungsplans" benachbarte Untergraphen des Eingabegraphen. Hierzu wird jedem Untergraphen ein Repräsentant zugeordnet, der anstelle des tatsächlichen Untergraphen verwendet wird. Da neue Repräsentanten ausschließlich durch Vereinigung zweier bereits existierender erzeugt werden, entspricht die "Erzeugungsstruktur" einem Binärbaum. Sie wird für eine effiziente Assoziation zwischen Repräsentant und Subgraph verwendet.

Abbildung 3 zeigt die Strukturen der Produktionen der Grapherkennungsgrammatik. Die Knoten- und Kantenmarkierungen werden wie Variablen behandelt, die abhängig von den zu erkennenden Referenzgraphen belegt werden müssen. Im Rahmen dieses Teilprojektes wurde bereits ein Prototyp zur Erzeugung des Zerlegungsplans implementiert.

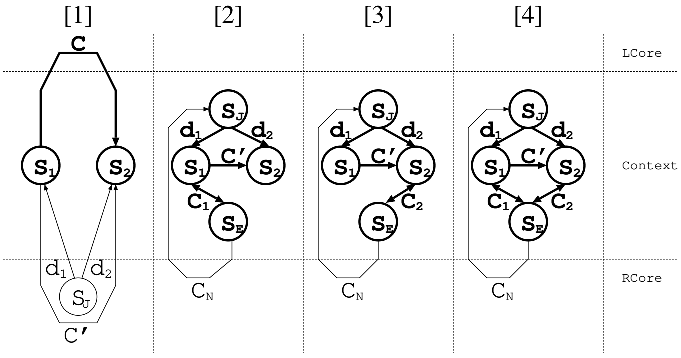


Abbildung 3: Strukturen der Produktionen der Grapherkennungsgrammatik

6 Verteilte Graphersetzung

Im Gegensatz zu Graphersetzung in einer sequentiellen Umgebung muß bei der verteilten Graphersetzung darauf geachtet werden, daß die Semantik des Graphersetzungssystems - definiert durch den Ableitungsbegriff - erhalten bleibt. Hierzu muß das verteilt arbeitende System bei der Durchführung eines Ersetzungsschrittes dafür Sorge tragen, daß immer nur unabhängige Produktionsinstanzen parallel ersetzt werden. Entsprechende Synchronisations- und Kommunikationsmechanismen müssen bereitgestellt werden.

Ein Graphersetzungssystem verarbeitet genau einen Eingabegraphen. Um größtmögliche Skalierbarkeit des verteilten Systems zu gewährleisten sollte der Eingabegraph nicht durch eine zentrale Instanz, die einen potentiellen Flaschenhals darstellen würde, verwaltet werden. Der Graph sollte vielmehr partitioniert und die Partitionen den einzelnen Verarbeitungseinheiten zugeordnet werden. Da sich die Struktur des Graphen während der Berechnungen des Graphersetzungssystems drastisch verändern kann, muß die Partitionierung *dynamisch*, d.h. zwischen den Ersetzungsschritten veränderbar, sein. Zur Realisierung müssen entsprechende Graphmigrationsmechanismen bereitgestellt werden.

Für die Bewertung der Partitionierung müssen noch entsprechende Kriterien gefunden werden. Mögliche Kandidaten sind

- minimale Verbindungskantenanzahl zwischen den einzelnen Partitionen,
- gleichmäßiges "Gewicht" der einzelnen Partitionen oder
- minimale Partitionierung von bereits erkannten Referenzmustern im Eingabegraphen.

7 Zusammenarbeit

Im Rahmen des Graduiertenkollegs ergab sich innerhalb des Teilprojektes sehr intensive Kooperation mit Barbara König. Ziel ist eine möglichst vollständige Realisierung des graphischen Kalküls SPIDER auf einer verteilten Graphersetzungsplattform.

Am Lehrstuhl für Informatik II von Prof. J. Eickel wurde im Rahmen der Dissertation [Vol92] das Syntaxanalyseverfahren $SB(\phi)$ für kontextfreie Graphgrammatiken entwickelt. Zur Beschleunigung der Analyse von massiven Graphstrukturen sollte das Verfahren für eine verteilte Umgebung angepasst werden. Dies kann durch Abbildung der einzelnen Analyseschritte auf Graphgrammatikproduktionen, welche ein verteiltes Graphersetzungssystem spezifizieren, erreicht werden. Die entsprechende Grammatik soll hierfür aus der, die Struktur beschreibenden, kontextfreien Grammatik generiert werden.

Im Rahmen der Zusammenarbeit mit dem molekularbiologischen Arbeitskreis von Dr. B. Steipe wird eine kontextfreie Graphgrammatik zur Modellierung von Sekundärstrukturelementen entwickelt. Sie soll zur Unterstützung der im Rahmen des DFG-Projektes[13] *"Strukturmotive in Proteinen; Definition, Extraktion und Analyse"* durchzuführenden Untersuchungen herangezogen werden.

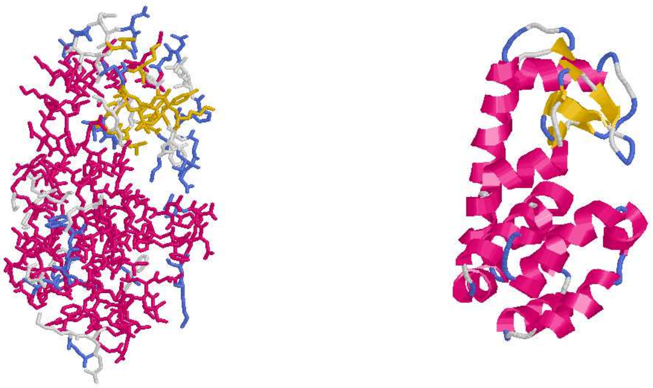


Abbildung 4: Ein Protein in zwei Abstraktionsformen (Bindungsstruktur/Sekundärstrukturelemente)

Abbildung 4 zeigt ein Protein[14] in zwei verschiedenen Darstellungsformen. In der Linken stehen die Bindungen zwischen den einzelnen Atomen im Vordergrund. Sie bilden die graphartige Struktur des Moleküls. Die rechte Darstellung zeigt die schematische Anordnung der Sekundärstrukturkomplexe. Das Finden derartige hierarchisch strukturierter Bereiche kann durch Graphsyntaxanalyse bewerkstelligt werden. Im weiteren soll die Eignung von Graphgrammatiken zur qualitativen Spezifikation von dynamischen molekularbiologischen Eigenschaften untersucht werden.

8 Ausblick

Die bisherigen Erfahrungen im praktischen Umgang mit Graphersetzungssystemen haben gezeigt, daß die Spezifikationen von hinreichend komplexen Anwendungen in ihrer bisherigen linearen Notation schnell undurchschaubar werden. Wünschenswert währe eine graphische Entwicklungsumgebung die sowohl Design (Graphgrammatikeditor) als auch Analyse (Visualisierung von Graphersetzungsprozessen) von entsprechenden Systemen unterstützt

[13] STE 563\4-2

[14] *HYDROLASE (O-GLYCOSYL)*, *PDB* Datensatz von *D.W.Heinz* und *B.W.Matthews*, visualisiert mit *RasMol2*

und die unproduktiven Transformationen zwischen graphischer und linearer Darstellung selbständig durchführt.

Im weiteren sollen einige komplexe Anwendungen spezifiziert und als Basis für genauere Leistungsanalysen herangezogen werden. Die Anwendungen werden sich hauptsächlich aus der oben dargestellten Zusammenarbeit mit dem molekularbiologischen Arbeitskreis ergeben.

Literatur

[BA78] C. Batini und A.d'Atri. Rewriting systems as a tool for relational data base design. In *Graph-Grammars and Their Application to Computer Science and Biology*, Nr. 73 aus LNCS, S. 139–153. Springer, 1978.

[Fu82] King Su Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Inc., 1982.

[Fu95] Jianghai Fu. Pattern Matching in Directed Graphs. In *Combinatorial Pattern Matching*, Nr. 937 aus LNCS, S. 64–77. Springer, 1995.

[JR91] D. Janssens und G. Rozenberg. Graph Grammar-Based Description of Object-Based Systems. In *Foundations of Object-Oriented Languages*, Nr. 489 aus LNCS. Springer, 1991.

[Kau83] Manfred Kaul. parsing of graphs in linear time. In G. Goos und J. Hartmanis, Hrsg., *Graph-Grammars and Their Application to Computer Science*, Nr. 153 aus LNCS, S. 206–218. Springer, 1983.

[Loy92] Joseph Patrick Loyall. *Specification of Concurrent Systems using Graph Grammars.* Dissertation, University of Illinois at Urbana-Champaign, Department of Computer Science, May 1992.

[Nag80] M. Nagl. Graph Rewriting and Automatic, Machine-Independend Programm Optimazation. In G. Goos und J. Hartmanis, Hrsg., *Graphtheoretic Concepts in Computer Science*, Nr. 100 aus LNCS, S. 55–69. Springer, 1980.

[RS95] J. Rekers und A. Schürr. A Parsing Algorithm for Context-Sensitive Graph Grammars. Technical report, Leiden University (NL), 1995. Available by: ftp.wi.leidenuniv.nl/pub/CS/TechnicalReports/1995/tr95-05.ps.gz.

[Tan95] Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice-Hall International, Inc., 1995.

[Vol92] Ulrich Vollath. *Generierung neuer Syntaxanalyseverfahren für Graphgrammatiken.* Dissertation, Technische Universität München, Institut für Informatik, August 1992.

1.3.17 Lastverwaltung auf Arbeitsplatzrechnern

Teilprojekt: Programmentwicklung für Parallelrechner und vernetzte Architekturen
Christian Röder

1 Motivation

Netze gekoppelter Arbeitsplatzrechner treten auf dem Gebiet der parallelen Datenverarbeitung in den letzten Jahren immer stärker in den Vordergrund. Verschiedene Varianten paralleler Programmierumgebungen wie PVM [BDG+95, SGDM94], p4 [BL94], NXLib [SBLL94] oder Implementierungen des MPI Standards [Mes93, Wal94] wurden entwickelt, die es erlauben, ein solches System als virtuellen Parallelrechner zu nutzen. Die Programmierumgebungen unterstützen dabei Strategien für eine globale Prozeßverwaltung, die Synchronisation der an der Anwendung beteiligten Prozesse und Kommunikation zwischen diesen. Die theoretisch mögliche Leistung eines parallelen Programms kann — abgesehen von architekturbedingten Engpässen — selten erreicht werden, da dessen Struktur oft nicht dazu geeignet ist, alle Betriebsmittel vollständig auszunutzen. Neben der Schwierigkeit, eine geeignete Aufteilung eines Problems zu finden, besteht die Notwendigkeit, lasterzeugende Programmteile auf leistungserzeugende Komponenten (z.B. Prozessoren) so zu verteilen, daß Leerlauf im System vermieden wird. Unter dem Begriff der **Lastverwaltung** (LV) versteht man allgemein die Überführung einer ungleichmäßig verteilten Systemauslastung — gleichgültig ob sie bereits zum Programmstart existiert oder erst während des Programmablaufes entsteht — in einen ausgeglichenen Zustand [Lud93].

Ziel dieser Arbeit ist es ein Lastverwaltungssystem (LVS) zu entwickeln, das die besonderen Eigenschaften eines Systems berücksichtigt, das aus gekoppelten heterogenen Arbeitsplatsrechnern zusammengesetzt ist und auf jedem solchen Rechenknoten mehrere Benutzer gleichzeitig Anwendungen zur Ausführung bringen können. Ein zentraler Bestandteil ist dabei die Entwicklung einer Lastbewertungs- und Entscheidungskomponente auf Grundlage einer Klassifikation von Lastmodellen (siehe §3).

Der vorliegende Bericht gliedert sich wie folgt: Zunächst werden in §2 die für die Lastverwaltung wichtigen Begriffe wiederholt. Es wird dabei auf die Problematik der Heterogenität und des Mehrbenutzerbetriebs im betrachteten System eingegangen. Die Lastbewertung, die Gegenstand einer praktischen Prototyprealisierung ist, orientiert sich an der in §3 vorgestellten Klassifikation von Lastmodellen. Den Abschluß bildet in §4 ein Ausblick auf die noch verbleibende Fertigstellung des Prototyps zur automatischen Lastverwaltung.

2 Lastverwaltung in heterogenen Systemen mit Mehrbenutzerbetrieb

2.1 Regelkreis der Lastverwaltung

Der Ablauf der Lastverwaltung orientiert sich an der Funktionsweise eines Regelkreisschemas [Lud93] und ist in Abb. 1 dargestellt.

Das zu regelnde System besteht aus dem (leistungserzeugenden) Rechensystem und den darauf auszuführenden (lasterzeugenden) Prozessen. Das LVS muß die Frage „wann müssen welche lasterzeugenden Objekte wohin verteilt werden?“ lösen. (Objekte sind in

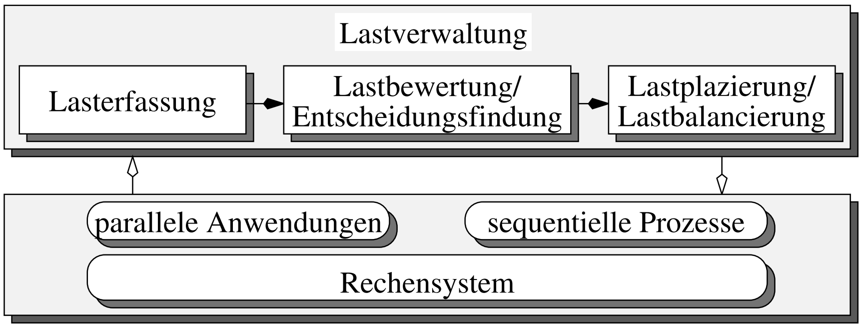


Abbildung 1: Regelkreis der Lastverwaltung

diesem Sinne nicht auf Prozesse beschränkt, sondern umfassen z.B. auch zu verarbeitende Daten oder Anfragen an ein Datenbanksystem.) Dazu wird die aktuelle Lastsituation des Systems ermittelt (Lasterfassung) und bewertet (Lastbewertung). Bei ungleicher Last der leistungserzeugenden Rechenkomponenten werden die Objekte so verteilt (Entscheidung: welches Objekt auf welche Komponente), daß eine ausgeglichene Last entsteht. Die Verteilung der Objekte kann vor deren Eintritt in das Rechensystem erfolgen (statische Lastplazierung) oder während sie schon aktiv im System agieren (dynamische Lastbalancierung). Entsprechend ihrer Integrationsart unterscheidet man anwendungs- und systemintegrierte LVS. Erstere haben den Vorteil, daß der Anwendungsentwickler detailliertes Wissen über die Anwendung ausnutzen kann, um effiziente Mechanismen zur LV mit dem Ziel der der Laufzeitminimierung zu integrieren. Allerdings müssen diese Mechanismen für jede Anwendung neu entwickelt und implementiert werden. Andererseits bietet ein systemintegriertes LVS mit dem Ziel der Durchsatzoptimierung einen allgemeineren Ansatz, da es unabhängig von einer bestimmten Anwendung agieren kann. Spezielles Wissen über die einzelnen Anwendungen wird dabei i.a. nicht ausgenutzt. Das für den Benutzer wichtigste Ziel der Laufzeitminimierung wird dem Ziel der Durchsatzoptimierung höchstens untergeordnet. Umfangreiche Klassifikationen und Übersichten über LVS finden sich in [CK88, Lud93, SHK95].

2.2 Anforderungen an das Lastverwaltungssystem

Allgemein wird von einem LVS erwartet, daß es das zu regelnde System selbst während seines eigenen Ablaufs kaum belastet. Der Zeitraum zwischen der Erfassung der Lastwerte, ihrer Bewertung und der resultierenden Aktivität bestimmt einen Zyklus des LVS. Während der Dauer eines Zyklus kann sich allerdings die Lastsituation im System schon wieder soweit verändert haben, daß die getroffene Entscheidung dem Ziel der LV entgegenarbeitet. Zur Stabilisierung des Regelkreises ist es also erforderlich, voreilige Entscheidungen zu vermeiden, damit diese später nicht revidiert werden müssen.

Das betrachtete Rechensystem besteht aus gekoppelten heterogenen Arbeitsplatzrechnern, die im Mehrbenutzer-/Mehrprozeßmodus betrieben werden. Im folgenden seien beispielhaft die dabei zu berücksichtigenden Eigenschaften erläutert. Eine ausführliche Diskussion findet sich im Vorjahresbericht [Röd96]. Die Architekturheterogenität der Arbeitsplatzrechner erfordert ggf. eine Konvertierung der Datenformate beim Nachrichtenaustausch. Durch die Leistungsheterogenität (z.B. unterschiedliche Leistungsfähigkeit der Prozessoren, unterschiedlicher Speicherausbau) der Arbeitsplatzrechner müssen die ermittelten Lastwerte entsprechend der Leistungsfähigkeit der Rechner skaliert werden. Im Mehrbenutzerbetrieb entsteht eine ungleiche Lastsituation nicht nur durch die parallele Anwendung, sondern auch durch die externen Prozesse (Benutzerprozesse, Aufgaben des Betriebssystem, allgemeine Dienstleistungen, etc.). Beiden Randbedingungen wird in jüngster Zeit vermehrt Aufmerksamkeit gewidmet [AALR94, MDF^{+}95, TNC94, ZHKS95].

Aus den angesprochenen Besonderheiten des betrachteten Systems (Heterogenität und Mehrbenutzerbetrieb) lassen sich weitere Anforderungen an das LVS ableiten:

1. Das LVS benötigt für verschiedene Architekturen und Betriebssysteme eine wohldefinierte Schnittstelle zur Lastwerterfassung. Lastwerte eines Arbeitsplatzrechners können dabei sowohl durch dessen Betriebssystem als auch durch die auf ihm zur Ausführung gebrachten Anwendungsteile zur Verfügung gestellt werden.

2. Die Lastbewertung und die Entscheidungsfindung sollten von der Lastwerterfassung entkoppelt werden. Dadurch erreicht man hohe Portabilität für verschiedene Architekturen und Programmierumgebungen.

3. Eine rein auf die parallele Anwendung zugeschnittene Erfassung der Lastwerte reicht nicht aus. Vielmehr müssen die sonstigen Aktivitäten an den Betriebsmitteln (z.B. verursacht durch externe Benutzerprozesse) von einer Lastbewertungskomponente berücksichtigt werden.

4. Das LVS erhält die Möglichkeit während des Ablaufs einer Anwendung neue Rechnerknoten in die Anwendung einzubinden. Somit muß die Lastbewertung auch auf Rechnerknoten stattfinden, auf denen im Moment keine Anwendungsteile ausgeführt werden.

3 Klassifikation von Lastmodellen

Für die Definition einer einheitlichen Schnittstelle zur Lastwerterfassung und für die Methoden zur Lastbewertung wurde im Rahmen der Arbeit eine Klassifikation der in der Literatur weitverbreiteten Lastmodelle erstellt. Die Rolle das Lastmodells und seine Einbettung in das LVS ist in Abb. 2 dargestellt.

3.1 Bestimmung der „Last“ eines Systems

Die Frage nach einer exakten Definition des Begriffs „Last“ ist schwer zu beantworten, da dieser von der Sichtweise und Zielsetzung der Problemstellung abhängig ist. So versteht man beispielsweise auf dem Gebiet der Leistungsanalyse unter „Last“ eine charakteristische Menge von Aufgaben die von einem System zu erfüllen ist. Dabei ist es i.a. gleichgültig wann diese Aufgaben ausgeführt werden. Im Gegensatz dazu ist es gerade das Ziel eines LVS, die augenblicklich im System befindliche „Last“ — die lasterzeugenden Objekte — zu verwalten und die Effektivität des Systems nach festgelegten Zielkriterien zu optimieren. Um dieses Ziel zu erreichen, muß zur Bestimmung der Last und dem anschließenden

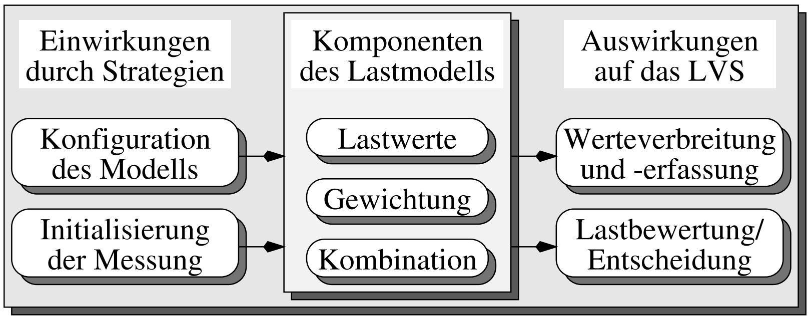


Abbildung 2: Einbettung und Nutzung des Lastmodells in das LVS

Vergleich der Lastsituation auf verschiedenen Rechenknoten ein Maß angegeben werden. Die Lastinformation wird in der Praxis gewöhnlich durch einen sogenannten *Lastindex* repräsentiert, der ein quantitatives Maß für die Last eines Betriebsmittels ist. Der Lastindex wird als eine nichtnegative Variable definiert, die den Wert 0 annimmt, falls das betrachtete Betriebsmittel unbeschäftigt ist, und immer größere Werte annimmt, je mehr Aufträge das Betriebsmittel zu bearbeiten hat [FZ87]. Im Rahmen dieser Arbeit wurde diese Definition um eine Hierarchie verfeinert. Auf der Ebene der Betriebsmittel stehen dem LVS *atomare* Lastindizes zur Verfügung. Durch die Kombination mehrerer solcher Indizes erhält man den knotenspezifischen *globalen* Lastindex. Dieser liefert eine Quelle für die zu treffenden Entscheidungen des LVS. Die gesamte Last des Systems läßt sich somit aus den globalen Indizes ermitteln. Die folgenden Ausführungen beschränken sich auf die einfachere Definition, falls auf die Unterscheidung verzichtet werden kann.

Zur Bestimmung der Last stehen zwei Arten von Informationsquellen zur Verfügung. Als statische Informationsquellen definieren wir solche Informationsquellen, die immer zur Verfügung stehen. So läßt sich beispielsweise die Anzahl der rechenbereiten Prozesse auf einem Arbeitsplatzrechner in der Regel immer bestimmen. Im Gegensatz dazu definieren wir dynamische Informationen als solche, die möglicherweise zur Verfügung stehen. Darunter fallen z.B. eine Beschreibung des Ablaufs der Anwendung einschließlich deren Betriebsmittelanforderungen oder historische Aufzeichnungen über das Anwendungsverhalten während früherer Abläufe. Man beachte also, daß statische Informationsquellen durchaus dynamisch veränderliche Werte liefern können.

3.2 Anforderungen an den Lastindex

Der Aufwand zur Bestimmung des Lastindex und dessen Einfluß auf die Qualität der Entscheidungen sind von zentraler Bedeutung für die Effizienz des LVS. In [ELZ86, FZ87, MW93] werden deshalb verschiedene Anforderungen an den Lastindex gestellt, die im folgenden kurz aufgelistet werden:

1. Der Lastindex soll eine qualitative Abschätzung über die augenblickliche Last im System liefern. Dadurch werden Aussagen wie „überlastet“ bedeutungsvoll.

2. Über den Lastindex sollen auch Aussagen über die Lastsituation in der nahen Zukunft möglich sein, da das Erreichen des Optimierungskriteriums mehr von der zukünfti-

gen als von der augenblicklichen Last abhängt (siehe §2.2: Verzögerung zwischen Lastwerterfassung und tatsächlicher Regelung).

3. Der globale Lastindex sollte stabil sein, d.h. kurzfristige starke Schwankungen der atomaren Lastindizes sollen keinen Einfluß auf die Entscheidungsfindung nehmen. Überreaktionen und folgende Korrekturen („Lastflattern“) können vermieden werden.

4. Der Lastindex und die erwartete Leistung stehen im Idealfall in linearer Beziehung. Aus der Last des Systems läßt sich dessen gerade verfügbare Leistung mit geringem Aufwand berechnen.

5. Der Lastindex soll einfach bestimmbar sein. Einerseits kann damit die durch das LVS verursachte Leistungseinbuße des Systems minimiert werden, andererseits erreicht man durch häufigere Messungen eine genauere Erfassung des Lastzustands.

6. Der Lastindex sollte die Last an allen kritischen Ressourcen repräsentieren. Informationen über die Last der CPU eignen sich zwar für rechenintensive Anwendungen, scheitern aber für speicherintensive Anwendungen.

7. Falls der Lastindex die Belastung an mehreren Ressourcen bestimmt, sollten die einzelnen Lastwerte unabhängig voneinander kontrollierbar sein.

Während die oben genannten Punkte in der Literatur schon weitgehend diskutiert wurden, wird hier vor dem Hintergrund des zu entwickelnden Prototyps noch ein weiteres Qualitätskriterium eingeführt. Ein systemintegriertes LVS sollte nicht nur die Durchsatzoptimierung als Ziel verfolgen, sondern auch auf die Laufzeitminimierung der Anwendung achten und dessen Anforderungen berücksichtigen:

8. Der Lastindex soll nicht nur durch Werte der leistungserzeugenden sondern auch durch Werte der lasterzeugenden Objekte bestimmt werden. Die Beobachtung der Ressourcenanforderungen einer Anwendung bestimmt die Menge und Art der zu erfassenden Lastwerte.

3.3 Klassifikation

Nach [Lud93] setzt sich ein Lastmodell aus drei Komponenten zusammen: den gemessenen Lastwerten, ihre Gewichtung zueinander und die Art ihrer Verknüpfung. Tabelle 1 (S. 202) gibt eine vollständige Übersicht über die Charakteristiken weit verbreiteter Lastmodelle.

Eine vollständige Diskussion von Tabelle 1 würden den Rahmen dieses Berichts überschreiten. Eine Diskussion ausgewählter Punkte — einschließlich Beispiele ihrer Verwendung — soll einen Einblick in die wichtigsten Merkmale der Lastmodelle geben. Eine Unterteilung der „Komponenten des Lastmodells“ erfolgt anhand allgemein gültiger Charakteristiken der gemessenen Werte, d.h. der atomaren Lastindizes, und optionalen Eigenschaften, die erst im Falle der Gewichtung und Kombination mehrerer Lastwerte zum knotenspezifischen globalen Lastindex angewendet werden.

Die allgemein gültigen Eigenschaften des Lastindex lassen sich wie folgt gliedern:

	Charakteristik	Kategorie
Komponenten des Lastmodells	**Eigenschaften des Lastindex**	
	Dimension:	- eindimensional - mehrdimensional
	Typ:	- CPU (Rechenaufwand) - MEM (Speicherbelastung) - I/O (Ein-/Ausgabe) - NET (Kommunikation) - Service (Dienstanforderungen) - anwendungsdefiniert
	Abtastzeitraum:	- Momentaufnahme - Intervallbeobachtung - komponentenabhängig
	Meßquelle:	- systembezogen - anwendungsbezogen - Mischform
	Komposition von Lastwerten	
	Kombination:	- gewichtete Linearkombination - boole'sche Auswahl - komplexe Funktion - keine Kombination
	Gewichtung:	- Anwendungs-Historie - vorhergehende LV-Entscheidungen - Systemeigenschaften - nicht anwendbar
Verwendung des Lastmodells	**Messung**:	- auf Anforderung - periodisch - implizit
	Verbreitung:	- auf Anforderung - nach Messung - adaptiv
	Modell Adaptivität	- statisch - adaptive Gewichtung - adaptive Typisierung - adaptive Kompostion

Tabelle 1: Klassifikation von Lastmodellen

- Die *Dimension* des Lastindex spiegelt die Anzahl der gemessenen Werte zur Bestimmung der Last wieder. Falls nur ein Wert erfaßt wird, heißt der Lastindex *eindimensional*, ansonsten ist er *mehrdimensional*. Beispielsweise wird als einziger Lastparameter in [HL95] die Anzahl der von einem Bildverarbeitungsprogramm pro Zeiteinheit verarbeiteten Pixel verwendet. In [AALR94] werden wird zur Bestimmung der Last eine in [WS88] entwickelte Formel verwendet, die sowohl die Anzahl der rechenbereiten Prozesse als auch das Zeitanteil, während dessen der Prozessor unbeschäftigt ist, berücksichtigt. In diesem Fall handelt es sich also um einen mehrdimensionalen Lastindex.

- Der *Typ* des Lastindex beschreibt diejenige Ressource bzw. den Aufgabenteil einer Anwendung, der auf die Laufzeit der Anwendung den größten Einfluß hat. Analog zur Unterscheidung zwischen rechen-, speicher-, E/A- oder kommunikationsintensiven Anwendungen kann der Typ des Lastindex durch CPU, MEM, I/O und NET bezeichnet werden. Handelt es sich z.B. um eine rechenintensive Anwendung, so wird (wie in [AALR94]) die Last durch CPU-bezogene Lastwerte bestimmt. Gleiches gilt für anwendungsintegrierte LVS, die beispielsweise die Anzahl der Schleifeniterationen pro Zeit (mit festem Rechenaufwand pro Schleifendurchlauf) als Lastindex verwenden. Das bereits oben zitierte Beispiel von [HL95] wird als anwendungsdefinierter Lastindex bezeichnet, da hier nicht nur die Verarbeitung der Pixel, sondern auch deren Speicherung und ihrer Kommunikation berücksichtigt wird.

- Der *Abtastzeitraum* legt fest, ob es sich beim Wert des Lastindex um einen momentanen Wert (z.B. Anzahl der augenblicklich belegten Speicherseiten) handelt, oder ob der Lastindex ein über die Zeit gemittelter Wert ist (z.B. durchschnittliche Anzahl der rechenbereiten Prozesse in der letzten Minute). Falls mehrere Werte zur Lastbestimmung erfaßt werden, so ist der Abtastzeitraum komponentenabhängig und die Charakteristik ist für jede Komponente anzugeben.

- Die *Meßquelle* des Lastindex bezeichnet den Ort, in dem die Lastwerte erfaßt werden. Typischerweise verwenden systemintegrierte LVS solche Lastwerte, die durch das System zur Verfügung gestellt werden (siehe [AALR94]). Andererseits verwenden anwendungsintegrierte LVS Werte, die durch die Anwendung selbst bestimmt werden (siehe [HL95]).

Werden zur Bestimmung der Knotenlast mehrere atomare Lastindizes verwendet, so müssen diese skaliert und kombiniert werden:

- Zur *Kombination* mehrerer Lastindizes werden in der Literatur hauptsächlich gewichtete (s.u.) Linearkombinationen verwendet (z.B. [Lud93]). Dabei werden die einzelnen Lastwerte nach ihrer Gewichtung durch eine Linearkombination so verknüpft, daß das Resultat als Last des Rechnerknotens aufgefaßt werden kann. Die boolesche Selektion kann als Spezialfall der Linearkombination betrachtet werden. Der Rechnerknoten gilt dann als überlastet, falls nur einer der Werte einen vorher festgelegten Sollwert überschreitet.

- Die *Gewichtung* der einzelnen Lastindizes dient einerseits dazu sie auf eine gemeinsame Basis zu skalieren, andererseits können dadurch ihre relativen Gewichte (Wichtigkeit) zueinander festgelegt werden. Skalierung und Gewichtung wird z.B. bei einer Kombination der augenblicklich belegten Speicherseiten und der durchschnittlichen

Anzahl der rechenbereiten Prozesse benötigt. Um die Last an verschiedenen Ressourcen im System miteinander vergleichen zu können, werden in [WS88] die gemessenen Lastwerte anhand von Leistungsindizes (Systemeigenschaft) der betrachteten Ressourcen skaliert.

Unter dem Stichpunkt „Verwendung des Lastmodells“ versteht man die Beziehung (Einbettung und Nutzung) zwischen dem Lastmodell und den restlichen Komponenten des LVS. Die Strategien des LVS und das verwendete Lastmodell hängen gegenseitig voneinander ab. Die notwendigen Charakteristiken seien an dieser Stelle nur kurz vorgestellt. Für die Effizienz und den eigenen Systemeinfluß eines LVS ist die Anzahl der Messungen von entscheidender Bedeutung (siehe §3.2). Falls die Lasterfassung über Softwarekomponenten erfolgt, so führt eine erhöhte Meßrate zu aktuelleren Werten auf Kosten der Leistungsminderung des Systems. Viele LVS messen die Lastwerte periodisch innerhalb fest vorgegebener Zeitintervalle. Adaptive Zeitintervalle werden beispielsweise dann verwendet, wenn sich die Meßrate an die Lastsituation anpaßt.

4 Zusammenfassung und Ausblick

In diesem Bericht wurde eine Klassifikation der in der Praxis verwendeten Lastmodelle vorgestellt. Ziel der Klassifikation ist es, ein Begriffsinstrumentarium zu schaffen, mit dessen Hilfe die wesentlichen Eigenschaften der Lastmodelle erfaßt werden können. Bereits existierende Lastmodelle können somit beschrieben und miteinander verglichen werden. Den Entwicklern von LVS eröffnet sich die Möglichkeit ihre Lastmodelle auf diese Eigenschaften hin zu untersuchen. Im Rahmen der laufenden Arbeiten des Teilprojekts „Programmentwicklung für Parallelrechner und vernetzte Architekturen“ dient diese Klassifikation als Grundlage der Prototyprealisierung einer Lastbewertungs- und Entscheidungskomponente für ein LVS in einem System aus gekoppelten heterogenen Arbeitsplatzrechnern. Einige wichtige Aspekte wurden in diesem Bericht vernachlässigt. Beispielsweise verursacht die in §2.2 angesprochene Möglichkeit der dynamischen Rekonfiguration der initialen Rechnerknotenmenge hohen Verwaltungsaufwand. Die Entscheidungskomponente sollte versuchen diese Kosten und den gewonnenen Nutzen gegeneinander abzuwägen.

Literatur

[AALR94] V.A.F. Almeida, J.N.C. Arabe, E.F. Loures und G.S. Rimolo. Scheduling Parallel Jobs on a Cluster of Heterogeneous Workstations. In *Proc. of the High Performance Computing Conference '94*, S. 103–108, Sep 1994.

[BDG+95] A. Beguelin, J. Dongarra, A. Geist, R. Manchek und V. Sunderam. Recent Enhancements to PVM. *International Journal of Supercomputing Applications and High Performance Computing*, 9(2), 1995.

[BL94] R.M. Butler und E.L. Lusk. Monitors, Messages, and Clusters: The p4 Parallel Programming System. *Parallel Computing*, 20(4):547–564, Apr. 1994.

[CK88] T. L. Casavant und J. G. Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. In *IEEE Transactions on Software Engineering*, S. 141–154, Feb 1988.

[ELZ86] D. L. Eager, E. D. Lazowska und J. Zahorjan. Adaptive Load Sharing in Homogeneous Distributed Systems. In *IEEE Trans. on Software Engeneering*, Bd. SE-12, S. 662–675, Mai 1986.

[FZ87] D. Ferrari und S. Zhou. An Empirical Investigation of Load Indices for Load Balancing Applications. In *Proc. Performance'87, The 12th Ann. Int. Symposium on Computer Performance Modeling, Measurement and Evaluation*, S. 515–528, 1987.

[HL95] M. Hamdi und C.-K. Lee. Dynamic Load Balancing of Data Parallel Applications in a Distributed Network. In *ACM Int. Conf. on Supercomputer*, S. 170–179, 1995.

[Lud93] T. Ludwig. *Automatische Lastverwaltung für Parallelrechner*. Reihe Informatik, Band 94. BI-Wissenschaftsverlag, 1993.

[MDF+95] B. Monien, R. Diekmann, R. Feldmann, R. Klasing, R. Lüling, K. Menzel, T. Römke und U.-P. Schroeder. Efficient Use of Parallel & Distributed Systems: From Theorie to Practice. In Jan van Leeuwen, Hrsg., *Computer Science Today, Recent Trend and Developments*, Bd. 1000 aus *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[Mes93] Message Passing Interface Forum. MPI: A Standard Message-Passing Interface. Technical report, University of Tennessee, Knoxville, TN, Nov. 1993.

[MW93] P. Mehra und B.W. Wah. Automated Learning of Workload Measures for Load Balancing on a Distributed System. In *1993 Int. Conf. on Parallel Processing, Vol. III*, S. 263–270. Syracuse University, CRC Press, Inc., 1993.

[Röd96] C. Röder. Lastverwaltung auf gekoppelten Arbeitsplatzrechnern. In P.P. Spies, Hrsg., *Graduiertenkolleg: Kooperation und Ressourcenmanagement in verteilten Systemen (Zwischenbericht zum Frühjahr 1996)*, S. 41–45. Institut für Informatik – Technische Universität München, 1996.

[SBLL94] G. Stellner, A. Bode, S. Lamberts und T. Ludwig. Developing Applications for Multicomputer Systems on Workstations. In Wolfgang Gentzsch und Uwe Harms, Hrsg., *HPCN, International Conference and Exhibition, Volume II*, Nr. 797 aus LNCS. Springer, Apr. 1994.

[SGDM94] V. Sunderam, G.A. Geist, J. Dongarra und R. Manchek. The PVM Concurrent Computing System: Evolution, Experiences and Trends. *Parallel Computing*, 20(4):531–545, Apr. 1994.

[SHK95] B.A. Shirazi, A.R. Hurson und K.M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. IEEE Computer Society Press, 1995.

[TNC94] S.W. Turner, L.M. Ni und B.H.C. Cheng. Time and/or Space Sharing in a Workstation Cluster Environment. In *Proc. of the Supercomputing'94*, S. 630–639, Nov. 14-18 1994.

[Wal94] D.W. Walker. The Design of a Standard Message Passing Interface for Distributed Memory Concurrent Computers. *Parallel Computing*, 20(4):657–673, Apr. 1994.

[WS88] A. Weinrib und S. Shenker. Greed is not enough: adaptive Load Sharing in Large Heterogeneous Systems. In *Proc. of the IEEE INFOCOM*, 1988.

[ZHKS95] Y. Zhang, K. Hakozaki, H. Kameda und K. Shimizu. A Performance Comparision of Adaptive and Static Load Balancing in Heterogenous Distributed Systems. In *Proc. of the 28th Simulation Symposium*, S. 332–340, Los Alamitos, CA, 1995. IEEE Comp. Soc. Press.

2 Stipendiaten des Kollegs

Name	Alter	1. berufsqualifizierender Abschluß	Monat /Jahr	Eintritt in GK	Zeitpunkt der Prom.
Angela Bücherl	29	TUM	11/95	1.3.96	
Dietrich Büsching	31	Universität Karlsruhe	9/94	1.4.95	
Martin Backschat	28	TUM	5/96	1.6.96	
Maximilian Frey	30	TUM	5/93	16.4.95	
Florian Fuchs	28	TUM	11/94	1.2.95	
Marc Fuchs	24	Universität Kaiserslautern	8/96	1.9.96	
Sascha Groh	28	TUM	5/94	1.2.95	
Barbara König	25	TUM	11/95	2.1.96	
Maximilian Lückenhaus	26	TUM	11/95	2.1.96	
Richard Mayr	25	TUM	11/94	15.2.95	
Maria-Athina Mountzia	27	TU Patra, Griechenland	7/92	1.4.95	
Martina Nöhmeier	27	TUM	11/94	1.2.95	
Dirk Ormoneit	28	TUM	5/94	1.2.95	
Markus Podolsky	26	TUM	5/96	20.6.96	
Bernhard Quendt	28	Universität Stuttgart	9/94	16.2.95	
Boris Reichel	29	TUM	5/95	1.6.95	
Christian Röder	28	TUM	11/94	1.2.95	

3 Auswahl der Stipendiaten

1. Für die Stipendien des Kollegs fanden zwei Ausschreibungen statt:

   Ausschreibung mit Termin 8.12.1994 brachte 38 Bewerbungen, davon 21 interne und 17 externe.

   Ausschreibung mit Termin 4.12.1995 brachte 8 Bewerbungen, davon 5 interne und 3 externe.

2. Voraussetzung für die Aufnahme in das Kolleg ist das Diplom oder ein gleichwertiger Abschluß

   - in Informatik oder
   - einem benachbarten Fach mit dem Vertiefungsgebiet Informatik

   an einer deutschen oder einer ausländischen wissenschaftlichen Hochschule mit wenigstens gutem Gesamtergebnis.

   Bewerber sollen über vertiefte Kenntnisse im Themenbereich des Kollegs verfügen.

   Die Auswahl unter den Bewerbern erfolgt durch die am Kolleg beteiligten Hochschullehrer auf der Grundlage der vorgelegten Zeugnisse, der gutachterlichen Stellungnahme von zwei Hochschullehrern und von Eignungsgesprächen.

4 Durchführung des Studienprogrammes

Das Studienprogramm des Kollegs ist so angelegt, daß die Kollegiaten schnell an den aktuellen Stand der Forschung herangeführt werden und in der Lage sein sollen, ihre Promotionsvorhaben nach drei Jahren erfolgreich abzuschließen. Es trägt den Anforderungen Rechnung, welche an die Kollegiaten gestellt werden: Sie sollen gemeinsam die Aufgabenstellungen des Kollegs bearbeiten und sie sollen selbständige, wissenschaftliche Leistungen zu den von ihnen behandelten, speziellen Themen erbringen.

Diesen zweifachen Anforderungen entsprechend ist für das Studienprogramm ein Rahmen festgelegt, der mit Pflichtveranstaltungen und mit individuell gewählten Veranstaltungen gefüllt wird. Der Rahmen sieht drei Phasen vor, die den drei Jahren der vorgesehenen Bearbeitungsdauer für Promotionsvorhaben und dem vorgesehenden Fortschreiten dieser Arbeiten angepaßt sind: Für die Eingangsphase sind 8 und für die Hauptphase 6 Semesterwochenstunden festgelegt. In der Endphase soll die Arbeit auf die Fertigstellung der Dissertation konzentriert werden; für diese Phase sind 3 Semesterwochenstunden festgelegt. In diesem Rahmen stellen die Kollegiaten ihre individuellen Studienprogramme in Abstimmung mit den Betreuern ihrer Dissertationen aus Pflicht– und Wahlpflichtveranstaltungen zusammen.

Pflichtveranstaltungen

Alle Kollegiaten sind verpflichtet, an den folgenden Verantaltungen des Kollegs teilzunehmen:

- Das Graduiertenkollegstreffen findet während der Vorlesungszeit regelmäßig einmal jede Woche als durchgehende, zentrale Veranstaltung des Kollegs statt. An ihm nehmen alle Kollegiaten und der Sprecher des Kollegs teil.

  Dieses interne Treffen ist das Forum für die Erörterung der anstehenden organisatorischen Fragen, für Vor– und Nachbereitungen weiterer Veranstaltungen und für Diskussionen über die Themen, mit denen die Kollegiaten befaßt sind.

- Treffen der Arbeitskreise und der Querschnittsthemen–Arbeitsgruppen des Kollegs sind interne Veranstaltungen, an denen jeweils Gruppen von Kollegiaten teilnehmen. Diese Treffen finden während der Vorlesungszeit regelmäßig einmal jede Woche und sonst nach Bedarf statt. Über die Themen, die in diesen Arbeitskreisen und –gruppen behandelt werden, ist in Kapitel 1.1 berichtet.

  Treffen der Arbeitsgruppen der Heimatlehrstühle der Kollegiaten sind kollegexterne Veranstaltungen, an denen einzelne Kollegiaten mit Berichten über ihre laufenden Arbeiten und über die Arbeiten des Kollegs teilnehmen.

- Das Kolloquium des Kollegs wird wegen der Verwandtschaft der behandelten Themen mit dem Sonderforschungsbereich 342 „Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen“ als gemeinsames SFB/GK–Kolloquium durchgeführt. Es findet während der Vorlesungszeit wöchentlich und hochschulöffentlich statt. In diesem Kolloquium tragen eingeladene Gäste über ihre laufenden Arbeiten vor.

- Das Berichtskolloquium des Kollegs findet einmal jährlich als Doktorandenkolloquium statt. Es ist eine fakultätsöffentliche Veranstaltung mit breiter Beteiligung. Jeder Kollegiat berichtet über seine laufenden Arbeiten und stellt sie mit ihren Ergebnissen und Zielen zur Diskussion.

Das Graduiertenkollegstreffen, die Mitarbeit in den Arbeitskreisen und –gruppen des Kollegs und das SFB/GK–Kolloquium werden mit je einer Semesterwochenstunde auf die Rahmenfestlegungen angerechnet.

Wahlpflichtveranstaltungen

Im folgenden sind die Wahlpflichtveranstaltungen aufgelistet, die den Kollegiaten für ihre individuellen Studienprogramme zur Verfügung stehen. Neben den angebotenen Vorlesungen sind die Seminare genannt, die für das Kolleg spezifische Themen behandeln und den Kollegiaten Möglichkeiten zur Mitgestaltung bieten.

Vorlesungen	**SWS**
Algebraische Theorie kommunizierender Prozesse	3
Automatisches Beweisen	3
Betriebssysteme I	3
Zentrale und verteilte Betriebssysteme	3
Breitbandnetze	3
Codegenerierung	2
Computergestützte Gruppenarbeit	2
CSCW-Systeme — Unterstützung von Gruppenarbeit	2
Elektronisches Publizieren	2
Funktionale Programmierung	2
Grundlagen der Programm- und Systementwicklung	3
Interpretation von Bildfolgen	2
KI und Robotik	2
Kommunikationsnetze I	3
Kommunikationsnetze II	3
Komponenten zum Aufbau von Rechnernetzen	2
Leistung von Parallelrechnern	3
Lernende Systeme: ein informationstheoretischer Zugang	2
Maschinelles Lernen	2
Mobile verteilte Systeme	3
Netz- und Systemmanagement	3
Netzkopplungen	3
Nichtsequentielle Systeme, nebenläufige Prozesse	3

Vorlesungen	**SWS**
Parallel und verteilte Algorithmen	4
Parallele Algorithmen	2
Parallele Komplexitätstheorie	2
Parallelrechner	3
Petrinetze	2
Programmiersysteme und -werkzeuge für Parallelrechner	3
Raumkognition: Repräsentation und Verarbeitung räumlichen Wissens	2
Rechnernetze und Rechnerkommunikation I	3
Rekursive Verfahren und hierarchische Datenstrukturen der numerischen	2
Semantik	3
Sensor- und kamerageführte Roboter	1
Sichere Rechensysteme	2
Syntaxanalyse	2
Symmetrische Multiprozessorsysteme	3
Verarbeitung unsicheren Wissens	2
Verteilte Anwendungen	3
Verteiltes Problemlösen	2
Wissensbasierte Systeme I	3
Wissensbasierte Systeme II	2
Wissensrepräsentation	3

Seminare	**SWS**
Ausgewählte Aspekte verteilter Betriebssysteme	2
Betriebssysteme	2
Bewegungsplanung für Fahrzeuge und Roboter	2
Computerunterstützte Verifikation paralleler Systeme	2
Effizientes Suchen mit Constraints	2
Elektronische Fachinformation	2
JAVA und objektorientierte Programmierung	2
KI/Kognition	1
Multi-Agenten Systeme	2
Objektorientierte Plattformen, Sprachen und Modelle für verteilte Systeme	2
Syntaktische Musterverarbeitung	2
Telekommunikation	3
Workflow-Management	2

5 Interne Erfolgskontrollen des Kollegs

Die in der Konzeption des Kollegs verankerte Form der Ausbildung, der Betreuung und der Erfolgskontrolle mit der zweifachen Integration der Doktoranden sowohl in das Kolleg als auch in die Arbeitsgruppen ihrer Heimatlehrstühle hat sich bisher bewährt und soll weitergeführt werden.

Diese zweifache Integration bedeutet für die Betreuung und für die Erfolgskontrollen der Doktoranden zunächst, daß sie regelmäßig und in kurzen Zeitabständen in den Arbeitsgruppen ihrer Heimatlehrstühle über ihre laufenden Arbeiten berichten sowie ihre Vorhaben und Ergebnisse zur Diskussion stellen, so daß permanente und kurzfristige Rückkoppelungen gewährleistet sind. Als Mitglieder des Kollegs sind die Doktoranden zudem an den Arbeitskreisen des Kollegs beteiligt, in denen sie den jeweils behandelten Themen entsprechend über ihre Arbeiten berichten. Schließlich findet einmal in jedem Jahr ein Berichtskolloquium mit Beiträgen aller Kollegiaten unter Beteiligung aller Hochschullehrer des Kollegs statt.

Diese gemischt dezentral/zentrale Betreuung und Erfolgskontrolle der Doktoranden trägt den Anforderungen Rechnung, die gestellt werden: Die Doktoranden sollen eine wissenschaftliche Leistung zu dem Thema, das sie bearbeiten, und zum gemeinsamen Thema des Kollegs leisten.

Von den 11 Stipendiaten, die dem Kolleg seit 1995 angehören, wird erwartet, daß sie ihre Promotionsvorhaben im Rahmen der 3–jährigen Bearbeitungszeit im Laufe des Jahres 1998 erfolgreich abschließen.

6 Gastwissenschaftlerprogramm

Bisher wurden lediglich Gäste zu Vorträgen im Kolloquium, das vom Kolleg zusammen mit dem SFB 342 durchgeführt wird, eingeladen. Die Gäste und die Themen ihrer Vorträge sind im folgenden aufgelistet.

Vorträge im Kolloquium

15.11.95	Prof. Dr. W. Reisig Humboldt-Universität Berlin	Verteilte Algorithmen: Modellierung und Analyse mit Petrinetzen
06.12.95	Prof. Dr. H.J. Müller Universität Bremen	Zur Korrelation von Anwendungsprofilen und Agentenmodellen -Eine Fallstudie-
20.12.95	Dr. B. Neumair Universität München	Realisierung von Agenten für integriertes Netz- und Systemmanagement
10.01.96	Prof. Dr. M. Nagl RWTH Aachen	Koordination kooperativer Entwicklung durch eine administrative Komponente und deren Spezifikation durch Graphersetzungssysteme
17.01.96	Dr. K. Fischer DFKI Saarbrücken	Ein Multiagentenansatz für das Fleet–Scheduling in verteilten Speditionen
24.01.96	R. Pollak IPVR,Universität Stuttgart	Auswirkungen verschiedener Informationsebenen auf die Effizienz der dynamischen Lastbalancierung
31.01.96	A. Erzmann Universität Hannover	Dynamische Lastverwaltung mit dem TDC–Programmiermodell
07.02.96	Prof. Dr. P. Levi IPVR, Universität Stuttgart	Kooperierende Agenten in der Robotik und Fertigung
14.02.96	Dr. W. Oed Cray Research GmbH, München	Das massiv–parallele Prozessorsystem CRAY T3E
15.05.96	Dr. T. C. Lüth Forschungszentrum Informatik, Karlsruhe	Multi–Agentensysteme in der Robotik
22.05.96	Dr. S. Hahndel Institut für Informatik der TUM	Das verteilte, verhandlungsbasierte Planungsverfahren NEDIP
05.06.96	Dr. G. Dreo–Rodosek Leibniz-Rechenzentrum München	Verteiltes, skalierbares DV-Management
19.06.96	Dr. H. Schlingloff Institut für Informatik der TUM	Modallogik und propositionaler μ-Kalkül – Grundlagen und Anwendungen
26.06.96	Dr. G. Weiss Institut für Informatik der TUM	Lernen in Mehragentensystemen
17.07.96	Prof. Dr. H.–W. Meuer Rechenzentrum der Universität Mannheim	Das TOP500 Projekt der Universitäten Mannheim und Tennessee zur Evaluierung des Supercomputer Marktes
29.01.97	Prof. Dr. Kurt Geihs FB Informatik der Universität Frankfurt	Dienstmanagement in offenen verteilten Systemen
19.02.97	Prof. Dr. Jochen Pfalzgraf Universität Salzburg und RISC – Linz	Logische Faserungen: Ein Ansatz zur Modellierung verteilter Logiksysteme (Grundlagen und erste Anwendungen)
26.02.97	Dr. Peter Buchholz Universität Dortmund	Strukturierte Techniken zur quantitativen Analyse dynamischer Systeme

7 Zwischenbilanz

Rund zwei Jahre nach Einrichtung des Kollegs lassen sich die bisherigen Erfahrungen wie folgt zusammenfassen:

- Die Konzeption des Kollegs mit der zweifachen Integration der Doktoranden in das Kolleg und in die Arbeitsgruppen ihrer Heimatlehrstühle hat sich für die Betreuung der Kollegiaten, für die Entwicklung der Zusammenarbeit im Kolleg und für die Entwicklung der Zusammenarbeit der Kollegiaten mit Gruppen im näheren und weiteren Umfeld des Kollegs bewährt und soll weitergeführt werden.

- Die Anforderungen, die an die Kollegiaten gestellt werden, einerseits gemeinsam zum Fortschritt des Verständnisses der Aufgabenstellung des Kollegs und ihrer Lösungen beizutragen und andererseits eine selbständige, wissenschaftliche Leistung zu den Themen, die sie bearbeiten, zu erbringen, sind hoch. Die bisherige Entwicklung des Kollegs hat die Vorzüge dieses integrativen Arbeitens, aber auch seine Grenzen deutlich gezeigt. Diese Erfahrungen spiegeln die Schwierigkeiten wider, die sich daraus ergeben, daß Spezialisierungen notwendig sind und gleichzeitig verengte, einseitige Sichten auf die gestellten Aufgaben entgegenzuwirken ist. Dieses Spannungsfeld ist jedoch charakteristisch für die Aufgaben der Informatik und insbesondere für die Aufgabenstellung „Kooperation und Ressourcenmanagement in verteilten Systemen“, in denen vielfältige, unterschiedliche Anforderungen und entsprechend unterschiedliche Lösungsansätze zusammentreffen. Diese Schwierigkeiten lassen sich nach den bisherigen Erfahrungen im Rahmen der Konzeption für die Arbeiten des Kollegs mit einem flexiblen, an die unterschiedlichen Anforderungen angepaßten Ausbildungs–, Betreuungs– und Erfolgskontrollprogramm meistern.

- Das Kolleg ist im Laufe der zwei Jahre seit seiner Einrichtung zu einem wichtigen Träger der Vernetzung der Arbeiten im Kolleg und der Arbeiten im näheren und weiteren Umfeld des Kollegs geworden. Es hat sich als außerordentlich fruchtbare integrierende und Synergien erzeugende Institution erwiesen.

  Die Kollegiaten wurden zügig an die Forschungsthemen, die sie bearbeiten, herangeführt und in die Gruppen, mit denen sie zusammenarbeiten, integriert. Von den 11 Stipendiaten, die dem Kolleg seit 1995 angehören, wird erwartet, daß sie ihre Promotionsvorhaben im Rahmen der 3–jährigen Laufzeit der Stipendien erfolgreich abschließen. Wenn diese Erwartung erfüllt wird, hat das Kolleg dazu beigetragen, die Promotionszeiten gegenüber den üblichen deutlich zu verkürzen und Nachwuchskräften einen Abschluß mit hoher wissenschaftlicher Qualifikation zu ermöglichen.