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

Zwischenbericht zum Frühjahr 1996

TUM-I9611
März 1996

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-03-1996-I9611-80/1.-FI

Typescript: ---

Druck: Mathematisches Institut und
Institut für Informatik der
Technischen Universität München

Graduiertenkolleg

Kooperation und Ressourcenmanagement in verteilten Systemen

Zwischenbericht zum Frühjahr 1996

Herausgeber:

Prof. Dr. P. P. Spies
Sprecher des Graduiertenkollegs
Institut für Informatik
Technische Universität München
80290 München

Tel.: +49 - 89 - 2105 - 8252
Fax: +49 - 89 - 2105 - 2037
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 ab Anfang 1996 drei 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 Teilprojekte zusammngefaßt, mit denen von den Anforderungen ausgewählter Anwendungsgebiete ausgehend Lösungsmethoden und –verfahren für die spezifischen 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 Teilprojekte zusammengefaßt, mit denen Methoden und Verfahren zur Realisierung paralleler und kooperativer Systeme auf Hardwarekonfigurationen mit vernetzten Stellenrechnern entwickelt werden. Dabei werden Grundlagenprobleme und unmittelbar einsetzbare Verfahren behandelt.

Der vorliegende Bericht gibt einen Überblick über den gegenwärtigen Stand der Arbeiten der Kollegiaten. Er spiegelt die unterschiedlichen Aufgabenstellungen, Ansätze und Ziele der einzelnen Teilprojekte wider. Nach der für das Kolleg gewählten Vorgehensweise sollen Fortschritte bei der Entwicklung verteilter Systeme für parallele und kooperative Problemlösungen dadurch erreicht werden, daß unterschiedliche, charakteristische Anwendungsanforderungen erfüllt und dabei als geeignet erkannte Lösungsmethoden und –verfahren zusammengeführt werden. Der Bericht spiegelt den Weg zu diesem Ziel wider und macht deutlich, daß für die angestrebte Zusammenführung noch große Anstrengungen erforderlich sind.

München, im März 1996 — Prof. Dr. P. P. Spies

Graduiertenkolleg: Kooperation und Ressourcenmanagement in verteilten Systemen

Zwischenbericht zum Frühjahr 1996

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
Kollegiat:	Dirk Ormoneit
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
Kollegiat:	Maximilian Frey
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 2.7:	Verteilte numerische Algorithmen auf Bäumen
Leitung:	Prof. Dr. Chr. Zenger
Kollegiat:	Anton Frank

Wahrscheinlichkeitsdichteschätzung mit Neuronalen Netzen

Teil-Projekt: Kooperierende Agenten in verteilten Systemen - Grundlagen

Dirk Ormoneit
e-mail: ormoneit@informatik.tu-muenchen.de

1 Einführung

In jüngster Vergangenheit haben entscheidungstheoretische Konzepte verstärktes Interesse als Mittel zur Formulierung des Kooperationsverhaltens in künstlichen Multiagentensystemen gefunden (z.B. [EMR95], [MW95]). Eine zentrale Idee ist hierbei die einer Nutzenfunktion[1], die jedem (heutigen und zukünftigen) Zustand der Welt einen gewissen Nutzenwert zuordnet. In einer statischen Betrachtung läßt sich so ein Marktmodell ableiten, in dem Gleichgewichtspreise den Austausch von Leistungen zwischen Agenten dominieren (siehe z.B. [Wel93]). Fast alle praktisch relevanten Probleme sind jedoch dynamischer Natur, so daß im Entscheidungsverhalten des Agenten außerdem die Unsicherheit über die zukünftige Entwicklung seiner Umwelt berücksichtigt werden muß. Eine entsprechende Risikoaversion kann so in die Nutzenfunktion des Agenten integriert werden, daß sich das Entscheidungsproblem auf die Maximierung des erwarteten heutigen und (eventuell diskontierten) zukünftigen Nutzens reduziert (Von-Neumann-Morgenstern-Erwartungsnutzen). Notwendig zur Berechnung dieses Erwartungswertes ist die Verfügbarkeit eines Wahrscheinlichkeitsmodells über den zukünftigen Zustand der Umwelt des Agenten[2], eventuell bedingt durch zusätzlich verfügbare Information. Idealerweise soll der Agent außerdem in der Lage sein, die entsprechenden Wahrscheinlichkeitsverteilungen selbst aus seiner Erfahrung durch Interaktion mit seiner Umwelt aufzubauen. Dies erleichtert wesentlich die Spezifikation des Systems und erhält den Agenten anpassungsfähig gegenüber eventuellen Veränderungen in seiner Umwelt. 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 Hilfen von sogenannten Gaussian-Mixture-Netzwerken.

2 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

[1] Eine Funktion $u : X \rightarrow \Re$ heißt eine *Nutzenfunktion, die die Präferenzenrelation $\succeq$ repräsentiert*, wenn gilt: $\forall x, y \in X : x \succeq y \Leftrightarrow u(x) \geq u(y)$. Jede Nutzenfunktion repräsentiert eine rationale (i.e. vollständige und transitive) Präferenzrelation auf X. Die Umkehrung gilt nicht im allgemeinen.

[2] In der Praxis wird es natürlich kaum möglich sein, eine Wahrscheinlichkeitsverteilung über die gesamte Umwelt eines Agenten zu formulieren. Es ist in der Tat hinreichend, alle entscheidungsrelevanten Einflüsse in einer variablen Größe zusammenzufassen, deren Verhalten der Agent modelliert. Ein prominentes Beispiel hierfür ist die Aggregierung von Erwartungen über zukünftige Einflußgrößen für die Verfügbarkeit eines Gutes in einem Marktpreis. Eine Wahrscheinlichkeitsverteilung über zukünftige Marktpreise ist in der Tat bei gegebener Nutzenfunktion hinreichend für die Entscheidungsfindung. In diesem Beispiel bewirkt der Marktmechanismus gleichzeitig die Separierung des Vorhersageproblems vom eigentlichen Entscheidungsproblem: In einem entsprechenden Markt existiert schon heute ein Preis für ein Gut, daß erst in der Zukunft produziert wird. Ein Produzent kann deshalb unter Sicherheit seinen Produktionsplan optimieren. Das Vorhersageproblem wird von Spekulanten bewältigt, die schon heute eine möglichst objektive Berücksichtigung zukünftiger Einflußfaktoren im Marktpreis garantieren. Eine interessante Idee ist die Übertragung eines derartigen Prinzips auf künstliche Multiagentensysteme.

weite 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]). Man erhält im Prinzip ein RBF-Netz mit einer Hidden-Schicht (den Gaussians) und einer linearen Ausgabeschicht. Entsprechend lassen sich zahlreiche Methoden der Modelloptimierung, insbesondere der populäre Backpropagationalgorithmus, problemlos auf das Modell anwenden. Voraussetzung hierfür ist lediglich die Definition einer geeigneten Fehlerfunktion, wie sie in Form der (negativen) Log-Likelihood der Modellparameter zur Verfügung steht. Diese Fehlerfunktion läßt sich dann wie gewöhnlich durch einen Gradientenabstieg oder andere Algorithmen minimieren.

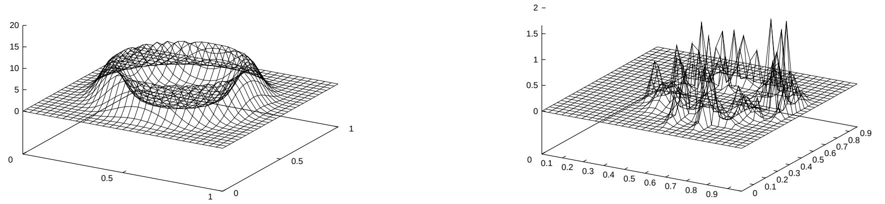


Abbildung 1: *Wahre Dichte (links) und unregularisierte Dichteschätzung (rechts).*

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 Ursache hierfür liegt im Fall der Gaussian Mixtures in der Existenz von Singularitäten in der Fehleroberfläche. Durch Konzentration der gesamten Wahrscheinlichkeitsmasse des Modells auf gerade die beobachteten Samples (die hohen Peaks im rechten Bild) kann der Fehler in trivialer Weise unendlich minimiert werden. Der eigentliche Grund für das Auftreten dieses Problems ist natürlich der Mangel an Wohldefiniertheit des Dichteschätzungsproblems aus einer endlichen Menge von Samples im allgemeinen. Die Behebung derartiger Schwierigkeiten wird im Zusammenhang mit neuronalen Netzen als Regularisierung bezeichnet. Es gibt verschiedene gebräuchliche Verfahren, von denen wir drei in den folgenden Abschnitten diskutieren wollen. Die ersten beiden sind sogenannte Bayessche Verfahren ([Mac91]), in denen man “bevorzugte” Lösungen des Optimierungsproblems in besonderer Weise auszeichnet. In der dritten Methode versucht man durch Mittelung mehrerer, unabhängig voneinander produzierter Vorhersagen die Instabilität des Schätzers zu minimieren.

2.1 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.

Im Falle der Bayesschen Regularisierung wendet man dann Standardmethoden an, um den Posterior der Parameter zu maximieren. Eine besonders einfache und effiziente Lösung ergibt sich für den populären EM Algorithmus (EM = Expectation Maximization). Man erhält einfache zusätzliche Terme in den Updategleichungen des Algorithmus, welche sich einfach und ohne Effizienzverlust implementieren lassen ([OT96]).

2.2 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 sogenannten "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. Der resultierende Algorithmus ist in der Realisierung dem regularisierten EM Algorithmus sehr ähnlich, nur daß man nun an gewissen Stellen, an denen man sonst die Parameterwerte updated, Daten aus Standardwahrscheinlichkeitsverteilungen generiert.[3]

2.3 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. Diese Divergenz zwischen den einzelnen Vorhersagen läßt sich zusätzlich verstärken, indem man die einzelnen Netze nicht auf identischen Daten, sondern auf resampleten (z.B. durch zufälliges Ziehen mit Zurücklegen) Instanzen des ursprünglichen Datensatzes trainiert. Voraussetzung für die Verbesserung durch die genannte Methode ist allerdings eine gewisse Instabilität der Prediktoren, wie sie für den Fall von neuronalen Netzen gegeben ist.

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 führt. Alle vorgeschlagenen Regularisierungsmethoden verbessern das Ergebnis jedoch statistisch signifikant gegenüber der unregularisierten Dichteschätzung.

3 Bedingte Dichten

Zur Charakterisierung von Wahrscheinlichkeiten, deren Ausprägung durch zusätzliche Informationen (dem Wert zusätzlicher Variablen) dominiert wird, verwenden wir das Konzept der bedingten Wahrscheinlichkeitsdichte.[4] Bedingte Dichten lassen sich mit dem sogenannten Conditional Density Estimation Network (CDEN, [NHFO94]) approximieren, dessen schematischer Aufbau in Abbildung 2 dargestellt ist.

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

[3] Eine bisher unbeantwortete Frage ist die nach der "korrekten" Wahl der Prior-Verteilung. Die Hyperparameter im Conjugate Prior spielen in der Tat die Rolle von suffizienten Statistiken für eine imaginäre Menge von zusätzlichen Daten. Man kann nun seine "Prior Beliefs" in Form von derartigen suffizienten Statistiken formulieren und deren Intensität durch Ausprobieren verschiedener Größen dieser imaginären Datenmenge variieren. Der wesentliche Vorteil dieses Vorgehens ist, daß jetzt nur noch ein Hyperparameter eingestellt werden muß. Ein Kriterium zur Auswahl eines optimalen Wertes ergibt sich entweder durch Cross-Validierung oder geeignete Bayessche Methoden.

[4] 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.

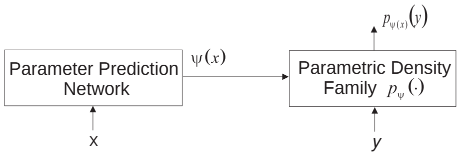


Abbildung 2: *Conditional Density Estimation Network (CDEN)*

legt außerdem eine Variante des Backpropagation-Algorithmus sowie anderer NN-Lernalgorithmen zur Optimierung nahe.

Je nach Problemstellung können verschiedene parametrische Dichten für $p_\psi(y)$ verwendet werden. Eine naheliegende Wahl sind Gaussian Mixtures, die wir ja schon zur Schätzung unbedingter Dichten verwendet haben. Interessanterweise stellt sich heraus, daß die sich ergebende Architektur unter Verwendung eines (verallgemeinert) linearen $\psi(x)$ äquivalent zum populären Modell der "Hierarchical Mixture of Experts" ([JJ92]) ist. Für diesen Fall lassen sich prinzipiell auch die zuvor diskutierten Bayesschen Methoden auf die Schätzung von bedingten Dichten anwenden. Man erhält im Prinzip entsprechende Varianten des EM und des Gibbs-Sampling Algorithmus ([JJ92], [PJT96]). Aufgrund der komplizierteren Struktur der Posterior-Verteilung stellt sich jedoch heraus, daß beide nicht ohne Zuhilfenahme komplizierterer Teilalgorithmen (Second-Order-Optimierungsmethoden bzw. Metropolis-Algorithmus) auskommen. In unseren Experimenten haben wir uns deshalb auf den allgemeineren Fall eines nichtlinearen $\psi(x)$ ohne die Anwendung Bayesscher Methoden konzentriert.

4 Zusammenfassung

Das Verhalten eines Agenten läßt sich als ein mathematisches Entscheidungsproblem unter Unsicherheit formalisieren. Zur Auflösung dieser Unsicherheit ist die Existenz eines Wahrscheinlichkeitsmodells über die Umwelt des Agenten erforderlich. Neuronale Netze bilden eine vielversprechende Grundlage zur Schätzung sowohl von unbedingten als auch von bedingten Wahrscheinlichkeitsdichten. Wesentliche Probleme entstehen jedoch aufgrund des Mangels an Wohldefiniertheit des Dichteschätzungsproblems. Für den Fall von gemeinsamen Dichten haben wir verschiedene Methoden untersucht, um diese Schwierigkeiten zu umgehen.

5 Ausblick & Zusammenarbeit mit anderen Teilprojekten

In der weiteren Arbeit gilt es zunächst, ähnliche Methoden wie die hier behandelten für den Fall der Schätzung von bedingten Wahrscheinlichkeitsdichten zu entwickeln. Einen vielversprechenden Ansatz hierzu bildet eine Variante des CDEN, in der eine parametrische Dichte aus der Familie der Exponentialverteilungen verwendet wird. Letztere haben den Vorteil, daß sie für eine vorgegebene Anzahl von Parametern eine maximale Entropie aufweisen. Im Sinne der Informatik wird also durch die Vorgabe einer Struktur so wenig wie möglich über die tatsächliche Verteilung der Daten ausgesagt. Die Evaluierung dieser Methode sowie ihr Vergleich mit den bestehenden Algorithmen erfordert eine sorgfältige Implementierung sowie umfangreiche Experimente. Desweiteren gilt es, existierende Marktmechanismen in Hinblick auf ihre Anwendbarkeit in Multiagentensystemen zu untersuchen und so unter Umständen zu einem praktisch implementierbaren Konzept für die Interaktion von Agenten in dynamischen Entscheidungssituationen zu gelangen.

In den anderen Teilprojekten des Graduiertenkollegs bieten sich zahlreiche Anwendungs- und Testmöglichkeiten der in dieser Arbeit vorgestellten Ideen. Obwohl natürlich prinzipiell keine Beschränkung auf spezielle Gebiete existiert, erscheint vor allem eine Anwendung auf die Themen "Kooperierende Agenten und autonome Robotersysteme" (F. Fuchs) sowie "Kooperierende Agenten im Netz- und Systemmanagement" (B. Quendt) attraktiv. Idealerweise wäre es möglich, die vorgestellte Theorie so weit voranzutreiben, daß sich ihre Implikationen unter einer der dort verwendeten Simulationsumgebungen praktisch erproben lassen.

Literatur

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

[EMR95] E. Ephrati, M.E.Pollack und J.S. Rosenschein. A Tractable Heuristic that Maximizes Global Utility through Local Plan Combination. In *Proceedings First International Conference on Multi-Agent Systems*, 1995.

[JJ92] M.I. Jordan und R.A. Jacobs. Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems 4*. Morgan Kaufman, 1992.

[Mac91] D. MacKay. *Bayesian Modelling and Neural Networks.* Dissertation, California Institute of Technology, Pasadena, 1991.

[MW95] T. Mullen und M.P. Wellman. A Simple Computational Market for Network Information Services. In *Proceedings First International Conference on Multi-Agent Systems*, 1995.

[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.

[OT96] D. Ormoneit und V. Tresp. Improved Gaussian Mixture Density Estimates Using Bayesian Penalty Terms and Network Averaging. In *Advances in Neural Information Processing Systems 8*, 1996.

[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.

[PJT96] F. Peng, R.A. Jacobs und M.A. Tanner. Bayesian Inference in Mixtures of Experts and Hierarchical Mixtures of Experts Models With an Application to Speech Recognition. *Journal of the American Statistical Association (accepted for publication)*, 1996.

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

Tradingagenturen als Grundlage eines Signalisierungssystems mit Dienstanbieterauswahl

Teil-Projekt: Ressourcenmanagement in breitbandigen ATM-Kommunikationsnetzen

Bernhard Quendt

e-mail: bq@lkn.e-technik.tu-muenchen.de

1 Einführung

Breitbandige Kommunikationsnetze sind hochgradig verteilte Systeme mit einer Vielzahl heterogener Ressourcen. Der Ressourcenbegriff umfaßt dabei im folgenden Übertragungs- und Vermittlungseinrichtungen, als auch Spezialressourcen für die Multi-Media-Kommunikation[1] und Leistungsmerkmale[2]. Einen multimedialen Kommunikationsdienst über diese Netze anzubieten, stellt bereits heute komplexe Anforderungen an das Signalisierungssystem [Ebe92]. Mit der Deregulierung auf dem Telekommunikationsmarkt wird die Komplexität weiter gesteigert durch:

- Die Existenz einer Vielzahl unterschiedlicher Anbieter für Ressourcen, Dienste und Leistungsmerkmale.
- Die Existenz einer Vielzahl unterschiedlicher Tarife und Tarifierungsregeln.
- Die variable Zahl angebotener Dienste und Ressourcen.
- Die dynamische Tarifierung, bei der die Preise an die Ressourcenbelastung angepaßt werden, um auf diesem Wege eine gleichmäßigere Auslastung zu erreichen.

Damit ein Kommunikationsteilnehmer einen größtmöglichen Nutzen aus der Deregulierung zieht, muß ein Signalisierungssystem folgende Funktionalität besitzen :

- Die Nutzung verschiedener Anbieter für die benötigten Ressourcen eines Kommunikationsdienstes muß unterstützt werden.
- Die Verfügbarkeit der Anbieter und deren aktuelle Angebote oder Tarifierungsregeln müssen bekannt sein.
- Die Anbieterauswahl durch den Kommunikationsteilnehmer muß möglichst transparent unterstützt werden.

Ersteres wird bereits durch Signalisierungsarchitekturen erfüllt, die auf einer seperaten Ruf- und Verbindungssteuerung [BMMM94], bzw. auf einer Trennung von Ruf-, Verbindungs- und Ressourcensteuerung [Mü95a] basieren. Es bleibt die Forderung nach einem Mechanismus zur Auswahl von Dienstanbietern unter Berücksichtigung seiner Preise. Diesen bereitzustellen, ist in dem beschriebenen Zusammenhang eine Grundaufgabe des zukünftigen Signalisierungssystems.

2 Dienstanbieterauswahl in Ansätzen für B-ISDN Signalisierungssysteme

Vorschläge der ITU für die Signalisierung im Breitbandnetz B-ISDN (CS1, [ATM94]) sehen den in Abschnitt 1 beschriebenen Auswahldienst bisher nicht vor. Zumindest was die Spezialressourcen angeht, wird er auch nicht benötigt: deren Funktionalität muß in den Endgeräten der Teilnehmer

[1] Spezialressourcen sind Zusatzeinrichtungen, die für die Multi-Media-Kommunikation zwischen heterogenen Endgeräten benötigt werden, z.B.Konverter oder Mischbrücken.

[2] Leistungsmerkmale sind Zusatzdienste, die den Komfort eines Basisdienstes erhöhen, z.B. Rufumleitung oder Rückruf.

zur Verfügung stehen. Gleiches gilt für die Vorschläge im Forschungsprojekt RACE II MAGIC [RAC93]: hier geht man davon aus, daß die Funktionalität der Spezialressourcen in den Vermittlungsknoten verfügbar ist.
Am weitesten fortgeschritten ist der Ansatz von H. Müller [Mü95a]. Die Auswahl unterschiedlicher Anbieter erfolgt dort unter Berücksichtigung ihrer Preise in der Ressourcen- und der Verbindungssteuerung. Bei der Auswahl der Übertragungswege werden entfernungsabhängige Schätzwerte angesetzt. Andere Tarifmodelle oder dynamisch angepaßte Preise werden nicht unterstützt.

3 Marktplätze, Tradingsysteme und Tradingagenturen

Die Ressourcenauswahl für einen Kommunikationsdienst ist vergleichbar mit Schedulingaufgaben in anderen Disziplinen, etwa im Bereich der Logistik oder der flexiblen Fertigung. Als Methoden kommen dort das **Kontrakt-Netz-Protokoll** [DS83] und das **Specification-Sharing-Protokoll** [GK94] zum Einsatz. Sie unterscheiden sich in der Durchführung der Angebotsrecherche. Das Kontrakt-Netz-Protokoll sieht einen Ausschreibungsprozeß vor, an dessen Ende der Dienstnutzer die Dienstanbieter kontaktiert, deren Angebote sammelt und auswertet. Beim Specification-Sharing-Protokoll tritt der Dienstanbieter mit seinen Angeboten aktiv an den Dienstnutzer heran. In beiden Fällen wird ein netzweiter Verteildienst (Broadcast) benötigt. Der praktische Einsatz ist damit auf Netze beschränkt, die nur kleine örtliche Ausdehnung und genügend Übertragungskapazität besitzen.
Kennzeichen beider Protokolle ist eine **direkte Angebotsrecherche**. Darunter soll im folgenden eine Angebotsrecherche verstanden werden, in die nur Dienstnutzer und Dienstanbieter involviert sind. Im Falle einer **indirekten Angebotsrecherche** erfolgt die Recherche durch eine dritte Instanz. Je nach Grad der Unterstützung von Dienstnutzern und Dienstanbieters durch die dritte Instanz kann weiter zwischen einer **passiven**, einer **semi-aktiven** und einer **aktiven, indirekten Angebotsrecherche** unterschieden werden.
Erfolgt eine indirekte Angebotsrecherche, ist kein Broadcast nötig. Dies schont Ressourcen im Netz. Der **elektronische Marktplatz** ist ein Konzept, das die passive, indirekte Angebotsrecherche unterstützt. Dienstanbieter speichern dort ihre Angebote, der Dienstnutzer erhält vom Markt auf Anfrage alle Angebote und selektiert daraus das für sich passende. Die Selektionsfunktionalität muß somit vom Dienstnutzer bereitgestellt werden.
Tradingsysteme unterstützen die semi-aktive, indirekte Angebotsrecherche. Sie werden zur Dienstevermittlung in offenen, verteilten Systemen eingesetzt [SPM95]. Auch sie kommen ohne einen netzweiten Broadcast aus, benötigen aber eine zentrale Vermittlungseinrichtung (Trader). Im folgenden ist ihre Funktionsweise kurz charakterisiert, für eine ausführliche Beschreibung sei auf [SPM95] verwiesen. Der Trader vermittelt zwischen Dienstanbietern (Exporteuren) und Dienstnutzern (Importeuren). Der Importeur richtet eine Anfrage in Form einer abstrakten Dienstbeschreibung an den Trader. Dieser sucht daraufhin den passenden Exporteur, der zuvor in einem Exportprozeß die von ihm angebotenen Dienste dem Trader bekanntgemacht hat.
Im Unterschied zum elektronischen Marktplatz stellt in Tradingsystemen der Trader die Funktionalität zur Diensteselektion bereit. Dadurch wird der Importeur entlastet.
Tradingagenturen (TA) gehen in ihrer Funktionalität weit über Tradingsysteme hinaus. Sie bilden ein neues Konzept zur aktiven, indirekten Angebotsrecherche: Importeure, Exporteure und auch andere Agenturen werden durch eine Vielzahl von Tradingdiensten bestangepaßt unterstützt. Darüberhinaus profitieren diese Agenturkunden indirekt von den Managementeinrichtungen der TA. Ihre Aufgabe ist es, den Betrieb der TA zu überwachen und damit die Qualität der Tradingdienste sicherzustellen. Im einzelnen sind dies Einrichtungen für das Anfragen-, Export-, Konfigurations-, Änderungs-, Leistungs- und Fehlermanagement.
Mit Hilfe von Tradingagenturen für Kommunikationsressourcen kann ein Signalisierungsystem die in Abschnitt 1 geforderte Funktionalität ohne große zusätzliche Netzbelastung erbringen. Anders als bei Verwendung des Kontrakt-Netz-Protokolles ist nicht für jeden Rufaufbau eine seperate Angebotsrecherche erforderlich. Sie wird in größeren Zeitabständen von den TA durchgeführt. Pro Rufaufbau wird durch die Nutzung der Tradingdienste lediglich auf deren Ergebnisse zugegriffen.

Im Unterschied zu elektronischen Märkten und Tradingsystemen bieten Tradingagenturen dem Signalisierungssystem eine Vielzahl von Tradingdiensten unterschiedlicher Qualität und Nutzungsgebühr. Gibt das Signalisierungssystem diese Auswahlmöglichkeit an den Kommunikationsteilnehmer weiter, kann dieser selbst entscheiden, wieviel er für den Tradingdienst bezahlen möchte.
In Abbildung 1 ist die Architektur von Tradingagenturen dargestellt. Sie besitzt einen modularen und hierarchischen Aufbau. Die Modularität zeigt sich in der Aufteilung der einzelnen Managementaufgaben auf die Managementeinrichtungen (ME). Dies unterstützt die Erstellung, die Wartung und die Änderung der TA. Außerdem vereinfacht sich die Verwaltung der ME, insbesondere die schnelle Fehlerlokalisierung und damit i.d.R. auch die schnelle Behebung des Fehlers.

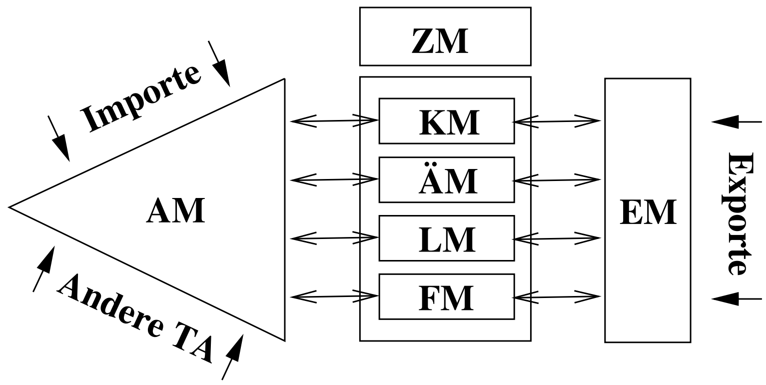


Abbildung 1: Die Architektur der Tradingagentur

Die hierarchische Organisation besteht aus drei Stufen und ergibt sich aus den Aufgaben der einzelnen ME. Die Managementeinheit Anfragemanagement [AM] überwacht die Anfragen der Importeure und anderer TA auf die Nutzung von Tradingdiensten. Vergleichbar dazu observiert das Exportmanagement [EM] die Abwicklung der Exporte. [AM] und [EM] werden durch die Managementseinheiten der zweiten Hierarchiestufe kontrolliert. Diese sind Konfigurations-, Änderungs-, Leistungs- und Fehlermanagement [KM], [ÄM], [LM], [FM]. Als letzte Hierarchiestufe verwaltet die zentrale Managementeinheit [ZM] die ME der zweiten Hierarchiestufe. Insbesondere kommt sie bei der Schlichtung möglicher Konflikte der ME zum Einsatz und regelt den direkten Zugriff anderer TA auf diese ME.
Die Komplexität der Managementaufgaben erfordert eine enge Kooperation der einzelnen Managementeinheiten. Ein Beispiel ist die Überlastabwehr in der TA durch Zusammenwirken von [AM] und [LM]. Ein weiteres Beispiel ist die Funktionalitätserweiterung der TA durch Tradingdienste anderer TA als gemeinsame Aufgabe von [ÄM] und [ZM].

4 Modellierung der Tradingagenturen durch ein MAS

Einen vielversprechenden Lösungsansatz für die Aufgaben der TA bietet das Konzept der Multiagentensysteme (MAS) aus der Domäne der Verteilten Künstlichen Intelligenz (VKI). MAS sind zunächst ein abstraktes Modellierungskonzept, das im wesentlichen auf der Identifikation von Rollen in der zu modellierenden Umgebung basiert. Rollen sind durch charakteristische Ziele und Verhaltensweisen gekennzeichnet und werden von Agenten (s.u.) ausgeführt. Beobachtet man die Auswirkung der Rollen auf die Agentengesellschaft, so erzielt man u.U interessante Rückschlüsse auf die eigentliche Aufgabenstellung, die andere Modellierungsansätze nicht ermöglichen. Einen plausiblen Grund dafür liefert K. Sundermeyer in [Sun93]. Er nennt als Vorteil des Ansatzes die natürliche und stark abstrahierende (und damit überschaubare) Problemmodellierung.
Wie oben beschrieben, werden die identifizierten Rollen durch Agenten ausgeführt. Die Rolle sollte dabei möglichst in ihrer Gesamtheit erfaßt werden. Dies verlangt nach gewissen Grundeigenschaften eines Agenten.

Nach M. Wooldridge [WJ95] zeichnen sich Agenten durch ihre Fähigkeit zu autonomen Handlungen, sowie durch Kommunikationsvermögen, Reaktivität und Eigeninitiative aus.
H. J. Müller identifiziert in [Mü95b] die explizite Wissensrepräsentation und die Fähigkeit zur Kommunikation als wesentliche Kennzeichen eines Agenten. Weiterhin unterscheidet er drei Klassen von Eigenschaften: Primärtechnologien (Aktorik, Sensorik, Kommunikationsfähigkeit), Sekundärtechnologien (Planungsfähigkeit, Lernfähigkeit, Konfliktbewältigung innerhalb eines Agenten) und Tertiärtechnologien (Partnermodellierung, Interagenten-Konfliktbewältigung). Analysiert man[3] einige existierende Agentensysteme [Kea95, HS95, Sho93], so lassen sich (meistens) die drei folgenden Eigenschaften feststellen:

Agentenaktionszyklus: Agenten handeln zielorientiert. Aus ihren Zielen und ihrem Wissen über die Umwelt generieren sie zunächst Handlungspläne zur Erfüllung dieser Ziele. Diese Pläne werden u.U noch weiter verfeinert, bevor sie ausgeführt werden. Nach der Ausführung hat sich die Umwelt des Agenten geändert. Durch eine Sensorikkomponente werden die Änderungen erfaßt. Es folgt die Auswertung und die Aktualisierung des Wissens über die Umwelt. Der Zyklus schließt sich, indem der Agent, basierend auf seinem neuen Wissen, erneut mit der Erzeugung von Plänen beginnt.

Kommunikationsfähigkeit: Agenten bedienen sich einer agentenunabhängigen Kommunikationssprache, die meist drei Schichten aufweist. Die erste Schicht definiert das verwendete Vokabular, insbesondere anwendungsspezifische Begriffe. Die zweite Schicht beinhaltet die syntaktischen und semantischen Regeln, nach denen Mitteilungen gebildet werden. Schicht 3 definiert linguistische Regeln, die verschiedene Mitteilungen zueinander in Verbindung setzen.

Kooperationsfähigkeit: Agenten lösen ihre Aufgaben unter Einbeziehung anderer Agenten. Eine Kooperationskomponente unterstützt dabei die Auswahl von Partneragenten, die Übertragung von Teilaufgaben und das Lösen von Konflikten.

Agenten, die für diese drei Fähigkeiten Methoden der Künstlichen Intelligenz (KI) einsetzen, insbesondere die explizite Wissensrepräsentation, werden in der Folge als Intelligente Agenten bezeichnet.
Die Managementeinheiten (ME) einer Tradingagentur (TA) legen Rollen gemäß obiger Definition fest. Die Erfüllung der zugeordneten Managementaufgaben ist ihr Ziel, das sie mit bestimmten Verhaltensweisen, insbesondere der engen Kooperation mit anderen ME, zu erreichen versuchen. Damit ist die Grundvoraussetzung für eine erfolgversprechende Modellierung der TA durch ein MAS und der ME durch Agenten erfüllt. Konkret erlaubt dies, die Ergebnisse der MAS-Forschung für die vorliegende Aufgabenstellung zu nutzen. Insbesondere lassen sich u.U Teile des Systemverhaltens ohne die Notwendigkeit aufwendiger Simulationen vorhersagen. Oder man kann auf existierende Strategien zurückgreifen, um die speziellen Anforderungen des Anwendungsgebiets zu erfüllen. Dazu zählen insbesondere :

Kritische Echtzeitanforderungen: der Rufaufbau sollte unter einer Sekunde liegen.

Knappe Netzressourcen: die zusätzliche Netzbelastung durch die Angebotsrecherche der TA muß minimal gahalten werden.

Hohe Zuverlässigkeit: die praktische Nutzbarkeit der Konzepte wird wesentlich von der Zuverlässigkeit der Ergebnisse abhängen, die die TA liefert. Insbesondere ist auf die Aktualität der Preise und die Verfügbarkeit der Anbieter zu achten.

Ökonomischer Nutzen: die Kosten für die Nutzung der Tradingdienste dürfen den zu erwartenden Gewinn durch die Dienstanbieterauswahl nicht erheblich schmälern.

[3] Die aufgeführten Eigenschaften sind konform zu Teilergebnisse des Arbeitskreises Multiagentendefinition (AK-MAD) des Graduiertenkollegs

5 Zusammenfassung

Mit der Tradingagentur (TA) wurde ein Konzept für eine Dienstevermittlung vorgestellt, das eine verbesserte Anpassung an die Importeure/Exporteure ermöglicht und eine strukturierte innere Verwaltung besitzt. Letzteres erlaubt es der TA, Tradingdienste prinzipiell auch unter hohen anwendungsspezifischen Anforderungen bereitzustellen. Eine mögliche Anwendung ist die Unterstützung des zukünftigen B-ISDN Signalisierungssystems bei der Auswahl von Ressourcen verschiedener Anbieter in der Rufaufbauphase.

Literatur

[ATM94] ATM Forum. *Overview of ITU-T B-ISDN CS1 and ATM Forum Phase 1 Signalling , Draft 0C*, Oktober 1994.

[BMMM94] H. Bussey, S. Minzer, P. Mouchtaris und S. L. Moyer. EXPANSE Software for Distributed Call and Connection Control. *Int. Journal of Communicatiosn Systems*, 7(2):149–160, April - Juni 1994.

[DS83] R. Davis und R. G. Smith. Negotiation as a methaphor for distributed problem solving. *Artificial Intelligence*, 20, 1983.

[Ebe92] J. Eberspächer. *Vorlesung Kommunikationsnetze 2*. Lehrstuhl für Kommunikationsnetze, TU München, 1992.

[GK94] M. Genesereth und S. Ketchpel. Software Agents. *Communications of the ACM*, 37(7):48 – 53, 1994.

[HS95] H. Haugeneder und D. Steiner. Cooperative Agents: Concepts and Applications. In *Agent Software Seminar*, London, 1995. UNICOM.

[Kea95] P. Kearney. Intelligent Agents and Personal Electronics. In *Agent Software Seminar*, London, 1995. UNICOM.

[Mü95a] H. Müller. A Signalling Framework for Multimedia Broadband Networks. In *Proceedings of 2nd IEEE MICC'95*, Langkawi, November 1995. IEEE.

[Mü95b] H. J. Müller. Zur Korrelation von Anwendungsprofilen und Agentenmodellen, eine Fallstudie. Vortrag anläßlich des SFB/GK - Kolloquiums der Fakultät Informatik der TU-München, Dezember 1995.

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

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

[SPM95] O. Spaniol, C. Popien und B. Meyer. *Dienste und Dienstvermittlung in Client/Server Systemen*, Jgg. 1 of *Thomson's aktuelle Tutorien*. International Thomson Publishing, Bonn, 1995.

[Sun93] K. Sundermeyer. Modellierung von Agentensystemen. In J. Müller, Hrsg., *Verteilte Künstliche Intelligenz — Methoden und Anwendungen*, Seiten 22–44. BI Wissenschaftsverlag, 1993.

[WJ95] M. Wooldridge und N. Jennings, Hrsg. *Intelligent Agents - ECAI - 94 Workshop in Agent Theories, Architectures and Languages*, Jgg. 890 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin, 1995.

Ein auf Intelligenten Agenten basierendes Modell für Netz- und Systemmanagement

Teil-Projekt: Kooperierende Agenten im Netz- und Systemmanagement

Maria-Athina Mountzia

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

1 Einführung

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, die Anzahl der angebotenen Dienste und der betroffenen Organisationsdomänen.

Verteilte Künstliche Intelligenz beschäftigt sich seit Jahren mit Problemen, für deren Lösung die Konzepte von *Cooperative Distributed Problem Solving (CDPS)* und intelligenten Agenten erfolgreich eingesetzt wurden. Demgemäß besteht unser Ziel darin, diese Konzepte für die Entwicklung einer Methodik zur Realisierung von komplexen, verteilten Aufgaben im Bereich des Netz- und Systemmanagements zu verwenden.

Die Methodik besteht aus folgenden Schritten. Zuerst wird die Aufgabe anhand von Kriterien in eine Anzahl von Teilaufgaben zerlegt. Jede dieser Teilaufgaben wird von einem intelligenten Agenten realisiert und alle intelligenten Agenten kooperieren zur Ausführung der gesamten Aufgabe. Das Konzept wird auf unsere Managementarchitektur abgebildet und mit den dabei existierenden Werkzeugen implementiert.

Das Hauptkonzept zur Abbildung der Methodik in Managementarchitekturen ist *Management by Delegation* ([GY95]), das auf den von uns vorgeschlagenen *flexiblen Agenten* basiert. *Flexible Agenten* erweitern die heutigen Managementmodelle, indem sie eine effektive und flexible Verteilung von Aufgaben und Funktionalität zwischen Managementeinheiten erlauben.

2 Motivation

Die Rollen in allen Managementsystemen sind: Manager und Agent. Bislang konnte man durch die Menge der beim Manager liegenden Funktionalität und seiner Fähigkeit, Managementoperationen zu initiieren, zwischen den beiden Begriffen unterscheiden. Trotzdem reichen die beiden Kriterien für die Bezeichnung einer Einheit als Manager oder Agent nicht aus. Die Managementarchitekturen ermöglichen inzwischen nicht nur die Delegierung von Managementfunktionalität vom Manager auf die Agenten, sondern auch die dynamische Änderung von Manager-Agenten-Rollen.

Es gibt zum Beispiel Agenten, die die *Remote Monitoring (RMON)* und *Manager to Manager (M2M)* Management Information Base (MIB) realisieren ([Wal91], [CMRW93]), was die Verlagerung von Managementfunktionalität auf Agenten und hierarchisches Management ermöglicht. Die von OSI definierte Managementarchitektur erlaubt eine flexible Zuteilung von Rollen zwischen den Managementeinheiten, hierarchische Strukturen und erweiterte Funktionalität, wie von den *Systems Management Functions (SMFs)* geliefert [ISO94].

Obwohl das Potential für dezentralisiertes Management vorhanden ist, weisen die meisten Managementsysteme nur ein zentralisiertes, plattform-basiertes, sehr aufwendiges Interaktionsparadigma. Da aber die Verteilung von Ressourcen und Diensten eines der größten Probleme in der Realisierung von Managementanwendungen ist, eignet sich ein zentralisierter Ansatz aus Leistungs- und Effektivitätsgründen nicht. Andererseits erfordert die dynamische Natur der heutigen verteilten Systeme auch ein flexibles Managementmodell, das die Anpassung des Systems an den jeweiligen Änderungen ermöglicht. Aus diesen Gründen konzentrieren wir uns auf die Konzepte von *Management by Delegation*, besonders die dynamische Verlagerung von Managementfunktionalität auf die Agenten und *Kooperatives Management* im Sinne der Kooperation unter den Agenten.

3 Intelligente Agenten

Um flexible Kooperation unter den Managementeinheiten zu haben, brauchen wir Agenten, die bei der Realisierung von Managementaufgaben intelligenter agieren. Deswegen untersuchen wir den Bereich von intelligenten Agenten, wo Eigenschaften wie Mobilität und Kooperation von großer Bedeutung sind.

In den letzten Jahren gab es großes Interesse an Agententechnologie. Agenten werden in unterschiedlichsten Bereichen benutzt, wie verteilten Systemen, Rechnergestützter Gruppenarbeit, Fertigungssystemen, Robotertechnik usw. 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.

Wooldridge und Jennings [WJ95] definieren einen Agenten als *ein Hardware- oder Softwaresystem, das folgende Eigenschaften aufweist:* Autonomie, soziales Verhalten, Reaktivität, Proaktivität. Dazu kommt eine strengere Definition, die von den folgenden Eigenschaften geprägt wird: Mobilität, Wahrheitstreue, Gutartigkeit, Rationalität.

Im allgemeinen ist ein Agent ein System, das Aufgaben erfüllt, die für den Benutzer von Interesse sind. Agenten sind besonders für Anwendungen geeignet die Verteilung, Kooperation und Heterogenität beinhalten und wo Skalierbarkeit ein sehr wichtiger Faktor ist. Agenten arbeiten mit verschiedenen Problemlösungsstrategien, und ermöglichen dadurch Wiederverwendbarkeit, daß ein Agent in vielen verschiedenen Anwendungen benutzt werden kann.

Da solche Eigenschaften bei Netz- und Systemmanagementanwendungen sehr wichtig sind, ist es wünschenswert, daß wir von Konzepten und Lösungen im Bereich von intelligenten Agenten zu profitieren versuchen, um Probleme im Bereich von Netz- und Systemmanagement zu lösen.

Wir untersuchen die Konzepte von CDPS, die die Zerlegung von komplexen Problemen und ihre Implementierung mit kooperierenden intelligenten Agenten ermöglichen. Dabei findet man spezielle Rahmenwerke zur Realisierung der Interaktion unter den Agenten und eine Methodik zur Strukturierung dieser Interaktionen. Beispiele solcher Rahmenwerke sind dezentralisierte Plattformen wie ARCHON [WJM94] und GRATE [JML+92], die zwei Zielen dienen: Erstens bieten sie die nötige Umgebung und Kontrolle zur Kooperation unter den Agenten und zweitens erlauben sie die Entwicklung einer Methodik zur Zerlegung einer globalen Aufgabe und zur Verteilung der entsprechenden Teilaufgaben. Diese Rahmenwerke sind in einer Anzahl von Industrieanwendungen eingesetzt worden, wie *electricity transportation management, fault diagnosis in an electricity network* und *particle accelerator control* ([CVJ92], [JCL95]).

Außerdem wurde bezüglich der Kooperations- und Kommunikationsaspekte zwischen intelligenten Agenten bereits viel Arbeit geleistet, die zur Entwicklung von z.B. *Knowledge Query Manipulation Language (KQML)* [FFMM93], *AgentTalk* usw. geführt hat. Im *KADS task modelling framework* [DOS94], findet man Modellierungstechniken zur Spezifizierung der Aufgaben, die von den Agenten übernommen werden sollen.

4 Methodik

In diesem Abschnitt wird der hier entwickelte Ansatz beschrieben. Gemäß des Konzepts von *CDPS* wird eine Aufgabe in eine Anzahl von Teilaufgaben zerlegt. Jede dieser Teilaufgaben wird von einem intelligenten Agenten gelöst. Alle intelligenten Agenten kooperieren zur Ausführung der gesamten Aufgabe. Das wird in der ersten Ebene der Abbildung 1 gezeigt.

Dieses Konzept muß auf unsere Managementarchitektur abgebildet werden. Das wird folgendermaßen gemacht: die Teilaufgaben werden als funktionale Einheiten beschrieben, die unter den Managementagenten verteilt werden. Die Managementagenten müssen eine Architektur besitzen, die es ihnen ermöglicht, die neue Funktionalität anzunehmen und auszuführen. Dieser Ansatz wird in der zweiten Ebene der Abbildung 1 dargestellt.

Es muß betont werden, daß die in der ersten Ebene gezeigten intelligenten Agenten nur die Konzepte anbieten, die wir zur Realisierung von *flexiblen Managementagenten* brauchen. *Flexible Agenten* sind die Instantiierung des Konzepts in unserer Umgebung. Das sind Agenten, die neue

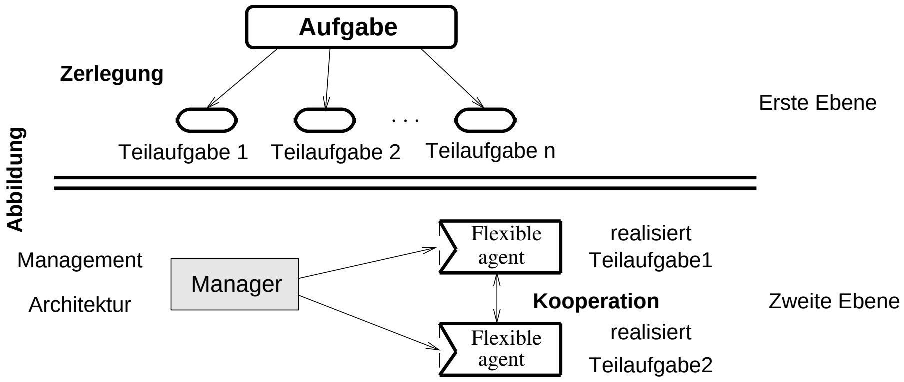


Abbildung 1: Lösungsansatz

Funktionalität bekommen können und die an der kooperativen Ausführung einer Managementaufgabe teilnehmen können. Das wird im Bereich des Netz- und Systemmanagements Delegierung von Funktionalität auf die Agenten (management by delegation) [GY95] genannt. Deshalb besteht der wichtigste Aspekt von Intelligenz und Flexibilität in unserem Umfeld in Funktionalität, die nicht bloß auf dem Agenten liegt und aufgerufen wird, sondern in Funktionalität, die den Managementeinheiten übergeben werden kann.

Die Vorteile dieses Ansatzes sind: (i) Modularität, (ii) geringerer Komplexität, (iii) höhere Leistung, (iv) Zuverlässigkeit, (v) Wiederverwendbarkeit der delegierbaren Funktionen. Am wichtigsten sind natürlich die Einführung der Kooperation unter Agenten sowie die Fähigkeit, Funktionalität dynamisch zu delegieren. Die Schritte der Methodik werden in Abbildung 2 gezeigt.

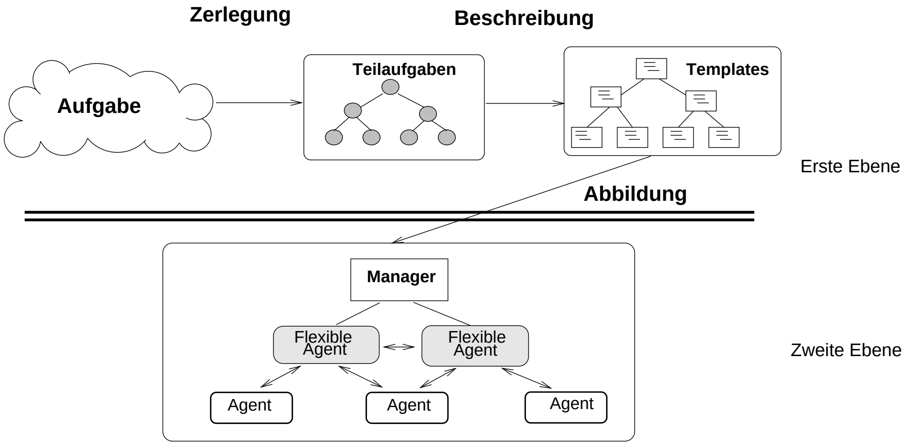


Abbildung 2: Methodik

4.1 Aufgabenzerlegung

Als erstes soll herauskristallisiert werden, ob die Managementaufgabe für eine verteilte Realisierung geeignet ist. Für die meisten Managementanwendungen ist das normalerweise der Fall.

Wenn eine verteilte Realisierung bevorzugt wird oder erforderlich ist, wird die Aufgabe in die entsprechenden Teilaufgaben zerlegt. Eine Analyse, was überhaupt eine Aufgabe (task) ist, und eine Beschreibungsmethodik können wir dem KADS entnehmen. Leider gibt es keinen Algorithmus zur Problemzerlegung. Die spezielle Art der Zerlegung basiert nur auf bestimmten Kriterien und ist natürlich anwendungs- und umgebungsspezifisch. Wir können aber folgende allgemeine Kriterien aufzählen:

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

4.2 Beschreibung des Multi-Agenten-Systems

Die aus der Zerlegung der globalen Aufgabe resultierenden Teilaufgaben müssen formal beschrieben werden. Dafür können wir das KADS Beschreibungsmodell benutzen. Daraus wählen wir die nötigen Informationen (Attribute): Eingabe- und Ausgabedaten, Quelle- und Zielagenten, die zu Kommunikationszwecken nötig sind. Die Funktionalität (die Teilaufgabe) wird durch das Attribut „Funktion“ spezifiziert. Um die Kooperationsmöglichkeiten zu beschreiben, definieren wir die Rollen des Agenten (master, slave, peer). Schließlich spezifizieren wir die Rahmenbedingungen, die die Ausführung der Teilaufgabe auslösen.

4.3 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.

Informationsmodell: Es gibt 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 über die Fähigkeiten anderer Agenten und das Kooperationsschema verfügen. Normalerweise ist es nicht notwendig, Information über die gesamte Menge von Agenten zu besitzen, sondern nur über eine relevante Teilmenge davon.

Kommunikationsmodell: Wegen der Kooperation unter den *flexiblen Agenten* brauchen wir 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)* ist ein Protokoll, das ein Message-Format und Message-Handling-Fähigkeiten anbietet, die auch für unseren Bedarf nötig sind.

Funktionsmodell: Es gibt auch neue Anforderungen bezüglich der Spezifizierung der Teilaufgaben und ihrer Delegierung auf die Agenten. Außerdem ist die Koordination der Teilaufgaben sehr wichtig.

Organisationsmodell: Die Manager-Agenten-Rollen sollen jetzt ganz dynamisch ausgetauscht werden. Bei der Bildung von Domänen sollen auch Gruppen von kooperierenden Agenten berücksichtigt werden.

Schließlich gibt es neue Anforderungen an die Architektur eines *flexiblen Agenten*. Wir untersuchen die Tauglichkeit existierender Ansätze wie ARCHON, dessen Bestandteile auf die Teile von Managementagenten abbgebildet werden können. Diese beinhalten das *High Level Communication Module (HLCM)*, das zu Kommunikationszwecken dient, das *Agent Information Management module (AIM)*, das Informationen über andere Agenten beinhaltet und das *Planning and Coordination Module (PCM)*, das für die kooperativen Interaktionen von Agenten zuständig ist.

5 Zusammenfassung und Ausblick

Wegen der Verteilung der Ressourcen, Dienste und Anwendungen ist der Übergang auf ein dezentralisiertes, flexibles Managementparadigma notwendig. Zu diesem Zweck benutzen wir Konzepte aus dem Bereich der intelligenten Agenten und entwerfen eine Methodik zur verteilten Abwicklung von komplexen Managementaufgaben.

Es gibt natürlich viele offene Fragen bezüglich der Realisierung des Konzepts; unter anderem: dynamische Verteilung der Funktionalität auf die Managementeinheiten und die damit verbundenen neuen Aspekte, Architektur eines *flexiblen Agenten* und Erweiterung der vier Funktionsbereiche von Managementarchitekturen.

6 Zusammenarbeit

Das in diesem Abschnitt beschriebene Teilprojekt wird in Zusammenarbeit mit dem von Prof. Hegering geleiteten Münchner Netzmanagement Team (MNM-Team) durchgeführt, das sich mit Fragestellungen im Bereich von Netz- und Systemmanagement beschäftigt. In diesem Rahmen besteht intensive Kooperation mit vielen anderen Mitgliedern der Gruppe, die zu Diskussionen und Veröffentlichungen bezüglich des Konzepts von kooperierenden Agenten im Netz- und Systemmanagement führt.

Im Rahmen des Graduiertenkollegs besteht Kooperation mit den Kollegiaten, die in anderen Bereichen von intelligenten, kooperierenden Agenten tätig sind. Dabei entstand eine Studie über das Wesen von Agenten, Agentensprachen und Protokollen. Unsere Absicht besteht darin, unsere Ergebnisse in einem gemeinsamen Papier zusammenzufassen.

Literatur

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

[CVJ92] D. Cockburn, L.Z. Varga und N.R. Jennings. Cooperating Intelligent Systems for Electricity Distribution. In *Proc. Expert Systems 1992 (Applications Track), Cambridge, UK*, 1992.

[DOS94] C. Duursma, O. Olsson und U. Sundin. KADS: Task Model Definition and Task Analysis Process. Deliverable M.5 2, ESPRIT Project P5248 KADS-II, August 1994.

[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.

[GY95] G. Goldszmidt und Y. Yemini. Distributed Management by Delegation. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, Juni 1995.

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

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

[JCL95] N.R. Jennings, J.M. Corera und I. Laresgotti. Developing Industrial Multi-Agent Systems. In *First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, CA.*, Seiten 423–430, Juni 1995.

[JML+92] N.R. Jennings, E.H. Mamdani, I. Laresgoiti, J. Perez und J. Corera. GRATE: A General Framework for Cooperative Problem Solving. *IEE-BCS Journal of Intelligent Systems Engineering*, 1(2), 1992.

[Wal91] S. Waldbusser. Remote Network Monitoring Management Information Base. RFC 1271, IAB, November 1991.

[WJ95] M. Woolridge und N.R. Jennings. Intelligent Agents: Theory and Practice. *Submitted to Knowledge Engineering Review*, Januar 1995.

[WJM94] T. Wittig, N.R. Jennings und E.H. Mamdani. ARCHON - A Framework for Intelligent Cooperation. *IEE-BCS Journal of Intelligent Systems Engineering - Special Issue on Real-time Intelligent Systems in ESPRIT*, 3(3):168–179, 1994.

Konfliktauflösung im Multi-Agenten Scheduling

Teilprojekt: Kooperierende Agenten und autonome Robotersysteme

Florian Fuchs

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

1 Einleitung

Wenn eine Aufgabenstellung durch kooperierende, (semi-)autonome Agenten gelöst werden soll, so müssen dabei Aspekte berücksichtigt werden, die bei einem vergleichbaren zentralen Ansatz keine oder nur eine untergeordnete Rolle spielen. Der dezentrale Ansatz bringt zum einen mit sich, daß den einzelnen Agenten i.a. kein vollständiges Wissen über die Umwelt und die anderen Agenten zur Verfügung steht. Die lokalen Wissensbasen der Agenten sind möglicherweise inkonsistent. Zum anderen besitzen die Agenten aufgrund der ihnen zugeteilten Teilaufgaben lokale Ziele. Die zu ihrer Erreichung lokal optimalen Lösungen sind aus globaler Sicht i.a. unvereinbar.

Eingeschränkte Sicht und Verfolgung lokaler Ziele, die i.a. konträre Ausprägungen besitzen, sind Eigenschaften, die typisch für eine dezentrale Problemlösung sind, bei traditionellen, zentral ausgerichteten Ansätzen dagegen keine Rolle spielen. Diese Eigenschaften führen dazu, daß sich zwischen Agenten Konflikte ergeben, die aufgelöst werden müssen. D.h. es müssen bei einem Multi-Agenten Ansatz Methoden zur Konfliktlösung integriert werden, um eine global konsistente Lösung zu erhalten.

Die Auflösung eines Konfliktes erfolgt durch die Anwendung einer Konfliktlösungsstrategie. Unterschiedliche Konflikttypen, Situationen in denen sie auftreten und verschiedene Szenarien erfordern i.a. auch unterschiedliche Strategien, um den Konflikt erfolgreich zu lösen. Ein Agent benötigt daher Konfliktlösungswissen, das es ihm ermöglicht, bei einem konkret gegebenen Konflikt geeignet zu reagieren.

Im Rahmen dieser Arbeit soll gezeigt werden, wie die Konfliktlösung in der Scheduling-Domäne algorithmisch erfaßt werden kann. Dazu ist es notwendig, zum einen Mechanismen zur Konflikterkennung und Strategieumsetzung zu entwickeln, und zum anderen eine geeignete Repräsentationsform für das Konfliktlösungswissen zu finden.

Zur Konfliktauflösung existieren zwei grundsätzlich verschiedene Ansätze: Zum einen ist es möglich, daß die Agenten zunächst unabhängig von einander lokale Pläne entwickeln. Sie prüfen diese Pläne anschließend auf Konflikte und verhandeln gegebenenfalls über Korrekturen der lokalen Pläne oder Zielrevidierungen, bis der Konflikt gelöst ist. Wir nennen diese Vorgehensweise *konfliktgesteuerten Planabgleich*.

Zum anderen können die Agenten auch gemeinsam einen Plan generieren. Sie gehen dabei hierarchisch vor, indem sie zunächst einen Grobplan erstellen und diesen dann schrittweise verfeinern. Für jeden einzelnen Schritt wird geprüft, ob durch ihn eine Konfliktsituation entsteht, und gegebenenfalls eine Alternative gesucht. D.h. Konflikte werden unmittelbar bei ihrer Entstehung eliminiert. Wir nennen diese Methode *gemeinsame Planentwicklung* — sie ist i.d.R. günstig für Domänen mit häufigen Konflikten.

Andere Arbeiten aus dem Bereich Konfliktauflösung in Multi-Agenten Systemen konzentrieren sich vor allem auf die Design-Domäne (siehe z.B. [LLC91], [KB91], [PG92]). Die Scheduling-Domäne im allgemeinen und Scheduling in der flexiblen Fertigung im speziellen weisen einige Besonderheiten auf, die darauf abgestimmte Konfliktlösungsmechanismen und -strategien erfordern. Im folgenden seien die wesentlichen herausgestellt:

- In der Scheduling-Domäne kann man nicht voraussetzen, daß Agenten uneingeschränkt kooperationswillig sind. Es können in solchen Fällen keine Ansätze verwendet werden in denen Agenten zur Auflösung eines Konfliktes eine gemeinsame Strategie, die entweder fest vorgegeben ist oder vor der eigentlichen Konfliktauflösung ausgehandelt wird, einsetzen. Im Scheduling existieren auch Szenarien, in denen die Strategien der jeweiligen Kooperationspartner unbekannt sind und die Kooperationswilligkeit der Konkurrenten unklar. D.h. ein

Agent wird i.d.R. seine eigenen Strategien nicht preisgeben, um zu verhindern, daß ein Konkurrent daraus Kapital schlagen kann. Genauso wird er nicht beliebig darauf vertrauen, daß sich das Verhalten eines Konkurrenten gegenüber dem bisher gezeigten nicht ändert. Es ist also ein Konfliktlösungsmechanismus notwendig, der es den Konfliktpartnern erlaubt, ihre Strategien vor dem Zugriff durch andere zu schützen.

- Scheduling als Spezialfall der Zuordnungsproblemklasse weist bei einer Zerlegung in Teilprobleme einen starken Verflechtungsgrad der Teilprobleme untereinander auf, es ist also i.d.R. mit einer großen Zahl von Konflikten zu rechnen. Deshalb ist von einer Konfliktauflösung durch eine zentrale Schlichtungsinstanz aus Effizienzgründen abzusehen.
- Die flexible Fertigung stellt eine hochdynamische Umgebung für ein Scheduling-System dar. Dynamik resultiert dabei vor allem aus unverhersehbaren Ereignissen wie Maschinen- oder Werkzeugfehlfunktionen und den dynamisch in das System eingebrachten Aufträgen, die eingeplant werden sollen. Die eingesetzten Konfliktlösungsmechanismen müssen auf diese Dynamik ausgerichtet sein.

Eine weitere Forderung an die Konfliktlösung ist die leichte Modifizierbarkeit von Konfliktlösungsstrategien, was eine explizite Repräsentation notwendig macht. Diese ist auch Voraussetzung für die Möglichkeit zur Adaption der Strategien an veränderte Umgebungsbedingungen zur Laufzeit, d.h. für die Einbringung eines gewissen Grades an Lernfähigkeit in den Agenten. Sie erlaubt weiterhin die Verwendung eines einheitlichen Kontrollmechanismus zur Konfliktlösung für unterschiedliche Szenarien.

2 Ein Konzept zur Konfliktauflösung

Im folgenden wird ein Ansatz zur Konfliktlösung für die Scheduling-Domäne vorgeschlagen, der die oben genannten Anforderungen erfüllt. Dabei wird zunächst kurz auf das zugrundegelegte Scheduling-Modell eingegangen, dann das verwendete Agentenmodell näher beschrieben und schließlich die Einbindung von Konfliktlösungsstrategien beschrieben.

2.1 Scheduling-Modell

Das zugrundegelegte Scheduling-Modell entspricht dem allgemeinen Job-Shop Scheduling, das im folgenden kurz charakterisiert wird. Für eingehendere Erläuterungen sei auf [Bla94] verwiesen.

Es existieren eine Menge von Tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$, eine Menge von Maschinen $\mathcal{M} = \{M_1, \ldots, M_m\}$ und eine Menge von Ressourcen $\mathcal{R} = \{R_1, \ldots, R_s\}$. Die Maschinen sind dediziert, d.h. spezialisiert auf eine Menge von Tasks. Die Zeitdauer, die zur Bearbeitung einer Task benötigt wird, ist von der eingesetzten Maschine abhängig. Maschinen und Ressourcen besitzen zugeordnete Kapazitäten.

Tasks aus der Menge $\mathcal{T}$ können eine Untermenge bilden — einen sogenannten Job. Ein Job J_j besteht aus den Tasks $T_{1j}, \ldots, T_{n_j j}$. Auf der Menge der einem Job zugeordneten Tasks ist eine Halbordnung definiert. Jeder Task ist eine Menge von Ressourcen zugeordnet. Jedem Job ist ein Einlastzeitpunkt und ein Termin zugeordnet.

Ein Zeitplan ist eine Zuweisung von Maschinen und Ressourcen an Tasks aus $\mathcal{T}$ über der Zeit[1], so daß folgendes gilt: 1. Maschinen- und Ressourcenkapazitäten werden eingehalten. 2. Alle Tasks eines Jobs werden nach dessen Einlastzeitpunkt bearbeitet. 3. Alle Tasks werden irgendwann bearbeitet. 4. Die Halbordnungen auf den Task-Mengen, die einen Job bilden, werden respektiert.

Das Modell sieht vor, daß Jobs dynamisch in das System eingelastet werden können (Online-Scheduling). Es ist deshalb keine feststehende Menge von Tasks zeitlich einzuplanen, sondern vielmehr erforderlich, daß zu beliebigen Zeitpunkten eintreffende Aufträge in den Planungsvorgang unmittelbar einbezogen werden.

[1] Im Online-Scheduling muß die Zuweisung zusätzlich auch in dem Sinne über der Zeit gefordert werden, daß die Zuweisung vor der zugewiesenen Startzeit, d.h. rechtzeitig erfolgen muß.

2.2 Agentenmodell

In diesem Abschnitt soll überblicksartig das eingesetzte Agentenmodell beschrieben werden. Die dabei herausgestellten Charakteristika eines Agenten gehen konform mit dem Verständnis des Agentenbegriffes, das im Rahmen eines Arbeitskreises des Graduiertenkollegs entwickelt wurde.

2.2.1 Rollenverständnis

Charakteristisch für Agenten ist eine rollenorientierte Modellierung. Die Rolle eines Agenten prägt seine Grundeinstellung, Ziele und Verhaltensweisen (siehe [Sun93]). Die Agenten sind in der Lage, eine bestimmte Rolle aus ihrem Rollen-Repertoire entsprechend den Erfordernissen der aktuellen Situation einzunehmen. In der vorliegenden Arbeit sind die beiden Rollen *Auftragsagent* und *Ressourcenagent* wesentlich, die weitgehend den beiden Rollen *manager* und *bidder* aus [Smi88] entsprechen.

Die Rollen weisen gewisse Freiheitsgrade auf, die der Agent nutzen kann, um unterschiedliche Strategien zu realisieren. Zum einen sind Variationen in den zur Ressourcen-Allokation verwendeten Verhandlungsprotokolle möglich und zum anderen sind gebotene und geforderte Preise (siehe 2.2.3) als Parameter der Protokolle frei wählbar.

2.2.2 Architektur

Abb. 1 zeigt die Grobstruktur eines konzeptionellen Agentenmodells, das für den vorgeschlagenen Ansatz zum verteilten Scheduling Verwendung finden soll.

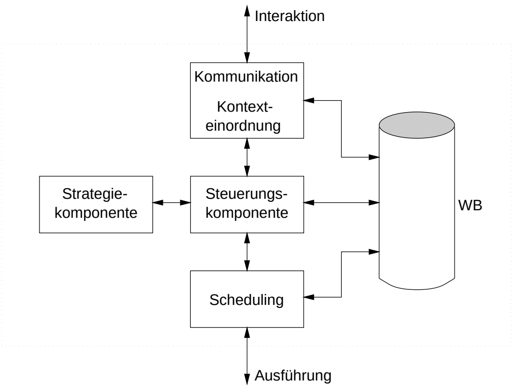


Abbildung 1: Agentenarchitektur

Der Agent kapselt in sich das Wissen, das er zur Erreichung seiner Ziele benötigt, und die für das Wissen notwendigen Verarbeitungsmechanismen. Das Wissen liegt explizit repräsentiert in einer lokalen Wissenbasis vor, die im wesentlichen Wissen über andere Agenten, Kommunikationsprotokolle, lokale Ziele und vorgegebene Planskelette enthält. Charakteristisch für den Verarbeitungsmechanismus eines Agenten ist das Durchlaufen von verschiedenen Schritten in einem Zyklus, wobei bei jedem Schritt auf die lokale Wissenbasis zugegriffen werden kann.

Die Steuerung des Zyklus erfolgt in einer zentralen Steuerungskomponente. Die Kommunikationsfähigkeit eines Agenten liefert die Voraussetzung, um in einem ersten Schritt des Zyklus Informationen über die Umwelt aufzunehmen. Aus einer empfangenen Nachricht werden Informationen extrahiert, die zur Aktualisierung der Wissenbasis genutzt werden. Weiterhin wird die Nachricht entsprechend ihres Typs und Inhalts in einen Kontext eingeordnet. Der Agent kann dann die für diesen Kontext vorgesehene Rolle einnehmen.

In den weiteren Schritten wird eine Reaktion auf die eingetroffene Nachricht erzeugt. Dabei wird sowohl auf die Wissenbasis als auch auf eine Scheduling- und eine Strategiekomponente zurückgegriffen. Die Scheduling-Komponente verwaltet die Zeitpläne, kann Vorschläge für Ressourcenbelegungen bewerten und ist in der Lage, Alternativzeitpläne zu finden. Die Strategiekomponente liefert den strategischen Beitrag zum Konfliktmanagement.

2.2.3 Koordination und Kooperation

Grundlage des Kooperationsmodells ist ein Zahlungssystem. Die Auftragsagenten erhalten für die Durchführung eines Auftrages eine gewisse Menge an Geldeinheiten (GE), über die sie frei verfügen können. Um einen Auftrag durchführen zu können, muß der Auftragsagent eine Menge von Ressourcen unter Berücksichtigung von zeitlichen Abhängigkeiten belegen. Für die Belegung durch die entsprechenden Ressourcenagenten hat er jeweils eine gewisse GE-Menge zu entrichten. Im Gegenzug entstehen den Ressourcenagenten Kosten für die Durchführung von Aktionen und auch die Instandhaltung der Ressourcen — d.h. insbesondere auch für stilliegende Ressourcen.

Für alle Leistungen, also Durchführung von Aufträgen und Verwendung von Ressourcen, existieren Standardpreise, die aber im konkreten Fall beliebig über- oder unterschritten werden können. Zum einen sind Preise Gegenstand von Verhandlungs- und Konfliktlösungsstrategien und zum anderen können Terminüberschreitungen o.ä. zu Preismodifikationen führen.

Sowohl Auftrags- als auch Ressourcenagent haben das Ziel, ihren jeweiligen Gewinn zu maximieren. Je nach eingesetzter lokaler Strategie gewinnt dieses Ziel mehr oder weniger Gewicht gegenüber einem möglicherweise existierenden übergeordneten Gesamtziel.

Die Auftragsagenten sind frei in der Verwendung der für einen bestimmten Auftrag erhaltenen GE-Menge. D.h. es ist nicht erforderlich, daß sie diese GE-Menge für die Ressourcenbelegung zur Erfüllung des entsprechenden Auftrages verwenden.

Die Belegung der Ressourcen geschieht auf dem Verhandlungsweg. Das dafür eingesetzte Verhandlungsprotokoll entspricht weitgehend dem *contract net* Protokoll [Smi88], bestehend aus den vier Phasen *Ausschreibung*, *Angebot*, *Auftragserteilung* und *Auftragsbestätigung*. Für ein Beispiel zum Einsatz dieses Protokolls in einem verteilten Planungsverfahren sei auch auf [Hah96] verwiesen.

2.3 Strategien zur Konfliktbewältigung

Im verteilten Scheduling sind Konflikte zu bewältigen, die aus inkonsistenten Belegungswünschen verschiedener Agenten für Ressourcen resultieren. Konkret bedeuted ein derartiger Ressourcenkonflikt die Anforderung ein und derselben Ressource durch wenigstens zwei Agenten zu überlappenden Zeitintervallen. Die Ursachen für diese Inkonsistenzen sind zum einen in den begrenzten lokalen Sichtweisen der Agenten und zum anderen auch in den unterschiedlichen lokalen Interessen zu sehen.

Um Ressourcenkonflikte aufzulösen, nutzen die Agenten ihre Fähigkeit zu kommunizieren. Kommunikationsprotokolle dienen dabei zur syntaktischen, semantischen und prozeduralen Festlegung der Verhandlung zwischen Agenten. Die für den Einsatz von Strategien interessanten Nachrichtentypen sind *Ausschreibung* und *Angebot*. Eine geeignete Belegung der bei diesen Nachrichtentypen vorhandenen *slots* für die Verhandlungsgegenstände Zeitintervall und Preis kann zur Steuerung der Konfliktauflösung und damit auch des Scheduling genutzt werden.

Zunächst muß man sich die Frage stellen, was eine Strategie leisten soll. Eine Strategie muß zum ersten die Entscheidung bringen, ob eine Ausschreibung bzw. ein Angebot akzeptiert werden soll. Dabei darf für eine leistungsfähige Strategie nicht nur ein Schwellwert festgelegt werden, den die Bewertung einer Ausschreibung bzw. eines Angebotes überschreiten muß. Neben einer Bewertung müssen auch noch andere Faktoren wie beispielsweise Nachfragedichten, Prioritäten und der bisherige Verhandlungsverlauf eingehen.

Falls ein Agent einen an ihn gerichteten Vorschlag ablehnt, muß er einen Gegenvorschlag erstellen. Diesen Gegenvorschlag erstellt er, indem er nach alternativen Zeitintervallen sucht und gleichzeitig den Preis so adaptiert, daß die Bewertung des Vorschlages für ihn günstig wird. Auch hier fließen wieder ähnliche Entscheidungskriterien wie zur Entscheidung über die Akzeptanz eines entgegengenommenen Vorschlages ein.

Eine Strategie stellt eine Abbildung von Daten, die dem Agenten über seinen Zustand und seine Umgebung bekannt sind, auf ein Paar aus Zeitintervall und Preis dar. Die vorliegenden Daten sind i.a. unvollständig und möglicherweise veraltet — ermöglichen also keine optimale Strategie und machen den Einsatz von heuristischem Wissen erforderlich.

Die wichtigsten Eingangsdaten für die Strategiekomponente des Agenten sind die Inhalte der eingegangenen Nachrichten (Ausschreibung bzw. Angebot). Daneben können aber auch noch andere Größen in die Strategie mit einfließen. Beispielsweise kann die Zahl der bereits bestrittenen Verhandlungsrunden auf die Kompromißbereitschaft Einfluß haben oder Information über den Verhandlungspartner einbezogen werden.

Strategien werden als Skripte repräsentiert, die durch die Strategiekomponente interpretiert werden. Die wesentlichen Elemente der verwendeten Skriptsprache sind Fallunterscheidung, Variablen und Aufrufe externer Funktionen. Die Variablen werden auch dazu benutzt, um Kontextinformation bereitzustellen. In diesen speziellen Systemvariablen werden z.B. Daten über die Nachricht, auf die reagiert werden muß, abgelegt. Desweiteren steht zur Nutzbarmachung der Funktionalitäten anderer Komponenten des Agenten eine Aufrufmöglichkeit für externe Funktionen zur Verfügung. Dadurch kann etwa die Scheduling-Komponente zur Bewertung eines Zeitplanes herangezogen werden, Daten aus der Wissenbasis abgefragt werden oder über die Kommunikationskomponente eine Nachricht versendet werden. Die Vorteile der Skriptrepräsentation sind in erster Linie die leichte Lesbarkeit durch den (menschlichen) Benutzer und die Ausdrucksstärke. Beides ist in Hinblick auf das Ziel, geeignete Strategien für ein bestimmtes Szenario zu finden, wichtig.

3 Zusammenfassung und Ausblick

In diesem Bericht wurde das Konzept eines Konfliktlösungsmechanismus für Multi-Agenten Systeme vorgestellt, der für die Scheduling-Domäne — speziell für den Einsatz zum Scheduling in der flexiblen Fertigung — geeignet ist. Die Konfliktlösungsstrategien sind explizit repräsentiert und privat, d.h. vor dem Zugriff durch andere Agenten geschützt. Damit ist es einem Agenten möglich, seine lokalen Interessen zu wahren.

Im weiteren soll eine prototypische Implementierung dieses Konzepts erstellt werden, die es erlaubt, in einer Simulationsumgebung die Wirksamkeit verschiedener Strategien in einem bestimmten Szenario zu testen. Zur Bewertung müssen entsprechende, objektivierbare Kriterien gefunden werden.

Literatur

[Bla94] J. Blazewicz. *Scheduling in Computer and Manufactoring Systems.* Springer-Verlag, 1994.

[Hah96] S. Hahndel. *Ein verteiltes verhandlungsgesteuertes Planungsverfahren für flexible Fertigungsumgebungen.* Dissertation, Technische Universität München, 1996.

[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*, Seiten 201–219. Springer, Berlin u.a., 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*, Seiten 183–200. Springer, Berlin u.a., 1991.

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

[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.

[Sun93] K. Sundermeyer. Modellierung von Agentensystemen. In J. Müller, Hrsg., *Verteilte Künstliche Intelligenz — Methoden und Anwendungen*, Seiten 22–44. BI Wissenschaftsverlag, 1993.

Agenteneinsatz in globalen Informationsräumen

Teil-Projekt: Kooperierende Agenten innerhalb Computergestützter Gruppenarbeit

Martina Nöhmeier

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

1 Motivation und Problembeschreibung

Die rasch fortschreitende Entwicklung des Internet und entsprechender Dienste haben zu vielfältigen Veränderungen und neuen Möglichkeiten im Bereich internationaler Kollaboration, Informations- bzw. Dokumentenaustausch und damit zum Aufbau eines immer dichter werdenden zusammenhängenden Informationsraumes geführt. Mit zunehmender Beteiligung an dieser Online-Gemeinschaft und wachsender Menge an bereitgestellten Dokumenten und Metainformationen wird deutlich, daß die Entwicklung von Werkzeugen, die eine sinnvolle Nutzung dieses erweiterten, globalen Informationsraumes erlauben, weit hinter dessen Fortschreiten hinterherhinkt. Nachdem die Weiterbenutzung von Konzepten, die für Wissensaustausch innerhalb klassischer Gruppenarbeit oder gar für einzelbenutzerorientiertes Arbeiten entwickelt wurden, für viele Aufgabenstellungen nicht mehr ausreichend ist, wurden Agenten als vielversprechende Alternative zur Lösung dieser Probleme vorgeschlagen [San94].

Ein dabei häufig verfolgter Ansatz war jedoch das isolierte Entwickeln von agentenbasierten Einzellösungen für ausgewählte Aufgabenstellungen. Da derzeit zudem noch kein akzeptiertes - auch in der Praxis - allgemein verwendbares Agentenmodell verfügbar ist, wurde jeweils relativ unabhängig von bestehenden Agentenkonzepten eine Lösung gesucht und geeignet erscheinende Teile der Realisierung als Agent deklariert. Ergebnis dieser Vorgehensweise sind verschiedene, meist prototypische Beispiel-Realisierungen für einzelne Problemstellungen, die sowohl in Agentenbegriffen, als auch in Konzeption, Mächtigkeit und Realisierung erheblich differieren. Da jedoch gerade innerhalb dieses immer dringlicher werdenden Gebiets des Informations- und Dokumentenaustausches in globalen Informationsräumen für sich allein stehende oder ausschließlich auf eine bestimmte Teilaufgabe abgestimmte Einzellösungen nicht effizient einsetzbar sind, sondern gerade die Interoperabilität von Werkzeugen eine große Rolle spielt, kann diese Vorgehensweise nicht zu befriedigenden Konzepten führen.

Es gibt jedoch kaum Arbeiten, die aufgabenübergreifend analysieren, welche Aufgabenstellungen beim Wissensaustausch in globalen Informationsräumen anfallen, welche Probleme zu lösen sind, an welchen Stellen und v.a. mit welchen Strategien Agenten verbesserte Lösungen erzielen könnten und wie ein Agentenkonzept (nicht eine isolierte Realisierung) für diesen Bereich beschaffen sein müßte, um eine konsistente Basis für die Lösung der verschiedenen anfallenden Aufgabenstellungen bieten zu können. Diese Lücke zu schließen ist das Ziel dieser Arbeit.

2 Vorgehensweise

Da die bisherigen Versuche agentenbasierter Unterstützung von Informationsaustausch in globalen Informationsräumen, die sich meist auf isolierte Einzelaufgaben spezialisierten[1] bei weitem keine befriedigende Lösung der Gesamtproblematik erwarten lassen, sondern Uneinheitlichkeit und zunehmend willkürliche Verwendung des Agentenbegriffes innerhalb dieses Teilbereiches vielmehr zu Verwirrung und teilweise sogar zum Infragestellen des Agentenkonzeptes an sich geführt haben, soll in diesem Teilprojekt nicht eine weitere agentenbasierte Einzellösung für eine exemplarisch gewählte Teilaufgabe entwickelt werden, sondern gerade die verschiedenartigen Aufgaben und Einsatzmöglichkeiten von Agenten innerhalb dieses Themenbereiches, sowie deren Abhängigkeiten voneinander ermittelt werden.

[1] z.B.Indexierung, Filterung von Electronic Mail, Informationssuche (Browsing/Anfrage)

Neben einer Analyse der Anforderungen, die Wissensaustausch in globalen Informationsräumen stellt, erfolgt eine Begutachtung derzeit existierender Teilkonzepte zur Unterstützung einzelner Aufgaben, um zu beurteilen, welche derzeit eingesetzten Verfahren sich gerade im Kontext von CSCW bewährt haben und auch für eine agentenbasierte Lösung von Bedeutung sein können und wo andererseits Probleme und Brüche innerhalb und zwischen einzelnen Aufgabenbereichen auftreten, die ein leistungsfähiges Agentenkonzept schließen sollte. Auf der Basis dieser Analysen können dann die aufgabenübergreifenden Anforderungen extrahiert werden, denen ein geeignetes Agentenkonzept in globalen Informationsräumen entsprechen muß und die Charakteristika einer solchen geeigneten Agentenmetapher beschrieben und konkretisiert werden.

3 Informationsaustausch in globalen Informationsräumen

3.1 Der Gruppen- und Informationsbegriff

Der Begriff *Information* bezeichnet i.a. zweckorientiertes Wissen. Für diese Arbeit relevant ist jedoch nur diejenige Informationsmenge, die bereits in Form eines elektronisch verfügbaren Dokumentes oder Datensatzes gespeichert ist bzw. sich darauf beziehende Metainformation.

Die innerhalb vernetzter Informationsräume enstandenen, zusätzlichen Formen des Informationsaustausches ermöglichen über den klassischen Begriff der Kollaboration in Gruppen hinaus die Herausbildung von Strukturen, die nicht mehr in allen Punkten dem *klassischen Gruppenbegriff*[2] entsprechen, aber dennoch ähnliche Interessen, Berufszweige, Projekte etc. aufweisen und aufgrund dieser Gemeinsamkeiten Interesse daran haben, Informationen in einer meist asynchronen (z.T. sogar anonymen) und häufig am Holprinzip orientierten Art und Weise auszutauschen.

3.2 Aufgaben

Für eine Analyse der notwendigen Eigenschaften und Kooperationsschnittstellen der Agenten verschiedener Funktionsbereiche ist ein wichtiger Schritt die Identifikation von Aufgaben, die im Zusammenhang mit dem Austausch von Information bzw. Metainformation in globalen Informationsräumen von Bedeutung sind sowie von Interdependenzen zwischen diesen Aufgaben. Beispiele für unterschiedliche, aber doch verflochtene Aufgabenbereiche sind:

* *Informationssuche* in verteilten, heterogenen Informationsquellen. Hierbei treten u.a. auch verschiedene Aufgaben aus dem Bereich der Informationsbeurteilung und -filterung auf.

* *Bereitstellung von Information*, die in Dokumenten- oder Datensatzform vorliegt (sowohl von Primär- als auch Sekundärinformation). Die Art, in der die bereitgestellte Information angeboten wird (Existenz bzw. Strukturierungsgrad von Metainformation, Zugriffsschnittstelle) ist in der Regel ausschlaggebend für die Prozesse, die nötig sind, um die Suche nach diesen Informationen zu unterstützen, wobei derzeit i.a. die Anbieterseite bei der Entwicklung von Unterstützungskonzepten nicht berücksichtigt wird.

* *lokale Verwaltung bzw. Pflege* von Dokumenten und Metainformation (Aufgaben aus dem Bereich der Datenpflege wie z.B. Versionsverwaltung, Aktualisierung), wobei hier sowohl Daten betroffen sein können, die nur für lokale Nutzung durch einen festgelegten Personenkreis bestimmt sind, aber auch Daten, die gleichzeitig für externen Zugriff zur Verfügung stehen.

* *Eigentlicher Informationstransfer*, angestoßen durch Informationsanbieter oder -nachfrager (Bring- oder Holprinzip), wobei die Austauschpartner bekannt, aber auch anonym sein können.

* *Notifikation*, die v.a. bei Informationsaustausch nach dem Holprinzip eine wichtige Rolle spielt.

* *Informationsbeobachtung*, sofern keine geeigneten Notifikationsmechanismen existieren.

All diese Aufgaben werden in der Regel getrennt voneinander analysiert, wobei die jeweiligen Anforderungen spezifisch für die Einzelaufgabe festgelegt und dementsprechend isolierte Konzepte

[2] Bearbeitung einer gemeinsamen Aufgabe in gemeinsamer Umgebung, Kenntnis der Teamzusammensetzung, Kommunikation untereinander, Koordination der Kommunikation und Aktionen innerhalb der Gruppe

für deren Unterstützung erarbeitet werden. Zwischen diesen Aufgaben bestehen jedoch deutliche Abhängigkeiten, deren Vernachlässigung zu Brüchen und Unverträglichkeiten sowohl in den unterstützten Abläufen, als auch in den dafür entwickelten Agentenkonzepten führen muß. Dies bedeutet, daß Aufgaben in einer Weise unterstützt werden, die für eine festgelegte Teilaufgabe geeignet ist, aber innerhalb der vor- und nachgelagerten Arbeitsabläufe zu Problemen führt, was häufig nicht nur ineffizientes Arbeiten, sondern auch mangelhafte Akzeptanz des Unterstützungskonzepts zur Folge hat. Agentenbasierte Unterstützungskonzepte für globale Informationsräume sind aber gerade auf eine breite Benutzerakzeptanz angewiesen, um ihre potentielle Leistungsfähigkeit entfalten zu können.

3.3 Probleme in globalen Informationsräumen

Bei der Untersuchung der o.g. Aufgabenbereiche stößt man auf zahlreiche problematische Aspekte, die beim Einsatz von Agentenkonzepten in globalen Informationsräumen berücksichtigt werden müssen. Im folgenden sollen einige Beispiele hierfür angesprochen werden:

Die, häufig durch organisatorische Maßnamen mitverursachte, dezentrale Speicherung von Wissen erschwert eine konsistente Informationsversorgung in kollaborativen Prozessen, in welchen dieses Wissen in einheitlicher Form benötigt wird. Zudem sind mit bisherigen Retrievalmethoden die über unterschiedliche Informationslager verteilten Informationen nur in atomisierter Weise zugänglich [GL95]. Es sind folglich Methoden nötig, um Interdependenzen zwischen angeforderden Informationen auszudrücken z.B. um den Kontext einer Anfrage durch vorausgegangene Anfragen zu spezifizieren [BPK+96].

Eine - infolge der Verfügbarkeit entsprechender technologischer Voraussetzungen und der damit wachsenden Ortsunabhängigkeit - zunehmende räumliche Entkoppelung, Ausdehnung und Anonymisierung von kollaborativen Vorgängen und Wissensaustausch schafft zusätzliche Probleme.

Ausreichende Netzwerkfähigkeit der Konzepte muß nicht nur für lokale sondern auch für Weitverkehrsnetze gewährleistet sein.

Die wachsende Anzahl an Teilnehmern am internationalen Informationsaustausch und die Unterschiede bezüglich des Kontexts, aus dem die jeweiligen Benutzer stammen, machen ausschließlich zentrale Lösungen, wie sie z.B. bei der Informationssuche [BPK+96] vielfach verfolgt werden schwierig und auf lange Sicht wenig erfolgversprechend, wenn man zusätzlich Probleme wie z.B. Generierung und ausreichendes Aktualisieren der entsprechenden Indexierungsinformation betrachtet.

Die Heterogenität der Information, z.B. bezüglich Format, Strukturiertheitsgrad, Erstellungs- bzw. Verwendungszweck, Metainformation etc. eröffnet ferner ein weites Feld an Komplikationen, v.a. wenn Informationen als Input für Folgeaktionen (z.B. Kombination von Suchergebnissen) [BPK+96]) oder weiterführende Dienste (z.B. NPS [BPK+96]) verwendet werden sollen. Zudem erschwert die Heterogenität der Information die Vergleich- bzw. Bewertbarkeit von Informationen, wie sie z.B. zum Filtern durch Agenten erforderlich wäre. Die Möglichkeit zur agentenbasierten Filterung von Information ist aufgrund der wachsenden, unmittelbar erreichbaren Informationsmenge in vielen Situationen wünschenswert, ist jedoch an das Vorhandensein geeigneter Beurteilungsgrundlagen (z.B. strukturierte Metainformation) gebunden.

Da mittlerweile ergänzend bzw. sogar alternativ zu klassischen Veröffentlichungsmöglichkeiten und Verbreitungskanälen zunehmend neue Wege der Veröffentlichung und Bereitstellung von Information genutzt werden (oft direkt am Ort der Entstehung z.B. basierend auf World Wide Web), haben immer mehr Personen die Möglichkeit, als Informationsanbieter aufzutreten, sind dabei jedoch mit Aufgabenstellungen (Pflege und Aktualisierung der angebotenen Daten, Bereitstellen von Metainformation, Suchschnittstellen etc.) konfrontiert, die einerseits nicht ihrem Fachgebiet entsprechen und zudem aufgrund der bisher fehlenden Unterstützung zeitaufwendig sind, andererseits aber einen entscheidenden Faktor für die Qualität der angebotenen Information und die langfristige Akzeptanz alternativer Veröffentlichungsmethoden darstellen. Verstärkt wird dieser Aspekt durch die ausgeprägte Dynamik des Informationsraumes, die teilweise zu einem raschen Veralten von Information und damit zu einem hohen Aktualisierungsaufwand führt und damit auch eine hohe Instabilität des Informationsangebotes verursacht. Gerade ein instabiles Informationsangebot und veraltete Informationsgegenstände (oft Sekundärinformation) erschweren nicht nur die Orientierung und das Erzielen von qualitativ zufriedenstellenden Suchergebnissen, sondern auch das Arbeiten mit Dokumenten, die über längere Zeiträume verfügbar sein müssen.

Die Verstärkte Anwendung des Holprinzips beim Austausch von Information kann schließlich bei mangelnden Notifikationsmechanismen zu Informationsmangel führen.

4 Anforderungen an ein Agentenkonzept

Aus den o.g. aufgabenspezifischen Problemaspekten, denen ein Agentenkonzept für den beschriebenen Anwendungsbereich Rechnung tragen muß, können Rahmenbedingungen für eine geeignete Agentenmetapher abgeleitet werden, anhand derer mögliche Agentenkonzepte auf ihre Einsetzbarkeit in globalen Informationsräumen geprüft werden können, wie z.B.:

* Ein geeignetes Agentenkonzept muß mit möglichst geringem Aufwand an die Dynamik der unterstützten Arbeitsprozesse und behandelten Dokumente bzw. Formate anpaßbar sein, ohne die Fähigkeit zur Behandlung älterer Informationstypen zu verlieren.

* Vor allem diejenigen Bestandteile von Agentensystemen, die die Schnittstelle zum Benutzer bilden, sollten nicht nur einfach und intuitiv handhabbar sein, sondern auch möglichst generisch aus der Problem- bzw. Aufgabenspezifikation erstellt werden können.

* Wichtig ist ferner die Berücksichtigung der Heterogenität der Koopertationspartner: Agenten sollten mit Agenten des gleichen Aufgabentyps bzw. des gleichen organisatorischen Kontexts, aber auch mit Agenten anderer Aufgabentypen bzw. abweichendem organisatorischen Kontext zusammenarbeiten können. Darüber hinaus müssen unmittelbar mit dem Benutzer in Kontakt stehende Agenten Kommunikationsschnittstellen aufweisen, die für den Anwender ein geeignetes Kommando-, Kontroll-, Meldungs- und Rückkopplungsformat anbieten und dieses in die jeweilige formale, von den Agenten verwendete Repräsentationsform umsetzen.

* Es müssen schließlich Mittel bereitgestellt werden, die es erlauben, die Beziehungen zwischen Agenten zu charakterisieren und hierarchisch strukturierte Agentengruppen zu modellieren.

5 Charakterisierung eines möglichen Agentenkonzeptes

Um eine aufgabennähere, übersichtlichere und intuitivere Vorgehensweise beim Entwurf von Agentensystemen zu gewährleisten, sollte eine rollenbasierte Agentenmetapher für die Beschreibung von Agentensystemen auf abstrakter Ebene verwendet werden. Die Rolle eines Agenten sollte dabei seine funktionelle und hierarchische Stellung in einer Agentenmenge widerspiegeln und kann zusätzlich z.B. durch Angabe von verschiedenen 'Scopes' bezüglich der behandelten Informationsgegenstände verfeinert werden [ABPS95].

Ein Agent soll eine Kapselung seiner Rollenspezifikation zusammen mit genau denjenigen Aktionsmustern, -zyklen, Wissensbestandteilen bzw. Wissenserwerbsmethoden repräsentieren, die er braucht, um seine spezifizierte Rolle auszuführen. Die Forderung der Kapselung ist nötig, um die Austauschbarkeit bzw. Erweiterbarkeit von Agenten zu gewährleisten. Zudem ist die Kapselung eine der Mindestanforderungen, die für eine in manchen Situationen notwendige oder zumindest vorteilhafte Mobilität von Agenten erfüllt sein muß.

Jeder Agent hat nur diejenige, für seine Auftragsausführung benötigte, Information lokal, die für die jeweiligen durchzuführenden Arbeitsgänge selbst spezifisch ist. Zusätzliches benötigtes Wissen (z.B. über die bearbeiteten Informationsgegenstände) kann über entsprechende Verfahren aus den Auftragsgegenständen ermittelt werden oder über entsprechende Kommunikationsschnittstellen und -protokolle von anderen Agenteneinheiten angefordert werden. Wissensbestandteile, bzw. die für deren Erwerb und Verarbeitung nötigen Verfahren sind jedoch in expliziter Weise im Agenten repräsentiert.

Umgebungsbedingungen und sonstige kontextspezifische Informationen, die sich entweder rasch ändern oder je nach Einsatzgebiet für ein und denselben Aufgabentyp unterschiedlich sein können, sollten nach Möglichkeit nicht fest im Agenten kodiert sein, sondern in separater und möglichst aufgabenunabhängig strukturierter Form gespeichert sein, um eine Parametrisierung oder zumindest eine problemlose Austauschbarkeit dieser Daten (z.B. durch den Anwender selbst) zu gewährleisten. Diese Forderung erleichtert zudem das Verwenden von Agenten in Form von rollenspezifischen Schablonen, die instanziiert und mit den aktuellen Kontextdaten an die jeweilige Situation angepaßt werden. Auf diese Weise kann eine bessere Mehrfachverwendung von den in Agenten gekapselten Aktionsmustern und Problemlösungsstrategien angestrebt werden.

Kommunikation zwischen Agenten sollte den Austausch von Kontextinformation erlauben (z.B. in strukturierter Form über Constraints [ABPS95]. Die Sprache selbst sollte jedoch nach Möglichkeit nicht vom jeweiligen Kontext abhängig sein.

Für die Ausführungsqualität vieler Agentenaufgaben (z.B. im Bereich Suche, Filterung oder Beobachtung von Dokumenten) ist ausschlaggebend, daß geeignete, möglichst strukturierte Metainformation über die zu bearbeitenden Gegenstände bereitgestellt werden kann. Diese sollte nicht, wie in bisherigen Ansätzen häufig realisiert, vom Ort des Metainformationsbedarfs aus automatisch generiert werden, sondern verstärkt von entsprechenden kooperierenden Agenteneinheiten am Ort der Informationsbereitstellung an den jeweiligen Informationsgegenstand gekoppelt in einem kommunizierbaren Format generiert und an dedizierten Brokerschnittstellen nur als (evtl. ergänztes) Duplikat gehalten werden. Die für jeden Bedarfsfall immer neue, automatische Generierung eines Metainformationsindex ist für große Informationsmengen in zentraler Form nicht effizient, führt aber bei dezentraler, voneinander unabhängiger Generierung oft zu unüberwindbaren, durch Heterogenität und unterschiedliche Aussagekraft bedingten Problemen bei einer kombinierten Verwendung von Metainformation verschiedener Indexe. Neben der oft nicht ausreichenden Aussagekraft dieser automatisch generierten Metainformation wird hierdurch zudem ein zu hoher Aufwand für die Aktualisierung der Metainformation verursacht, sowie abweichende bzw. sogar widersprüchliche Metainformation für z.B. ein und dasselbe Dokument generiert.

6 Kooperation mit anderen Projekten

Eine wichtige Problematik, die sich aus den bisherigen Anforderungen an Informationsagenten deutlich abzeichnet, ist die Fragestellung, welche Spezifikationsmethoden geeignet sind, um auf ausreichend abstrakter Ebene Agentenverhalten und -kooperation zu spezifizieren, so daß aus der Spezifikation möglichst generisch entsprechende Agenteninstanzen bzw. Verhaltenssimulationen erzeugt werden können. Der Themenbereich von Barbara König könnte wertvolle Hinweise liefern, inwiefern Graphgrammatiken hierbei vorteilhaft eingesetzt werden können.
Alternativ zu dem in der Literatur häufig zitierten Ansatz, die notwendigen i.a. mißverständlichen Attribute eines "Universalagenten" aufzulisten, soll in Zusammenarbeit mit den Projekten des 1. Teilbereiches weiter daran gearbeitet werden, mit Hilfe der für jeweils einen Anwendungsbereich aufgabenübergreifend ermittelten Anforderungen bzw. Potentiale von Agenten gemeinsame Rahmenbedingungen zu identifizieren, die eine Verwendung des Agentenbegriffes rechtfertigen und Aufschluß darüber geben, welche Vorteile der Einsatz des Agentenkonzeptes tatsächlich gegenüber anderen Softwareparadigmen bietet.
Durch die viermonatige Mitwirkung an der Weiterentwicklung eines Constraint-basierten Brokersystems für Information-Retrieval im Kontext eines Netzwerk-Publishing Systems (RXRC Grenoble, [BPK+96]) konnte zudem die Bedeutung vorgeschlagener Agenteneigenschaften in einem konkreten Anwendungskontext nachvollzogen werden.

Literatur

[ABPS95] J.-M. Andreoli, U. M. Borghoff, R. Pareschi und J. H. Schlichter. Constraint Agents for the Information Age. *J. Universal Computer Science*, 1(12):762–789, Dezember 1995.

[BPK+96] U. M. Borghoff, R. Pareschi, H. Karch, M. Nöhmeier und J. H. Schlichter. Constraint-Based Information Gathering for a Network Publication System. In *Proc. 1st Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, UK, April 1996.

[GL95] M. Grötschel und J. Lügger. Aufbau elektronischer Informations- und Kommunikationsstrukturen. Technical Report TR 95-10, Konrad-Zuse Zentrum für Informationstechnik Berlin, September 1995.

[San94] J.A. Sanchez. User Agents in the Interface to Digital Libraries. In *Proceedings of Digital Libraries '94*, Seiten 217–218, College Station, TX, Juni 1994.

Verteilte Generierung von Objekterkennungsprogrammen

Teil-Projekt: Bildverstehen in verteilten Systemen

Dietrich Büsching

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

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. Im Rahmen meiner Dissertation möchte ich einen an empirischen Untersuchungen der Eignung von Merkmalen oder Operatoren orientierten Ansatz verfolgen.

Die automatisch zu generierenden Algorithmen sollen sich an vorgegebenen Erkennungsstrategien orientieren. Solche Erkennungsstrategien sind nur für eine bestimmte Klasse von Objekterkennungsaufgaben unter bestimmten Voraussetzungen verwendbar. Die in Abschnitt 2 beschriebene Pixelklassifikation verwendet beispielsweise nur Operatoren, die im Bild sehr lokal arbeiten. Diese Strategie kann daher nur angewendet werden, wenn die zu erkennenden Objekte durch lokale Informationen (z.B. ihre Farbe) unterscheidbar sind. Solche Einschränkungen haben den Vorteil, daß der zu untersuchende Raum von möglichen Objekterkennungsprogrammen wesentlich verkleinert wird. Trotzdem ist der erforderliche Rechenaufwand für die Generierung eines Programmes enorm und macht eine Parallelisierung erforderlich.

In den Abschnitten 2 und 3 werden Generierungsverfahren für zwei einfache Erkennungsstrategien beschrieben und Ergebnisse dargestellt. Im Abschnitt 4 werden die Grundzüge einer komplexeren Erkennungsstrategie beschrieben, auf die sich die weiteren Arbeiten konzentrieren sollen.

2 Generierung von Pixelklassifikatoren

Bei der Pixelklassifikation wird jeder Bildpunkt anhand lokaler Informationen einer Klasse zugeordnet. Diese Informationen können der Grauwert am Bildpunkt oder bei Farbbildern die Intensitätswerte in den einzelnen Farbkanälen sein. Es ist aber auch möglich, durch sogenannte Texturoperatoren Informationen im näheren Umfeld eines Bildpunktes zusammenzufassen. Ein möglicher Texturoperator berechnet beispielsweise die Standardabweichung der Grauwerte in einem Rechteck um jeden Bildpunkt.

Eine Pixelklassifikation wird häufig bei der Auswertung von Luft- und Satellitenbildern verwendet. Hier können Bereiche von besonderem Interesse (Felder, Wälder etc.) einfach gefunden und weiter analysiert werden. Eine andere interessante Anwendung der Pixelklassifikation liegt in der Qualitätskontrolle in der Produktion. Materialfehler werden häufig durch farbliche Veränderungen angezeigt oder können durch Texturfilter erkannt werden.

Für die Erzeugung von Pixelklassifikatoren liegen bereits einige Arbeiten vor [Ame93]. Es wurde nicht versucht, in diesem Bereich wesentliche Neuerungen einzuführen. Das für die Generierung verwendete Verfahren wird hier dennoch beschrieben, da es Grundlage für das interessantere Verfahren der Pixel- und Regionenklassifikation ist.

Die Hauptprobleme bei der Konstruktion eines Pixelklassifikators sind die Auswahl für die Aufgabe relevanter Merkmale und die Bildung eines Klassifikators. Für die Auswahl von Merkmalen muß ein Kriterium gewählt werden, mit dem eine Menge von Merkmalen bewertet werden

Anzahl Workstations	1	2	4	8
Dauer Merkmalsauswahl (s)	1593	836	753	388

Tabelle 1: Rechenzeit für Merkmalsauswahl für unterschiedliche Anzahlen von Workstations in der virtuellen Maschine von PVM

kann [Kit86]. Als sehr aussagefähiges (aber rechenintensives) Kriterium wurde die Fehlerrate eines trainierten Klassifikators gewählt. Damit nicht alle möglichen Teilmengen von Merkmalen bewertet werden müssen, wird eine heuristisch eingeschränkte Suche im Raum aller Teilmengen durchgeführt. Die verwendete Heuristik ist die "sequential forward selection" [Kit86]. Die verteilte Implementierung dieser Suche wird im folgenden beschrieben. Für die Klassifikation wird ein mehrschichtiges Perzeptron verwendet, da dies lernfähig ist und sehr schnell klassifizieren kann.

Zur Strukturierung der Merkmalssuche wurde eine Master-Slave Architektur benutzt. Ein Koordinatoragent liest die vorliegenden Trainingsbilder und die vom Benutzer vorgegebenen Klassifizierungen ein. Er berechnet die aus den Bildern abgeleiteten Merkmale z.B. durch Verwendung von Texturfiltern. Die Zahl der Arbeiteragenten entspricht der Zahl der vorhandenen Merkmale. Zunächst sendet der Koordinatoragent jedem der Arbeiteragenten die Werte eines (jeweils unterschiedlichen) Merkmals und die vorgegebenen Klassifikationen der entsprechenden Pixel. Jeder Arbeiteragent berechnet nun durch Training eines mehrschichtigen Perzeptrons die mit dem vorliegenden Merkmal erreichbare Erkennungsqualität und sendet diese an den Koordinatoragenten zurück. Der Koordinatoragent wählt das beste Merkmal aus und sendet die Werte dieses Merkmals an alle Arbeiteragenten, die dieses Merkmal noch nicht erhalten haben. Nur diese Arbeiteragenten berechnen nun im nächsten Schritt die Erkennungsqualität anhand der insgesamt bei ihnen lokal vorliegenden Merkmale. Dieser Prozeß wird iterativ fortgesetzt. Zur Implementierung wurde das Werkzeug PVM 3.3 eingesetzt. Bei einer Anwendung mit 12 vorgebenen Merkmalen (siehe Abbildung 1) wurden die in Tabelle 1 aufgeführten Rechenzeiten gemessen (gemittelt über drei Programmläufe). Man erkennt, daß der Speedup nicht optimal ist. Ein Grund dafür sind aus dem Rechenprozeß ausscheidende Arbeiteragenten, deren Merkmal bereits ausgewählt wurde. Andere Gründe sind Kommunikationszeiten und die variable Trainingszeit für die verwendeten Perzeptronen. Da in jeder Iteration der Merkmalsauswahl gewartet werden muß, bis alle Merkmalsgüten berechnet sind, warten viele Arbeiteragenten untätig auf die Konvergenz der entsprechenden Perzeptrone, die Auswahl des besten Merkmals und die Zusendung der Merkmalswerte.

Ein Beispiel für die Anwendung der Pixelklassifikation bei der Unterscheidung texturierter Oberflächen wird in Abbildung 1 gegeben.

3 Pixel- und Regionenklassifikation

Soll anhand einer Pixelklassifkation ein Objekt in einem Bild gefunden werden, so ist es leicht möglich, daß Bereiche des Hintergrundes lokal die gleichen Eigenschaften wie das gesuchte Objekt haben und daher zu Fehlern führen können. Die Regionenklassifikation bietet Möglichkeiten zu einer weiteren Unterscheidung von Objekten. Dazu werden zusammenhängende Gebiete von Pixeln bestimmt, die bei der Pixelklassifikation der Zielklasse zugeordnet wurden. Merkmale der Form dieser Gebiete (Größe, Kompaktheit etc.) bieten nun Möglichkeiten, gezielter die gesuchten Objekte auszuwählen.

Die Generierung eines Regionenklassifikators wird ähnlich wie die Generierung eines Pixelklassifikators durchgeführt. Allerdings hängt die Regionenklassifikation stark von der vorhergehenden Pixelklassifikation ab. Je nachdem, ob eher falsche Negative (Objektpixel, die als Nichtobjektpixel klassifiziert werden) oder falsche Positive (Nichtobjektpixel, die als Objektpixel klassifiziert werden) bei der Pixelklassifikation in Kauf genommen werden, verändert sich die Aufgabe der Regionenklassifikation (vgl. Abb. 2). Um möglichst gute Gesamtergebnisse zu erzielen, wird aufbauend auf einer Reihe von unterschiedlichen Pixelklassifikatoren mit dem oben beschriebenen Merkmalsauswahlsverfahren jeweils der beste Regionenklassifikator konstruiert. Schließlich wird

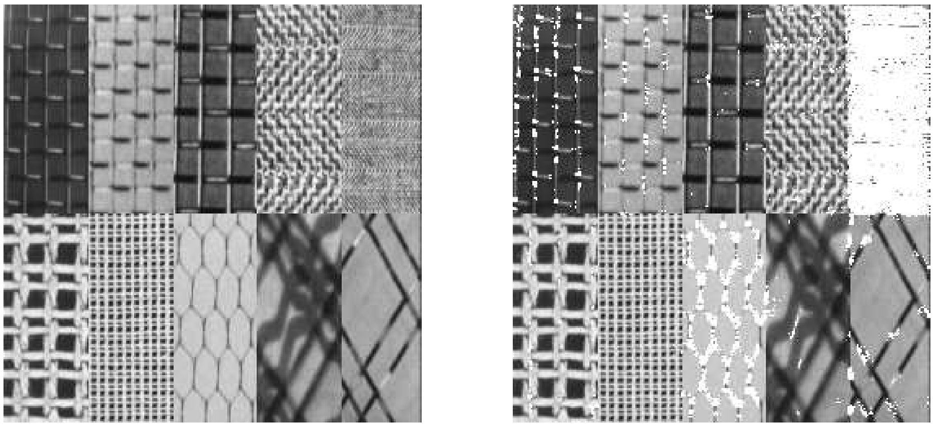


Abbildung 1: Zehn verschiedene Texturen; Originalbild und Ausgabe Pixelklassifikator (Objektpixel weiß) - der Bereich rechts oben war als Zielobjekt vorgegeben.

Anzahl Workstations	2	4	8
Rechenzeit (s)	1065	605	419

Tabelle 2: Rechenzeit für Konstruktion des Regionenklassifikators bei unterschiedliche Anzahlen von Workstations in der virtuellen Maschine von PVM

die Kombination von Pixel- und Regionenklassifikation mit der geringsten Fehlerrate ausgewählt.

Zur Erzeugung von Pixelklassifkatoren mit unterschiedlicher Gewichtung von falschen Positiven und falschen Negativen wurde folgendes Verfahren verwendet. Beim bei der Pixelklassifikation verwendeten Perzeptron sind zwei Ausgabeeinheiten vorhanden. Die eine zeigt das Vorhandensein von Objektpixeln, die andere das Vorhandensein von Nichtobjektpixeln an. Dadurch, daß ein Pixel nur dann als Objektpixel klassifiziert wird, wenn der entsprechende Ausgabewert um einen Betrag δ_{min} größer als der Wert der anderen Ausgabeeinheit ist, können durch Variation von δ_{min} Pixelklassifikatoren mit unterschiedlichen Anteilen von falschen Positiven und falschen Negativen erzeugt werden.

Da einerseits der Parameter δ_{min} variiert wird und andererseits für jeden dadurch erzeugten Pixelklassifikator mit Hilfe einer Merkmalsauswahl ein Regionenklassifikator erzeugt wird, ergibt sich ein zweistufig hierarchisches paralleles Programm. Ein Koordinatoragent erzeugt Instanzen von Arbeiteragenten 1. Stufe mit jeweils unterschiedlichem Parameter δ_{min}. Diese verwenden zur Auswahl von Regionenmerkmalen Arbeiteragenten 2. Stufe nach dem oben beschriebenen Verfahren zur verteilten Merkmalsauswahl. In Tabelle 2 sind Rechenzeiten für das weiter unten beschriebene Problem der Erkennung von Ästen bei 4 verschiedenen Werten von δ_{min} aufgeführt.

Die Pixel- und Regionenklassifikation wurde an zwei Beispielanwendungen getestet. Bei der einen sollen Äste in Brettern detektiert werden. In Abbildung 2 ist ein Originalbild und die Ausgaben von Pixelklassifikatoren mit unterschiedlichem δ_{min} zu sehen. Es ist deutlich zu erkennen, wie mit zunehmendem δ_{min} die Anzahl der falschen Positiven abnimmt. Für jeden dieser Pixelklassifikatoren wurden auf einer Menge von 26 Bildern die besten Regionenklassifikatoren bestimmt. Während für $\delta_{min} = 0.0$ eine Fehlerzahl von 938 falschen Positiven und 5 falschen Negativen auftrat, konnten bei $\delta_{min} = 0.4$ 29 falsche Positive und kein falscher Negative erreicht werden.

Das zweite untersuchte Anwendungsproblem ist Teil eines Projektes zur Analyse von endoskopischen Bildern, das zusammen mit dem Klinikum Rechts der Isar der TU München durchgeführt wird. Eine Teilaufgabe ist dabei die Erkennung des hinteren Endoskopteils bei rückwärtsgerichteter Blickrichtung. Das Endoskop hat eine metallische Farbe, die sich meist gut vom Gewebe abhebt. Mit Hilfe der beschriebenen Verfahren wurden Klassifikatoren für die Pixel- und Regio-

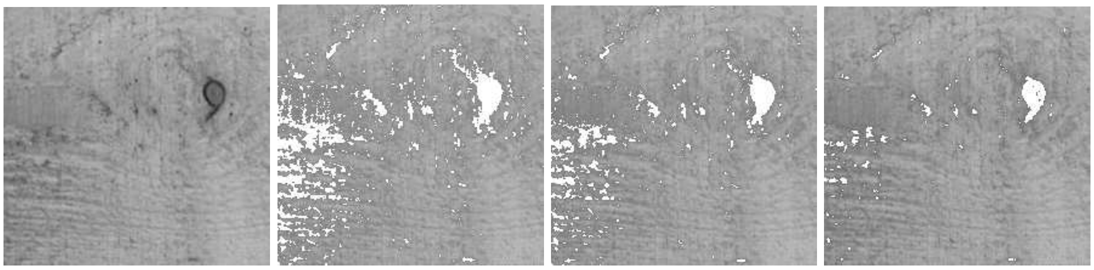


Abbildung 2: Originalbild und Pixelklassifikation mit Variationen von δ_{min} 0,0, 0,4, und 0,8 (von links nach rechts)

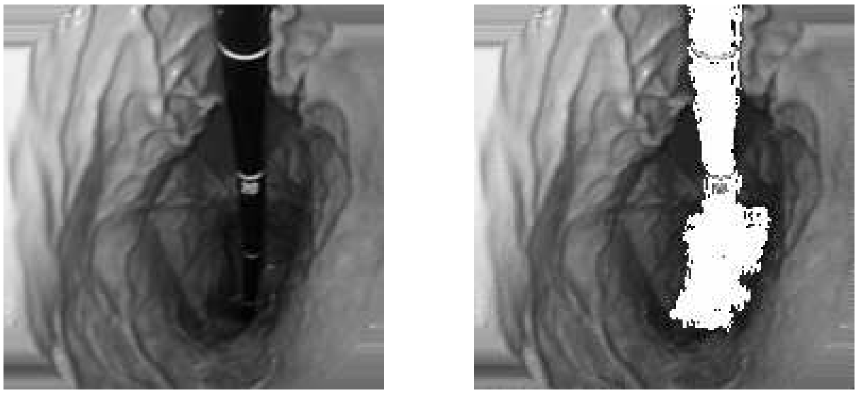


Abbildung 3: Detektion eines Endoskops mit dem generierten Verfahren

nenklassifikation erzeugt. Auf der Gesamtmenge aller vorhandenen Bilder (25 mit Endoskop und 156 ohne Endoskop) wurde eine Fehlerzahl von 0 falschen Negativen und 27 falschen Positiven erreicht. Ein zuvor manuell geschriebenens auf einer Kantendetektion basierendes Programm hatte 0 falsche Positive und 14 falsche Negative als Ergebnis. Da für die Aufgabenstellung falsche Positive nicht so kritisch wie falsche Negative waren, steigerte das automatisch generierte Verfahren die Leistungsfähigkeit des Gesamtsystems und wird in diesem weiter eingesetzt. In Abbildung 3 sind Bilder des Endoskops und der detektierten Regionen zu sehen.

4 Objektsuche mit Modellen

Durch Modelle wird der räumliche Aufbau von Objekten aus einfachen Primitiven wie z.B. Kanten beschrieben. Bei der Erkennung von Objekten in einer Fertigungsumgebung bietet es sich an, dafür CAD-Modelle zu verwenden [LMZ95]. Translationen und Rotationen des Objektes im Raum führen zu Veränderungen in der Objekterscheinung. Die Form von Objektflächen oder anderen Objektteilen, die durch Regionen ähnlicher Bildpixel gekennzeichnet sind, ändert sich dabei, so daß die im vorhergehenden Abschnitt beschriebene Erkennungsstrategie nicht verwendbar ist. Modelle bieten nun die Möglichkeit, das Aussehen eines Objektes unter verschiedenen Transformationen vorherzusagen. Allgemein wird zur Erkennung eine Hypothese über die Identität und die Lage eines Objektes in der beobachteten Szene benötigt. Die Plausibilität einer Hypothese hängt davon ab, wie gut das anhand des Objektmodells und der Lagehypothese vorausgesagte Aussehen des Objektes mit dem tatsächlichen Bild übereinstimmt.

Diese Sicht der Objekterkennung führt zu einer Aufteilung der Gesamtaufgabe in Hypothesengenerierung und -verifikation. Bei der Hypothesengenerierung wird häufig eine kleine Zahl von Modellmerkmalen korrespondierenden Bildmerkmalen zugeordnet und darauf aufbauend eine La-

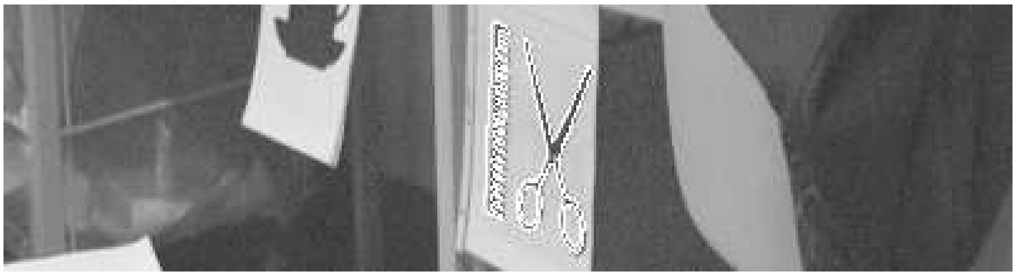


Abbildung 4: Detektion eines Piktogrammes - weiß: gefundener Objektumriss

gehypothese berechnet. Die Anzahl der möglichen Hypothesen nimmt für einfache Merkmale wie Punkte oder Polygonsegmente kubisch mit dem Produkt der Anzahlen von Modellmerkmalen und Bildmerkmalen zu. Die Generierung und Verifikation aller möglichen Hypothesen ist daher für komplexe Objekte und Szenen zu zeitaufwendig. In den geplanten und zum Teil durchgeführten Untersuchungen sollen Mechanismen zur Auswahl von Modellmerkmalen entwickelt werden. Die Anzahl der möglichen Hypothesen kann z.B. eingeschränkt werden, wenn Gruppierungen von Merkmalen (z.B. drei im Verlauf einer Kontur aufeinanderfolgende Ecken) stabil über eine große Zahl von Ansichten des Objektes extrahiert werden können. Falls solche Gruppierungen in allen Ansichten des Objektes stabil gefunden werden können, nimmt die Anzahl der zu untersuchenden Hypothesen stark ab. Durch Farbmerkmale kann ebenfalls für viele Objekte die Anzahl der sinnvollen Hypothesen stark eingeschränkt werden. Die Auswahl solcher besonders gut geeigneter Merkmale wird dadurch erleichtert, daß die Erkennungsaufgabe auf die Suche eines Objektes eingeschränkt ist. Um die Stabilität und Güte von Merkmalen verläßlich bestimmen zu können, wird ihr Vorhandensein in einer größeren Anzahl von künstlich erzeugten oder realen Bildern überprüft. Diese Untersuchungen sind größtenteils komplementär zu den im SFB 331 Teilprojekt L9 in der gleichen Forschungsgruppe bearbeiteten Fragestellungen, die sich in neueren Arbeiten [LMZ95] stärker auf Verfeinerung und Verifikation gefundener Hypothesen konzentrieren. Ein Anwendungsbeispiel ist in Abbildung 4 zu sehen. Dabei wurde ein in [Bü96] beschriebener Algorithmus zur Detektion von Gruppierungen von Punktmerkmalen verwendet. Durch die Merkmalsauswahl konnte die Rechenzeit für die Objektsuche im wesentlich größeren Originalbild von 35,8 s auf 6,1 s gesenkt werden. Bei den bisher durchgeführten Experimenten mit wenigen Merkmalstypen an relativ einfachen Objekten lag die für die Merkmalsauswahl benötigte Rechenzeit auf einer einzelnen Workstation in der Größenordnung von einigen Minuten. Für komplexere Objekte wird dies stark zunehmen. Wie bei den anderen Erkennungsstrategien soll die Programmgenerierung dann auf einem Pool von Workstations durchgeführt werden.

Literatur

[Ame93] J. Amelung. Ein Verfahren zur effizienten Merkmalsauswahl für die Texturfehleranalyse. In *Proccedings DAGM 1993 Lübeck*, Seiten 552–559, Berlin, 1993. Springer Verlag.

[Bü96] D. Büsching. Efficiently finding bitangents. Eingereicht zur 13th International Conference on Pattern Recognition, August 96, Wien,. 1996.

[Kit86] J. Kittler. Feature selection and extraction. In T.Y. Young und K.S. Fu, Hrsg., *Handbook of Pattern Recognition and Image Processing*, Seiten 59–83, Orlando, FL, 1986. Academic Press.

[LMZ95] S. Lanser, O. Munkelt und Ch. Zierl. Robust Video-based Object Recognition using CAD Models. In U. Rembold, R. Dillmann, L.O. Hertzberger und T. Kanade, Hrsg., *Intelligent Autonomous Systems IAS-4*, Seiten 529–536. IOS Press, 1995.

Konzepte zur agentenbasierten Parallelisierung in der Bildanalyse

Teil-Projekt: Bildverstehen in verteilten Systemen

Maximilian Lückenhaus
e-mail: lueckenh@informatik.tu-muenchen.de

1 Einleitung

Je komplexer die Anwendungen, umso spürbarer werden die durch einen herkömmlichen, sequentiellen Ansatz bedingten Restriktionen. Um diese Einschränkungen zu überwinden, ist eine Parallelisierung auf Anwendungs- und Systemebene unumgänglich. Bildanalyse gehört unzweifelhaft zu einer Gruppe komplexer Anwendungen, die aufgrund der hohen Ressourcenanforderungen in Bezug auf Rechenleistung und Speicherplatz, eine Parallelisierung nahelegt. Daher ist es ein Ziel der hier beschriebenen Arbeit, eine optimale Basis für die parallele Ausführung von Bildverarbeitungssequenzen zu schaffen. In den folgenden Abschnitten wird auf die Motivation und den speziellen Ansatz, auf die bisherigen Ergebnisse, sowie die Verknüpfungspunkte der Arbeit mit anderen Projekten eingegangen. Schließlich wird in einem Ausblick die weitere Vorgehensweise dargestellt.

2 Motivation und Aufgabenstellung

2.1 Agenten als Basis der Parallelisierung

Im Zusammenhang mit verteilten Systemen rücken in den letzten Jahren zunehmend agentenbasierte Lösungen in den Mittelpunkt des Interesses. Einmal korrespondiert das Agentenkonzept mit einer typischen Eigenschaft verteilter Systeme, nämlich der Strukturierung in eine Menge weitgehend autonomer, miteinander kommunizierender/kooperierender Komponenten. Darüberhinaus erfüllt es die besonders für Anwendersysteme typische Forderung nach Flexibilität. Die Möglichkeit der einfachen Hinzu- bzw. Wegnahme einzelner Agenten, sowie der Neudefinition ihrer Rolle, erlaubt es, ein System flexibel an die Anforderungen konkreter Problemstellungen anzupassen.

Der Begriff ,,agentenbasierte Parallelisierung" ist im Zusammenhang mit der hier betrachteten Arbeit in zweifacher Hinsicht zu sehen:

In einem ersten Schritt soll mit Hilfe eines Multiagentensystems aus einer gegebenen Sequenz von Bildverarbeitungsoperatoren ein äquivalenter *paralleler Ausführungsplan* aufgestellt werden. Hierfür ist die Fähigkeit zu komplexer Kommunikation (Absprache, Votieren,...) bei Agenten eine günstige Voraussetzung, um eine Koordination verschiedener Planungsinstanzen herbeizuführen. Im zweiten Schritt soll dann die *parallele Durchführung* der Bildverarbeitungsoperatoren durch Agenten erfolgen. Die Parallelisierung soll dabei zunächst auf Prozeßebene stattfinden. Erst für den Fall, daß diese Granularitätsstufe nicht ausreicht, soll in einem weiteren Schritt der Nutzen einer Parallelisierung auf Pixelebene untersucht werden.

Insgesamt läßt sich also eine Schichtung des Multiagentensystems bzgl. des Rollenverhaltens aufstellen (siehe Abbildung 1).

2.2 Bildanalyse als Anwendungsfeld der Parallelisierung

Multiagentenansätze wurden bereits erfolgreich in verschiedenen Gebieten der Informatik verwendet, sowohl zur Aufgabenplanung (z.B. eine verteilte Planung für Herstellungsprozesse in [Hahndel et al. 95]), als auch zur Ausführung von Operatorsequenzen (z.B. Multiagentensystem zur Objektverfolgung in [Tan et al. 95]). Daher erscheint es vielversprechend, entsprechende Multiagentenkonzepte auch in Hinblick eines parallelisierten Bildanalysesystems zu untersuchen.

Damit hierfür ,,das Rad nicht neu erfunden werden muß", baut die vorliegende Arbeit auf dem Bildanalysesystem *HORUS* auf. Die Entwicklung von *HORUS* begann 1988 am Lehrstuhl IX (Prof.

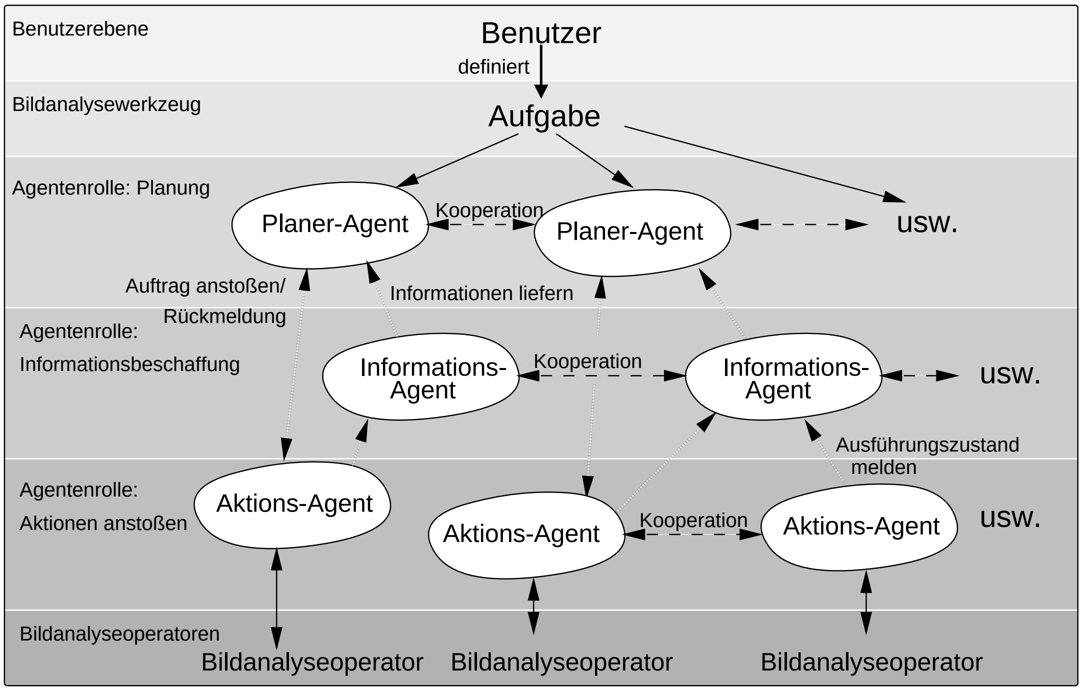


Abbildung 1: Rollenspezifische Schichtung des Multiagentensystems

B. Radig) und seit 1990 wird es in Forschung und Industrie erfolgreich eingesetzt. Obwohl bei dessen Design bereits eine spätere verteilte Ausführung berücksichtigt wurde und seither auch einige Vorarbeiten in Bezug auf eine Parallelisierung entstanden sind (siehe z.B. [Langer 90]), basiert die aktuelle Implementierung im wesentlichen auf einem sequentiellen Ansatz. Dieses Bildanalysesystem in eine parallelisierte Form zu transformieren, ist eines der primären Ziele. Nach Abschluß der Transformation, können anhand einer repräsentativen Anwendung die Vor- und Nachteile der verwendeten Konzepte bzgl. Verteilung/Parallelisierung evaluiert werden. So ließe sich z.B. untersuchen, inwieweit eine agentenbasierte Parallelisierung die interaktive Entwicklung von Bildanalyseanwendungen, wie sie in *HORUS* durch spezielle Werkzeuge unterstützt wird, verbessern kann.

3 Stand der Arbeit und Verknüpfungen zu anderen Projekten

3.1 Zerlegung der Aufgabenstellung

Betrachtet man die Aufgaben, die bei der Parallelisierung eines umfangreichen Anwendersystems anfallen, so können diese grob in drei Phasen eingeteilt werden (siehe Abbildung 2).

3.2 Analysephase

Um das während der Analyse gesammelte Material über Systemstruktur und Zusammenspiel der Komponenten in Bezug auf eine spätere Parallelisierung bewerten zu können, bedarf es eines adäquaten Modells. Also wurde als Ausgangsbasis ein modulares Modell entworfen, das es erlaubt, Systemkomponenten bzgl. ihres Autonomiegrades zu klassifizieren. Als Merkmal wird dabei die Form der Abhängigkeit verwendet, in der eine Systemkomponente zu anderen steht. Eine Abhängigkeit bzgl. eines Aufrufparameters bei einem Bildverarbeitungsoperator ist z.B. für die Parallelisierung weniger kritisch zu bewerten, als die Benutzung gemeinsamen Speichers (incl.

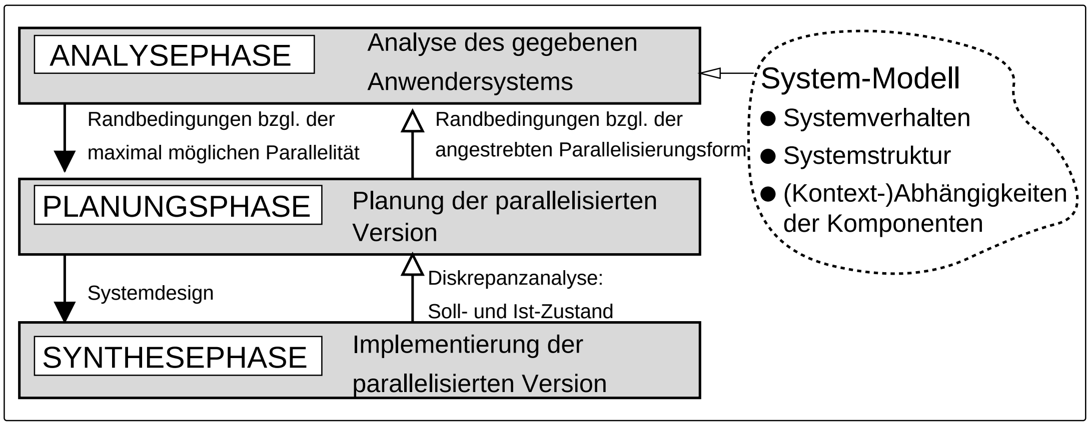


Abbildung 2: Phasen der Parallelisierung eines Anwendersystems

Schreibzugriffe) zusammen mit anderen Komponenten. Entsprechend wäre im ersten Fall der Autonomiegrad höher als im zweiten (je autonomer/unabhängiger eine Komponente, desto problemloser läßt sie sich parallelisieren). Langfristiges Ziel ist es, diese Klassifizierung mit Hilfe einer geeigneten Wissensbasis, die die notwendigen Komponentenbeschreibungen beinhaltet, zu automatisieren.

In diesem Zusammenhang lassen sich die Erfahrungen nutzen, die der Kollegiat Maximilian Frey bzgl. der (halb-)automatischen Erzeugung eines Kausalitätsnetzes aus einem parallelen Programmablauf gesammelt hat (siehe [Frey 95]).

Als Ergebnis der Analysephase ergeben sich also die Kontextabhängigkeiten der Systemkomponenenten. Diese bilden die Randbedingung bzgl. der maximal erreichbaren Granularität bei der Parallelisierung und liefern die Grundlage für die Planung des verteilt ablaufenden Anwendersystems, d.h., sie legen Synchronisationspunkte und Ablaufstrukturen fest.

3.3 Planungsphase

Beim Design des parallelisierten/verteilten Anwendersystems ergeben sich aufgrund der gewählten Konzepte zusätzliche Randbedingungen. In unserem Fall also durch die Wahl eines hierarchisch strukturierten Multiagentensystems (vgl. auch Abbildung 1). Das Modell der Analysephase sollte auch diese Randbedingungen möglichst gut miteinbeziehen, d.h., an dieser Stelle ergibt sich eine Rückkopplung zur Analysephase. Welche Auswirkungen dies im vorliegenden Fall hat, ist u.a. Gegenstand dieser Arbeit.

Neben strukturellen Entscheidungen müssen beim Entwurf auch das Verhalten und die Kooperation der Komponenten festgelegt werden. Um Systemkomponenten bzgl. ihrer dynamischen Abhängigkeiten beurteilen zu können, ist es auch notwendig, parallele/verteilte Abläufe zu modellieren. Es soll daher untersucht werden, wie geeignet Aktoren für eine derartige Modellierung sind. Zu diesem Zweck lassen sich Ergebnisse mit der Kollegiatin Barbara König austauschen, die im Rahmen des Graduiertenkollegs u.a. Aktormodelle als Spezifikationsbasis verteilter Systeme betrachtet.

3.4 Synthesephase

Der ausgereifte Systementwurf dient dann als Grundlage der Implementierung. Die Implementierungsentscheidungen sind naturgemäß zu diesem frühen Stadium der Arbeit noch nicht getroffen worden. U.a. soll untersucht werden, ob sich die am Lehrstuhl XIII (Prof. P. P. Spies) für die Top-down-Konstruktion verteilter Systeme entwickelte, imperative Programmiersprache InSEL (siehe z.B. [Spies et al. 93]) als Hilfsmittel für die Implementierung eignet.

4 Ausblick

Die nächsten Arbeiten werden sich u.a. darauf konzentrieren, das Modell zur Autonomiebestimmung von Systemkomponenten in eine ausgereifte Form zu bringen. In einem zweiten Schritt kann dann versucht werden, das bislang mittels Prozeß- und Schichtungsdiagrammen analysierte *HORUS*-Bildanalysesystem als eine Instanz des ausgereiften Modells darzustellen. Dadurch wäre eine Basis für das konkrete Design einer parallelisierten Version geschaffen, wodurch eine erste ,,Tauglichkeitsprüfung" des aufgestellten Modells ermöglicht würde.

Weitere Arbeiten betreffen Entwurfs- und Implementierungsentscheidungen, sowie strategische Überlegungen, welche Anwendung sich am besten eignet, den Nutzen einer agentenbasierten Parallelisierung eines Bildanalysesystems zu überprüfen.

Literatur

[Frey 95] Maximilian Frey, *Ein Konzept zur Fehlersuche in parallelen Programmen unter Verwendung von temporaler Logik*, in: Berichtskolloquium des ,,Graduiertenkolleg Kooperation und Ressourcenmanagement in verteilten Systemen", Technische Universität München, 1995

[Hahndel et al. 95] S. Hahndel und Paul Levi, *Modeling Manufacturing Systems*, S. 25-32 in: Tutorial ,,Task oriented Agent Robot Systems" at the 4. International Conference on Intelligent Autonomous Systems, Karlsruhe, 1995

[Langer 90] Wolfgang Langer, *Optimale Prozedurauswahl bei HORUS in einem verteilten System*, Diplomarbeit, Technische Universität München, Institut für Informatik, 1990

[Spies et al. 93] P.P. Spies, C. Eckert, D. Marek, H.-M. Windisch, *Konzeptionell struktuierte Verteilte Systeme*, S. 495-509 in: P. P. Spies (Hrsg.), EURO-ARCH '93 (Proceedings), Springer-Verlag, Reihe: Informatik aktuell, 1993

[Tan et al. 95] Chew Lim Tan, Chiun Min Pang, Worthy N. Martin, *Transputer implementation of a multiple agent model for object tracking*, S. 1197-1204 in: Pattern Recognition Letters, Elsevier Science B.V., North-Holland, Vol. 16, No. 11, Nov. 1995

Model checking und Bisimulation bei nebenläufigen Systemen mit unendlichen Zustandsräumen

Teil-Projekt: Verteilte Algorithmen – Spezifikation, Modellierung, Korrektheit

Richard Mayr

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

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] [SCM93].

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 $a \xrightarrow{\alpha} b$, 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 2.1 Sei A eine Menge von Konten, Act eine Menge atomarer Aktionen und (A, θ) ein LTS, wobei $\theta : A \longrightarrow pow(Act \times A)$ die Transitionen bestimmt. Es sei

$$\theta_\alpha(a) := \{x \mid (\alpha, x) \in \theta(a)\}$$

Man schreibt $a \xrightarrow{\alpha} a'$ für $a' \in \theta_\alpha(a)$. Eine binäre Relation $\sim$ auf A ist eine Bisimulation, wenn

$$\begin{array}{ll} \forall a, b \in A \text{ s.t.} a \sim b\ \forall \alpha \in Act. & (a \xrightarrow{\alpha} a' \ \Rightarrow \exists b \xrightarrow{\alpha} b'.\ a' \sim b') \ \wedge \\ & (b \xrightarrow{\alpha} b' \ \Rightarrow \exists a \xrightarrow{\alpha} a'.\ a' \sim b') \end{array}$$

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 $\stackrel{a}{\Rightarrow} := (\xrightarrow{\tau})^* \xrightarrow{a} (\xrightarrow{\tau})^*$ für $a \in Act$ und

$$\stackrel{\hat{a}}{\Rightarrow} := \begin{cases} \stackrel{a}{\Rightarrow}, & \text{für } a \neq \tau \\ (\xrightarrow{\tau})^*, & \text{für } a = \tau \end{cases}$$

Eine binäre Relation R über den Zuständen eines LTS ist eine *schwache Bisimulation*, wenn

$$\begin{array}{ll} \forall s_1, s_2 \in S \text{ s.t.} s_1 R s_2\ \forall a \in Act. & (s_1 \xrightarrow{a} s_1' \ \Rightarrow \exists s_2 \stackrel{\hat{a}}{\Rightarrow} s_2'.\ s_1' R s_2') \ \wedge \\ & (s_2 \xrightarrow{a} s_2' \ \Rightarrow \exists s_1 \stackrel{\hat{a}}{\Rightarrow} s_1'.\ s_1' R s_2') \end{array}$$

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. Die anderen Ergebnisse sind von Milner [Mil89], Jančar [Jan94], Taubner [Tau89] Hirshfeld, Jerrum und Moller [YHM94].

	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. 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 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.

3.1 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.

Voraussetzung: $\forall t \in T.\, |{\cdot}t| = 1$

Formeln (F,G,H, ...):$F ::= s \geq k \mid s \leq k \mid \neg G \mid G \wedge H \mid G \vee H \mid \Box G \mid \Diamond G$
wobei $s \geq k (s \leq k)$ bedeutet, daß auf der Stelle s mindestens (höchstens) k Marken liegen. Sei Σ eine Markierung eines communication free nets N. Das Problem $(N, \Sigma) \models F$ ist PSPACE-complete. 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.

3.2 μ-Kalkül

Im Gegensatz zu der im letzten Abschnitt beschriebenen branching time Logik ist Model checking mit dem modalen μ-Kalkül unentscheidbar für BPP [Esp95a]. 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.

4 Ausblick

Geplant sind weitere Forschungen in folgenden Bereichen:

- Weiterentwicklung von Tableauverfahren für den linearen und modalen μ-Kalkül.
- 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.

5 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

[Esp95a] Javier Esparza. Decidability of Model Checking for Infinite-State Concurrent Systems. *Acta Informatica*, 1995.

[Esp95b] Javier Esparza. Petri Nets, Commutative Context-Free Grammars and Basic Parallel Processes. In Horst Reichel, Hrsg., *Fundamentals of Computation Theory*, number 965 in LNCS. Springer Verlag, 1995.

[GH91] J. F. Groote und H. Hüttel. Undecidable equivelences for basic process algebra. Technical Report ECS-LFCS-91-169, University of Edinburgh, August 1991.

[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 STACS94, LNCS 775*. Springer Verlag, 1994.

[Jan95] P. Jančar. Undecidability of Bisimilarity for Petri Nets and related problems. *Theoretical Computer Science*, 1995.

[JB96] A. Mader J. Bradfield, J. Esparza. An effective tableau system for the linear time μ-calculus. In *Proceedings of ICALP96*. Springer Verlag, 1996.

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

[MN94] R. Mayr und T. Nipkow. Higher-Order Rewrite Systems and their Confluence. Technical Report TU-I9433, TU München, August 1994.

[SCM93] Y. Hirshfeld S. Christensen und F. Moller. Bisimulation Equivalence is Decidable for Basic Parallel Processes. In E. Best, Hrsg., *Proceedings of CONCUR 93*, number 715 in LNCS. Springer Verlag, 1993.

[Tau89] D. Taubner. Finite representation of CCS and TCSP programs by automata and Petri nets. In *LNCS 369*. Springer Verlag, 1989.

[YHM94] M. Jerrum Y. Hirshfeld und F. Moller. A polynomial algorithm for deciding bisimulation of normed context free processes. Technical report, LFCS report series 94-286, Edinburgh University, 1994.

Lastverwaltung auf gekoppelten Arbeitsplatzrechnern

Teil-Projekt: Programmentwicklung für Parallelrechner und vernetzte Architekturen

Christian Röder

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

1 Einleitung

Auf dem Gebiet der parallelen Datenverarbeitung traten in den letzten Jahren immer stärker Netze gekoppelter Arbeitsplatzrechner in den Vordergrund. Um ein solches System als Parallelrechner zu nutzen, wurden verschiedene Varianten paralleler Programmierumgebungen, wie PVM, P4 oder Linda entwickelt. Parallele Programmierumgebungen unterstützen dabei Strategien für eine globale Prozeßverwaltung, die Synchronisation der an der Anwendung beteiligten Prozesse und Kommunikation zwischen diesen. Parallele Programme zeichnen sich durch die Aufspaltung einer Aufgabe in mehrere, weitgehend unabhängige Teilaufgaben aus. Verbleibende Abhängigkeiten werden mit Hilfe der Kommunikation (z.B. Austausch von Zwischenergebnissen) oder Synchronisation (z.B. Reihenfolgesicherung der Teilaufgaben) aufgelöst. Die eigentliche Schwierigkeit bei der Formulierung eines parallelen Programms besteht in der Identifizierung geeigneter Teilaufgaben.

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** versteht man allgemein die Überführung einer ungleichmäßig verteilten Systemauslastung – gleichgültig ob sie bereits zum Programmstart existiert oder erst während des Programmablaufs entsteht – in einen ausgeglichenen Zustand. Auf dem Gebiet der Parallelrechner wird diesem Problem große Aufmerksamkeit gewidmet, da das vorrangige Ziel aufgrund der zu erreichenden Leistungsvorgaben eine Minimierung der Laufzeit einer Anwendung ist. Für eine effiziente Nutzung gekoppelter Arbeitsplatzrechner als Parallelrechner muß dieses Problem analysiert und es müssen Konzepte zu dessen Lösung angegeben werden.

1.1 Mechanismen der Lastverwaltung

Die Funktionsweise eines Lastverwaltungssystems entspricht den typischen Abläufen eines Regelkreises: ein Meßfühler erfaßt den aktuellen Zustand des Systems (die Führungsgröße ist die aktuelle Last im Rechensystem), eine Reglereinheit ermittelt die Abweichung vom Sollzustand (der optimalen Maschinenausnutzung) und versucht, über ein Stellglied (Komponente zur Lastverschiebung) diesen wieder herzustellen. Einen ersten Ansatz zur Klassifizierung von Lastverwaltungssystemen liefern Casavant und Kuhl [CK88], der durch Ludwig [Lud93] weiter verfeinert wird. Ein wichtiges Kriterium zur Unterscheidung ist die Art der Integration der Lastverwaltung. Viele in der Literatur beschriebenen Verfahren beschäftigen sich mit anwendungsintegrierten Mechanismen. Diese haben den Vorteil, daß der Programmierer die relevanten Eigenschaften der Anwendung kennt und die Lastverwaltung seinen Bedürfnissen optimal anpassen kann. Demgegenüber steht der Nachteil, daß für jede Anwendung die Lastverwaltung neu zu entwickeln und zu integrieren ist. Systemintegrierte Lastverwaltungsverfahren sorgen – für den Benutzer transparent – für eine automatische Umverteilung der Last. Entscheidungen darüber, wann und wo Last plaziert bzw. verschoben wird, werden von der augenblicklichen Belastung des Gesamtsystems beeinflußt.

Um die Funktionsweise eines Lastverwaltungsalgorithmus analysieren zu können, muß für das Rechensystem, ein System aus gekoppelten Arbeitsplatzrechnern, ein Modell entwickelt werden. Die Unterteilung in drei Teilmodelle stellt das notwendige Begriffsinstrumentarium zur Verfügung: (1) das *Rechnermodell* charakterisiert die relevanten Architektureigenschaften des Systems; (2) das

Benutzungsmodell beschreibt die Struktur und Benutzung der verwendeten Programmsysteme; und (3) das *Lastmodell* definiert den Begriff der Last für ein gegebenes Rechner- und Benutzungsmodell.

1.2 Gekoppelte Arbeitsplatzrechner für parallele Anwendungen

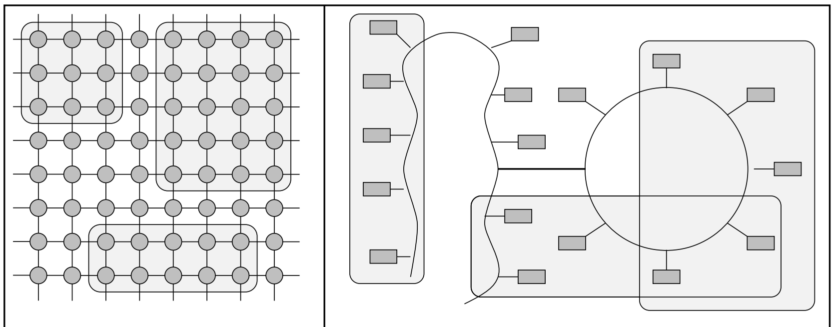


Abbildung 1: Unterscheidungsmerkmale: Parallelrechner versus gekoppelte Arbeitsplatzrechner

Will man – ganz allgemein – bereits existierende Problemlösungen aus dem Bereich der Parallelrechner auf Systeme gekoppelter Arbeitsplatzrechner übertragen, treten sowohl aus der Sicht der Hardware als auch bei der vorwiegend verwendeten Nutzungsweise beider Plattformen einige grundlegende Unterschiede auf. Lösungsansätze müssen deswegen, auch aufgrund ihrer Zielsetzungen, im allgemeinen neu überdacht werden. Folgende Punkte untergliedern die auffälligsten Unterscheidungsmerkmale beider Alternativen und sind in Abbildung 1 veranschaulicht:

Heterogenität versus Homogenität: Während für die in einer regelmäßigen Topologie angeordneten Rechenknoten in Parallelrechnern identische Prozessoren mit gleicher Betriebssoftware verwendet werden, bestehen die einzelnen Arbeitsplatzrechner (in beliebiger Anordnung) i.a. aus unterschiedlichen Prozessoren (mit verschiedenen Hauptspeicherkapazitäten) und unterschiedlichen Betriebssystemen. Zwei der daraus entstehenden Probleme seien beispielhaft herausgegriffen: (1) die Leistungsindizes der unterschiedlichen Prozessortypen können großen Einfluß auf die Laufzeit einer parallelen Anwendung haben; (2) die Konventionen zur Datenrepräsentation (z.B. beim Austausch von Zwischenergebnissen) müssen berücksichtigt werden. Die Ein-/Ausgabe erfolgt bei Parallelrechnern über dedizierte Ein-/Ausgabeknoten, womit das Verhalten von Ein-/Ausgabeoperationen von jedem Knoten aus gleich ist. Bei gekoppelten Arbeitsplatzrechnern ist die Systembelastung durch eine Ein-/Ausgabeoperation davon abhängig, ob der Prozeß auf einen lokalen oder nur einen entfernten Hintergrundspeicher Zugriff hat.

Lose Kopplung versus enge Kopplung: Von enger Kopplung bei Parallelrechnern spricht man, wenn die Kommunikation der Prozesse über gemeinsamen, von jedem Prozessor aus zugreifbaren Speicher erfolgt. Bei loser Kopplung erfolgt die Kommunikation explizit über den Austausch von Nachrichten. In einem verteilten System, wie es gekoppelte Arbeitsplatzrechner darstellen, erfolgt die Kommunikation über vergleichsweise langsame Übertragungsmedien, die zudem durch andere Benutzer mitverwendet werden. Falls allerdings der Rechenaufwand pro Kommunikationsoperation hoch ist (grobgranularer Parallelismus), kann die Leistungsfähigkeit gekoppelter Arbeitsplatzrechner mit der von Parallelrechnern verglichen werden.

Mehrprogramm/Mehrbenutzer-Betrieb versus Einprogramm/Einbenutzer-Betrieb: Neben der exklusiven Belegung des Parallelrechners durch einen Benutzer ist die vorwiegend verwendete Nutzungsmethode das *space sharing*. Zwar können dabei mehrere Benutzer gleichzeitig auf dem Rechner Anwendungen ablaufen lassen, allerdings wird einem Benutzer vor dem Programmstart eine feste Partition der ihm zur Verfügung stehenden Rechnerknoten zugeteilt, die sich während des gesamten Programmablaufs nicht mehr ändert. Der gleichzeitige Ablauf mehrerer Anwendungen eines einzelnen Benutzers auf einer Partition ist i.a. nicht möglich. Im Gegensatz dazu zeichnen sich Arbeitsplatzrechner gerade durch ihre Fähigkeit des Multitasking (Verarbeitungsprinzip *time sharing*) aus. Zum einen besteht zusätzlich zur Last, die durch eine parallele Anwendung entsteht, eine externe Last anderer Benutzer, die direkten Einfluß auf die Laufzeit der Anwendung hat. Andererseits kann sich während der Laufzeit die Hardwarekonfiguration ändern, wenn ein „fremder“ Benutzer einen in die Anwendung integrierten Rechner exklusiv für sich selbst beansprucht.

Der Einsatz von Lastverwaltung in homogenen Multiprozessorsystemen setzt voraus, daß mehr Prozesse einer parallelen Anwendung als Rechenknoten existieren. Diese Voraussetzung entfällt, da Lastverwaltung in einer heterogenen Umgebung mit Mehrprogramm/Mehrbenutzer-Betrieb auch dann sinnvoll ist, wenn beispielsweise zur initialen Plazierung von 4 Prozessen einer parallelen Anwendung 6 Rechenknoten mit unterschiedlicher Belastung zur Auswahl stehen.

2 System- und Lastmodelle

Ein wichtiger Punkt der Arbeit ist die Definition eines Lastmodells für eine heterogene Umgebung bestehend aus gekoppelten Arbeitsplatzrechnern. Zunächst werden zwei gängige Beispiele für Lastdefinitionen aus den theoretischen Gebieten der Lastverwaltung hervorgehoben. In 2.3 werden Kriterien zur Erstellung eines geeigneten Lastmodells angegeben.

2.1 Last in Wartesystemen

Nach Jessen/Valk [JV87] ist ein Wartesystem (queueing system) ein System, in dem aktuell nicht ausführbare Aufträge warten, ohne abgebrochen zu werden. Aufträge warten z.B. auf die Daten einer Ein/-Ausgabeoperation oder auf die Rechnerkernzuteilung durch den Scheduler in einem Wartekanal. Die Bedieneinheit ist die Instanz, die die Ausführung der Aufträge übernimmt. Der Begriff der Last eines Wartesystems wird dann wie folgt definiert.

Definition 2.1 *(Last eines Wartesystems)*
Die **Last** $u(t)$ *in einem Wartesystem ist die Summe der zum Zeitpunkt* t *von der Bedieneinheit noch abzuarbeitenden Bedienzeit.*

Zu jedem Zeitpunkt erhält man somit eine Last des Rechnersystems. Ziel der Lastverwaltung muß es nach obiger Definition sein, auf allen Rechnerknoten dieselbe Summe der Restbedienzeiten für die dort vorhandenen Aufträge zu erhalten. Der Übergang vom Einprogramm/Einbenutzer-Betrieb zum Mehrprogramm/Mehrbenutzer-Betrieb schafft für die Lastverwaltung neue Optimierungskriterien. Im Einprogrammbetrieb bringt die Lastverwaltung ersichtlich eine Minimierung der Verweilzeit, wohingegen im Mehrprogramm- und/oder Mehrbenutzerbetrieb sowohl eine Verbesserung von Durchsatz als auch von Verweilzeit erreicht werden soll, die allerdings a priori nur ungenau quantifiziert werden kann. Im letzteren Fall muß außerdem auf eine faire Gleichbehandlung aller Benutzer eingegangen werden. Ludwig [Lud93] zeigt, daß die Einschätzung dessen, was den definierten Begriff der Last am genauesten charakterisiert, äußerst schwierig ist. Das Problem besteht darin, daß der Begriff der Last zu einem gegebenen Zeitpunkt unter Bezugnahme auf das künftige Geschehen im Rechensystem definiert ist und somit keine analytische a priori Betrachtung der Lastverwaltung durchgeführt werden kann.

2.2 Last in Systemen mit verteiltem Speicher

Werden Rechnersysteme und Anwendungen mit Hilfe von Graphen beschrieben, so kann die Lastverwaltung als Problem der Grapheinbettung betrachtet werden. Heiss und Dormanns definieren in [HD93] einen Prozessorgraph durch PCG = (P, E^P), wobei P eine Menge identischer Prozessorelemente, E^P eine Menge bidirektionaler Prozessorverbindungen und $\gamma(p,q)$ die Kommunikationsverzögerung der Verbindungen ist. Die Kantengewichte $\gamma(p,q)$ können zur Definition einer Abstandsmetrik innerhalb des PCG zwischen beliebigen Paaren von Prozessorknoten herangezogen werden. Entsprechend wird ein paralleles Programm als Prozeßgraph modelliert: TIG = (T, E^T), wobei T eine Menge von Teilaufgaben und E^T eine Menge von Kommunikationsbeziehungen ist.

Mit $\alpha(i,j)$ wird der Kommunikationsaufwand (Anzahl der zu versendenden Dateneinheiten) beschrieben. Ziel ist es, eine Abbildung $\pi : T \to P$ zu finden, so daß die Kommunikationskosten minimiert werden.

Definition 2.2 *(Last eines Graphen)*
Die **Last** *eines Knotens im Prozessorgraph PCG ist definiert als die Anzahl der ihm zugewiesenen Teilaufgaben des Prozeßgraphen TIG: load(p) := $|T_p|$, wobei $T_p \subset T$.*

Für den Fall unterschiedlicher Prozessorgeschwindigkeiten $\mu(p)$, $p = 1, \ldots, n$, und verschiedenen Größen der Teilaufgaben (Anzahl der auszuführenden Befehle) $\beta(t)$, $t = 1, \ldots, m$, wird die Definition der Last eines Prozessorknotens erweitert zu:

$$load(p) := \frac{1}{\mu(p)} \cdot \sum_{t \in \pi^{-1}(p)} \beta(t)$$

Falls zur Laufzeit der Anwendung Teilaufgaben verschoben werden sollen, entspricht dies einer graphischen Neueinbettung des sich dynamisch ändernden Prozeßgraphen in den Prozessorgraphen. Auch hier besteht wie in 2.1 das Problem, daß die Größe der Teilaufgaben a priori nicht bekannt ist, da die Anzahl der auszuführenden Befehle unter anderem von der Eingabe abhängig sein kann. Dem Problem des Mehrprogrammbetriebs kann dadurch Rechnung getragen werden, daß die Funktion $\mu(p)$ abhängig von der Anzahl der externen Prozesse dynamisch veränderlich ist.

2.3 Lastmodelle und Lastindizes

Dem in 2.1 und 2.2 genannten Problem (Definition der Last im Bezug auf zukünftiges Geschehen) wird in der Praxis durch die Verwendung eines Lastmodells entgegengetreten. Gemäß Ludwig [Lud93] wird der Begriff des Lastmodells wie folgt definiert.

Definition 2.3 *(Lastmodell)*
Als **Lastmodell** *bezeichnet man die Menge der zur Lastbewertung herangezogenen Meßgrößen mit ihren relativen Gewichtungen zueinander und die Art ihrer Verknüpfung.*

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 [Zho88]. Die Information über die Last eines Betriebsmittels ist ein sich sehr schnell ändernder Bestandteil des Systemzustands und dementsprechend schnell veraltet. Eine periodische Aktualisierung des Wertes des Lastindex ist notwendig, um während des Vorgangs der Lastverwaltung auf sich ändernde Lastverhältnisse reagieren zu können. Die Länge der Aktualisierungsperiode ist entscheidend, da zu kurze Perioden den Aufwand der Lastverwaltung und damit den Einfluß auf des System erhöhen, und zu lange Perioden Informationen schneller altern lassen und die Reaktionszeit des Lastverwalters herabsetzen.

Nach [SHK95] sollte ein Lastindex folgende Eigenschaften besitzen: (1) Der Lastindex sollte nicht nur die CPU-Anforderungen eines Prozesses berücksichtigen, sondern auch seine Ein-/Ausgabe- und Speicheranforderungen. (2) Der Lastindex muß quantitativ die qualitative Abschätzung der Last eines Knoten widerspiegeln. (3) Der Lastindex sollte zur Vorhersage der zukünftigen Last verwendbar sein, da die Antwortzeit eines Auftrags eher durch die zukünftige als durch die aktuelle Last eines Knoten beeinflußt wird. (4) Der Lastindex soll (relativ) stabil sein. Große Schwankungen sollen ignoriert werden. (5) Der Lastindex sollte in direkter Beziehung zur Leistung des Systems stehen.

Während durch die Arbeit von Zhou [Zho88] für homogene Multiprozessorsysteme die Länge der Warteschlange rechenbereiter Prozesse als geeigneter Lastindex anerkannt wurde, zeigen Zhang et al. [Z+95] in einem neueren Ansatz für eingeschränkt heterogene Rechensysteme (Prozessoren mit unterschiedlicher Geschwindigkeit), daß durch die Charakterisierung der Last durch Kombination mehrerer Systemparameter wesentlich bessere Ergebnisse bei der Lastverwaltung erzielt werden.

3 Ausblick

3.1 Entwurf und Implementierung eines Lastverwaltungsmechanismus

Ziel der Arbeit ist es, nach dem nun abgeschlossenen Entwurf und der begonnenen Implementierung eines Mechanismus zur Lastverwaltung durch eine ausreichend große Anzahl von Experimenten ein angemessenes Lastmodell für heterogene Rechensysteme zu entwickeln. Die Lastverwaltung sollte mehr als eine Anwendung (time- und spacesharing!) berücksichtigen, da ansonsten zwei unabhängige Lastverwalter entgegengesetzte Entscheidungen treffen könnten. Sowohl initiale Prozeßplazierung als auch dynamische Lastverwaltung (Daten- und/oder Prozeßmigration) werden unterstützt. Eine adaptive Lastbewertung entsprechend dem Verhalten der Anwendungen (Ein-/Ausgabe-, Kommunikations-, Rechen- und Hauptspeicherintensität) wird angestrebt. Anwendungen, die selbst für die Lastverwaltung sorgen wollen, können vom systemintegrierten Lastverwalter den globalen Systemzustand erfragen, um somit bessere Entscheidungen treffen zu können.

3.2 Kooperationen inner- und außerhalb des Graduiertenkollegs

Durch die Eingliederung des Themas in den zweiten Teilbereich des Graduiertenkollegs („Programmiermodelle und Ressourcenmanagement für verteilte Systeme“) ergibt sich eine inhaltliche Nähe zum Teilprojekt „Konstruktion heteromorph paralleler Systeme“. Für Experimente eignen sich reale Anwendungen, z.B. aus dem Teilbereich „Verteilte numerische Algorithmen auf Bäumen“.

Innerhalb des SFB 342 - Teilprojekt A1 („Integration von Werkzeugumgebungen und Rechnerarchitektur zur Parallelisierung“) ist eine Werkzeugumgebung für die parallele Programmierumgebung PVM geplant [LWB+95]. In Kooperation mit diesem Projekt, kann der dort entwickelte Monitor als Hilfsmittel zur Lastdatenerfassung verwendet werden.

Literatur

[CK88] T. L. Casavant und J. G. Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. In *IEEE Transactions on Software Engineering*, Seiten 141–154, Feb 1988.

[HD93] Hans-Ulrich Heiss und Marcus Dormanns. Task Assignment by Self-Organizing Maps. Technical Report Interner Bericht 17/93, Fakultät für Informatik, Universität Karlsruhe, Mai 1993.

[JV87] Eike Jessen und Rüdiger Valk. *Rechensysteme (Grundlagen der Modellbildung)*. Studienreihe Informatik. Springer Verlag, Berlin, 1987.

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

[LWB+95] T. Ludwig, R. Wismüller, R. Borgeest, S. Lamberts, C. Röder, G. Stellner und A. Bode. *The Tool-Set* – an Integrated Tool Environment for PVM. In J. Dongarra, M. Gengler, B. Touracheau und X. Vigouroux, Hrsg., *Proceedings of EuroPVM'95 Short Papers*, Lyon, France, Sep 1995. Ecole Normale Supérieure de Lyon.

[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.

[Z+95] Yongbing Zhang et al. A Performance Comparision of Adaptive and Static Load Balancing in Heterogenous Distributed Systems. In *Proceedings of the 28th Simulation Symposium*, Seiten 332–340, Los Alamitos, CA, 1995. IEEE Comp. Soc. Press.

[Zho88] S. Zhou. A Trace-Driven Simulation Study of Dynamic Load Balancing. In *IEEE Transactions on Software Engineering*, Seiten 14(9):1327–1341, September 1988.

Verwendung temporallogischer Spezifikationen zur Lokalisierung von Fehlern und Leistungsengpässen in parallelen Programmen

Teil-Projekt: Anwendungsnahe, integrierte Entwicklungsumgebung für die effiziente Nutzung heterogener verteilter Systeme

Maximilian Frey

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

1 Einführung

Sowohl die Fehlersuche als auch die Lokalisation von Leistungsengpässen in parallelen und verteilten Programmen gestaltet sich schwieriger als bei sequentiellen Programmen. Die Gründe dafür sind der Nichtdeterminismus bei parallelen Programmen, die Abwesenheit von globalen Zuständen bei verteilten Programmen sowie eine größere Vielfalt an Parametern, die das Leistungsverhalten eines parallelen und verteilten Programms beeinflussen.

Zwei Gründe machen die Entwicklung paralleler und verteilter Programme nötig: Einerseits gibt es inhärent verteilte Probleme. Zum anderen gibt es Anforderungen an die Leistung des Programms, die nur durch Parallelisierung und Ablauf auf mehreren miteinander verbundenen Prozessoren oder Rechnern erreicht werden kann. Neben der Korrektheit steht nun speziell für die zweite Klasse von parallelen und verteilten Programmen die Erfüllung von Leistungsanforderungen im Vordergrund. Bei funktionalen Programmen im Sinne von [Pnu86] werden diese meistens als Speedup oder Effizienz mit einer unteren Schranke, bei reaktiven Programmen als nach unten beschränkte Zeitdauer zwischen der Anfrage an das System und der Antwort des Systems beschrieben.

Ausgangspunkt der Fehlersuche und der Suche nach Leistungsengpässen ist, daß beim Testen des Programms nach der Implementierung in einem Testfall Korrektheitsanforderungen bzw. Leistungsanforderungen nicht erfüllt sind: Falls eine Korrektheitsanforderung nicht erfüllt ist, stellt der Programmierer aufgrund der Symptome des fehlerhaften Ablaufs eine Hypothese über die Ursache des Fehlers auf und überprüft die Hypothese. Falls sie richtig ist, ist der Fehler lokalisiert. Sonst muß der Programmierer eine andere mögliche Hypothese entwerfen (siehe Abbildung 1 und [Mye79]).

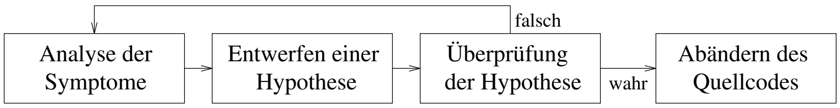


Abbildung 1: Prozeß der Lokalisierung von Fehlern

Falls eine Leistungsanforderungen nicht erfüllt ist, stellt der Programmierer oftmals mit Hilfe einer Visualisierung des Programmablaufs eine Hypothese über die Ursachen eines Leistungsengpasses auf. Ebenso sollte nun diese Hypothese überprüft werden. Erst wenn eine richtige Hypothese gefunden ist, kann der Programmierer den Leistungsengpaß eliminieren (siehe Abbildung 2).

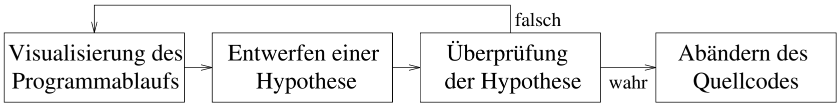


Abbildung 2: Ablauf der Lokalisierung von Leistungsengpässen

Wie aus den Abbildungen 1 und 2 ersichtlich, ist das Vorgehen bei der Lokalisierung von Leistungsengpässen sehr ähnlich zur Vorgehensweise bei der Lokalisierung von Fehlern (siehe auch

[WOK96]). In den folgenden Abschnitten wird ein Konzept vorgestellt, das zur Überprüfung von Hypothesen über Fehler als auch über Leistungsengpässe verwendet werden kann.

Dazu wird im nächsten Abschnitt die Verwendung von temporallogischen Spezifikationen zur Lokalisierung von Fehlern und Leistungsengpässen motiviert. In Abschnitt 3 wird ein Konzept vorgestellt, das eine automatische Überprüfung von Hypothesen beschrieben in temporallogischen Spezifikationen ermöglicht. Schließlich gibt Abschnitt 4 eine kurze Zusammenfassung und eine Beschreibung der Zusammenarbeit mit anderen Teilprojekten.

2 Motivation

Die Überprüfung einer Hypothese über die Ursachen eines Fehlers geschieht meistens unter Verwendung eines interaktiven Debuggers (cf. [WOK96]). Dazu wird festgelegt, welche Breakpoints während weiterer Programmabläufe gesetzt und welche Variablen an den Breakpoints inspiziert werden müssen, um die Richtigkeit der Hypothese zu zeigen. Dabei ist die Festlegung der Breakpoints und zu inspizierenden Variablen bei Hypothesen mit komplexen globalen Eigenschaften durchaus schwierig und fehleranfällig.

Bei der Überprüfung von Hypothesen über die Ursachen eines Leistungsengpasses werden normalerweise Visualisierungen des Programmablaufes verwendet. Es werden anwendungsspezifische Ereignisse definiert um das Auftreten bestimmten Variablenwerte und Ausführungen von Funktionsrümpfen in der Visualisierung darstellen zu können. Weiterhin ist es nötig, spezifische Auswertungs- und Visualisierungsfunktionen für diese anwendungsspezifischen Ereignisse zu definieren, um mit Visualisierungen weiterer Programmabläufe die Richtigkeit der Hypothese zu validieren. Eine solche Umsetzung einer komplexen Hypothese über Leistungsgrößen eines Programms ist ebenfalls schwierig und fehleranfällig.

Eine automatische Überprüfung komplexer globaler Hypothesen über Leistungsgrößen unterstützt den Programmierer bei der Lokalisierung von Fehlern und Leistungsengpässen. Um solche Eigenschaften des Programms, die die Hypothese widerlegen, automatisch überprüfen zu können, muß eine formale Spezifikationssprache verwendet werden, die hinreichend mächtig ist. Des weiteren sollte die Sprache es erlauben, einzelne Eigenschaften des Programms auf eine andere Weise als sie im Programm spezifiziert sind anzugeben. Auf Grund dieser Kriterien wird im folgenden eine Spezifikationssprache basierend auf temporaler Logik verwendet.

Zur Spezifikation von Leistungseigenschaften sind Leistungsgrößen nötig, die die Nutzung von Ressourcen durch das Programm beschreiben. In parallelen und verteilten Systemen sind i.allg. für die Analyse der Leistung die Betrachtung von Speicher-Ressourcen, Kommunikations-Ressourcen und Rechen-Ressourcen ausreichend. Dazu sind die benötigte Zeit, die Anzahl an Synchronisationen, die Anzahl an Kommunikationen, die Größe der bei Kommunikation übertragenen Daten, die Anzahl an Zugriffen auf den gemeinsamen Speicher, die Anzahl an Zugriffen auf den lokalen Speicher, und die Anzahl an I/O-Operationen interessante Leistungsgrößen. Es sollte unterschieden werden, ob ein Leistungsengpaß in der Applikation, im Betriebssystem oder in der Hardware lokalisiert ist. Da es einem Anwendungsprogrammierer meistens nicht möglich ist das Betriebssystem oder die Hardware für seine Applikation zu verändern, kann er die Einflüsse des Betriebssystems und der Hardware deshalb nur durch Veränderung des Algorithmus oder durch Abänderung der Abbildung von Programmobjekten auf Hardware-Objekten (Mapping) berücksichtigen. Aus diesem Gründen sollten die Spezifikation es ermöglichen, zwischen Leistungsgrößen der Applikation unabhängig von Hardware und Betriebssystem und Leistungsgrößen abhängig vom Mapping zu unterscheiden. Im folgenden werden nur solche Leistungsgrößen betrachtet, die unabhängig vom Mapping sind.

3 Konzept

Das im folgenden beschriebene Konzept (siehe auch [FW94, Fre95, Fre96] und Abbildung 3) automatisiert die Überprüfung einer Hypothese über die Ursache eines Fehlers oder Leistungsengpasses. Der Programmierer beschreibt dazu die Eigenschaften des Programms, die, wenn sie erfüllt sind, seine Hypothese widerlegen, in einer temporallogischen Spezifikationssprache. Die Überprüfung, ob diese Spezifikation in einem Programmablauf erfüllt ist, geschieht automatisch.

Dazu werden während des Ablaufs Ereignisse, wie beispielsweise Erzeugung von Tasks[1], Senden und Empfangen von Nachrichten, Zugriffe auf gemeinsame Variablen oder Betreten und Verlassen von Funktionsrümpfen zusammen mit Parameter, wie Task-Identifikatoren, Reihenfolgen von Variablenzugriffen oder Zeitdauern aufgezeichnet. Die während eines Programmablaufs aufgezeichneten Ereignisse liefern die Information zur Erzeugung eines partiell geordneten Modells (timed State Action Net oder tSAN), das nur die Abhängigkeiten zwischen Tasks des ursprünglichen Programmablaufs enthält, die nicht zufällig sind sondern Synchronisationspunkte darstellen. Dadurch repräsentiert das tSAN eine Äquivalenzklasse von Abläufen. In dieser Äquivalenzklasse kann sich ein Ablauf befinden, der fehlerhaft ist, obwohl der ursprünglichen Programmablauf richtig war. Bei der Überprüfung, ob die temporallogische Spezifikation in einem tSAN erfüllt ist, kann dennoch ein solcher Fehler gefunden und lokalisiert werden.

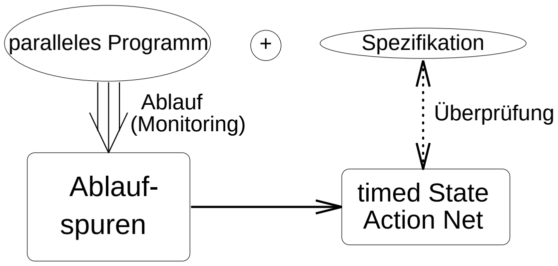


Abbildung 3: Fehlersuche mit temporaler Logik

In den folgenden Teilabschnitten werden tSANs, die temporallogische Spezifikationssprache und die Überprüfung überblicksartig dargestellt.

3.1 Timed State Action Nets

Ein timed State Action Net ist ein endliches Kausalnetz, dessen Knoten spezielle Semantik besitzen und als lokale Zustände bzw. Aktionen bezeichnet werden.

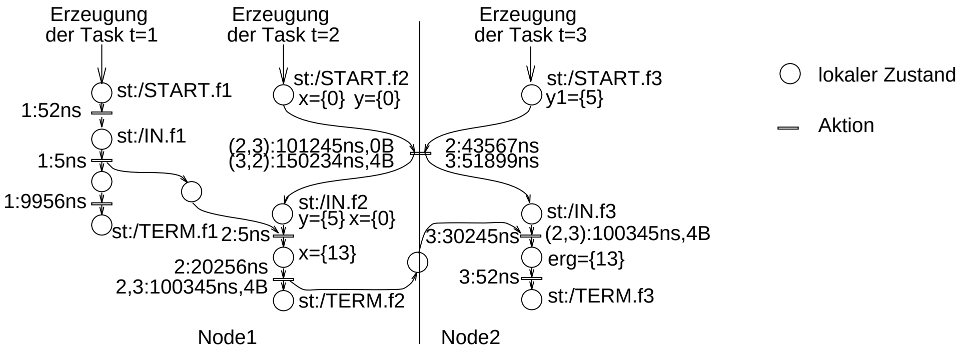


Abbildung 4: Beispiel eines tSAN

Eine Aktion a repräsentiert dabei Ausführungen von Anweisungen des Quellcodes, wobei deren Semantik fordert, daß sie gleichzeitig ausgeführt werden. a beschreibt weiterhin für jede Task die Zeitdauer, die die Task an der Ausführung der Aktion beteiligt ist und für jedes Paar von Tasks (t1,t2) die Kommunikationszeit und die Menge der übertragenen Daten aller Kommunikationen von t1 nach t2.

Ein lokaler Zustand s enthält lokale Informationen bezüglich einer Task, wie den Identifikator der Task, Funktionen die an s ausgeführt werden und Variablenwerte. Dabei kann eine gemeinsame

[1]Der Begriff Task wird synonym zu „thread" verwendet und beschreibt eine parallele Aktivität.

Variable mehr als einen Wert in s besitzen, wenn sie von einer anderen Task kausal unabhängig zu s beschrieben wird. In diesem Fall kann die Variable in s den alten und den neuen Wert annehmen.

3.2 Temporallogische Spezifikationssprache

Die temporallogische Spezifikationssprache gliedert sich in eine lokale Logik und eine temporale Logik:

Die lokale Logik beschreibt Eigenschaften lokaler Zustände und ist eine propositionale Logik. Sie enthält Konstrukte zur Spezifikation von Leistungsgrößen, wie die Ablaufzeit, die bisherige Anzahl an Synchronisationen, die bisherige Anzahl an Kommunikationen und die Größe der bisher übertragenen Daten in einem lokalen Zustand. Eine Formel der lokalen Logik kann erfüllt, unerfüllt oder undefiniert sein. Eine Formel ist beispielsweise in einem lokalen Zustand undefiniert, falls sie eine Variable enthält, die in der ,,aktuell" ausgeführten Funktion des lokalen Zustands nicht sichtbar ist.

Die temporale Logik ist eine Erweiterung der Logik in [Rei88] und beschreibt Eigenschaften von timed State Action Nets. Sie enthält Formeln der lokalen Logik als atomare Formel und temporallogische Operatoren. Die Semantik von temporallogischen Formeln ist definiert über Slices, die globale Zustände des tSAN repräsentieren.

not(t1:(in.f1) and t2:(in.f1)) if t1 != t2

T= sum_t(t1, runtime(last(t1,term.f1))), p=tasks(last(t1,term.f2),term.f1)
t0:(term.f1 and lb==0 → 2∗runtime(me)<3∗T/p)

Abbildung 5: Temporallogische Spezifikation

Ein Beispiel für eine temporallogische Spezifikation findet sich in Abbildung 5. Die erste Formel beschreibt, daß sich nicht zugleich eine Task t1 innerhalb einer Funktion f1 und eine Task t2 ebenfalls innerhalb von f1 aufhalten darf, wenn t1 und t2 unterschiedliche Tasks sind (t1!=t2). Dadurch wird gegenseitiger Ausschluß zwischen Tasks spezifiziert, die f1 ausführen. In der zweiten Formel ist T definiert als die Summe der Zeitdauern aller Tasks, in denen sie bis zum Ende der Funktion f1 rechnend sind. p ist die Anzahl an Tasks, die f1 beendet haben, bevor eine Task t1 eine Funktion f2 beendet hat. p gibt somit von Ende von f2 in t1 aus gesehen die Anzahl aller Tasks an, die bisher f1 ausgeführt haben. Die zweite Formel spezifiziert, daß in einer Task t0 am Ende der Funktion f1, falls die Variable lb den Wert 0 hat, die Zeit, in der t0 bisher rechnend war kleiner ist als das 1.5 fache der durchschnittlichen Zeit, die alle Tasks bis zum Ende von f1 rechnend waren.

3.3 Überprüfung

Das Ziel der Überprüfung ist zu zeigen, daß eine temporallogische Formel in einem tSAN nicht erfüllt ist und eine Teilstruktur des tSAN anzugeben, die eine Begründung des Fehlers angibt.

Dazu müssen die lokalen Zustände und Slices des tSAN berechnet werden, in denen die Formel nicht erfüllt ist. Dies geschieht in den folgenden Schritten:

1. Die Formel wird negiert, da die lokalen Zustände und Slices interessant sind, in denen die Formel und ihre Teilformeln nicht erfüllt sind.

2. Der Formelbaum der negierten Formel wird aufgebaut. Dieser Formelbaum stellt die zentrale Datenstruktur dar, die zur Überprüfung der Formel verwendet wird. Er ist in seiner Struktur sehr ähnlich zu einem Syntaxbaum und enthält in den Blattknoten Formeln der lokalen Logik, in den inneren Knoten temporallogische Operatoren.

3. Bevor die Überprüfung der Formel durchgeführt werden kann, werden die Leistungswerte der in der Spezifikation vorkommenden Leistungsgrößen für alle Zustände berechnet.

4. Der Formelbaum wird bottom-up ausgewertet:

(a) Es wird für alle lokalen Zustände des tSAN überprüft, ob eine Formel f eines Blattknoten b in einem lokalen Zustand s erfüllt ist. Wenn f in s erfüllt ist, werden alle Slices, die s enthalten in b eingefügt.

(b) Die inneren Knoten des Formelbaums werden ausgewertet, indem ihre Slices unter Verwendung der Slices der Nachfolgerknoten und einer Funktion berechnet werden, die nur von der temporallogischen Operation des Knotens abhängt.

(c) Falls sich nach der Auswertung des Formelbaums ein Slice in der Wurzel des Formelbaums befindet, ist die Spezifikation nicht erfüllt.

4 Zusammenfassung und Ausblick

Es wurde ein Konzept zur automatischen Lokalisierung von Fehlern und Leistungsengpässen in parallelen und verteilten Programmen unter Verwendung von temporallogischen Spezifikationen vorgestellt. Ein wichtiger Aspekt dieses Konzepts ist, daß komplexe globale Eigenschaften mit Leistungsgrößen spezifiziert werden können.

Das Konzept ist noch nicht vollständig implementiert. Diese Implementierung soll prototypisch fertiggestellt werden. Weiterhin soll das Konzept erweitert werden, so daß Hypothesen spezifizierbar und überprüfbar sind, die die in Abschnitt 2 angegebenen und von der Hardware und dem Betriebssystem abhängigen Leistungsgrößen enthalten. Bei reaktiven Systemen stellt sich die Frage nach der Repräsentation von unendlichen Abläufe.

Das sich diese Arbeit mit Verifikation und Spezifikation von verteilten Systemen beschäftigt, ergibt sich eine inhaltliche Nähe zum Teilprojekt ,,Verteilte Algorithmen – Spezifikation, Modellierung, Korrektheit". Außerdem ergibt sich bei der Frage, wie Leistungsgrößen abhängig vom Betriebssystem und der Hardware aufgezeichnet werden sollen, eine Zusammenarbeit mit den Teilprojekten ,,Programmentwicklung für Parallelrechner und vernetzte Architekturen" und ,,Konstruktion heteromorph paralleler Systeme". Zur Validierung des Konzepts und der Spezifikationssprache wird eine enge Zusammenarbeit mit den Teilprojekten angestrebt, die sich mit Anwendungen beschäftigen.

Literatur

[Fre95] M. Frey. Performance Debugging of Parallel Programs with Temporal Logic Specifications. In *Proceedings of the 21th EUROMICRO Conference, Como, Italy*, Seiten 90–98. IEEE, September 1995.

[Fre96] M. Frey. Debugging Parallel Programs using Temporal Logic Specifications. In *Proceedings of the First IFIP Workshop on Software Engineering for Parallel and Distributed Systems, Berlin, Germany*, March 1996. to appear.

[FW94] M. Frey und A. Weininger. A Temporal Logic Language for Debugging Parallel Programs. In *Proceedings of the 20th EUROMICRO Conference, Liverpool, England*, Seiten 170–178. IEEE, September 1994.

[Mye79] G.J. Myers. *The Art of Software Testing.* Wiley, New York, 1979.

[Pnu86] A. Pnueli. Specification and Development of Reactive Systems. In H.-J. Kugler, Hrsg., *Information Processing 86, Dublin, Ireland*, Seiten 845–858. IFIP, Elsevier, Amsterdam, September 1986.

[Rei88] W. Reisig. Temporal Logic and Causality in Concurrent Systems. In *Concurrency 88*, LNCS 335, Seiten 121–139. Springer, Berlin, 1988.

[WOK96] R. Wismüller, M. Oberhuber und J. Krammer. Interactive Debugging and Performance Analysis of Massively Parallel Applications. *Parallel Computing.*, 1996. to appear.

Entwicklung eines Ressourcenmanagements für Verteilt-Parallel-Kooperative-Systeme mit einem Graphersetzungssystem

Teil-Projekt: Konstruktion heteromorph paralleler Systeme

Sascha Groh

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

1 Einleitung

In den letzten Jahren sind große Fortschritte auf dem Gebiet der parallelen Programmierung für verteilte Hardwareplattformen erzielt worden. Für die Klasse der Anwendungen, die in voneinander unabhängige Teilberechnungen zerlegt werden können, ist eine effiziente Realisierung und somit eine hohe Auslastung der Ressourcen der verteilten Hardwareplattform möglich (z.B. [Pol95]). Aber neben dieser Klasse gibt es ein weites Feld von kooperativen Anwendungen, die mit den zur Verfügung stehenden Programmierhilfsmitteln, wie z.B. [Inm84], nicht einfach beschreibbar sind oder nicht effizient realisiert werden können. Diese Anwendungen zeichnen sich dadurch aus, daß eine Parallelisierung möglich ist, die einzelnen parallelen Teile aber nicht unabhängig voneinander ihre Berechnungen vorantreiben können, sondern miteinander kooperieren müssen. Dies führt zu dem Problem, daß nicht a priori eine statische Verteilung, auch anwendungsintegrierte Lastverteilung genannt, der parallelen Teile erfolgen kann. Die gängige Alternative, die Anwendung in uniforme Repräsentationen zu zerlegen und mit einer systemintegrierten Lastbalanzierung eine effiziente Realisierung zu erhalten, nutzt die Hardwareressourcen zwar aus, achtet dabei aber weniger auf eine effiziente Lösung der gestellten Aufgaben (Anwendungen). Das Ziel muß es daher sein, eine anwendungsangepaßte Realisierung vorzunehmen, die flexibel und für den Anwender transparent ist, und die für jede Teilaufgabe die beste Realisierung mit der gegebenen Hardware findet, dabei aber sowenig Verwaltungsaufwand wie möglich verursacht (siehe auch [Spi95]).

2 Ressourcenmanagement in VPK-Systemen

Um die in Kapitel 1 dargestellten Ziele zu erreichen, ist es notwendig, das Wissen über die Anwendungen dem Ressourcenmanagement zur Verfügung zu stellen. Desweiteren muß das Management in der Lage sein, alle Realisierungsalternativen in einem realen System überblicken zu können, diese unter Zuhilfenahme der gelieferten Anwendungsinformationen auszuwerten und eine Realisierung vorzunehmen. Entsteht während der Laufzeit des Systems eine mögliche, effizientere Realisierung für eine Anwendung, so hat das Management dies zu erkennen, die Kosten für den Wechsel (Transformation) eines Systemteils zu dieser neuen Realisierung mit den daraus entstehenden Vorteilen abzuwägen, und gegebenenfalls diese Transformation vorzunehmen.

Dies muß alles zur Laufzeit des Systems erfolgen. Wobei der Aufwand für das Management möglichst gering gehalten werden muß, da die Ressourcen, die für das Management verwendet werden, nicht mehr den Berechnungen, die das System erledigen soll, zur Verfügung stehen.

2.1 Beschreibung des Systems mit Graphersetzungen

Um das Ressourcemanagement auf eine fundierte Grundlage zu stellen, ist es notwendig, eine Möglichkeit zu finden, die Abhängigkeiten in dem System, die zur Verfügung stehenden Realisierungsmöglichkeiten und die notwendigen Entscheidungsinformationen zu beschreiben. Hierzu wird ein Graphersetzungssystem, wie in [SW92] spezifiziert, verwendet.

2.1.1 Das Graphenersetzungssystem

Die Knoten des Graphen stellen Ressourcen auf unterschiedlichen Abstraktionsebenen dar. Jeder Graph hat zwei ausgezeichnete Knoten, auf der einen Seite den Knoten „abstraktes System“ und auf der anderen Seite den Knoten „reales System“. Diese beiden Knoten stehen für die höchste und die tiefste Realisierungsebene. Zwischen diesen beiden Knoten bilden sich, mithilfe der Graphersetzungsregeln, über die Laufzeit des Systems die einzelnen Abstraktionsebenen.

Die gerichteten Kanten in diesem Graph haben unterschiedliche Aufgaben. Die eine Aufgabe besteht in der Verklammerung der einzelnen Abstraktionsebenen. Eine Kante beschreibt auf diese Weise die vertikale-Ressourcen-Beziehung, also die Beziehung zwischen einem Objekt a_e der Abstraktionsebene e, welches das Objekt a_{e-1} auf der Abstraktionsebene $e-1$ als Ressource benötigt. Neben diesen Kanten gibt es in dem Graphen auch Kanten, die die Abhängigkeitsverhältnisse innerhalb einer Abstraktionsebene beschreiben. Eine Kante bedeutet in diesem Sinne, daß für den Fortschritt der Berechnung des Objektes a_e, das Objekt b_e benötigt wird. Diese Abhängigkeit wird auch als horizontale Ressourcenbeziehung bezeichnet.

Sowohl die Knoten, als auch die Kanten in diesem Graphen sind attributiert. Auf die einzelnen Aufgaben der Attribute soll hier nicht weiter eingegangen werden. Im Groben läßt sich sagen, daß einige Attribute, vor allem Knotenattribute für die vollständige Beschreibung des Systems notwendig sind, wie z.B. aktuelle Wertbelegungen von Registern etc. Andere Attribute, vornehmlich Kantenattribute machen Aussagen über die Gewichtung von Abhängigkeitsverhältnissen. Dieser Punkt wird in Kapitel 2.5 nochmals aufgegriffen.

Die Graphersetzungsregeln überführen einen Graphen G_1 in einen Graphen G_2 durch die Ersetzung eines Untergraphen U_1 in G_1 durch einen Untergraphen U_2. Der Untergraph U_1 spiegelt dabei eine in dem realen System existierende Realisierung eines Objektes O, des „höchsten Knotens“ des Untergraphen U_1 wider. Der Untergraph U_2 beschreibt eine (alternativ mögliche) Realisierung für das abstrakte Objekt O. Durch die Anwendung einer Graphersetzungsregel wird eine Aktion des Ressourcenmanagement beschrieben. Durch die bei jeder Graphersetzungsregel angegebene Einbettungsregel wird beschrieben, zu welchen Knoten des neuen Untergraphen U_2 die äußeren Kanten des zu ersetzenden Untergraphen U_1 laufen.

2.1.2 Verhalten des Systems – Vorgänge in dem Graphen

Jeder attributierte Graph beschreibt das System vollständig mit all seinen Hardwareeigenschaften und seinen Anwendungen zu einem Betrachtungszeitpunkt t. Jeder Fortschritt der Berechnung des Systems spiegelt sich in Veränderungen des Graphen wider. Auf diese Weise ist es möglich, alle Vorgänge in dem zu verwaltenden System anhand des Graphen zu beschreiben. Jeder Vorgang, der nicht von dem Ressourcenmanagement vorgenommen wird, spiegelt sich in Veränderungen der Attribute des Graphen wider. Veränderungen, die von dem Ressourcenmanagement durchgeführt werden, sind durch Graphersetzungsregeln beschrieben und überführen einen Graphen G_1 in einen Graphen G_2, bei dem von der Verwaltung Realisierungsaufgaben wahrgenommen worden sind.

Anhand von dieser Beschreibungsmethode lassen sich die Aufgaben des Ressourcenmanagement aus der Einleitung zum Kapitel 2 wie folgt beschreiben. Die Entscheidungsinformationen werden durch den Graph mit seinen Attributen beschrieben. Die Möglichkeit eine Ersetzungsregel anwenden zu können stellt die Möglichkeit dar, eine Realisierungsalternative zur Verfügung zu haben. Die Auswahl der Regel entspricht der Entscheidung für eine Alternative. Die Anwendung der Regel entspricht der Durchführung der ausgewählten Managementaufgabe.

2.2 Grundlagen für die Entscheidungen des Ressourcenmanagements

Als Grundlage für die Entscheidungen des Ressourcenmanagement dienen die Ressourceneigenschaften. Hierbei kann zwischen vier verschiedenen Ressourceneigenschaften differenziert werden.

Die ebeneninvarianten Eigenschaften, also Eigenschaften, die unabhängig von der betrachteten Abstraktionsebene gelten müssen, beschreiben, welche alternativen Repräsentationen für ein Objekt in einem konkreten System zur Auswahl stehen. In dem Graphenersetzungssystem werden diese Eigenschaften durch die Graphersetzungsregeln beschrieben. Es können nur Ressourcen als

Repräsentation verwendet werden, für die es eine Ersetzungsregel gibt, die das zu realisierende oder transformierende Objekt als linke Seite enthält und die Ressourcen als rechte Seite.

Die zweite wesentliche Eigenschaft ist die Bindung einer Ressource. Ist die Ressource ungebunden, so steht sie zur Realisierung von Objekten zur Verfügung. Ist sie hingegen gebunden, so kommt sie nicht mehr als Realisierungsalternative in Betracht. In diesem Fall ist zusätzlich wichtig, an welches Objekt eine Ressource gebunden ist. In dem Graph wird diese Eigenschaft durch Kanten und Teilgraphen repräsentiert. Alle Ressourcen, die sich in dem Teilgraph unter der Wurzel „freie Ressourcen" befinden, haben die Eigenschaft frei, alle anderen Ressourcen sind gebundene Ressourcen. An welches Objekt eine Ressource gebunden ist, wird durch vertikale Kanten dargestellt.

Die dritte Klasse von Ressourceneigenschaften sind ihre ebenenspezifische Eigenschaften. Diese Eigenschaften beschreiben die Abhängigkeiten eines Objekts von anderen Objekten auf derselben Ebene. Diese Eigenschaften werden in dem Graphen durch Abhängigkeitskanten dargestellt. Der Grad der Abhängigkeit wird durch ein Kantenattribut beschrieben. Nutzt zum Beispiel eine rechenfähige Komponente eine andere rechenfähige Komponente, so existiert zwischen diesen beiden Knoten in dem Graphen eine Abhängigkeitskante, deren Attribut Auskunft über die Nutzungshäufigkeit etc. gibt.

Die vierte und somit letzte wesentliche Eigenschaft einer Ressource sind ihre Klasseneigenschaften. Diese Eigenschaften beschreiben spezifische Eigenschaften einer Ressource, wie zum Beispiel ihre Rechenleistung, oder ihre Zugriffszeiten. In dem Graphen sind diese Eigenschaften durch Knotenattribute repräsentiert.

2.3 Ressourcenmanagement als Graphersetzungssystem

Mit den geschaffenen Grundlagen ist es möglich, das Ressourcenmanagement mithilfe eines Graphersetzungssystems zu beschreiben. Der Graph beschreibt das zu verwaltende System zu einem Zeitpunkt t vollständig (Kapitel 2.1.1). Mithilfe der Graphersetzungsregeln können die Maßnahmen, die von einem Ressourcenmanagement vorgenommen werden, beschrieben werden (Kapitel 2.1.2). Als Grundlage für die Entscheidungen dienen die Eigenschaften, wie in Kapitel 2.2 beschrieben. Die Verwaltungsstrategien werden durch das Ersetzungssystem und dessen Strategie für die Auswahl von Ersetzungsregeln festgelegt.

Durch den Aufbau eines Graphersetzungssystem ist es damit möglich ein Ressourcenmanagement zu entwickeln und zu beschreiben. Der Vorteil liegt darin, daß hiermit eine konzeptionelle Trennung zwischen Bereitstellung von Informationen, Mechanismus und Strategie vorliegt und eine formale Verhaltensbeschreibung möglich ist. Die daran anschließenden Schritte dienen der systematischen Integration der Mechanismen, Strategien und Bereitstellungsfunktionen in das System.

2.3.1 Beschreibung der Fähigkeiten des Managements

In einem ersten Schritt wird das Ressourcenmanagement durch ein Graphersetzungssystem spezifiziert. Der Vorteil dieses Vorgehens liegt darin, daß sämtliche zu berücksichtigen Abhängigkeiten für einen Managementschritt anhand des Graphen erkannt werden können. Auf diese Weise läßt sich schon in dieser Phase überprüfen, ob auch alle Abhängigkeiten und Folgen eines Schritts berücksichtigt wurden. Auch läßt sich auf dieser Ebene viel leichter die Managementstrategie in Form von Auswahlregeln formal beschreiben. Diese Methode kann auch verwendet werden, um eine neue Verwaltungsstrategie in ein bestehendes System aufzunehmen.

2.3.2 Explizite Repräsentation der Managementinformationen

Das in Kapitel 2.3.1 spezifizierte Management basiert auf Graphen. In dem später real laufenden System wird es diesen Graphen explizit nicht mehr geben. In ihm enthaltene Informationen müssen dem Management aber für die Entscheidungsgrundlage zur Verfügung gestellt werden. Diese Aufgabe wird in diesem Schritt vorgenommen, indem in den Graphen/System Knoten aufgenommen werden, die Verwaltungsinformationen repräsentieren.

2.3.3 Darstellung des Ressourcenmanagements

Analog zu dem gemachten Schritt im letzten Kapitel, in dem die Managementinformation als Knoten in den Graphen/System aufgenommen wurden, gilt es in diesem Schritt das Ressourcenmanagement in den Graphen/System aufzunehmen. Dies entspricht dem Vorgang das Graphersetzungssystem in das System einzubetten. Im einzelnen sind hierzu Instanzen zu schaffen, die Ersetzungsregeln ausführen den Auswahlalgorithmus auf Basis der in Kapitel 2.3.2 explizit gemachten Informationen durchführen können.

Um die erste Aufgabe zu realisieren, kann versucht werden, sich auf schon vorhandene Transformationen, für die schon Knoten geschaffen wurden, zu stützen. Ist dies nicht möglich, so muß für diese Transformation eine Realisierung (Implementierung) geschaffen werden.

Für die zweite Aufgabe ist es notwendig, den Auswahlalgorithmus für Ersetzungsregeln auf verschiedene Knoten (Instanzen) zu verteilen. In dem „realen System“ wird auf jedem Rechner ein Teil des Ressourcenmanagements durchgeführt und somit auch ein Teil des Auswahlalgorithmus. In dem externen Ersetzungssystems gab es hierfür nur eine Instanz, die alle Informationen über das System hatte. Durch die Verwendung einer verteilten Hardwarebasis ist dies nicht möglich.

Eine sich daran anschließende weitere Aufgabe besteht in der Festlegung auf welchen in Kapitel 2.3.2 explizit gemachten Informationen diese Auswahlalgorithmen aufbauen sollen. Auch dieser Schritt ist eine Folge der physikalischen Verteiltheit. Die neu geschaffenen Auswahlinstanzen haben keinen Zugriff auf alle im Graphen/System vorhandenen Informationen. Daher ist eine Beschränkung auf wesentliche Informationen notwendig, falls dies bei dem Schritt in Kapitel 2.3.1 noch nicht geschehen ist.

2.3.4 Flexibilisierung des Managements

Um den Verwaltungsoverhead so minimal wie möglich und das System hochgradig flexibel halten zu können, wie dies auch in Kapitel 1 beschrieben wurde, werden in diesem Schritt auch Transformationen auf den im letzten Kapitel eingeführten Verwaltungsinstanzen beschrieben. Diesem Ansatz liegt die Idee zu Grunde, daß nicht alle Verwaltungsinstanzen, die durch die Verteiltheit des Systems entstehen, in der Lage sein müssen, alle Managementfunktionen auszuführen und alle eventuell benötigten Informationen ansammeln müssen. Es ist sogar sehr wahrscheinlich, daß viele Instanzen nur einen sehr geringen Teil der Managementfunktionen beherrschen müssen.

Um für die Beschreibung dieser Vorgehensweise wieder die Graphen anzuwenden heißt dies, daß in gewissen Teilen des Graphen nur eine beschränkte Zahl von Ersetzungsregeln potentiell zur Anwendung kommen können. Daraus folgt natürlich, daß die für diesen Teil des Graphen verantwortliche Auswahlinstanz (Verwalterinstanz) auch nur über eine eingeschränkte Anzahl von Transformationsmöglichkeiten und Informationen verfügen muß.

Um dies zu erreichen wird die Möglichkeit geschaffen, auch Verwaltungsinstanzen zu transformieren. Reichen die Fähigkeiten einer Instanz nicht mehr aus, wird sie in eine mächtigere Instanz transformiert. Die dazu nötigen Regeln sollen in diesem Schritt erarbeitet werden. Danach müssen für diese neuen Regeln die Schritte in den Kapitel 2.3.2 und 2.3.3 nochmals ausgeführt werden.

2.4 Der Übergang von dem Graphersetzungssystem zu einem realisierten Ressourcenmanagement

Der Übergang von dem Graphersetzungssystem zu einem realisierten Ressourcenmanagement findet nicht in einem großen Schritt statt, sondern ist in den in Kapitel 2.3 beschriebenen Einzelschritten enthalten. Zum Beispiel ist bei der Darstellung des Ressourcenmanagements in Kapitel 2.3.3 eine explizite Beschreibung der Strategie notwendig. Diese wird in den Graphen aufgenommen, indem für sie Instanzen angegeben werden müssen, die ihre Realisierung beschreiben. Damit ist ihre Realisierung aber auch schon festgelegt und der Programmiercode ablesbar.

Unter diesem Gesichtspunkt ist die Entwicklung des Graphersetzungssystems nicht von der des Ressourcenmanagements abzukoppeln, sondern vielmehr ist das Graphersetzungssystem die Grundlage und der Ausgangspunkt, um eine effiziente Ressourcenverwaltung für VPK-Systeme entwickeln zu können. Daher muß der erste Schritt auf diesem Weg sein, die Grundlagen für das

Graphersetzungssystem zu schaffen, sprich die ersten einfachen Transformationen aufzustellen, die das reale System beschreiben. Danach kann damit begonnen werden, die Fähigkeiten des Managements, wie in Kapitel 2.3.1, beschrieben zu spezifizieren. Wenn dies geschehen ist, beginnt bei den weiteren Schritten des Graphersetzungssystem auch die Implementierung der realen Ressourcenverwaltung.

2.5 Bewertung von Informationen

Ein noch nicht beschriebenes Problem dieses Ansatzes ist es, die Gewichtigkeit der Kanten zueinander zu bewerten. In vielen Fällen sollte eine Bewertung nicht notwendig sein, da die Auswahl der Ersetzungsregeln auch ohne eine Bewertung auskommen kann. Aber dies ist nicht in allen Fällen möglich. In diesen Fällen stehen dem Ressourcemanagement Alternativen zur Verfügung, d.h. mehrere verschiedene Ersetzungsregeln können angewendet werden. Hierzu ist es notwendig, die Kanten in dem System zu bewerten.

Dieses Problem soll durch die Einführung von Kräfte gelöst werden, wie sie aus der Physik bekannt sind. In Analogie gibt es daher in einem Graphen anziehende und abstoßende Kräfte. Diese Kräfte gehen von bestimmten festgelegten Objekten aus und können nur entlang der Kanten wirken. Ihre Stärke wird durch die Bewertung der Kante bestimmt. Der Vorteil dieses Ansatzes liegt in der Addierbarkeit der Kräfte und der einfachen Darstellbarkeit.

3 Zukünftige Arbeiten und Zusammenarbeit

Die nächsten Arbeiten werden sich darauf konzentrieren, die Graphersetzungsregeln für ein eingeschränktes System zu entwickeln. Diese Einschränkungen sollen in der Beschränkung auf wesentliche Objekte des abstrakten Systems bestehen und in einer Einschränkung der betrachteten Komponenten des realen Systems. Darauf aufbauend soll versucht werden, die am Lehrstuhl in den Projekten EVA ([Rad96]) und ADAM ([Win96]) geleisteten Vorarbeiten in dem Gebiet des Ressourcenmanagement, in diesen Ansatz zu integrieren. Weitere Anstrengungen müssen in dem Gebiet der Attribute, wie in Kapitel 2.5 dargelegt, unternommen werden.

Für die Entwicklung des hier vorgestellten Graphersetzungssystems ist die Zusammenarbeit innerhalb des Kollegs sehr hilfreich. Auf der einen Seite sind Ansprechpartner vorhanden, die sich mit Fragen, die das Graphersetzungssystem im Allgemeinen betreffen, weiterhelfen können. Auf der anderen Seite sind Ansprechpartner vorhanden, wenn es um Verwaltungsstrategien oder um mögliche Zusammenhänge in einer Anwendung und den daraus ablesbaren Informationen für ein Ressourcenmanagement geht. Durch die Nähe zu dem SFB342, hier besonders des Teilbereiches A8, wird der Kreis der kompetenten Ansprechpartner nochmals erweitert.

Literatur

[Inm84] Inmos Ltd., Englewood Cliffs. *Occam Programming Manual*, prentice hall. Auflage, 1984.

[Pol95] Rainer Pollak, Hrsg. *A Hierarchical Load Balancing Environment for Parallel and Distributed Supercomputer*, September 1995.

[Rad96] Ralf Radermacher. *Eine Ausführungsumgebung mit integrierter Lastverteilung für verteilte und parallele Systeme.* Dissertation, Technische Universität München, 1996.

[Spi95] P.P. Spies. Sprachkonzepte für die Konstruktion Verteilter Systeme. Technischer Bericht, Technische Universität München, 1995.

[SW92] A. Schürr und B. Westfechtel. Graph Grammars and Graph Rewriting Systems. Tech. Rep. AIB 92–15, RWTH Aachen, 1992.

[Win96] H.-M. Windisch. The Distributed Programming Language INSEL - Concepts and Implementation. In *High-Level Programming Models and Supportive Environments HIPS'96*, 1996.

Verteilte semantikgesteuerte Graphersetzung

Teil-Projekt: Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für verteilte Programmierung

Boris Reichel

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

1 Motivation

Im Rahmen dieses Teilprojekts wurden die Einsatzmöglichkeiten von Graphgrammatiken zur Spezifikation massiver Probleme untersucht. Hierbei hat sich unter anderem herausgestellt, daß sich Graphgrammatiken, wegen ihrer *intuitiven Darstellungsform*, sehr gut zur Beschreibung biochemischer Prozesse und der in den Datenstrukturen inhärenten Möglichkeiten von nebenläufiger Verarbeitung, eignen. Im speziellen wurde in Zusammenarbeit mit der Bioinformatik-Gruppe von Dr. B. Steipe, die sich mit Proteinfaltungsproblemen beschäftigt, der Einsatz von Graphgrammatiken als adäquates Modell zur Beschreibung der zu untersuchenden Problemstellungen erarbeitet. Innerhalb des Graduiertenkollegs hat sich in mehreren Sitzungen mit Barbara König und Sascha Groh herausgestellt, daß Graphgrammatiken sich außerdem wegen ihrer klaren und formal eindeutigen Repräsentation von *statischen und dynamischen Strukturen* ausgezeichnet zur Spezifikation von komplexen Systemen und der Vorhersage lokaler Eigenschaften eignen. Die aus dem Compilerbau bekannten Verfahren der Kontextberechnung lassen sich beispielsweise für die Vorhersage von Zusammenhängen zwischen einzelnen Systemkomponenten - Knoten der Graphen - verwenden, die dann für eine günstige Platzierung herangezogen werden können.

In allen Fällen hat sich gezeigt, daß im allgemeinen nur *konstextsensitive Graphgrammatiken* mit einer hinreichend komplexen *Einbettungsbeschreibung* und *Einbettungsprädikaten*, die sich auf *syntaktische und semantische Informationen* des Kontextgraphen beziehen, zur natürlichen Spezifikation der Systeme benötigt werden. Aus diesem Grund wird in diesem Teilprojekt, im Gegensatz zu anderen Arbeiten [Vol92, Dö95], die die Grammatiken für eine effiziente Interpretation durch ein *Graphersetzungssystem* stark einschränken, versucht die Beschränkungen so gering wie möglich zu halten.

Da die Graphen der repräsentierten Datenstrukturen der Probleme im allgemeinen sehr umfangreich sind - in biochemischen Problemstellungen im allgemeinen mehrere tausend Atome (Knoten) -, entsteht bei der Realisierung eines für die Praxis relevanten Graphersetzungssystems ein großes Effizienzproblem. Um dieses zu lösen wird in diesem Teilprojekt die *verteilte Realisierung* von Graphersetzungssystemen näher untersucht. Die verteilte Realisierung eines allgemeinen Graphersetzungssystems bietet schließlich eine *universelle verteilte Plattform* für beliebige durch Graphgrammatiken spezifizierbare Probleme.

In einem *monolitischen Graphersetzungssystem* führt eine sequentiell arbeitende Graphersetzungseinheit die Ersetzungssschritte durch. Im Gegensatz dazu führen in einem *verteilten Graphersetzungssystem* mehrere Graphersetzungseinheiten die Ersetzungsschritte *nebenläufig* durch. Da die Graphersetzungseinheiten einen *gemeinsamen* Graphen bearbeiten, müssen sie, um die Konsistenz des Graphen zu wahren, untereinander Informationen austauschen. Durch die Integration dieser Kommunikation in das verteilte Graphersetzungssystem erhält man für Spezifikationen mit Graphgrammatiken ein *implizites Kommunikationsmodell.* Dies hat gegenüber *expliziten Kommunikationsmodellen* den Vorteil, daß die gesamte Kommunikation vom Graphersetzungssystem aus der Spezifiaktion automatisch generiert wird, was eine Steigerung der Zuverlässigkeit zur Folge hat.

Der Bericht ist in drei Abschnitte gegliedert. In Abschnitt 2 werden semantikgesteuerte Graphersetzungssysteme und ihre Semantik informell eingführt. In Abschnitt 3 wird auf die durch eine verteilte Realisierung entstehende Synchronisationsproblematik dargestellt. Abschnitt 4 befaßt sich mit einer möglichen verteilten Realisierung eines Graphen.

2 Semantikgesteuerte Graphersetzungssysteme

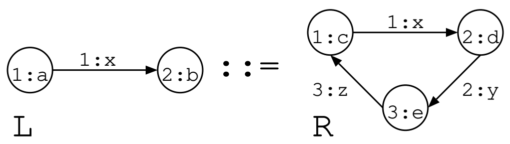
Analog zu Wortersetzungssystemen werden in diesem Abschnitt **semantikgesteuerte Graphersetzungssysteme** eingeführt.

Wie bei Wortgrammatiken besteht eine **Graphgrammatik** Γ im wesentlichen aus einer Menge von Markierungen Σ, einer Menge von Produktionen Π und einem Axiom S. Die Unterscheidung zwischen Kanten und Knoten in einem Graphen führt zu einer Partitionierung von Σ in die Mengen der Knotenmarkierungen Σ_E und Kantenmarkierungen Σ_V.

Auch **Produktionen** lassen sich ähnlich wie in Wortgrammatiken darstellen. Sie bestehen zum einen aus einer linken und einer rechten Seite L und R, wobei L und R keine Zeichenketten sondern gerichtete knoten- und kantenmarkierte Graphen sind. Da innerhalb von Graphen Begriffe wie "links" und "rechts" von einem Teilgraphen, welche die Einbettung bei einem Ersetzungsschritt im Wortfall implizit vorgeben, nicht existieren, muß bei Graphersetzungssystemen die Einbettungsbeschreibung E des zu R isomorphen, neu erzeugten Teilgraphen $\tilde{R}$ explizit angegeben werden. Hierzu muß auf Komponenten des **Wirtsgraphen** - Graph, dessen Teilgraph der zu L isomorphe Graph $\tilde{L}$ ist - zugegriffen werden können.

Die **Einbettungsbeschreibung** (bzw. **Einbettungsrelation**) E ist eine Relation zwischen Mengen von Einbettungsknoten - Knoten des Wirtsgraphen von $\tilde{L}$ - und Kantenbeschreibungen mit Knoten aus $\tilde{R}$. Hierzu existieren in der Literatur die verschiedensten Ansätze, die sich allerdings prinzipiell in die zwei Kategorieen *lokal* und *global* einordnen lassen [Sch92]. Im Gegensatz zu globalen Einbettungsbeschreibungen die für alle Produktionen des Systems gelten, können lokale für jede Produktion einzeln definiert werden. Die durchgeführten Untersuchungen haben gezeigt, daß für eine natürliche Problemspezifikation nur der lokale Ansatz hinreichend flexibel ist. Da die **Einbettungsbeschreibung** E auf einer Klasse von Wirtsgraphen G anwendbar sein muß, werden die Einbettungsknotenmengen durch Einbettungsterme und einen Interpretationsstartknoten $v_l \in V_L$ beschrieben.

Ein **Einbettungsterm** t^e ist eine statische Pfadbeschreibung für Graphen. In der Beispielproduktion in Abbildung 1 steht $t1$ für den Pfad "Einlaufende x-markierte gefolgt von auslaufender y-markierter Kante". Durch Anwenden einer, von der Belegung von G und $\tilde{L}$ abhängigen, **Einbettungsterminterpretation** $I_{G,\tilde{L}}$ auf den Term t^e und den zu v_l isomorphen Knoten $\tilde{v}_l \in \tilde{L}$ wird eine Knotenmenge $I_{G,\tilde{L}}[t^e](\tilde{v}_l) \subseteq G \setminus \tilde{L}$ erzeugt. Im Ableitungsbeispiel in Abbildung 2 gilt $I_{G1,\tilde{L}}[t1](4) = \{1,2\}$ und $I_{G2,\tilde{L}}[t1](6) = \{1,2,9\}$. Die zweite Komponente der Einbettungsrelation beschreibt welche Kanten zwischen der so erzeugten Knotenmenge und dem, zu $v_r \in V_R$ isomorphen, generierten Knoten $\tilde{v}_r$ erzeugt werden müssen. In Abbildung 2 folgt beispielsweise aus dem Element der Einbettungsbeschreibung $(t1, 1, 0, z, 1)$ die Erzeugung der Kanten 11 und 12.




```
t1 = (Oy(Ix)),t2 = (Ix),
t3 = (|t2|=1),
E = {(t1,1,0,z,1),(t2,1,I,y,2)},
C = {(t3,1)}
```

Abbildung 1: Spezifikation einer Produktion

Es hat sich außerdem herausgestellt, daß in sehr vielen Problemstellungen die Anwendbarkeit einer Produktion von der **semantischen Information** ihrer Umgebung abhängt. So ist beispielsweise bei der Proteinfaltungsproblematik das, durch eine Produktion beschreibbare, Aufbrechen einer Wasserstoffbrückenbindung hochgradig vom eigenen Energieniveau sowie den Umgebungseinfüssen abhängig. Um diese Informationen in der Spezifikation berücksichtigen zu können, müssen den Knoten und Kanten **Attribute** und **Attributwerte** zugewiesen werden können. Die Abhängigkeit einer Produktionsanwendug vom Wirtsgraphen wird durch das **Einbettungsprädikat** C dargestellt. Das Einbettungsprädikat besteht aus einem **Einbettungsterm** t^c und einem Interpretationsstartknoten $v_l \in V_L$. Analog zur Einbettungsbeschreibung wird der Term t^c relativ zu dem zu v_l isomorphen Knoten $\tilde{v}_l$ durch die **Einbettungsterminterpretation** $J_{G,\tilde{L}}$ für Einbettungsprädikate ausgewertet. Das in Abbildung 1 dargestellte Prädikat zeigt ein rein syntaktisches Prädikat. Die Interpretation $J_{G,\tilde{L}}[t3](1)$ liefert genau dann *true*, wenn mit dem zu Knoten 1 isomorphen Knoten in G genau ein Knoten durch eine einlaufende x-markierte Kante verbunden ist.

Die **Ersetzungssemantik** eines semantikgesteuerten Graphersetzungssystems wird über die Grammatik und den zugehörigen Ableitungsbegriff definiert. Ein Graph G' ist aus G mittels Produktion p **ableitbar**, wenn G einen zu L_p isomorphen Teilgraphen $\tilde{L}$ und G' einen zu R_p isomorphen Teilgraphen $\tilde{R}$ enthält, und $G \setminus \tilde{L} = G' \setminus \tilde{R}$ ist und für $J_{G,\tilde{L}}[t^c](v_l) = true$ gilt. Die Berechnung des Systems terminiert, wenn keine Ableitungsschritte mehr durchführbar sind.

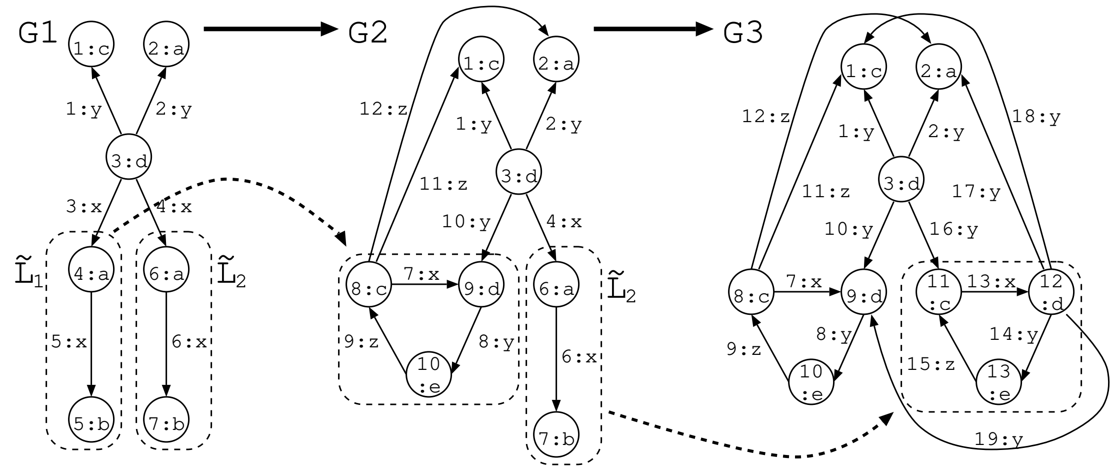


Abbildung 2: Zwei Schritte einer Ableitungssequenz

Im allgemeinen existieren in einem Graphen G eine Menge von Teilgraphen $\tilde{L}_i \subseteq G$ mit zugehörigen Produktionen $p_i \in \Pi$, die obige Bedingungen erfüllen und somit **potentielle Ableitungsstellen** markieren. Im Beispiel in Abbildung 2 erfüllen die durch die Knotenmengen $\{4, 5\}$ und $\{6, 7\}$ definierten Teilgraphen alle Bedingungen. Da die Reihenfolge der Ableitungsschritte nicht durch das Graphersetzungssystem definiert wird, ist der Ablauf im allgemeinen **nichtdeterministisch**. Es können allerdings auch deterministische Problemstellungen wie sackgassenfreie Syntaxanalyse von graphartigen Strukturen spezifiziert werden. Hierzu müssen lediglich die Einbettungprädikate der vorgegebene Grammatik so um Kontextabhängigkeit erweitert werden, daß die Anwendung der einzelnen Produktionen nicht in Sackgassen führen kann [Vol92]. Bedingungen für Grammatiken, die konfluente Graphersetzungssysteme beschreiben, werden in [Loy92] diskutiert.

Durch die verteilte Realisierung eines Graphersetzungssystems kann es vorkommen, daß mehrere Graphersetzungseinheiten potentielle Ableitungsstellen finden, die sich überlappen. Die Überlappung kann unter Umständen zu gegenseitigem Ausschluß der zugehörigen Ableitungsschritte führen. Um diese Problematik formal handhaben zu können, wird der Begriff der Ableitungsspezifikation eingeführt. Eine **Ableitungsspezifikation** A enthält alle Informationen, die für einen konkreten Ableitungsschritt benötigt werden. Dies ist im speziellen der zu L isomorphen Teilgraphen $\tilde{L}$ und der von der Einbettungsinterpretation benötigte Einbettungsteilgraphen $G^e \subseteq G$.

3 Probleme bei verteilter Graphersetzung

Im folgenden Abschnitt wird die verteilte Realisierung eines Graphersetzungssystems näher betrachtet. Hierbei wird im speziellen auf die entstehende Synchronisationsproblematik zwischen einzelnen Ersetzungsschritten, die nebenläufig durch verschiedene Graphersetzungseinheiten ausgeführt werden könnten, eingegangen.

Wie bereits erläutert, kann es vorkommen, daß in einem Graphen G mehrere potentielle Ableitungsstellen mit zugehörigen Ableitungsspezifikationen $A_i = (\tilde{L}_i, G_i^e)$ existieren. Auf der Menge der Ableitungsspezifikationen werden, zur Klassifikation von potentiell nebenläufig durchführbaren Ableitungsschritten, die Relationen **unkritisch**, **potentiell kritisch** und **kritisch** wie folgt definiert. Es gelte $A_n \neq A_m$.

- A_n**unkritisch**$A_m \;\Leftrightarrow\; ((\tilde{L}_n \cup G_n^e) \cap (\tilde{L}_m \cup G_m^e) = \emptyset$
  "Zwei Ableitungsspezifikationen sind unkritisch, wenn sich die zugehörigen Ersetzungsschritte gegenseitig nicht beeinflussen"
- A_n**potentiell kritisch**$A_m \;\Leftrightarrow\; (\tilde{L}_n \cap \tilde{L}_m = \emptyset \wedge (G_n^e \cap (\tilde{L}_m \cup G_m^e \neq \emptyset) \vee (\tilde{L}_n \cup G_n^e) \cap G_m^e \neq \emptyset))$
  "Zwei Ableitungsspezifikationen sind potentiell kritisch, wenn sich die zugehörigen Ersetzungsschritte gegenseitig beeinflussen könnten"
- A_n**kritisch**$A_m \;\Leftrightarrow\; (\tilde{L}_n \cap \tilde{L}_m \neq \emptyset)$
  "Zwei Ableitungsspezifikationen sind kritisch, wenn sich die zugehörigen Ersetzungsschritte auf jeden Fall gegenseitig beeinflussen"

Wenn zwei Ableitungsspezifikationen **unkritisch** sind, können sich die entsprechenden Ableitungsschritte weder durch das Löschen von $\tilde{L}$ noch durch die Einbettung von $\tilde{R}$ gegenseitig beeinflussen. Sie können also immer nebenläufig ausgeführt werden.

Wenn zwei Ableitungsspezifikationen **potentiell kritisch** sind, kann die Einbettung des einen Ableitungsschritts durch Ausführung des anderen verändert werden. Dies ist genau dann der Fall, wenn Kanten und Knoten erzeugt werden die die Einbettung der jeweils anderen Ableitungsspezifikation erweitern oder Einbettungsknoten gelöscht werden. Durch Hintereinanderausführung der zugehörigen Ableitungsschritte erhält man eine erste, für verteilte Umgebungen nicht immer optimale, Strategie zur Konfliktauflösung. Hierbei wird nach Ausführung des ersten Ableitungsschrittes die Einbettung für den zweiten entsprechend der Einbettungsvorschrift erweitert. Im Beispiel in Abbildung 2 sind die zu $\tilde{L}_1$ und $\tilde{L}_2$ gehörenden Ableitungsspezifikationen potentiell kritisch. Durch die Ausführung des ersten Ableitungsschrittes vergrößert sich die Einbettungsknotenmenge für den zweiten Ableitungsschritt. Daraus folgt schließlich die Asymmetrie des dritten Graphen.

Da sich die Einbettungen in vielen Fällen nicht beeinflussen, ist es sinnvoll nach Kriterien zu suchen, die dies garantieren. Wenn die beiden Ableitungsschritte jeweils Knoten- und Kantenmarkierungen erzeugen, die in der anderen Einbettung nicht betrachtet werden, können sich überlappende Einbettungen gegenseitig nicht verändern, was eine nebenläufige Ausführung der Ableitungsschritte zulässt. Um die Effizienz des Systems nicht zu vermindern, sollten die Kriterien in einer *statischen* Produktionsanalyse überprüfbar sein.

Wenn zwei Ableitungsspezifikationen **kritisch** sind, beeinflussen sich die entsprechenden Ableitungsschritte durch Löschen von Elementen des jeweils anderen $\tilde{L}$. Es kann also immer nur einer von beiden ausgeführt werden.

4 Realisierung verteilter Graphen

Um ein Graphersetzungssystem verteilt zu realisieren, müssen alle Graphersetzungseinheiten auf den gemeinsam zu verarbeitenden Graphen zugreifen können. Da der Graphersetzungsmechanismus immer nur einen *zusammenhängenden lokal begrenzten* Teilgraphen verändert, ist es sinnvoll, den gesamten Graphen in eine Menge von solchen Teilgraphen zu partitionieren, die jeweils einer Graphersetzungseinheit zugeordnet werden.

Aus der Sicht einer Graphersetzungseinheit besteht der zu verarbeitende Graph also aus einem **lokalen** und einem **nichtlokalen** Anteil. Um den Kommunikationsaufwand bei nichtlokalen Zugriffen, die durch Einbettungsauswertung entstehen können, zu minimieren, ist es sinnvoll die Randbereiche zwischen den einzelnen Teilgraphen redundant in den beteiligten Teilgraphen zu verwalten. Da in dieser Arbeit nur Einbettungstermen für Pfade fester Länge spezifiziert werden können, kann die maximale Pfadlänge als obere Schranke für die Tiefe der redundanten Bereiche verwendet werden.

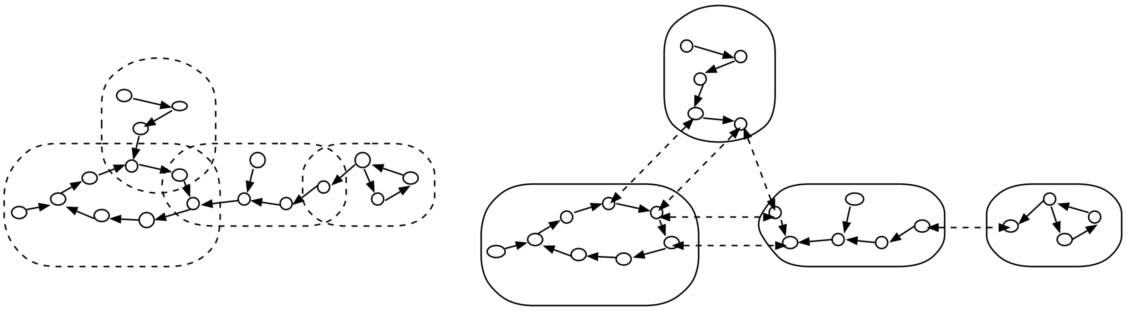


Abbildung 3: Ein verteilter Graph

Durch die redundante verteilte Datenhaltung entstehen allerdings **Kohärenzprobleme**. Da die Graphersetzungseinheiten die den Knoten und Kanten zugeordneten Attribute nur lesen und nicht verändern, wird der Verwaltungsaufwand hierfür entsprechend vereinfacht. Für die strukturelle Information bleibt die Problematik allerdings vollständig erhalten. Um sie zu lösen müssen Mechanismen eingeführt werden, die einzelne Komponenten exklusiv einer Ersetzungseinheit zuordnen können.

5 Zusammenarbeit mit anderen Teilprojekten

In diesem Abschnitt sollen kurz die Anknüpfungspunkte mit anderen Teilprojekten des Graduiertenkollegs sowie die Zusammenarbeit mit der Bioinformatik-Gruppe dargestellt werden.

Innerhalb des Graduiertenkollegs werden Graphgrammatiken von Barbara König und Sascha Groh verwendet. Da Barbara König sich im gleichen Teilprojekt mit der Spezifikation von Aktorsystemen durch Graphgrammatiken beschäftigt, ist hier in naher Zukunft mit einer engeren Kopplung der Arbeiten zu rechnen. Im speziellen bietet ein verteiltes Graphersetzungssystem eine geeignete Plattform für die Realisierung ihrer Studien. Mit Sascha Groh, der sich mit der Spezifikation von Ressourcenverwaltung durch Graphgrammatiken beschäftigt, ist ebenfalls ein reger Gedankenaustausch bezüglich der Vorhersagbarkeit von Systemverhalten im Gange.

Externe Zusammenarbeit besteht mit der Bioinformatik-Gruppe um Dr. B. Steipe. Ziel der Forschungsarbeiten ist das Verständnis makromolekularer Prozesse im allgemeinen, sowie die Vorhersagbarkeit von Proteinfaltungen im speziellen. Durch die Komplexität dieser Problemstellungen sind momentan selbst auf leistungsfähigen Parallelrechnern hinreichend genaue Simulationen auf atomarer Ebene nur für relativ kurze Zeitintervalle möglich. Graphgrammatiken bieten die Möglichkeit die Probleme auf einem höheren Niveau zu beschreiben und die den Problemen zugrundeliegende strukturelle Information bei der Auswertung zu berücksichtigen. Auch hier bietet ein verteiltes Graphersetzungssystem eine geeignete Plattform zur Simmulation der Prozesse, was zu einem völlig neuen Verständnis der zugrundeliegenden Abläufe führen kann.

Literatur

[Dö95] Heiko Dörr. *LNCS 922 — Efficient Graph Rewriting and Its Implementation.* Springer, 1995.

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

[Sch92] Andy Schürr. Graphgrammatiken und Graphersetzungssysteme (Vorlesungsskript). Technical report, RWTH Aachen, 1992.

[Vol92] Ulrich Vollath. *Generierung neuer Syntaxanalyseverfahren für Graphgrammatiken.* Dissertation, Technische Universität München, Institut für Informatik, August 1992.

Beschreibung und Implementierung von verteilten objekt-orientierten Programmiersprachen durch Graphgrammatiken

Teil-Projekt: Verteilte Realisierung von Spezifikationsmodellen aus dem Compilerbau und Compiler für verteilte Programmierung

Barbara König

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

1 Verteilte objekt-orientierte Programmiersprachen und Graphgrammatiken

Bei der Programmierung verteilter Systeme ist—wie bei sequentieller Programmierung—Abstraktion und Modularität zur leichteren Wartbarkeit und Wiederverwendbarkeit von entscheidender Bedeutung.

Diese Softwareentwurfs-Kriterien werden von verteilten objekt-orientierten Programmiersprachen unterstützt, die zur Zeit Gegenstand lebhafter Forschung sind ([AWY93]). Eine der bekanntesten Sprachen ist das Aktormodell von Hewitt und Agha ([Agh86]).

Ein Aktorsystem kann in natürlicher Weise mit einer Graphgrammatik beschrieben werden ([JR90, Jan93]) und mit einem Graphersetzungssystem implementiert werden. Graphen haben gegenüber textueller Aufschreibung eines Programms den Vorteil, daß sie die Verteiltheit und Dynamik des Systems und die Lokalität der einzelnen Komponenten besser wiedergeben können. Aus einem Graphen, der einen Zustand des Aktorsystems wiedergibt, können Hinweise für Plazierung und Migration der Aktoren bei einer Implementierung gewonnen werden.

Graphgrammatiken können mit Hilfe algebraischer Spezifikationen und Kategorientheorie ([Kor93, BS93]) elegant beschrieben werden. Diese kategorientheoretische Beschreibung kann auch auf Aktorsysteme ausgedehnt werden und führt zu einer formalen Semantik, mit deren Hilfe man Beweise über Eigenschaften eines Systems führen kann.

2 Problemstellungen

Beim Entwurf großer verteilter Systeme ist es wünschenswert, wenn gewisse Konzepte unterstützt werden, wie z.B. *Modularität*, *Wiederverwendbarkeit von Code*, *Synchronisations- und Koordinationsmechanisen*, Nachweis von *Invarianten*. Dabei ergeben sich u.a. folgende Fragestellungen:

- Bei verteilten objekt-orientierten Sprachen gab es bislang Probleme, *Vererbung* und *Synchronisationsbedingungen* zu verbinden ([MY93]). Es ist zu untersuchen, wie Vererbung in Aktorsysteme integrierbar ist, inwiefern diese "Vererbungs-Anomalie" auftaucht und wie man sie vermeiden kann.

- Modularität kann mit Hilfe von *offenen Aktorsystemen* erreicht werden ([Agh86]). Es ist zu untersuchen, wie man die Komposition offener Aktorsysteme auf der Ebene der Graphgrammatiken beschreiben kann.

- In verteilten Systemen müssen Koordinationsmechanismen unterstützt werden, die z.B. dazu dienen, das Auftreten von Deadlocks verhindern. (Zur Einbindung dieser Mechanismen in Aktorsysteme siehe [FA93].) Es wäre wünschenswert, wenn man aus höheren Spezifikationen von Koordinationsmechanismen Synchronisationsbedingungen auf Aktorebene generieren könnte.

Zur Realisierung verteilter objekt-orientierter Sprachen können *verteilte Graphersetzungssysteme* verwendet werden (siehe auch Abschnitt 3). Damit kann man alle in diesen Sprachen beschreibbaren verteilten Probleme auf das Problem der verteilten Graphersetzung reduzieren und die bei der Verteilung entstehenden Schwierigkeiten an einer zentralen Stelle bewältigen.

Ein Compiler hat dann die Aufgabe, ein Programm in Graphgrammatik-Produktionen zu übertragen.

Durch Realisierung verteilter Anwendungen soll die Praktikabilität und Effizienz der Implementierung einer verteilten objekt-orientierten Sprache durch ein Graphersetzungssystem überprüft werden.

3 Kooperation mit anderen Teilnehmern des Graduiertenkollegs

Ich arbeite seit Januar 1996 im Graduiertenkolleg und baue gerade Kontakte zu anderen Kollegiaten auf.

Mit Boris Reichel, der im selben Teilprojekt arbeitet, wird sich eine enge Zusammenarbeit im Bereich Graphgrammatiken ergeben. Sein verteiltes Graphersetzungssystem möchte ich zur Implementation von Aktorsystemen benutzen.

Geeignete Implementierungssprachen sind die Grundlage aller verteilten Systeme. Daher könnten Aktorsysteme verwendet werden, um Agenten aus dem Teilbereich *Kooperierende Agenten in verteilten Anwendungen* oder Teile des objekt-basierten INSEL-Systems des Teilprojekts *Konstruktion heteromorph paralleler Systeme* zu beschreiben.

Literatur

[Agh86] Gul A. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, Massachusetts, 1986.

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

[BS93] K. Barthelmann und G. Schied. Graph-Grammar Semantics of a Higher-Order Programming Language for Distributed Systems. In *Graph Transformations in Computer Science*, Seiten 71–85. Springer, 1993. LNCS 776.

[FA93] S. Frølund und G. Agha. A Language Framework for Multi-Object Coordination. In *Proceedings of ECOOP 1993.* Springer-Verlag, 1993. LNCS 707.

[Jan93] D. Janssens. ESM-Systems and the composition of their computations. In *Graph Transformations in Computer Science*, Seiten 203–217. Springer, 1993. LNCS 776.

[JR90] D. Janssens und G. Rozenberg. Graph Grammar-Based Description of Object-Based Systems. In *Foundations of Object-Oriented Languages*, Seiten 341–404. Springer, 1990. LNCS 489.

[Kor93] M. Korff. Single Pushout Transformations of Equationally Defined Graph Structures with Applications to Actor Systems. In *Graph Transformations in Computer Science*, Seiten 234–247. Springer, 1993. LNCS 776.

[MY93] Satoshi Matsuoka und Akinori Yonezawa. Analysis of Inheritance Anomaly in Object-Oriented Concurrent Programming Languages. In *Research Directions in Concurrent Object-Oriented Programming*, Kapitel 4. MIT Press, Cambridge, Massachusetts, 1993.

Hierarchisch strukturierte numerische Algorithmen auf Workstation-Netzen

Teil-Projekt: Verteilte numerische Algorithmen auf Bäumen

Anton Frank
e-mail: frank@informatik.tu-muenchen.de

1 Einleitung

In der numerischen Mathematik gibt es eine große Anzahl von Problemstellungen, die eine Verarbeitung von derart umfangreichen Datenmengen erfordern, daß ein normaler Arbeitsplatzrechner zur Lösung des Problems nicht ausreicht. Somit ist es erforderlich, entweder das Potential spezieller Vektor- bzw. Parallelrechner oder das eines Netzwerks aus Workstations — evtl. sogar gemeinsam — zu nutzen. Schwierigkeiten treten hier sowohl bei der notwendigen Umstrukturierung der Algorithmen und Datenorganisation als auch beim Softwareentwurf oder der Verwaltung der vorhandenen Berechnungsressourcen auf. Daher gibt es gerade im Bereich der numerischen Simulation im Rahmen des technisch-wissenschaftlichen Hochleistungsrechnens einen großen Bedarf an neuen Methoden zur Behandlung dieser Problemstellungen.

Dieser Bericht soll einen Überblick über die laufenden Forschungsaktivitäten in diesem Bereich am Lehrstuhl für Ingenieuranwendungen in der Informatik und numerische Programmierung (Prof. Zenger) und einen Ausblick auf die zukünftigen Arbeiten im Rahmen des Graduiertenkollegs geben.

2 Vorangegangene Arbeiten

2.1 Numerische Algorithmen

Im Bereich der numerischen Simulation werden physikalische Eigenschaften oft durch Systeme partieller Differentialgleichungen beschrieben. Das Lösen partieller Differentialgleichungen erfordert die Lösung großer dünnbesetzter Gleichungssysteme, was in akzeptabler Zeit mit hinreichender Genauigkeit nur iterativ zu bewerkstelligen ist. Neuartige Verfahren zur iterativen Lösung solcher Gleichungssysteme wurden in den letzten Jahren entwickelt und weiter verbessert. Neben den klassichen SOR- und CG-Verfahren haben vor allem die sogenannte Mehrgittermethode und ihre Erweiterung zur algebraischen Mehrgittermethode sowie allgemein Multilevelverfahren an Bedeutung gewonnen. Durch eine stufenweise Verarbeitung der Daten wird eine Unabhängigkeit der Konvergenzrate von der Anzahl der Unbekannten erreicht, was bei der Behandlung großer Gleichungssysteme sehr vorteilhaft ist. Trotz dieser Bemühungen auf mathematisch-algorithmischer Seite sind viele Probleme so umfangreich, daß sie aufgrund ihres Laufzeit- und Speicherplatzbedarfs nur unter großem Aufwand an Rechenleistung und Speicherkapazitäten berechenbar sind. Gerade in den Ingenieurwissenschaften ist die Optimierung dieser Lösungsverfahren von herausragender Bedeutung, da sie auf dem Feld der numerischen Simulation den Hauptanteil der gesamten Berechnungen ausmachen. Als Beispiele seien Anwendungen aus der Baustatik [HS94], der Elektrotechnik [AR94] sowie der Fluidmechanik [GHZ93] genannt, wie sie auch im Bayerischen Forschungsverbund für technisch-wissenschaftliches Hochleistungsrechnen (FORTWIHR) untersucht werden.

2.2 Hierarchische Strukturen und Parallelisierung

Versucht man nun, solche Berechnungen durch Parallelisierung zu beschleunigen, so stellt man fest, daß sich die herkömmlichen Algorithmen dafür nur bedingt eignen, da Datenabhängigkeiten im Entwurf nicht berücksichtigt sind. Ein möglicher Ansatz für eine günstigere Organisation der Daten ist die Aufteilung in voneinander unabhängige, separat berechenbare Teile.

Die sogenannte Divide-and-Conquer-Methode, bestehend aus den drei Phasen Aufteilung (divide), Berechnung (conquer) und Zusammenfügung (merge), führt bei rekursiver Fortsetzung zu

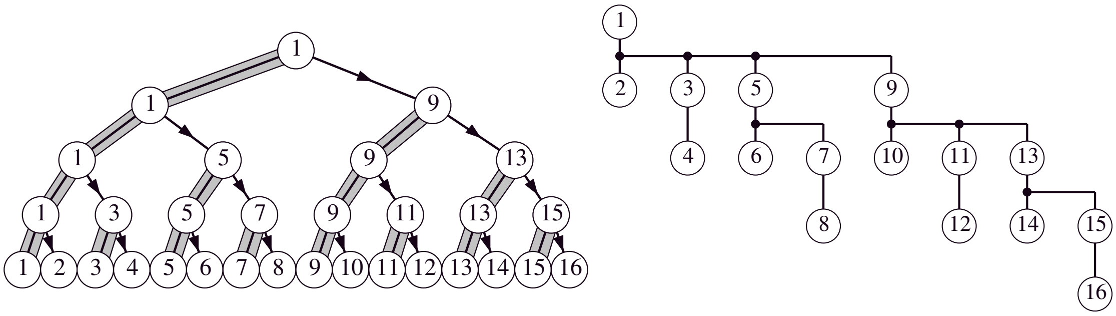


Abbildung 1: Hierarchische Berechnungsabfolge und entsprechendes baumartiges Netzwerk

einer hierarchischen, durch Bäume beschreibbaren Struktur. Hierarchische Strukturen erleichtern Entwurf und Analyse paralleler Algorithmen und ermöglichen die einfache Verwendung adaptiver Strategien.

Im Gegensatz zu voneinander unabhängig berechenbaren Daten, wie sie z. B. in der Computergraphik beim Raytracing vorliegen, kommt im Fall der numerischen Algorithmen der Merge-Phase eine größere Bedeutung zu. Hier ist besonders darauf zu achten, daß der Laufzeitgewinn bei der Parallelisierung durch die zusätzlich erforderlichen Berechnungen nicht wieder verlorengeht.

Da meist ein vorgegebenes Rechengebiet vorliegt, ist es naheliegend, dieses Gebiet nach festgelegten Kriterien in Teilgebiete zu unterteilen. Wird dieses sogenannte Gebietszerlegungsverfahren rekursiv fortgesetzt, so spricht man von rekursiver Substrukturierung. Hierarchische Methoden finden außer zur Lösung partieller Differentialgleichungen z. B. auch Verwendung bei der Beschleunigung der Finite-Elemente-Methode, zur effizienten Berechnung der mehrdimensionalen numerischen Quadratur oder bei der Interpolation zur Datenreduktion [Fra95].

Einige der oben genannten Verfahren wurden bereits auf ihr Potential hinsichtlich der Parallelisierung untersucht ([Gri95], [GG94], [GN] u. a.). Es zeigt sich, daß auf Parallelrechnern sehr gute Ergebnisse in Bezug auf Speedup und Effizienz erzielt werden können. Auf Workstation-Netzen erreicht man ebenfalls gute, teilweise sogar mit Parallelrechnern vergleichbare Performanceraten, wobei man hierzu aber auf eine exklusive Benutzung der Workstations angewiesen ist.

2.3 Workstation-Netze

Gegenüber spezialisierten Parallelrechnern haben Workstation-Netze, welche in Anschaffung und Wartung billiger und in der Handhabung einfacher sind und in der Regel insgesamt über mehr Speicher verfügen, in letzter Zeit an Attraktivität gewonnen. Es müssen nun Methoden gefunden werden, die den speziellen Anforderungen von Workstation-Netzen gerecht werden. Im Vergleich zu Parallelrechnern ist festzustellen, daß bei Workstation-Netzen das Kommunikationsmedium den ausschlaggebenden Engpaß darstellt. Folglich muß das Hauptaugenmerk in diesem Fall speziell auf der Minimierung der Kommunikation oder der Erhöhung der Bandbreite liegen. Gerade das üblicherweise zur Vernetzung im lokalen Bereich verwendete Ethernet eignet sich nur bedingt für die oben genannten Problemstellungen, da es auf einer Bus-Struktur basiert und ein kollisionsbehaftetes Zugriffsprotokoll verwendet. Durch Einbau einer zweiten Ethernet-Karte in die Workstations kann eine doppelte Anbindung über zusätzliche Busse eingeführt werden, so daß mehrere parallel laufende Kommunikationen stattfinden können, was zu einer höheren Bandbreite führt. Es ist damit möglich, das Netz bzgl. der Topologie hierarchisch anzuordnen (s. Abb. 1). Bei Divide-and-Conquer-Algorithmen übernimmt der erste Rechner die Aufteilung der Daten, übergibt eine Hälfte an einen zweiten Rechner und berechnet die andere Hälfte selbst. Nach Beendigung der Berechnung ist dieser Rechner auch für das Zusammenfügen beider Teilergebnisse zuständig. Sind die Teilprobleme unterschiedlich rechenintensiv, so kann die Auslastung der Rechner im Extremfall sehr stark voneinander abweichen, was bedeutet, daß auch hier das Ressourcenmanagement von großer Bedeutung ist. Eine baumartig vernetzte Workstation-Topologie kann auch in eine hypercube-artige Topologie integriert werden, was neben einer allgemeineren Verwendbarkeit aufgrund der zyklischen Verbindungen zu einer besseren Fehlertoleranz führt [Pfa95].

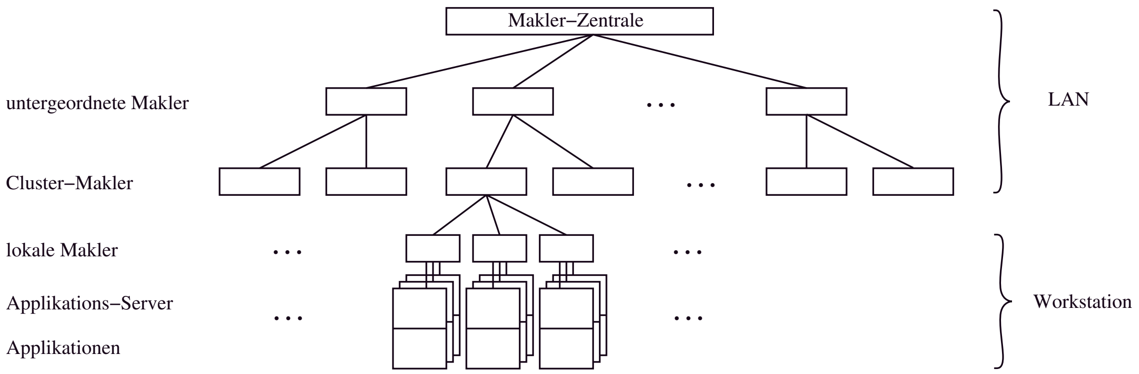


Abbildung 2: Hierarchische Organisation des Maklersystems zur dynamischen Lastverteilung

2.4 Funktionale Sprachen zur Programmentwicklung

Generell problematisch ist bei verteilten Anwendungen der sehr viel höhere Entwicklungsaufwand, da sich die Fehlersuche in parallelen Programmen sehr kompliziert gestaltet und es bisher noch keine befriedigenden computergestützten Programmierumgebungen gibt. Bei imperativen Programmiersprachen sind komplexe Abhängigkeiten und auftretende Seiteneffekte nur schwer zu überblicken, da das Programm durch die Abarbeitungsreihenfolge und nicht durch logische Zusammenhänge bestimmt wird. Daher bietet es sich an, funktionale Sprachkonzepte zur Beschreibung paralleler Algorithmen zu verwenden [Zen92]. Die am Lehrstuhl für solche Fragestellungen entwickelte funktionale Sprache FASAN ermöglicht es dem Programmierer, unter Verwendung von elementaren Programmodulen (z. B. numerischen Grundalgorithmen) auf einfachere Weise ein paralleles Programm zu erstellen [EPZ]. Besonders gut eignet sich dieser Ansatz für numerische Divide-and-Conquer-Algorithmen, da spezielle hierarchische Datenflußmechanismen, sogenannte Kabelbäume, eingesetzt werden. Dieses Konzept wird im Rahmen des Sonderforschungsbereichs 342 erforscht.

2.5 Ein System zur dynamischen Lastverteilung

Um die zur Verfügung stehenden Ressourcen möglichst optimal auszunutzen, ist eine intelligente Verteilung der zu berechnenden Teilprobleme auf die vorhandenen Rechner ein entscheidender Punkt. 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 vor allem als Arbeitsplatzrechner genutzt werden. Dieser Benutzerbetrieb darf durch die zusätzliche Last der Rechnungen nicht beeinträchtigt werden. Da das Laufzeitverhalten der Prozesse nicht exakt vorhersagbar ist, ist statt manueller oder statischer Lastverteilung der Einsatz eines dynamischen Systems von Vorteil.

Aus diesem Grund wurde ein hierarchisch organisiertes System zur dynamischen Lastverteilung auf Workstation-Netzen entwickelt [BPZ], das sich auch sehr gut für hierarchisch strukturierte numerische Algorithmen eignet. Dieses System orientiert sich an ökonomischen Vorgängen und simuliert dazu Waren, Kunden und eine Hierarchie von Maklern (s. Abb. 2). Als Kunden fungieren die Berechnungsaufträge, die Waren sind die benötigten Ressourcen, wie z. B. Rechnerleistung oder Speicherplatz. Die Aufgabe der Makler ist es, die Zuteilung der Waren an die Kunden zu organisieren, d. h. die Makler ermitteln die am besten geeignete Maschine und organisieren die Migration dorthin. In diesem simulierten Wirtschaftskreislauf steht für die Berechnungsaufträge virtuelles Geld zur Verfügung. Die Verteilung der Aufträge erfolgt nach ökonomischen Gesichtspunkten, Steuerungsmechanismen sind die Kosten, die den Aufträgen durch Ressourcenmiete, Maklercourtage oder Transportkosten bei Kommunikation oder Migration entstehen.

Durch die hierarchische Anordnung erreicht man eine einfache Skalierbarkeit des Lastverteilungssystems. Auch adaptive Strategien, die bei statischer Lastverteilung zu einer stark unausgeglichenen Verteilung führen können, werden hier viel besser berücksichtigt. Ein weiterer Vorteil liegt in der Trennung von Applikation und Maklersystem, da dadurch kaum Veränderungen in den Programmcodes der Applikationen vorgenommen werden müssen.

3 Zielvorstellung und Ausblick

Die bisherigen Tests beschränken sich auf die Simulation von Problemen, die unter Verwendung der rekursiven Substrukturierung gelöst werden. Da die Aufgabe hier überall gleich geartet ist, läuft auf allen Knoten dasselbe Programm (SPMD). Daher migrieren bisher ausschließlich Daten. Dies kann durch Programm-Migrationen ausgeweitet werden, was insbesondere beim Einsatz unterschiedlicher numerischer Löser auf verschiedenen Daten notwendig werden kann. Auch sollen in Zukunft die Migrationen statt durch die Makler durch separate Transporteinheiten durchgeführt werden.

Die hierarchische Struktur des Ansatzes ermöglicht es auch, die Lastverteilung über Local-Area-Netzwerke hinaus vorzunehmen. Da die Netzwerke untereinander in ihrer logischen Struktur hierarchisch angeordnet sind, sind globale Makler in der Lage, auch Überblick über Wide-Area-Netze zu besitzen. Bei großen Netzen kann sich eine dynamische Maklerhierarchie als vorteilhaft erweisen. Hierzu eignen sich wohl insbesondere Konzepte, die der Theorie kooperierender Agenten entnommen sind. Auch die Berücksichtigung vorhandener besonderer Netztopologien oder vernetzter Spezialcomputer, wie Parallel- oder Vektorrechner, bei der Ressourcenverwaltung kann zu einer effizienteren Behandlung paralleler numerischer Algorithmen führen. Dies ist leicht in ein hierarchisches System, in dem solche Informationen lokal gehalten und bei Bedarf nach oben weitergereicht werden, integrierbar.

Darüber hinaus bietet es sich an, die Strategien des Maklersystems durch umfangreichere Simulation von Wirtschaftskreisläufen zu verbessern. Z. B. kann mit Einführung einer Art Vertragssystem durch Ressourcenplanung langfristig eine bessere Verteilung erreicht werden. Da numerische Programme oft mehrmals hintereinander gleichartig ablaufen, können so die gewonnenen Lastverteilungsinformationen der ersten Läufe in das weitere Ressourcenmanagement einfließen.

Außerdem ist es möglich, die Entwicklung paralleler Programme und die Automatisierung der Verteilung zumindest für numerische Divide-and-Conquer-Algorithmen zu verbessern, wenn man eine allgemeine Schnittstelle zwischen Applikation und Berechnungsressourcen spezifiziert. Diese Spezifikation kann dann über eine funktionale Sprachbeschreibung vom Lastverteilungssystem zur automatischen Ressourcenverwaltung genutzt werden.

Das Ziel in Bezug auf die numerische Simulation ist die Beschleunigung der dazu verwendeten speziellen Algorithmen, insbesondere der Mehrgitterverfahren, durch Parallelisierung, automatische Verteilung, Ressourcenmanagement und neue Netztopolgien, um Probleme zu berechnen, die bisher nur unter unverhältnismäßig hohem Aufwand zu bewältigen sind. Durch die inhaltliche Nähe zu den im Rahmen des FORTWIHR behandelten Themen ist eine Fülle von industrierelevanten Problemstellungen vorhanden [HHSZ94]. Der Entwurf verbesserter Algorithmen, auch unter Berücksichtigung der Lastverteilungs- und Ressourcenmanagementproblematik, wird nach wie vor ein wichtiger Punkt bleiben, da die Algorithmen ebenfalls an die sich ständig ändernden Anforderungen bezüglich Datenverwaltung, Kommunikation usw. angepaßt werden müssen.

4 Kooperationen mit anderen Teilprojekten

Im Rahmen des Graduiertenkollegs eröffnen sich mehrere interessante Ansatzpunkte für Kooperationen. Im Bereich des Ressourcenmanagements bieten insbesondere die von C. Röder untersuchten Lastverwaltungsstrategien auf gekoppelten Arbeitsplatzrechnern Möglichkeiten zur Zusammenarbeit. Die automatische Unterstützung bei der Fehlersuche durch temporale Logik (M. Frey) könnte bei der Entwicklung paralleler Programme mit Hilfe funktionaler Sprachen von Nutzen sein. Die Ergebnisse der Kollegiaten, die mit dem Entwurf kooperierender Agenten betraut sind, können z. B. in den Entwurf eines intelligenten Maklersystems einfließen, bei dem die Makler als kooperierende Agenten modelliert werden, um das gesamte Ressourcenmanagement flexibler und damit effektiver zu gestalten.

Darüber hinaus bestehen Kooperationsmöglichkeiten innerhalb des SFB 342, wo man sich ebenfalls mit hierarchisch strukturierten numerischen Algorithmen befaßt. Die Ergebnisse wiederum können in die Optimierung der Simulationsverfahren, wie sie im Rahmen des FORTWIHR entwickelt werden, einfließen.

Literatur

[AR94] M. Aschauer und H. Regler. Parallelization of the Placement Tool GORDIAN with a Divide-and-Conquer Method. In M. Brehm und Ch. Schaller, Hrsg., *Overview of Research on the Parallel Computer SNI-KSR at the Leibniz-Rechenzentrum München*, Jgg. 9401, 1994.

[BGR94] H.-J. Bungartz, M. Griebel und U. Rüde. Extrapolation, Combination, and Sparse Grid Techniques for Elliptic Boundary Value Problems. *Computational Methods Appl. Mech. Engrg.*, 116:243–252, 1994.

[BGRZ94] H.-J. Bungartz, M. Griebel, D. Röschke und C. Zenger. Pointwise Convergence of the Combination Technique for the Laplace Equation. *East-West Journal of Numerical Mathematics*, 2:21–45, 1994.

[BPZ] M. Backschat, A. Pfaffinger und C. Zenger. Economic-Based Dynamic Load Distribution In Large Workstation Networks. Erscheint als SFB-Bericht.

[EPZ] 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.

[Fra95] A. Frank. Hierarchische Polynombasen zum Einsatz in der Datenkompression mit Anwendung auf Audiodaten. Diplomarbeit, Institut für Informatik, TU München, 1995.

[GG94] T. Grauschopf und M. Griebel. Parallelization of a Multigrid Algorithm on the KSR1. In M. Brehm und Ch. Schaller, Hrsg., *Overview of Research on the Parallel Computer SNI-KSR at the Leibniz-Rechenzentrum München*, Jgg. 9401, 1994.

[GHZ93] M. Griebel, W. Huber und C. Zenger. A Fast Poisson Solver for Turbulence Simulation on Parallel Computers Using Sparse Grids. In E.H. Hirschel, Hrsg., *Flow Simulation with High-Performance Computers I, Notes on Numerical Fluid Mechanics*, Jgg. 38, Seiten 101–113. Vieweg Verlag, Braunschweig, 1993.

[GN] 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.*

[Gri95] M. Griebel. Parallel Domain-Oriented Multilevel Methods. *SIAM Journal of Scientific Computing: SISC*, 16(5):1105–1125, 1995.

[HHSZ94] W. Huber, R. Hüttl, M. Schneider und C. Zenger. Distributed Numerical Simulation on Workstation Networks. In M. Griebel und C. Zenger, Hrsg., *Numerical Simulation in Science and Engieneering, Proceedings of the FORTWIHR Symposium oh High Performance Scientific Computing in Munich, June 17-18, 1993*, Jgg. 48 of *Notes on Numerical Fluid Mechanics*, Seiten 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.

[Pfa95] A. Pfaffinger. Parallel Communication on Workstation Networks with Complex Topologies. In *Mitteilungen - Gesellschaft für Informatik e. V., Parallel-Algorithmen, -Rechenstrukturen und -Systemsoftware (PARS)*, Jgg. 14, Seiten 227–233, 1995.

[Zen92] C. Zenger. Funktionale Programmierung paralleler Algorithmen für numerische und technisch-wissenschaftliche Anwendungen. In R. Rannacher G. Bader, G. Wittum, Hrsg., *GAMM-Seminar über numerische Algorithmen auf Transputer-Systemen, Heidelberg, 31.5.-1.6.1991.* Teubner, Stuttgart, 1992.