DFG-Antrag auf Gewährung einer Sachbeihilfe für das Projekt

Multi-Logik-Systeme als Basis für heterogene Spezifikation und Entwicklung

Bernd Krieg-Brückner
Hans-Jörg Kreowski
Fachbereich 3, Mathematik/Informatik
Universität Bremen

27. Juli 1999

Inhaltsverzeichnis

1 Allgemeine Angaben

Antrag auf Gewährung einer Sachbeihilfe, Neuantrag

1.1 Antragsteller

Bernd Krieg-Brückner, Prof. Dr. rer. nat.
Professor für Programmiersprachen, Übersetzer und Softwaretechnik
Universität Bremen
Studiengang Informatik des Fachbereichs Mathematik/Informatik

Dienstliche Adresse
Universität Bremen
Fachbereich 3, Mathematik/Informatik
Postfach 33 04 40
28334 Bremen
Tel. (0421)218-3660
Fax (0421)218-3054
E-Mail bkb@informatik.uni-bremen.de

Hans-Jörg Kreowski, Professor Dr.-Ing.
Professor für Theoretische Informatik
Universität Bremen
Studiengang Informatik des Fachbereichs Mathematik/Informatik

Dienstliche Adresse
Universität Bremen
Fachbereich 3, Mathematik/Informatik
Postfach 33 04 40
28334 Bremen
Tel. (0421)218-2956
Fax (0421)218-4322
E-Mail kreo@informatik.uni-bremen.de

1.2 Thema

Multi-Logik-Systeme als Basis für heterogene Spezifikation und Entwicklung

1.3 Kennwort

Multiple

1.4 Fachgebiet und Arbeitsrichtung

Fachgebiet:	Praktische und Theoretische Informatik
Arbeitsrichtung:	Theorie der Programmierung, Semantik, Logik, formale Methoden und Werkzeuge, sichere Systeme, korrekte Programmentwicklung

1.5 Voraussichtliche Gesamtdauer

4 Jahre

1.6 Antragszeitraum

2 Jahre

1.7 Gewünschter Beginn der Förderung

1. 1. 2000

1.8 Zusammenfassung

Formale Methoden sind für die Entwicklung korrekter Software insbesondere in sicherheitskritischen Bereichen bedeutsam. Bei Softwareentwicklungsprojekten werden oft für verschiedene Zwecke mehrere Sprachen und Werkzeuge gleichzeitig in die Entwicklung eingebracht. Um die Wirksamkeit vielfältiger Konzepte und Methoden innerhalb einer Systementwicklung zu gewährleisten, müssen sie semantisch verträglich sein.

Basierend auf der neuen international standardisierten Spezifikationssprache CASL (Common Algebraic Specification Language) soll ein Graph von Logiken entwickelt werden, der Teilsprachen und Erweiterungen von CASL sowie exemplarisch andere Spezifikationssprachen umfasst, mit denen bereits erfolgreich Anwendungen entwickelt wurden (CSP, OBJ, Larch, Z).

Zudem soll eine heterogene Spezifikationssprache entwickelt werden, in der Spezifikationen aus verschiedenen Logiken kombiniert werden können. Die heterogene Spezifikationssprache soll generisch, d.h. parametrisiert mit einem beliebigen Logik-Graphen, entworfen und dann mit dem bereits entwickelten Logik-Graph instantiiert werden. Eine wichtige Fragestellung ist dabei das Verhältnis von Spezifikation-im-Kleinen (einzelne Moduln), Spezifikation-im-Großen (Strukturierung von Spezifikationen und Programmen) und heterogener Struktur (Übersetzungen des Logik-Graphs).

Der Logik-Graph ist zudem die semantische Basis für die heterogene Kombination von Methoden und Werkzeugen. Dabei sollen sowohl Werkzeuge für die heterogene Sprache entwickelt als auch bestehende Werkzeuge für einzelne Sprachen des Logik-Graphs integriert werden, so dass ein Logik-, Methoden- und Werkzeuggraph entsteht.

2 Stand der Forschung, eigene Vorarbeiten

2.1 Stand der Forschung

2.1.1 Formale Spezifikation und formale Methoden

Für den Einsatz formaler Methoden zur Entwicklung sicherer Systeme besteht ein Bedarf an Sprachen zur formalen Spezifikation, um mit Hilfe einer mathematisch orientierten Modellierung und präzisen Definition der Anforderungen Klarheit über die Funktionalität der Systeme zu erhalten. Die verwendeten Spezifikationssprachen müssen durch eine formale, d.h. mathematisch fundierte Semantik abgesichert sein. Damit ist auch formal definiert, was die Korrektheit einer Implementierung bzgl. einer Spezifikation bedeutet. Die Korrektheit kann bewiesen oder durch die Benutzung von Transformationen garantiert werden. Dadurch wird die Entwicklung korrekter und sicherer Systeme mit formalen Methoden unterstützt.

Dem Einsatz und der Entwicklung von Sprachen für ausführbare Spezifikationen bzw. die Programmierung auf sehr hohem Abstraktionsniveau (funktionale bzw. logische Programmiersprachen) kommt strategisch eine besondere Bedeutung zu: Sie erlauben eine frühzeitige Ent-

wicklung von Prototypen. Derartige Sprachen und ihre effiziente Übersetzung sind aber auch ein wichtiger erster Schritt zur Überzeugung des Praktikers, dass der Einsatz formaler Methoden realistisch ist und neben Korrektheit auch Produktivität und Wiederverwendbarkeit fördert.

Es existiert derzeit eine Vielfalt von z.T. geringfügig, z.T. erheblich unterschiedlichen Spezifikationssprachen. Eine Standardisierung ist bisher nur für wenige spezialisierte Sprachen wie VDM [Jon90] oder Z [Spi89, SB93] versucht worden. Keine Sprache erfüllt alle Anforderungen aus der Praxis; dies ist u.U. auch nicht erwünscht, um eine zu große Komplexität zu vermeiden. Diese Erkenntnis führte innerhalb der Common Framework Initiative (CoFI, [Mos97c, CoF]) zu dem Vorhaben, einen Standard zu entwickeln, der eine ganze Familie von Sprachen umfasst, die Einschränkungen und/oder Erweiterungen einer zentralen Sprache, der Common Algebraic Specification Language (CASL, [CoF98]) sind (siehe auch Abschnitt 2.2.5).

2.1.2 Strukturierung von Spezifikationen und Programmen

Es hat sich gezeigt, dass die Strukturierung von Spezifikationen und Programmen (Entwicklung-im-Großen) in der Praxis eine große Rolle spielt, um die Komplexität der Aufgabenstellung beherrschen zu können, indem man sich jeweils auf überschaubare Teilaspekte konzentriert. Bei formalen, insbesondere axiomatischen Spezifikationsmethoden, ist oft die Struktur einer Anforderungsspezifikation wesentlich verschieden von jener der Entwurfsspezifikationen oder fertigen Programme. Man versucht, eine Spezifikation aus (Teilen von) anderen zusammenzusetzen bzw. sie aus abstrahierten Schemata durch Parametrisierung abzuleiten (siehe z.B. CLEAR [BG77], ASL [Wir86], OBJ3 [GW88], Larch [GHG+93]). So entsteht ein Geflecht von Vererbungsrelationen der Wiederverwendung, das dem aus objekt-orientierten Sprachen ähnlich ist, hier aber sehr viel stärker semantische Verwandtschaften repräsentiert.

2.1.3 Logiken und Institutionen

Eine Spezifikationssprache umfaßt syntaktische Elemente wie Signaturen, Axiome und Beweis-Kalküle sowie semantische Elemente wie Modelle und semantische Erfülltheit von Axiomen. Damit liegt einer Spezifikationssprache eine Institution im Sinne von Burstall und Goguen [GB92] bzw. eine Logik im Sinne von Meseguer [Mes89] zugrunde. Zentral ist die Aussage, dass Wahrheit und Beweisbarkeit invariant gegenüber Notationswechsel sind. Damit ist es möglich, die Strukturierung von Spezifikationen-im-Großen unabhängig von der zugrunde liegenden Logik zu betrachten [ST88, ST86, Mos98c].

2.1.4 Heterogene Kombination von Sprachen, Methoden und Werkzeugen

Bei dem Einsatz formaler Methoden zur Entwicklung korrekter Software, insbesondere bei großen Softwareentwicklungsprojekten, an denen viele Personen und möglicherweise auch mehrere Entwicklungsteams beteiligt sind, kann man nicht davon ausgehen, dass eine einzige formale Spezifikationssprache mit einer einheitlichen Entwicklungsumgebung zum Einsatz kommt. Für verschiedene Zwecke werden von unterschiedlichen Beteiligten mehrere Sprachen und Werkzeuge gleichzeitig in die Entwicklung eingebracht. Um die Wirksamkeit vielfältiger Konzepte und Methoden innerhalb einer Systementwicklung zu gewährleisten, müssen sie semantisch verträglich sein. Oft sind mehrere verschiedene Logiken und Sprachen gleichzeitig oder in verschiedenen Phasen der Entwicklung im Spiel: heterogene Spezifikationen ("horizontal") bzw. heterogene Entwicklungen ("vertikal"). So sind die Kombination von einzelnen Systemteilen mit formalen Spezifikationen in unterschiedlichen Logiken oder die Bearbeitung einer Spezifikation mit verschiedenen Werkzeugen, die nur für eine eingeschränkte Logik anwendbar sind oder verschiedene Notationen verwenden, sehr reale praktische Probleme.

Eine zentrale Aufgabe zukünftiger Grundlagenforschung im Bereich formaler Methoden ist daher zu untersuchen, wie verschiedene Methoden, Sprachen und Logiken und die darauf beruhenden Werkzeuge koexistieren und welche Wechselwirkungen beim gemeinsamen Einsatz auftreten können (vgl. auch Abschnitt 2.2.8).

Heterogene Spezifikationssprachen basieren auf einem Begriff der Abbildung zwischen Logiken [GB92, Mes89]. Es haben sich zwei Typen von Abbildungen herausgebildet: einerseits Abbildungen, die eine Logik in einer anderen repräsentieren oder kodieren, andererseits Abbildungen, die komplexere Logiken auf ihre Bestandteile projizieren [Tar96]. Die Technik des "Borrowing" [CM97] erlaubt, Teile einer Logik wie Modelltheorie oder Beweiskalkül entlang einer Repräsentationen zu liften. Dies ist entscheidend für die Wiederverwendung von Werkzeugen.

Die Forschung über (semantisch fundierte) heterogene Spezifikationssprachen bewegt sich überwiegend noch im Bereich der theoretischen Grundlagen – es existieren nur wenige Publikationen über dieses Gebiet [AC94, Dia98, Bor99, Tar00, BCL96, CBL99]. Erste praktische Anwendung war die Entwicklung der heterogenen Sprache CafeOBJ [DF96], basierend auf der Theorie von [Dia98]. CafeOBJ basiert auf einem Würfel von acht Logiken. Allerdings sind diese Logiken eng an bedingten Gleichungslogiken orientiert, während für problemorientierte Anforderungsspezifikationen oft mächtigere Logiken erforderlich sind. CafeOBJ beinhaltet auch kein handhabbares Konzept von logik-unabhängiger Strukturierung-im-Großen, wie es etwa in [Mos00] entwickelt wird. Zudem geht die Theorie von [Dia98] von Logik-Projektionen zwischen den beteiligten Logiken aus, während unsere Auffassung sich eher mit der von Tarlecki [Tar00] deckt, dass auch Logik-Repräsentationen eine entscheidende Rolle spielen, insbesondere, wenn ein größerer Graph mit heterogeneren und komplexeren Logiken als bei CafeOBJ verwendet wird (siehe Logik-Graph in Abschnitt 2.2.7).

[Tar00] gibt – als bisher einzige umfassendere Untersuchung der mathematischen Grundlagen einer heterogenen Spezifikationssprache – eine gute Richtung vor, die im beantragten Projekt sowohl theoretisch weiterentwickelt als auch – in Wechselwirkung damit – praktisch nutzbar gemacht werden soll.

2.1.5 DFG-Schwerpunktprogramm

Einige Projekte des DFG-Schwerpunktprogramms "Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen" bearbeiten ebenfalls die Fragestellung von Multi-Logik-Systemen. Der Schwerpunkt des hier beantragten Projekts liegt, anders als im Schwerpunktprogramm, nicht auf der ingenieurwissenschaftlichen Anwendung, sondern auf Erforschung der semantischen Grundlagen von Multi-Logik-Systemen und einer semantisch fundierten Integration von Werkzeugen.

2.2 Eigene Vorarbeiten

2.2.1 Spezifikationssprache SPECTRAL

Die Bremer Entwicklung der Spezifikationssprache SPECTRAL [KBS91] fußt einerseits auf ExtendedML [ST90], anderseits auf langjährigen Erfahrungen beim Sprachentwurf (Ada [IBH+79], PAnndA-S im ESPRIT Projekt PROSPECTRA [HKB93]), algebraische Spezifikation im Rahmen der ESPRIT WG COMPASS (s.u.). Einige Konzepte von SPECTRAL sind in den Entwurf der Sprache CASL (siehe 2.2.5) eingeflossen.

Eigentlich handelt es sich bei SPECTRAL um eine Sprachfamilie, da die einzelnen Moduln prinzipiell in verschiedenen Logiken formuliert sein können. Damit stellt sich das Problem der theoretischen Grundlagen für die heterogene Kombination (2.1.4 und s.u.).

2.2.2 Ausführbare Spezifikationen als logische Programme höherer Ordnung

(Programm-)Spezifikationen sind Formulierungen abstrakter Anforderungen an ausführbare Programme. Im allgemeinen ist die Entwicklung von Programmen aus Spezifikationen sehr aufwendig und schwer zu automatisieren. Im DFG-Projekt EXSPEC (Dr. Z. Qian, AG Krieg-Brückner, vgl. [QW92, QW94, Qia94, Qia95, Wan94]) wurde untersucht, wie sich die in den letzten Jahren neu entwickelten Spezifikationsparadigmen erster und höherer Ordnung auf ihre ausführbaren Kerne mit relativ gutem operationalen Verhalten beschränken und solche Kerne zu einem logischen Programmierparadigma mit Nebenbedingungen ("Constraints") vereinigen lassen. Der Ansatz unterstützt *rapid prototyping* von Spezifikationen und kann die Entwurfsphase von Programmen mit formalen Methoden deutlich verkürzen.

2.2.3 Comprehensive Algebraic Approach to System Specification and Development

An der ESPRIT Basic Research Working Group COMPASS (a COMPrehensive Algebraic Approach to System Specification and development, 1989-96) mit insgesamt 20 europäischen Partnern waren Prof. Kreowski und Prof. Krieg-Brückner (Koordinator) beteiligt. Diese Working Group brachte die europäischen Wissenschaftler auf dem Gebiet der eigenschaftsorientierten Spezifikationen zusammen und war ein sehr gutes Forum, um theoretische Fragestellungen zu diskutieren und Ergebnisse zu verbreiten [BKL$^+$91, CGK$^+$97, GC95].

2.2.4 Foundations of Systems Specification

Prof. Kreowski ist Initiator und ehemaliger Sprecher der Working Group 1.3 (Foundations of Systems Specification) der International Federation of Information Processing (IFIP), der Dachorganisation aller wissenschaftlichen Informatik-Fachorganisationen; Prof. Krieg-Brückner ist Mitglied. Hier wird die Arbeit von COMPASS fortgesetzt.

Ein wichtiges Arbeitsergebnis der Working Group ist der State-of-the-Art-Report über Algebraic Foundations of Systems Specification [AKKB99], der von E. Astesiano, H.-J. Kreowski und B. Krieg-Brückner herausgegeben wurde und in Kürze erscheint. Der Band enthält drei Kapitel von Bremer Koautoren [CMR99, EK99, BKB99], deren Thematik in engem Zusammenhang zum beantragten Vorhaben steht.

2.2.5 Standardisierung einer Familie von Spezifikationssprachen

Im Rahmen der Working Group COMPASS und auf Initiative der IFIP Working Group 1.3 wurde 1995 beschlossen, ein einheitliches Rahmenwerk für Spezifikationssprachen zu entwerfen und zur internationalen Standardisierung vorzuschlagen (die sog. "Common Framework Initiative for Algebraic Specification and Development" (CoFI)). Diese Entwicklung ist von großer Bedeutung (vielleicht vergleichbar mit der Standardisierung von Algol 60), da auch außerhalb der derzeitigen WG andere internationale Arbeitsgruppen einbezogen werden, um eine maximale Verbreitung zu erreichen. Seit 1998 wird die CoFI Working Group mit Reisemitteln durch die Europäische Union gefördert.

Die wesentliche Idee ist, nicht eine allumfassende Sprache, sondern eine Familie von Spezifikationssprachen zu entwickeln, die als Teilsprachen der Basissprache oder Erweiterungen davon in Beziehung zueinander stehen. Entscheidend ist, dass auf diese Weise existierende Sprachen und vor allem auch existierende Werkzeuge zueinander in Relation gesetzt und, nach geeigneter Übersetzung, integriert verwendet werden können. Nur ein internationaler Standard wird auf Dauer bei Anwendungen in der Industrie erfolgreich sein und zu weitverbreiteten Werkzeugen und Entwicklungsumgebungen führen.

Der Entwurf der zentralen Sprache dieser Familie, der "Common Algebraic Specification Language" (CASL) [CoF98] sowie die Definition von Teilsprachen [Mos97a] konnten 1998 abgeschlossen werden. Als nächstes sollen *higher-order*, reaktive und objekt-orientierte Erweiterungen von CASL entwickelt werden.

Prof. Krieg-Brückner hat innerhalb von CoFI als Leiter der CoFI-Sprachentwurfsgruppe eine zentrale Verantwortung. Die Arbeitsgruppen der Antragsteller haben an der Semantik von CASL mitgearbeitet [CoF99] und mit zahlreichen internen Papieren [Mos97a, Mos98e, HKBM98, RM99b, Mos97b, Mos98b, Mos98c, Mos98f, RM99a, MR99, Mos98d] und Veröffentlichungen [CHKBM97, Mos98a, MKK98, ABKB$^+$] zur Entwicklung von CASL beigetragen. Für die Erweiterung von CASL um Logik höherer Stufe wurde ein erster Entwurf vorgelegt [HKBM98], der allerdings noch keine Polymorphie umfasst.

2.2.6 Nebenläufige und objekt-orientierte Sprachen

Im DFG-Projekt COOFL (Prof. Krieg-Brückner, Dr. Markus Roggenbach) wird untersucht, wie sich funktionale, nebenläufige und objekt-orientierte Spezifikation bzw. Programmierung miteinander vereinen lassen. Diese Konzepte sollen sowohl auf der Programmiersprachenebene, als auch auf der Spezifikationssprachenebene (basierend auf der Spezifikationssprache CASL) kombiniert werden. Die hierbei (oder auch in anderen Arbeitsgruppen) entstehenden (nebenläufigen bzw. objekt-orientierten) Erweiterungen von CASL sollen als Knoten in den zu untersuchenden Graphen von Logiken und Sprachen aufgenommen werden.

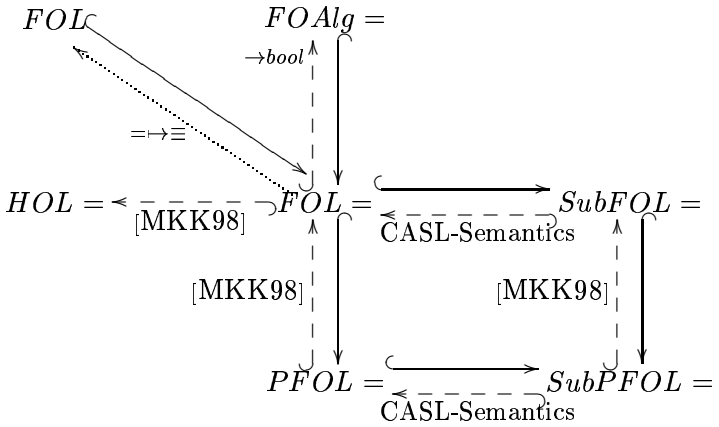
2.2.7 Logik-Graph und theoretische Grundlagen

Zu der konkreten Untersuchung von Logiken im Gebiet der algebraischen Spezifikation einschließlich der Anwendung kategorieller Methoden existieren umfangreiche Forschungsarbeiten der AG Kreowski, besonders des für die Mitarbeit im Projekt vorgesehenen Mitarbeiters, Dr. Till Mossakowski. Während der Gewährung eines Stipendiums der Studienstiftung des deutschen Volkes entstanden die Vorarbeiten [KM95, Mos96b, Mos96a, Mos96c, CMR99]. In diesen Arbeiten werden verschiedene Logiken exemplarisch miteinander verglichen, Anforderungen an Abbildungsbegriffe formuliert und derartige Abbildungsbegriffe (insbesondere für Repräsentationen) eingeführt. Ferner wurde an der Kombination von Logiken und dem Zusammenspiel zwischen Logik-Repräsentationen und -Projektionen gearbeitet [Mos96d, MTP97, MTP98, SSCM00]. Auf diese Weise wurde bereits ein erster Logik-Graph (vgl. auch [Mos97a, Mos99]) entwickelt, in dem verschiedene Logiken systematisch miteinander in Beziehung gesetzt werden und der eine Navigation beim Übergang in eine andere Logik bzw. Sprache der Familie erlaubt. Jeder Übergang entspricht einer korrekten Übersetzung und erlaubt dadurch u.a., Werkzeuge für die Ziellogik auch für Formeln der Quelllogik zu verwenden.

Diese Technik wurde auch bei der Definition von Teilsprachen für CASL [Mos97a] übernommen und ist die Grundlage für künftige Werkzeugentwicklungen. Abbildungen 1 und 2 zeigen zwei Logik-Graphen, die in diesem Zusammenhang entstanden sind und die in dem zu entwickelnden Logik-Graph Verwendung finden sollen.

2.2.8 UniForM Workbench

Die Software-Entwicklungsumgebung der Zukunft muss eine Kombination von Werkzeugen anbieten, wobei jedes Werkzeug einen scharf eingegrenzten Aufgabenbereich effizient bearbeitet bzw. für bestimmte Typen von Zielsystemen besonders geeignet ist. Innerhalb des inzwischen abgeschlossenen BMBF-Projekts UniForM Workbench (AG Krieg-Brückner zusammen mit akademischen und industriellen Kooperationspartnern [Kri99, KPO$^+$99]) wurde exemplarisch gezeigt, wie Methoden in logisch konsistenter Weise kombiniert und Transformationswerkzeuge



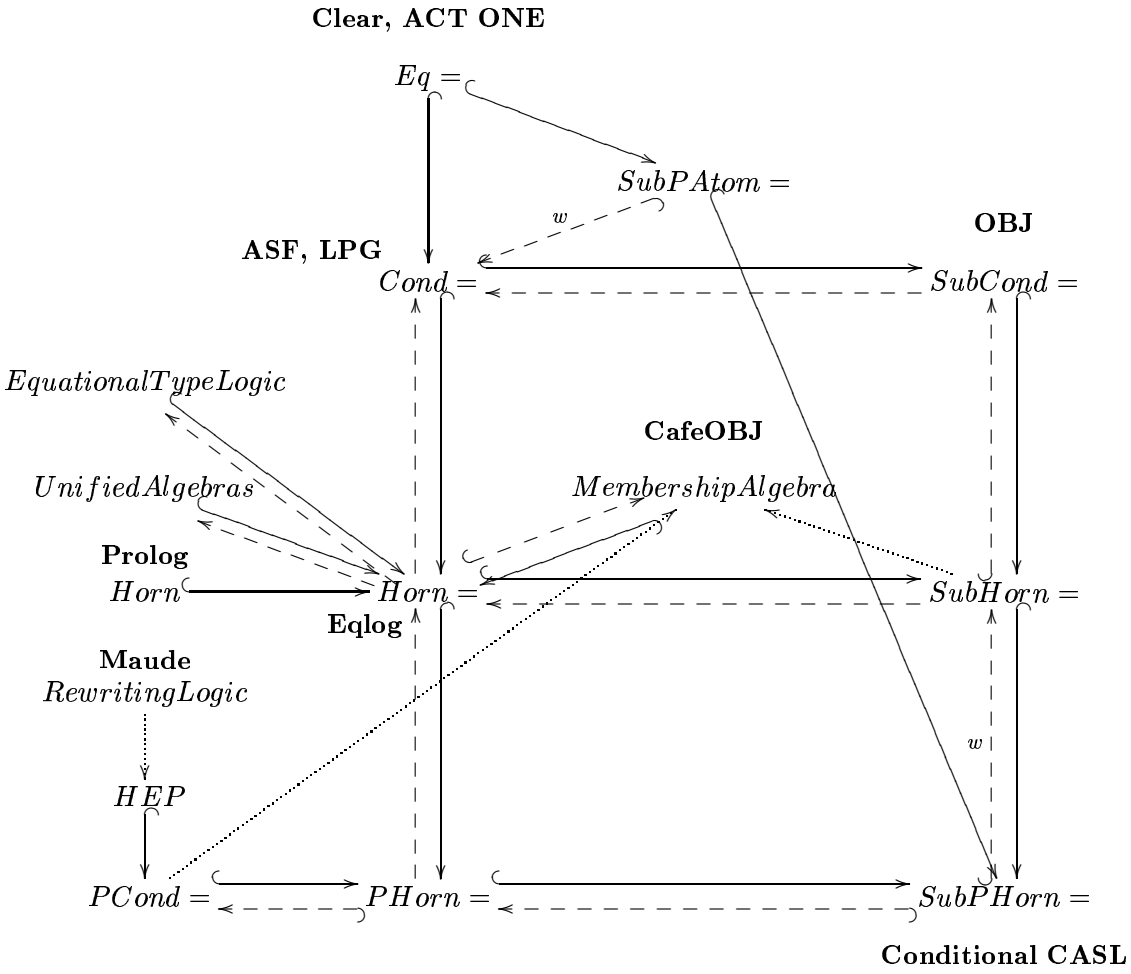
Legende für die Benennung der Knoten (Logiken):

FOL	Logik erster Stufe
HOL	Logik höherer Stufe
=	mit Gleichheit
P	mit Partialität
Sub	mit Subsorten

Legende für die Kanten:

⋯→	1. Wiederverwendung von Theorembeweisern
– – →	2. Bewahrung freier Konstruktionen + 1.
⊂– – →	3. Bewahrung loser Semantik + 1. + 2.
⊂⟶	4. Syntaktische Teilsprache + 1. + 2. + 3.

Abbildung 1: Beispiel-Logikgraph aus [Mos97a]: Logiken erster Stufe



Legende für die Benennung der Knoten (Logiken):

Eq	nur Gleichungen
Atom	nur atomare Formeln
Cond	bedingte Gleichungen
Horn	Hornformeln
=	mit Gleichheit
P	mit Partialität
Sub	mit Subsorten
HEP	Reichels HEP-Theorien [Rei87]

Legende für die Kanten: siehe Abb. 1

Abbildung 2: Beispiel-Logikgraph aus [Mos97a]: Horn-Logiken

korrekt entwickelt werden können. Die industrielle Nutzbarkeit der UniForM Workbench wurde anhand einer Fallstudie aus der industriellen Praxis nachgewiesen, der Entwicklung einer dezentralen Steuerungs- und Sicherungseinrichtung für Regionalbahnen.

Gegenüber der Neuentwicklung von eigenen Werkzeugen für jede Spezifikationssprache bietet eine Codierung bzw. Repräsentation in einer Logik wie Isabelle/HOL [Pau90, Pau94] den Vorteil der Wiederverwendbarkeit, Kombinierbarkeit und Integrierbarkeit von Logiken und Werkzeugen. Im Rahmen des UniForM-Projekts wurden die Sprachen CSP [TW97], Z [LKK+98] und CASL [MKK98] in Isabelle/HOL codiert. Das Problem der Kombination wurde aber erst exemplarisch für CSP und Z gelöst [Fis98]. Die UniForM Workbench bietet eine sehr gute Plattform für Experimente der Logik- und Werkzeug-Kombination.

Das Arbeiten mit den Logik-Codierungen erleichtert ein generisches graphischen Benutzerinterface [KLMW97], das eine Darstellung von Theorien, Beweiszielen, Regelmengen für Simplifikations- und Tableaux-Beweise und natürlich auch bewiesenen Theoremen als Icons auf dem Bildschirm ermöglicht.

2.2.9 GRACE

Seit einiger Zeit entwickeln Mitglieder der AG von Prof. Kreowski zusammen mit Forschungsgruppen in Berlin, Erlangen, München, Oldenburg und Paderborn die graph- und regelzentrierte Spezifikationssprache GRACE, deren Ziel ist, heterogene Spezifikationen mit verschiedenen Graphtransformationsansätzen zu unterstützen (siehe [KKS97, AEH+99, KK99]). Die Problematik ist ähnlich gelagert wie die beim Logik-Graphen, weil Graphtransformationsansätze als spezielle Rewrite-Logiken aufgefaßt werden können.

3 Ziele und Arbeitsprogramm

3.1 Ziele

Mit der Existenz verschiedener Logiken in heterogenen Spezifikationen bzw. heterogenen Entwicklungen entsteht die Forderung nach einem logischen Rahmenwerk, in dem verschiedene Logiken in semantisch fundierter Weise heterogen kombiniert werden können. Zusätzlich soll die (heterogene) Strukturierung von Spezifikationen auch zu einer davon abgeleiteten Strukturierung von Beweisen führen, um Einzelbeweise möglichst getrennt voneinander führen und wiederverwenden zu können (analog zur getrennten Übersetzung). Zusammengefaßt entsteht also der Bedarf nach Multi-Logik-Systemen und nach geeigneten Abbildungen zwischen Logiken zum Notations- bzw. Semantik- oder Paradigmen-Wechsel.

Die Vorarbeiten in diesem Bereich sollen in der Theorie weiter ausgearbeitet und, praktisch angewandt, ausgehend von der zentralen Sprache CASL auf die gesamte CoFI-Sprachfamilie und auch auf andere gängigen Spezifikationssprachen wie Larch, OBJ, Z ausgedehnt werden. Dadurch soll der für einige Logiken bereits existierende Logik-Graph (siehe die Graphen in Abschnitt 2.2.7) erweitert werden. Die Logiken, die CASL, den Teilsprachen und den nebenläufigen und objekt-orientierten Erweiterungen (insbesondere der Kombination von CASL+CSP) sowie Larch, OBJ und Z zugrunde liegen, sollen systematisch zueinander in Beziehung gesetzt werden und Methoden ihrer heterogenen Kombination in einer Spezifikationssprache entwickelt werden. Dies ist erstens nützlich, um Spezifikationsprobleme in einer Logik angemessener Mächtigkeit formulieren zu können (bei zu mächtigen Logiken werden die Software-Werkzeuge evtl. schwächer). Zweitens wird dadurch ermöglicht, Spezifikationsprobleme aus einer gegebenen Logik in eine andere, mit Software-Werkzeugen unterstützte Logik zu übersetzen. Drittens können Spezifikationssprachen durch Übersetzungen kombiniert werden, und ein Netzwerk von Spezifikationssprachen und Übersetzern dazwischen kann selbst als Grundlage heterogener Spezifikation und Entwicklung begriffen werden.

3.2 Arbeitsprogramm

Das Arbeitsprogramm umfasst den Antragszeitraum von vier Jahren und ist in zwei Teile gegliedert: die konzeptionelle Ebene und die Werkzeug-Ebene, die jeweils von einem Mitarbeiter abgedeckt werden sollen. Beide Teile sind aufeinander abgestimmt: die im konzeptionellen Bereich entwickelten Resultate sollen direkt in die Werkzeugentwicklung einfließen. Es wurde darauf geachtet, dass auf der Werkzeug-Ebene unmittelbar mit der Arbeit begonnen werden kann, indem zunächst Werkzeuge auf Grundlage bereits ausgearbeiteter Konzepte entwickelt werden (vgl. dazu den Zeitplan).

Nach zwei Jahren sollen auf der konzeptionellen Ebene die Arbeiten zu den Übersetzungen (3.2.1, 3.2.2, 3.2.3) und deren Eigenschaften (3.2.4) abgeschlossen sein. Auf der Werkzeugebene sollen die logik-unabhängigen aber homogenen (da für eine beliebige aber feste Logik ausgelegten) Werkzeuge (3.2.8, 3.2.9) fertiggestellt sowie mit der Implementierung der Übersetzungen 3.2.10) begonnen worden sein.

Arbeitsprogramm: konzeptionelle Ebene

Auf der konzeptionellen Ebene geht es einerseits um die Entwicklung von konkreten Übersetzungen zwischen Logiken und Spezifikationssprachen. Untersucht werden sollen mehrere gängige und in der Praxis bereits eingesetzte Spezifikationssprachen in ihrem Verhältnis zu CASL bzw. CASL-Teilsprachen.

Andererseits soll es um die Untersuchung der Übersetzungen auf bestimmte Eigenschaften gehen, die für die Wiederverwendung von Spezifikationsbibliotheken und Werkzeugen und für die Entwicklung einer heterogenen Spezifikationssprache wichtig sind.

Zudem soll auch das Problem der Kompatibilität von CASL mit seinen Erweiterungen untersucht werden. Hierfür kann neben CASL höherer Stufe [HKBM98] ggf. auch auf eine oder mehrere reaktive und objekt-orientierte Erweiterungen von CASL zurückgegriffen werden, die von der *COFI task group* "Reaktive Systeme" entwickelt werden.

3.2.1 Logik-Graph

Der Logik-Graph soll CASL, CASL höherer Stufe (HOCASL), deren relevante Teilsprachen, die (derzeit noch in der Entwicklung befindlichen, siehe auch Abschnitt 2.2.6) nebenläufigen und objekt-orientierten Erweiterungen von CASL sowie alle wichtigen Repräsentationen und Projektionen zwischen diesen Logiken enthalten.

Die Kanten des Logik-Graphen sind entweder Logik-Repräsentationen, d.h. Codierungen einer Logik in eine andere, oder Logik-Projektionen auf eine Teillogik (Genaueres siehe Abschnitt 3.2.4). Der Logik-Graph umfasst damit nur die Ebene der Spezifikation-im-Kleinen — die CASL-Konstrukte zur Strukturierung-im-Großen sind logik-unabhängig.

Bei der Entwicklung von Repräsentationen müssen die Interaktionen der verschiedenen Konzepte (sowohl der statischen Konzepte wie Untersorten oder Partialität, als auch der dynamischen Konzepte wie parallele Kombination von Prozessen oder nichtdeterministische Auswahl) und die jeweilige Ausdrucksmächtigkeit (Gleichungslogik, Logik erster Stufe, Logik höherer Stufe) der Teilsprache aufeinander abgestimmt abgestimmt werden. Die Spannung besteht hier zwischen Einfachheit der Codierung und der Einhaltung der Restriktionen der jeweiligen Teilsprache.

Neben den CoFI-Sprachen sollen auch die Logiken anderer Spezifikationssprachen in den Logik-Graph mit einbezogen sowie dazu korrespondierende Teilsprachen oder Erweiterungen von CASL ausgezeichnet werden. Inwieweit darüber hinaus ein eigener Spezifikationssprachen-Graph sinnvoll ist, der auch die Übersetzungen von Strukturierungskonzepten-im-Großen umfasst, muss untersucht werden.

3.2.2 Übersetzungen von und nach anderen Spezifikationssprachen

Die Übersetzung einer Spezifikationssprache in eine Sprache des Logik-Graphs hat zum vorrangigen Ziel, bestehende Spezifikationsbibliotheken wiederverwenden zu können. Daher sollen die Spezifikationssprachen mit den umfangreichsten Bibliotheken und Anwendungen ausgewählt und in eine geeignete Sprache aus dem Logik-Graph übersetzt werden. Neben der semantischen Korrektheit ist für diesen Typ von Übersetzungen ein wichtiges pragmatisches Kriterium die Lesbarkeit der übersetzten Bibliotheken. Ggf. müssen die Übersetzungen entsprechend fein auf eine lesbare Ausgabe abgestimmt werden.

Die umgekehrte Richtung, also die Übersetzung von Sprachen des Logik-Graphs in andere Spezifikationssprachen hat die Wiederverwendung von Werkzeugen zum Ziel. Zwar könnte theoretisch für jede Logik im Graph ein eigener Kalkül und eigene Werkzeuge entwickelt werden. Praktisch ist dies aber zu arbeitsaufwendig; die Wiederverwendung von Werkzeugen ist unumgänglich. Dabei können Übersetzungen in anderen Sprachen auch mit den Kanten aus dem Logik-Graph komponiert werden. Z.B. kann eine Übersetzung von OBJ3 in das totale Hornklausel-Fragment von CASL kombiniert werden mit einer Übersetzung innerhalb von CASL-Teilsprachen, die Untersorten eliminiert (vgl. Abb. 2), um so z.B. bedingte Termersetzungs-Systeme nutzen zu können.

Für diesen Typ von Übersetzungen ist das pragmatische Kriterium, wie gut man entlang der Übersetzung Beweise führen kann. Optimal wäre eine Umkehrbarkeit der Formelübersetzung, um Zwischenschritte (*subgoals*) eines Beweises in der Quelllogik darstellen zu können. Typischerweise sind die Formelübersetzungen jedoch nicht surjektiv, so dass keine Umkehrung existiert. Oft kann jedoch eine Umkehrung gefunden werden, die als Ziel eine geringfügige Erweiterung der ursprünglichen Quelllogik besitzt, so dass die Zwischenschritte lesbarer dargestellt werden können als bei einer Anzeige in der Ziellogik. Beispielsweise kann die Übersetzung von CASL nach Logik erster bzw. höherer Stufe [MKK98] umgekehrt werden, indem man die CASL zugrunde liegende Logik um Variablen erweitert, die undefiniert sein können.

Die zugrunde liegenden Logiken der im folgenden beschriebenen Spezifikationssprachen sollen in den Logik-Graph (und damit auch in die weiter unten entwickelte heterogene Spezifikationssprache) integriert werden. Die im folgenden skizzierten Übersetzungen betreffen aber zudem auch noch die Übersetzung der Sprachkonstrukte zur Spezifikation-im-Großen.

CASL und CSP Es ist zu erwarten, dass zumindest eine der nebenläufigen Erweiterungen von CASL auf einer Kombination von CASL mit dem von Tony Hoare entwickelten *Communicating sequential processes* (CSP) [Hoa85] basiert. In diesem Fall sollen die Einbettungen von CASL und CSP in CASL+CSP sowie die Projektionen von CASL+CSP auf CASL und CSP auf ihre Eigenschaften, insbesondere im Hinblick auf die heterogene Sprache, untersucht werden. CSP bietet sich als Paradigma der Nebenläufigkeit deshalb gut an, weil in Bremen bereits eine Codierung von CSP in Isabelle/HOL entwickelt wurde [TW97].

Von der *CoFI task group* "Reaktive Systeme" ist in jedem Fall mindestens eine nebenläufige Erweiterung von CASL zu erwarten. Falls es keine auf CSP basierende Erweiterung gibt, muss ggf. CSP durch ein anderes Paradigma der Nebenläufigkeit ersetzt werden.

OBJ3 und CaféOBJ OBJ3 [GW88, GMW$^+$] ist eine auf Gleichungstheorien mit Untersorten basierende Mischung aus Spezifikations- und High-Level-Programmiersprache, die aufgrund ihrer guten Ausführbarkeit für viele Anwendungen vor allem im Bereich des *rapid prototyping* benutzt wurde. CaféOBJ [DF98, DF96] ist die Nachfolgesprache von OBJ3, die auch objekt-orientierte (über *hidden sorted algebra*) und reaktive (über *rewriting logic*) Spezifikationen unterstützt.

Zwischen OBJ3 und einer geeigneten Teilsprachen von CASL sollen Übersetzungen konstruiert werden. Für die Spezifikation-im-Kleinen geht es dabei hauptsächlich um den Vergleich

der unterschiedlichen Untersorten-Konzepte von OBJ3 und CASL, basierend auf der Vorarbeit [Mos99]. Die Strukturierungskonzepte von OBJ3-im-Großen unterscheiden sich von den CASL-Strukturierungskonzepten, da OBJ3 Namen nach ihrem Ursprung unterscheidet, während CASL dem *same name – same thing*-Prinzip folgt. Hier muss die Übersetzung geeignete Umbenenungen vornehmen, um die Semantik zu erhalten.

Die Übersetzungen zwischen CASL und OBJ3 können in einem weiteren Schritt zu Übersetzungen zwischen Erweiterungen von CASL (z.B. CASL+CSP) und CaféOBJ erweitert werden.

Larch Larch [GHG+93] ist eine algebraische Spezifikationssprache, deutlich mächtiger als OBJ3: Während OBJ3 und CaféOBJ3 auf bedingter Gleichungslogik basieren, erlaubt Larch Logik erster Stufe. Diese Mächtigkeit ermöglicht es, dass Larch zur formalen Spezifikation und Verifikation von Software eingesetzt wird. Es gibt Interface-Sprachen zu vielen Programmiersprachen (C, C++, Modula 3, Smalltalk, Standard-ML, CORBA); deswegen ist eine Querübersetzbarkeit zwischen CASL und Larch besonders interessant.

Auf der Ebene der Spezifikation-im-Kleinen geht es hauptsächlich um die Ausweisung einer geeigneten Teilsprache von CASL, die dann relativ eng zu Larch korrespondiert. Für die Spezifikation-im-Großen sind die Konstrukte von Larch verglichen mit denen von CASL sehr einfach, so dass eine Übersetzung von CASL nach Larch etwas trickreich arbeiten oder aber die Verwendung der CASL-Konstrukte einschränken muss.

Z Z [Spi89] ist eine Sprache, die viel zur Modellierung von Softwareprojekten eingesetzt wird. Anders als bei CASL oder Larch wird nicht eine Klasse von Modellen spezifiziert (lose Semantik), sondern ein ganz bestimmtes mengentheoretisches Modell. Allerdings kann man auch in Z Schemata hinschreiben, die abstrakte Eigenschaften spezifizieren, die also zu Spezifikationen mit loser Semantik korrespondieren. Dies kann bei der Übersetzung von CASL nach Z ausgenutzt werden. Schwierigkeiten wird hier die Behandlung der Sorten machen, die in Z immer als Schema-Parameter auftauchen müssen.

Die Hauptschwierigkeit bei der Übersetzung von Z nach CASL ist dagegen die Modellierung der Mengentheorie. Den Ansatz von Kirchner und Mosses [KM] finden wir nicht so interessant, da er auf Hornklauseln beschränkt ist und vor allem das CASL-Typsystem nicht nutzt – es wäre ein eigener Typcheck nötig. Wir favorisieren eine der beiden folgenden Möglichkeiten:

- Übersetzung von Z nach um Axiomenschemata erweitertes CASL. Generische Typen in Z werden auf parametrisierte Spezifikationen abgebildet, und Mengenkomprehensionen auf lokale Definitionen von Mengen (also eine Art λ-Lifting).
- Übersetzung von Z nach CASL höherer Stufe. Die Potenzmengenbildung entspricht Prädikatstypen höherer Stufe, und die Komprehension der λ-Abstraktion für Prädikate.

Generell bleibt das Problem, dass Z auf ungetypter Mengentheorie basiert und die Typen lediglich ein Hilfsmittel sind. Ggf. muss man sich auf eine Teilsprache von Z beschränken, um eine semantisch korrekte Übersetzung zu erhalten.

3.2.3 CASL-Teilsprachen

Ein erster Entwurf einer einheitlichen Definition von und Notation für eine ganze Reihe von CASL-Teilsprachen existiert bereits (siehe [Mos97a] und Abb. 1 und 2). Diese Teilsprachen sollen als Quell- und Zielsprache für Übersetzungen von und nach anderen Spezifikationssprachen sowie auch innerhalb des Logik-Graphs der CASL-Teilsprachen und -Erweiterungen verwendet werden. Es kann aber passieren, dass während der Arbeit am Logik-Graph und an den Übersetzungen neue Teilsprachen definiert werden müssen, um eine möglichst genaue Korrespondenz zwischen

einer gegebenen Spezifikationssprache und der jeweiligen CASL-Teilsprache zu erhalten. Wichtig ist eine möglichst syntaktische Charakterisierbarkeit der Teilsprachen, um sie für Werkzeuge leicht erkennbar zu machen.

3.2.4 Eigenschaften von Repräsentationen und Projektionen

Die Übersetzungen zwischen Logiken sollen in beiden Richtungen nicht nur syntaktische Übersetzungen, sondern auch semantisch fundiert sein. Technisch bedeutet dies für die Übersetzung der zugrunde liegenden Logiken eine Realisierung als Logik-Repräsentationen (*map of institution* [Mes89], *institution representation* [Tar96]).

Eine wichtige Eigenschaft von Logik-Repräsentationen ist die Surjektivität der Modellübersetzung, auf der die *borrowing*-Technik [CM97] basiert: ein Theorembeweiser für die Ziellogik kann auch für die Quelllogik der Logik-Repräsentation verwendet werden (mit dem Erhalt der Korrektheit und Vollständigkeit). Für die Übersetzung von CASL nach Logik erster bzw. höherer Stufe [MKK98] wurde dies bereits realisiert. Die *borrowing*-Technik wird auch für die Codierung von Logiken des Logik-Graphs in Isabelle/HOL benutzt. (Eine Codierung ist technisch nichts anderes als eine Logik-Repräsentation.)

Eine weitere wichtige Eigenschaft der für die Übersetzungen benutzten Logik-Repräsentationen ist die Vertauschbarkeit von CASL-Strukturierungskonstrukten mit der Übersetzung entlang einer Repräsentation. Die Ergebnisse von [Bor99] sind hier vielversprechend, decken aber nur Logik-Repräsentationen (keine Projektionen) und auch noch nicht alle Konstrukte von CASL-im-Großen ab. So ist z. B. die Kompatibilität mit dem Konstrukt zur freien Erzeugung von Datentypen eine nicht-triviale Frage (auch im Falle der Einbettung von CASL nach CASL höherer Stufe). Vermutlich müssen die Anforderungen an den Repräsentationsbegriff noch verschärft werden, um eine Vertauschbarkeit zu erreichen.

Es ist zu erwarten, dass sich eine Hierarchie von Bedingungen ergibt, die verschieden genaue Repräsentation bzw. verschieden starken Erhalt semantischer bzw. beweistheoretischer Eigenschaften bedeuten.

Vergleichbares gilt für die Projektionen, die vor allem als Projektionen auf Teilsprachen vorkommen. Eine Projektion ist technisch ein *institution morphism* [GB92]. Ein wichtiges praktisches Problem ist hier die Entwicklung einer Standard-Bibliothek von Spezifikationen [RM99a, MR99], die zwecks Kompaktheit in einer mächtigen Sprache formuliert ist, die aber auf möglichst uniforme Weise auch auf Teilsprachen projizierbar oder zumindest mit Teilsprachen-Bibliotheken verträglich sein soll. Hier ist die Frage interessant, ob eine Spezifikation ohne Informationsverlust entlang einer Projektion auf eine Teilsprache projiziert werden kann. Die Eigenschaft "ohne Informationsverlust" soll in logik-unabhängiger Weise formuliert werden.

Die Repräsentationen und Projektionen des Logik-Graphs sollen auf die oben skizzierten Eigenschaften überprüft und ggf. so modifiziert werden, dass sie die Eigenschaften tatsächlich haben.

3.2.5 Generische heterogene Spezifikationssprache

Die generische heterogene Spezifikationssprache soll auf einem beliebigen, aber jeweils festen Graphen von Logiken, Logik-Repräsentationen und -Projektionen basieren. Spezifikationen-im-Kleinen können in einer beliebigen Logik des Graphen geschrieben werden. Als Strukturierungsmechanismen-im-Großen stehen nicht nur die üblichen, aus CASL bekannten (homogenen) Konstrukte zur Strukturierung von Spezifikationen innerhalb einer Logik (wie Erweiterung, Umbenennung, Vereinigung, Instantiierung) zur Verfügung, sondern auch heterogene Konstrukte zur Übersetzung von Spezifikationen entlang der Repräsentationen und Projektionen des Graphen in andere Logiken. Repräsentationen und Projektionen mit verschieden guten Eigenschaften müssen evtl. mit verschiedenen Sprachkonstrukten unterstützt werden.

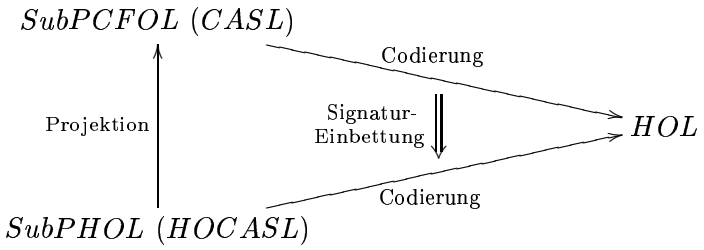


Abbildung 3: Eine Codierungs-Abbildung (*representation map*)

Die Semantik von CASL-im-Großen (strukturierte und Architektur-Spezifikationen sowie Bibliotheken) ist logik-unabhängig formuliert worden [CoF99, Mos98c, Mos00]. Damit stehen die Strukturierungsmechanismen von CASL auch für die heterogene Spezifikationssprache zur Verfügung.

Es ist geplant, für Teilsprachen einen Begriff von Repräsentations-Einbettung zu entwickeln, der es ermöglicht, die zugehörigen heterogenen Sprachkonstrukte implizit zu lassen, weil die Semantik der Spezifikation in der Teilsprache die gleiche ist wie in der umfassenderen Sprache. Entscheidend ist hier die Vertauschbarkeit von homogenen CASL-Strukturierungskonstrukten mit den heterogenen Konstrukten zur Übersetzung entlang der Repräsentationen des Logik-Graphs (vgl. Abschnitt 3.2.4).

3.2.6 Kompatibilität von Projektionen und Repräsentationen mit Codierungen

Da verschiedene Sprachen des Logik-Graphs in Isabelle/HOL kodiert werden sollen, stellt sich die Frage, ob die Codierungen (die ja spezielle Logik-Repräsentationen sind) mit den Logik-Repräsentationen und mit den Projektionen des Logik-Graphs kompatibel sind (und was Kompatibilität überhaupt bedeuten soll). Für die Kompatibilität von Projektionen wurde in [Tar96, MTP97, MTP98, Tar00] ein Ansatz entwickelt, der auf seine Tauglichkeit näher untersucht werden soll. Für Repräsentationen muss erst noch ein geeigneter Kompatibilitätsbegriff entwickelt werden. Da die Kompatibilität (zumindest im Falle der Projektionen) eine zusätzliche Signatur-Einbettungs-Komponente für die Projektion erfordert (die Projektion wird dadurch zur sog. Codierungs-Abbildung, vgl. Abb. 3), muss diese Komponente auch jeweils in den Logik-Graph integriert werden.

3.2.7 Beispielspezifikationen

Die Verwendbarkeit der heterogene Sprache soll durch die Entwicklung von adäquaten Beispielspezifikationen belegt werden. Insbesondere sollen Beispielspezifikationen entwickelt werden, die die Interaktion der verschiedenen Konzepte der verschiedenen verwendeten Logiken demonstrieren. Dazu sollen u.a. Teile des Sicherheitssystems des Bremer autonomen Rollstuhls [KBRCM98, LMKB98] (z.B. die virtuellen Sensoren) spezifiziert werden.

Arbeitsprogramm: Werkzeug-Ebene

Das Arbeitsprogramm für die Werkzeug-Ebene ist so strukturiert, dass mit der Entwicklung von Werkzeugen begonnen wird, für die die zugrunde liegenden Konzepte bereits klar entwickelt sind. Im weiteren Verlauf des Projekts sollen dann die Ergebnisse der Konzeptentwicklung unmittelbar in die Werkzeugentwicklung einfließen und ggf. auch wieder auf die Konzeptentwicklung zurückwirken.

3.2.8 Logik-unabhängige statische Analyse von CASL-im-Großen

CASL-im-Großen umfasst strukturierte und Architektur-Spezifikationen. Es existieren bereits mehrere Parser dafür (u.a. auch ein in Bremen auf der Basis von Isabelle entwickelter) sowie zwei Werkzeuge für die statische semantische Analyse (eines wurde in Saarbrücken, eines in Bremen entwickelt), die aber nicht logik-unabhängig sind.

Aufgabe der statischen Analyse ist nicht nur die statische Prüfung auf Wohlgeformtheit und Erzeugung des globalen *environments* (gemäß der CASL-Semantik), sondern auch die Erzeugung von Beweisverpflichtungen, die die Definiertheit der Semantik garantieren (z.B. bei der Definition von *views*.)

Die Semantik von CASL-im-Großen ist unabhängig von der zugrunde liegenden Logik definiert [Mos98c, CoF99]. Damit ergibt sich die Möglichkeit, auch die statische Analyse von CASL-im-Großen als generisches Werkzeug zu realisieren, das mit der logik-unabhängigen statischen Analyse für die Basis-Spezifikationen parametrisiert ist (technisch realisiert wird dies als Funktor in Standard ML). Eine erste konkrete Instantiierung ergibt sich dann mit der in Bremen entwickelten statischen Analyse für CASL-Basisspezifikationen. Später können dann Instantiierungen mit CASL-Erweiterungen folgen.

An die statische Analyse von CASL-im-Großen schließt sich in natürlicher Weise eine grafische Darstellung der Abhängigkeiten zwischen den Spezifikationen einer Bibliothek an. Es gibt verschiedene Abhängigkeiten, nämlich Instantiierung einer (benannten) Spezifikation, Verwendung einer Spezifikation als Parameter- oder Import-Spezifikation in einer anderen, generischen Spezifikation sowie Verwendung einer Spezifikation in einer Architektur-Spezifikation.

Für die Darstellung des Modulgraphen kann das in Bremen entwickelte Werkzeug daVinci verwendet werden, für das innerhalb der UniForM Workbench bereits Schnittstellen zu den funktionalen Sprachen Standard ML und Haskell existieren. Der Modulgraph wird dann Bestandteil eine Entwicklungsgraphen (letzterer umfasst auch die Verfeinerungsschritte in Richtung ausführbares Programm).

3.2.9 Erweiterung von Isabelle-im-Großen

Aufgrund der umfangreichen Bremer Erfahrungen soll vorrangig Isabelle als Beweiswerkzeug benutzt werden. Für CASL-Basisspezifikationen wurde bereits eine Repräsentation in Isabelle/HOL entwickelt [MKK98]. Um auch strukturierte und Architektur-Spezifikationen in Isabelle/HOL darstellen zu können, müssen die Strukturierungsmechanismen für Isabelle-Theorien erweitert werden. Derzeit sind nur Theorie-Erweiterungen und -Vereinigungen realisiert, während CASL auch Umbenennungen, *hiding* und Parametrisierung kennt.

Diese Erweiterung von Isabelle betrifft den die Korrektheit garantierenden Isabelle-Kern und ist von der Größenordnung her ein eigenes Projekt. Deshalb soll vorrangig mit der Gruppe von Prof. Tobias Nipkow, TU München, derart kooperiert werden, dass die dort bestehenden Überlegungen einer Erweiterung von Isabelle kompatibel mit den Anforderungen von CASL-im-Großen gemacht werden.

Falls diese Kooperation nicht oder nicht rechtzeitig zum gewünschten Erfolg führt, wäre auch eine Tiefenkodierung von CASL denkbar, bei der die Semantik der CASL-Strukturierungskonzepte in einer Mengentheorie ausgedrückt wird.

3.2.10 Der Logikgraph wird zum Werkzeuggraph

Der Werkzeuggraph besteht zunächst aus den Implementierungen der Repräsentationen und Projektionen des Logikgraphs. Als zugrunde liegendes Format kann das für CoFI benutzte *tool interchange format* verwendet werden. Für die in Bremen entwickelte Codierung von CASL in Isabelle/HOL existieren bereits Bibliotheken zum Parsen von CASL-Spezifikationen und zur

Umwandlung in das *tool interchange format*. Die Übersetzungen sollen in die UniForM Workbench integriert werden.

Basierend auf diesen Werkzeugen sollen als nächstes einige der Übersetzungen in andere Spezifikationssprachen implementiert werden (vgl. Abschnitt 3.2.2). Auf diese Weise wird ein Anschluss zu bestehenden Werkzeugen wie Termersetzern (OBJ3, CaféOBJ), Theorembeweisern (Larch Prover) und Modelcheckern (CSP) hergestellt. Darüber hinaus sollen weitere Übersetzungen zu Werkzeugen, die direkt zu CASL-Teilsprachen korrespondieren, implementiert werden (z. B. Theorembeweiser für Logik erster Stufe und Termersetzungssysteme). Der Logik-Graph wird so eingebettet in einen Werkzeug-Graph, der ebenfalls innerhalb der UniForM Workbench dargestellt werden kann.

3.2.11 Integration in die UniForM Workbench

Zu Beginn des Projektes sollen zunächst Werkzeuge für OBJ3, CafèOBJ und Larch sowie die für CASL existierenden Werkzeuge (die Bremer Codierung von CASL in Isabelle/HOL, die CASL-Versionen von KIV und Inka) in die UniForM Workbench [Kri99, KPO+99] eingefügt werden (ggf. auch weitere Werkzeuge für CSP und Z, neben den in der Workbench schon integrierten Werkzeugen HOL-CSP und HOL-Z). Dies ist aber noch keine echte Werkzeug-Integration, da noch keine gemeinsamen Datenformate und auch noch keine Querübersetzungen existieren.

Im weiteren Verlauf soll dann das Datenmodell der UniForM Workbench um Datentypen für die verschiedenen Teilsprachen und Erweiterungen von CASL, für die anderen Spezifikationssprachen und für die Repräsentationen, Projektionen und Übersetzungen erweitert werden.

Sobald die Übersetzungswerkzeuge vorliegen, sollen sie basierend auf dem erweiterten Datenmodell in die Workbench integriert werden.

3.2.12 Darstellung des Logik- und Werkzeuggraphen

Der Logik- und Werkzeug-Graph mit den CASL-Teilsprachen und -Erweiterungen und natürlich auch dessen Erweiterung um die Übersetzungen anderer Spezifikationssprachen von und nach CASL soll (mit dem in Bremen entwickelten Werkzeug daVinci) graphisch dargestellt werden. Für eine gegebene Spezifikation soll der Knoten des Graphen, in dem die Spezifikation lebt, zusammen mit einem relevanten Teilgraphen zur Navigation dargestellt werden. Zudem sollen die möglichen Übersetzungen in andere Sprachen anwählbar sein. Interessant wäre auch, für eine gegebene Quell- und Zielsprache sich die Menge der mögliche Pfade anzeigen lassen zu können (vgl. Dortmunder Electronic Tool Integration Platform [ETI97]). Dazu ist eine echte Erweiterung der UniForM Workbench notwendig.

3.2.13 Statische Analyse der heterogenen Sprache

Die Werkzeuge für die heterogene Sprache sollen generisch realisiert werden, und zwar parametrisiert über einen Logik-Graph sowie über Werkzeuge für die Behandlung der Knoten und Kanten des Graphen. Ein Werkzeug zur Behandlung eines Knotens ist eine Analysewerkzeug für Spezifikationen-im-Kleinen, die in der betreffenden Logik geschrieben wurden. Ein Werkzeug zur Behandlung einer Kante ist ein Übersetzungswerkzeug (vgl. Abschnitt 3.2.10).

Es muss also einerseits die statische Analyse von Spezifikationen-im-Großen tatsächlich logikunabhängig implementiert und andererseits für die heterogenen Konstrukte die jeweilige Übersetzung aufgerufen werden.

Durch Kopplung mit den anderen Werkzeugen kann zu einer heterogenen Spezifikation der von ihr benutzte Teil des Graphen sowie Übersetzungsmöglichkeiten in andere Sprachen aufgezeigt (und auf Wunsch auch ausgeführt) werden.

3.2.14 Heterogene Beweise

In der Regel wird der Benutzer bzw. die Benutzerin jedoch eine heterogene Spezifikation nicht "flachklopfen" wollen, indem alle Teilspezifikationen in eine einheitliche Spezifikationssprache übersetzt werden (schon allein, weil dabei -im Falle der Projektionen– Informationen verloren gehen können). Wenn jedoch Beweise in Isabelle/HOL geführt werden sollen, ist eine einheitliche Codierung von heterogenen Spezifikationen in Isabelle/HOL sinnvoll. Diese kann durch eine Kombination kompatibler Codierungen der einzelnen Logiken entstehen – die heterogene Sprache muss dafür also auf diejenigen Projektionen und Repräsentationen eingeschränkt werden, die mit den Codierungen in Isabelle/HOL kompatibel sind (vgl. Abschnitt 3.2.6).

Die mittels dieser Methode geführten Beweise sind jedoch noch sozusagen homogene Beweise, da allesamt in Isabelle/HOL geführt (wenn auch in den Codierungen verschiedener Logiken). Daneben soll auch die Möglichkeit heterogener Beweise, d.h. Verwendung verschiedener Beweiswerkzeuge, realisiert werden.

Ein Weg, dies umzusetzen, ist die Einbettung anderer Werkzeuge als Orakel in Isabelle/HOL. Es soll aber auch die Möglichkeit untersucht werden, Beweise, die mit unterschiedlichen Werkzeugen geführt werden, als Objekte in der UniForM Workbench zu repräsentieren. Diese Objekte können dann innerhalb von Isabelle/HOL oder anderen Beweisern (über die UniForM Workbench als Orakel) importiert werden. Dies hätte den Vorteil, dass verschiedene Beweiswerkzeuge gleichberechtigt koexistieren können.

3.2.15 Gesamt-Integration und Test

Basierend auf den weiteren (ggf. noch von anderen zu entwickelnden) Werkzeugen für CASL höherer Stufe, reaktives und objekt-orientiertes CASL soll dann alle Werkzeuge in die UniForM Workbench integriert und mittels einer kleinen Fallstudie getestet werden.

3.3 Untersuchungen am Menschen

entfällt

3.4 Tierversuche

entfällt

3.5 Gentechnologische Experimente

entfällt

4 Beantragte Mittel

4.1 Personalbedarf

Da das beantragte Projekt sowohl in seinem theoretischen als auch seinem praktischen Teil umfangreich ist (und gerade in ihrer Verbindung seine Kraft liegt), werden Mittel für zwei wissenschaftliche Mitarbeiter für zunächst zwei Jahre auf zwei vollen BAT IIa-Stellen beantragt. Es ist vorgesehen, für eine dieser Stellen Herrn Dr. Till Mossakowski einzustellen, der den theoretischen und konzeptionellen Teil des Projekts weitgehend selbständig durchführen soll. Zudem soll er den anderen Mitarbeiter bei der Entwicklung der Werkzeuge anleiten, mit der Möglichkeit zur Promotion. Herr Mossakowski wird derzeit durch die Universität Bremen mit einem Postdoktoranden-Stipendium gefördert, das im September 1999 endet.

Die Durchführung des Projektes erfordert im konzeptionellen Bereich einen Mitarbeiter mit genauen und vertieften Kenntnissen im Bereich des Entwurfs und der Semantik von Spezifikationssprachen sowie im Bereich von Beweiswerkzeugen. Im Bereich der Werkzeugentwicklung ist ein Mitarbeiter mit guter sowohl theoretischer (Spezifikationssprachen) als auch praktischer (Beweiswerkzeuge, funktionale Programmierung) Qualifikation erforderlich. Bei der derzeitigen Marktlage ist aller Voraussicht nach hier nur für eine volle Stelle ein entsprechend qualifizierter Bewerber/Bewerberin zu bekommen.

4.2 Wissenschaftliche Geräte

entfällt

4.3 Verbrauchsmaterial und Versuchstiere

entfällt

4.4 Reisen

Zur Zusammenarbeit mit den unter 5.2 genannten Wissenschaftlern sind innerhalb der ersten zwei Jahre folgende Reisen geplant:

...

4.5 sonstige Kosten

...

5 Voraussetzungen für die Durchführung des Vorhabens

5.1 Zusammensetzung der Arbeitsgruppe

Neben den Antragstellern arbeiten innerhalb der Forschungsgruppe Krieg-Brückner im Studiengang Informatik des Fachbereichs Mathematik/Informatik an der Universität Bremen Dr. Markus Roggenbach (DFG-Projekt COOFL), Dr. Christoph Lüth (Isabelle/HOL, generische grafische Benutzerschnittstelle) und Dr. George Russell (UniForM Workbench) an Fragestellungen, die eng mit dem beantragten Projekt zusammenhängen.

5.2 Zusammenarbeit mit anderen Wissenschaftlern

1. Prof. Dr. Andrzej Tarlecki, Polnische Akademie der Wissenschaften, Warschau (heterogene Sprache)
2. Dr. Hélène Kirchner, LORIA-CNRS & INRIA Lorraine, Nancy (Termersetzung und Beweisen)
3. Prof. Dr. Tobias Nipkow, TU München (Isabelle-im-Großen)
4. Dr. Dieter Hutter, Heiko Mantel, Axel Schairer, DFKI GmbH, Saarbrücken (heterogene Beweise)

5.3 Auslandsbezug

Mit den ersten beiden der unter 5.2 genannten Wissenschaftlern aus dem Ausland bestehen Kooperationen (u.a. Veröffentlichung gemeinsamer Arbeiten), die im Rahmen des beantragten Projekts fortgesetzt werden sollen. Zudem ergeben sich internationale Kooperationen, die mit dem beantragten Vorhaben in Zusammenhang stehen, auch aus der Mitarbeit in der CoFI Working Group.

5.4 Apparative Ausstattung

Die beantragten Arbeitsplatzrechner (vgl. Abschnitt 4.5) können in das vorhandene Rechnernetz eingebunden werden. Die im bestehende Rechnernetz vorhandenen leistungsfähigen Rechner werden bereits für andere Projekte fest genutzt.

5.5 Laufende Mittel für Sachausgaben

Laufende Kosten für Schreibbedarf etc. werden im Rahmen vorhandener Haushaltsmittel der AGs Kreowski und Krieg-Brückner von der Universität Bremen zur Verfügung gestellt.

5.6 Sonstige Voraussetzungen

Schreib- und Organisationsarbeiten können von den Sekretariaten der Arbeitsgruppen der Antragsteller in angemessenen Umfang übernommen werden.

6 Erklärungen

6.1

Ein Antrag auf Finanzierung dieses Vorhabens wurde bei keiner anderen Stelle eingereicht. Wenn wir einen solchen Antrag stellen, werden wir die Deutsche Forschungsgemeinschaft unverzüglich benachrichtigen.

6.2

Der Vertrauensmann der DFG an der Universität Bremen ist über die Antragstellung informiert.

6.3 entfällt

7 Unterschrift

Bremen, den

Prof. Dr. B. Krieg-Brückner

Prof. Dr. H.–J. Kreowski

8 Verzeichnis der Anlagen

Prof. Bernd Krieg-Brückner

- Lebenslauf
- Verzeichnis der wissenschaftlichen Veröffentlichungen seit 1997
- Ausgewählte Publikationen: [ABKB+, CHKBM97, MKK98, KPO+99]

Prof. Hans-Jörg Kreowski

- Lebenslauf
- Verzeichnis der wissenschaftlichen Veröffentlichungen seit 1994
- Ausgewählte Publikationen: [KKS97, AEH+99]

Dr. Till Mossakowski

- Personalfragebogen
- Ausgewählte Publikationen: [Mos96b, Mos96a, MTP98]

Literatur

[ABKB+] E. Astesiano, M. Bidoit, B. Krieg-Brückner, H. Kirchner, P. D. Mosses, D. Sannella, and A. Tarlecki. Casl – the common algebraic specification language. *Theoretical Computer Science*. To appear.

[AC94] E. Astesiano and M. Cerioli. Multiparadigm specification languages: a first attempt at foundations. In C.M.D.J. Andrews and J.F. Groote, editors, *Semantics of Specification Languages (SoSl 93)*, Workshops in Computing, pages 168–185. Springer Verlag, 1994.

[AEH+99] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schürr, and G. Taentzer. Graph transformation for specification and programming. *Science of Computer Programming*, 34:1–54, 1999.

[AKKB99] E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. Springer, 1999. 616 pages.

[BCL96] G. Bernot, S. Coudert, and P. Le Gall. Towards heterogeneous formal specifications. In *AMAST 96*, volume 1101 of *Lecture Notes in Computer Science*, pages 458–472. 1996.

[BG77] R. M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1045–1058. Cambridge, 1977.

[BKB99] D. Basin and B. Krieg-Brückner. Formalization of the development process. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, pages 521–562. Springer, 1999.

[BKL+91] M. Bidoit, H.-J. Kreowski, P. Lescannes, F. Orejas, and D. Sannella. *Algebraic System Specification and Development. A Survey and Annotated Bibliography*, volume 501 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.

[Bor99] T. Borzyszkowski. Moving specification structures between logical systems. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 13th International Workshop, WADT'98, Lisbon, Portugal, April 1998, Selected Papers*, volume 1589 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1999.

[CBL99] S. Coudert, G. Bernot, and P. Le Gall. Hierarchical heterogeneous specifications. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 13th International Workshop, WADT'98, Lisbon, Portugal, April 1998, Selected Papers*, volume 1589 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 1999.

[CGK+97] M. Cerioli, M. Gogolla, H. Kirchner, B. Krieg-Brückner, Z. Qian, and M. Wolf. *Algebraic System Specification and Development. A Survey and Annotated Bibliography, 2nd Edition*, volume 3 of *BISS monographs*. Shaker Verlag, 1997.

[CHKBM97] Maura Cerioli, Anne Haxthausen, Bernd Krieg-Brückner, and Till Mossakowski. Permissive subsorted partial logic in CASL. In Michael Johnson, editor, *Algebraic methodology and software technology: 6th international conference, AMAST 97*, volume 1349 of *Lecture Notes in Computer Science*, pages 91–107. Springer-Verlag, 1997.

[CM97] M. Cerioli and J. Meseguer. May I borrow your logic? (transporting logical structures along maps). *Theoretical Computer Science*, 173:311–347, 1997.

[CMR99] M. Cerioli, T. Mossakowski, and H. Reichel. From total equational to partial first order logic. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specifications*, pages 31–104. Springer Verlag, 1999.

[CoF] CoFI. The Common Framework Initiative for algebraic specification and development, electronic archives. Notes and Documents accessible from http://www.brics.dk/Projects/CoFI.

[CoF98] CoFI Language Design Task Group. CASL – The CoFI Algebraic Specification Language – Summary. Documents/CASL/Summary, in [CoF], October 1998.

[CoF99] CoFI Semantics Task Group. CASL – The CoFI Algebraic Specification Language – Semantics. Note S-9 (version 0.95), in [CoF], March 1999.

[DF96] R. Diaconescu and K. Futatsugi. Logical semantics of CafeOBJ. Technical report, JAIST, 1996. IS-RR-96-0024S.

[DF98] R. Diaconescu and K. Futatsugi. *CafeOBJ Report*. AMAST series. World Scientific, Singapore, 1998.

[Dia98] R. Diaconescu. Extra theory morphisms for institutions: logical semantics for multi-paradigm languages. *J. Applied Categorical Structures*, 6:427–453, 1998.

[EK99] Hartmut Ehrig and Hans-Jörg Kreowski. Refinement and implementation. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, pages 201–242. Springer, 1999.

[ETI97] Special section on the electronic tool integration platform. *International Journal on Software Tools for Technology Transfer*, 1:9–63, 1997.

[Fis98] C. Fischer. How to combine Z with a process algebra. In J. P. Bowen, A. Fett, and M. G. Hinchey, editors, *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users, Berlin, Germany, 24–26 September 1998*, volume 1493 of *Lecture Notes in Computer Science*, pages 5–23. Springer-Verlag, 1998.

[GB92] J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992. Predecessor in: LNCS 164, 221–256, 1984.

[GC95] M. Gogolla and M. Cerioli. What is an abstract data type after all? *Lecture Notes in Computer Science*, 906:499–523, 1995.

[GHG+93] John V. Guttag, James J. Horning, S. J. Garland, K. D. Jones, A. Modet, and J. M. Wing. *Larch: Languages and Tools for Formal Specification.* Springer Verlag, New York, N.Y., 1993.

[GMW+] J. Goguen, J. Meseguer, T. Winkler, K. Futatsugi, P. Lincoln, and J.-P. Jouannaud. Introducing OBJ. Technical Report SRI-CSL-88-8, Computer Science Lab, SRI International, August 1988; revised version from 24th October 1993.

[GW88] J. A. Goguen and T. Winkler. Introducing OBJ3. Research report SRI-CSL-88-9, SRI International, 1988.

[HKB93] Berthold Hoffmann and B. Krieg-Brückner. *Program development by specification and transformation: the PROSPECTRA methodology, language family, and system*, volume 680 of *Lecture Notes in Computer Science.* Springer-Verlag, New York, 1993.

[HKBM98] Anne Haxthausen, Bernd Krieg-Brückner, and Till Mossakowski. Subsorted partial higher-order logic as an extension of CASL. Note L-10, in [CoF], October 1998.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[IBH+79] J. D. Ichbiah, J. G. P. Barnes, J. C. Heliard, B. Krieg-Brückner, O. Roubine, and B. A. Wichmann. Reference manual and rationale for the Ada programming language. *ACM SIGPLAN Notices*, 14(6), June 1979.

[Jon90] C. B. Jones. *Systematic Software Development Using VDM.* Prentice Hall, 1990.

[KBRCM98] B. Krieg-Brückner, T. Röfer, H.-O. Carmesin, and R. Müller. A taxonomy of spatial knowledge for navigation and its application to the bremen autonomous weelchair. In Ch. Freska, Ch. Habel, and F. K. Wender, editors, *Spatial Cognition*, volume 1404 of *Lecture Notes in Artificial Intelligence*, pages 373–397. Springer Verlag, 1998.

[KBS91] B. Krieg-Brückner and D. Sannella. Structuring specifications in-the-large and in-the-small: Higher-order functions, dependent types and inheritance in Spectral. In *Proc. TAPSOFT 91*, volume 494 of *Lecture Notes in Computer Science*, pages 313–336. Springer Verlag, 1991.

[KK99] Hans-Jörg Kreowski and Sabine Kuske. Graph transformation units and modules. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rosenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2. World Scientific, Singapore, 1999.

[KKS97] Hans-Jörg Kreowski, Sabine Kuske, and Andy Schürr. Nested graph transformation units. *International Journal of Software Engineering and Knowledge Engineering*, 7:479–502, 1997.

[KLMW97] Kolyang, C. Lüth, T. Meier, and B. Wolff. TAS and IsaWin: Generic interfaces for transformational program development and theorem proving. In M. Bidoit and M. Dauchet, editors, *Proceedings of the Seventh International Joint Conference on the Theory and Practice of Software Development (TAPSOFT'97)*, pages 855–858. Springer-Verlag LNCS 1214, 1997.

[KM] H. Kirchner and P. D. Mosses. Algebratic specifications, higher-order types and set-theoretic models. To appear.

[KM95] H.-J. Kreowski and T. Mossakowski. Equivalence and difference of institutions: Simulating horn clause logic with based algebras. *Mathematical Structures in Computer Science*, 5:189–215, 1995.

[KPO+99] B. Krieg-Brückner, J. Peleska, E.-R. Olderog, D. Balzer, and A. Baer. The UniForM Workbench, a universal development environment for formal methods. In *FM99: World Congress on Formal Methods*, volume 1709 of *Lecture Notes in Computer Science*, pages 1186–1205. Springer-Verlag, 1999.

[Kri99] B. Krieg-Brückner. Uniform perspectives for formal methods. In *International Workshop on Current Trends in Applied Formal Methods, Boppard 1998*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[LKK+98] C. Lüth, E. W. Karlsen, Kolyang, S. Westmeier, and B. Wolff. HOL-Z in the UniForM-workbench – A case study in tool integration for Z. In J. P. Bowen, A. Fett, and M. G. Hinchey, editors, *ZUM'98: The Z Formal Specification Notation, 11th International Conference of Z Users, Berlin, Germany, 24–26 September 1998*, volume 1493 of *Lecture Notes in Computer Science*, pages 116–134. Springer-Verlag, 1998.

[LMKB98] A. Lankenau, O. Meyer, and B. Krieg-Brückner. Safety in robotics: The bremen autonomous wheelchair. In *Proc. AMC 98, 5th Int. Workshop on Advanced Motion Control, Coimbra, Portugal*, pages 524–529. 1998.

[Mes89] J. Meseguer. General logics. In *Logic Colloquium 87*, pages 275–329. North Holland, 1989.

[MKK98] T. Mossakowski, Kolyang, and B. Krieg-Brückner. Static semantic analysis and theorem proving for CASL. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 1998.

[Mos96a] T. Mossakowski. Different types of arrow between logical frameworks. In F. Meyer auf der Heide and B. Monien, editors, *Proc. ICALP 96*, volume 1099 of *Lecture Notes in Computer Science*, pages 158–169. Springer Verlag, 1996.

[Mos96b] T. Mossakowski. Equivalences among various logical frameworks of partial algebras. In H. Kleine Büning, editor, *Computer Science Logic. 9th Workshop, CSL'95. Paderborn, Germany, September 1995, Selected Papers*, volume 1092 of *Lecture Notes in Computer Science*, pages 403–433. Springer Verlag, 1996.

[Mos96c] T. Mossakowski. *Representations, hierarchies and graphs of institutions.* PhD thesis, Bremen University, 1996.

[Mos96d] T. Mossakowski. Using limits of parchments to systematically construct institutions of partial algebras. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 379–393. Springer Verlag, 1996.

[Mos97a] Till Mossakowski. Sublanguages of CASL. Note L-7, in [CoF], December 1997.

[Mos97b] Till Mossakowski. Subsorting and structured specification. Note S-3, in [CoF], March 1997.

[Mos97c] Peter D. Mosses. CoFI: The Common Framework Initiative for Algebraic Specification and Development. In *TAPSOFT '97, Proc. Intl. Symp. on Theory and Practice of Software Development*, volume 1214 of *LNCS*, pages 115–137. Springer-Verlag, 1997.

[Mos98a] T. Mossakowski. Colimits of order-sorted specifications. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 1998.

[Mos98b] Till Mossakowski. Cocompleteness of the CASL signature category. Note S-7, in [CoF], February 1998.

[Mos98c] Till Mossakowski. Institution-independent semantics for CASL-in-the-large. Note S-8 (revised), superseded by Note S-10, in [CoF], October 1998.

[Mos98d] Till Mossakowski. Standard annotations for parsers and static semantic checkers – a proposal. Note T-6 (revised), in [CoF], September 1998.

[Mos98e] Till Mossakowski. Two "functional programming" sublanguages of CASL. Note L-9, in [CoF], March 1998.

[Mos98f] Till Mossakowski. Version control and registration for CASL libraries. Note M-5, in [CoF], September 1998.

[Mos99] T. Mossakowski. Translating OBJ3 to CASL: the institution level. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 13th International Workshop, WADT'98, Lisbon, Portugal, April 1998, Selected Papers*, volume 1589 of *Lecture Notes in Computer Science*, pages 198–214. Springer-Verlag, 1999.

[Mos00] T. Mossakowski. Specification in an arbitrary institution with symbols. In C. Choppy, D. Bert, and P. Mosses, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, volume 1827 of *Lecture Notes in Computer Science*, pages 252–270. Springer-Verlag, 2000.

[MR99] Till Mossakowski and Markus Roggenbach. The datatypes REAL and COMPLEX in CASL. Note M-7, in [CoF], April 1999.

[MTP97] T. Mossakowski, A. Tarlecki, and W. Pawłowski. Combining and representing logical systems. In E. Moggi and G. Rosolini, editors, *Category Theory and Computer Science, 7th Int. Conf.*, volume 1290 of *Lecture Notes in Computer Science*, pages 177–196. Springer Verlag, 1997.

[MTP98] T. Mossakowski, A. Tarlecki, and W. Pawłowski. Combining and representing logical systems using model-theoretic parchments. In F. Parisi Presicce, editor, *Recent trends in algebraic development techniques. Proc. 12th International Workshop*, volume 1376 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 1998.

[Pau90] L. C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.

[Pau94] L. C. Paulson. *Isabelle - A Generic Theorem Prover.* Number 828 in LNCS. Springer Verlag, 1994.

[Qia94] Zhenyu Qian. Higher-order equational logic programming. In ACM, editor, *Proceedings of 21st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 254–267, New York, NY, USA, 1994. ACM Press.

[Qia95] Zhenyu Qian. Unification of higher-order patterns in linear time and space. *Journal of Logic and Computation*, 6(3):315–341, 1995.

[QW92] Zhenyu Qian and Kang Wang. Higher-order E-unification for arbitrary theories. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 52–66, Washington, USA, 1992. The MIT Press.

[QW94] Zhenyu Qian and Kang Wang. Modular AC unification of higher-order patterns. In *Proc. Int. Conf. on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*, pages 105–120. Springer, 1994.

[Rei87] H. Reichel. *Initial Computability, Algebraic Specifications and Partial Algebras.* Oxford Science Publications, 1987.

[RM99a] Markus Roggenbach and Till Mossakowski. Basic datatypes in CASL. Note M-6, in [CoF] (a new version is due in November, see also Note L-12), July 1999.

[RM99b] Markus Roggenbach and Till Mossakowski. Proposal of some annotations and literal syntax in CASL. Note L-11, in [CoF], March 1999.

[SB93] S. Stepney and R. Barden. Annotated Z Bibliography. 50:280–313, June 1993.

[Spi89] J. M. Spivey. *The Z Notation.* Prentice Hall, 1989.

[SSCM00] A. Sernadas, C. Sernadas, C. Caleiro, and T. Mossakowski. Categorical fibring of logics with terms and binding operators. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems 2*, Studies in Logic and Computation, pages 295–316. Research Studies Press, 2000.

[ST86] D. T. Sannella and A. Tarlecki. Extended ML: an institution-independent framework for formal program development. In *Proc. Workshop on Category Theory and Computer Programming*, volume 240 of *Lecture Notes in Computer Science*, pages 364–389. Springer, 1986.

[ST88] D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

[ST90] D. Sannella and A. Tarlecki. Extended ML: Past, present and future. In H. Ehrig, K. P. Jantke, F. Orejas, and H. Reichel, editors, *Proceedings of Recent Trends in Data Type Specification*, volume 534 of *Lecture Notes in Computer Science*, pages 297–322. Springer, 1990.

[Tar96] A. Tarlecki. Moving between logical systems. In M. Haveraaen, O. Owe, and O.-J. Dahl, editors, *Recent Trends in Data Type Specifications. 11th Workshop on Specification of Abstract Data Types*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer Verlag, 1996.

[Tar00] A. Tarlecki. Towards heterogeneous specifications. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems 2, 1998*, Studies in Logic and Computation, pages 337–360. Research Studies Press, 2000.

[TW97] H. Tej and B. Wolff. A corrected failure-divergence model for CSP in Isabelle/HOL. In John Fitzgerald, Cliff B. Jones, and Peter Lucas, editors, *FME'97: Industrial Applications and Strengthened Foundations of Formal Methods (Proc. 4th Intl. Symposium of Formal Methods Europe, Graz, Austria, September 1997)*, volume 1313 of *Lecture Notes in Computer Science*, pages 318–337. Springer-Verlag, 1997.

[Wan94] K. Wang. *Higher-Order Constraint Logic Programming.* PhD thesis, Universität Bremen, 1994.

[Wir86] M. Wirsing. Structured algebraic specifications: A kernel language. *Theoretical Computer Science*, 42:123–249, 1986.