DFG-Antrag auf Gewährung einer Sachbeihilfe für das Projekt

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

—Verlängerungsantrag—

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

30. April 2002

Inhaltsverzeichnis

1 Allgemeine Angaben

Antrag auf Gewährung einer Sachbeihilfe, Verlängerungsantrag

1.1 Antragsteller

Bernd Krieg-Brückner, Prof. Dr. rer. nat.
Professor für Programmiersprachen, Übersetzer und Softwaretechnik
geb. 15.2.1949, Deutscher
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

Privatadresse
Rehsprung 28
28355 Bremen
Tel. (0421)251024

Hans-Jörg Kreowski, Professor Dr.-Ing.
Professor für Theoretische Informatik
geb. 10.8.1949, Deutscher
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

Privatadresse
Zollpfad 7
28865 Lilienthal
Tel. (04298)8880

Till Mossakowski, Dr. Ing.
Wissenschaftlicher Mitarbeiter im DFG-Projekt MULTIPLE

geb. 25.10.1967, Deutscher
Universität Bremen
Studiengang Informatik des Fachbereichs Mathematik/Informatik

Dienstliche Adresse
Universität Bremen
Fachbereich 3, Mathematik/Informatik
Postfach 33 04 40, 28334 Bremen
Tel. (0421)218-4683, Fax: (0421)218-3054
E-Mail till@informatik.uni-bremen.de

Privatadresse
Dobbenweg 9
28203 Bremen
Tel. (0421)3379314

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

Das Vorhaben wird seit 1. Juli 2000 von der DFG gefördert.
Eine Förderung bis zum 31.10.2004 wird beantragt.

1.6 Antragszeitraum

2 Jahre

1.7 Gewünschter Beginn der Förderung

Datum der bisherigen Bewilligung: 10. 04. 2000

Die Personalmittel reichen voraussichtlich bis 31. 10. 2002 (kostenneutrale Verlängerung)

Die Sachmittel reichen voraussichtlich bis 31. 10. 2002

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) wurde im ersten Teil des Projekts ein Graph von Logiken entwickelt, der Teilsprachen und Erweiterungen von CASL sowie exemplarisch andere Spezifikationssprachen umfasst, mit denen bereits erfolgreich Anwendungen entwickelt wurden (CSP, OBJ, Larch, Z), siehe auch den beigefügten Zwischenbericht.

Im zweiten Teil soll es nun um die Implementierung einer darauf basierenden heterogenen Spezifikationssprache gehen. Dazu sollen existierende Analyse- und Beweiswerkzeuge integriert und ein umfassenden Beweismanagement entwickelt werden. Der Logik-Graph ist so die semantische Basis für die heterogene Kombination von Methoden und Werkzeugen.

2 Stand der Forschung, eigene Vorarbeiten

2.1 Stand der Forschung

Siehe beigefügten Zwischenbericht.

2.2 Eigene Vorarbeiten/Arbeitsbericht

Siehe beigefügten Zwischenbericht.

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 im ersten Teil des Projekts ausgearbeiteten semantischen Grundlagen sollen noch an einzelnen Stellen abgerundet, vor allem aber implementiert werden. Am Ende soll eine Entwicklungsumgebung für eine heterogene Spezifikationssprache entstanden sein, die auf dem ausschnittsweise im Zwischenbericht gezeigten Logikgraph basiert. Dazu sollen existierende Analyse- und Beweiswerkzeuge integriert und ein umfassenden Beweismanagement entwickelt werden. Der Logik-Graph ist so die semantische Basis für die heterogene Kombination von Methoden und Werkzeugen. Eine kleine Fallstudie soll die praktische Anwendbarkeit belegen.

3.2 Arbeitsprogramm

Das Arbeitsprogramm deckt den Antragszeitraum von zwei Jahren sowie die noch verbleibende Zeit aus dem ersten Antrag ab. Es umfasst fünfzehn Teilaufgaben, von denen die ersten fünf zur konzeptionellen Ebene gehören und die Ergebnisse der ersten Projektphase abrunden. Zehn Teilaufgaben bilden dann die Werkzeug-Ebene. Beide Ebenen sollen jeweils von einem Mitarbeiter abgedeckt werden und sind aufeinander abgestimmt. Gegenüber der ersten Phase des Projekts wird sich in der Verlängerung der Schwerpunkt von der konzeptionellen auf die Werkzeug-Ebene und die Ausarbeitung von Beispielen verschieben (d.h. der erste Mitarbeiter soll auch Implementierungsarbeiten übernehmen, insbesondere die Ableitung von Algorithmen aus den Ergebnissen der konzeptionellen Arbeit). Siehe auch den Zeitplan.

Arbeitsprogramm: konzeptionelle Ebene

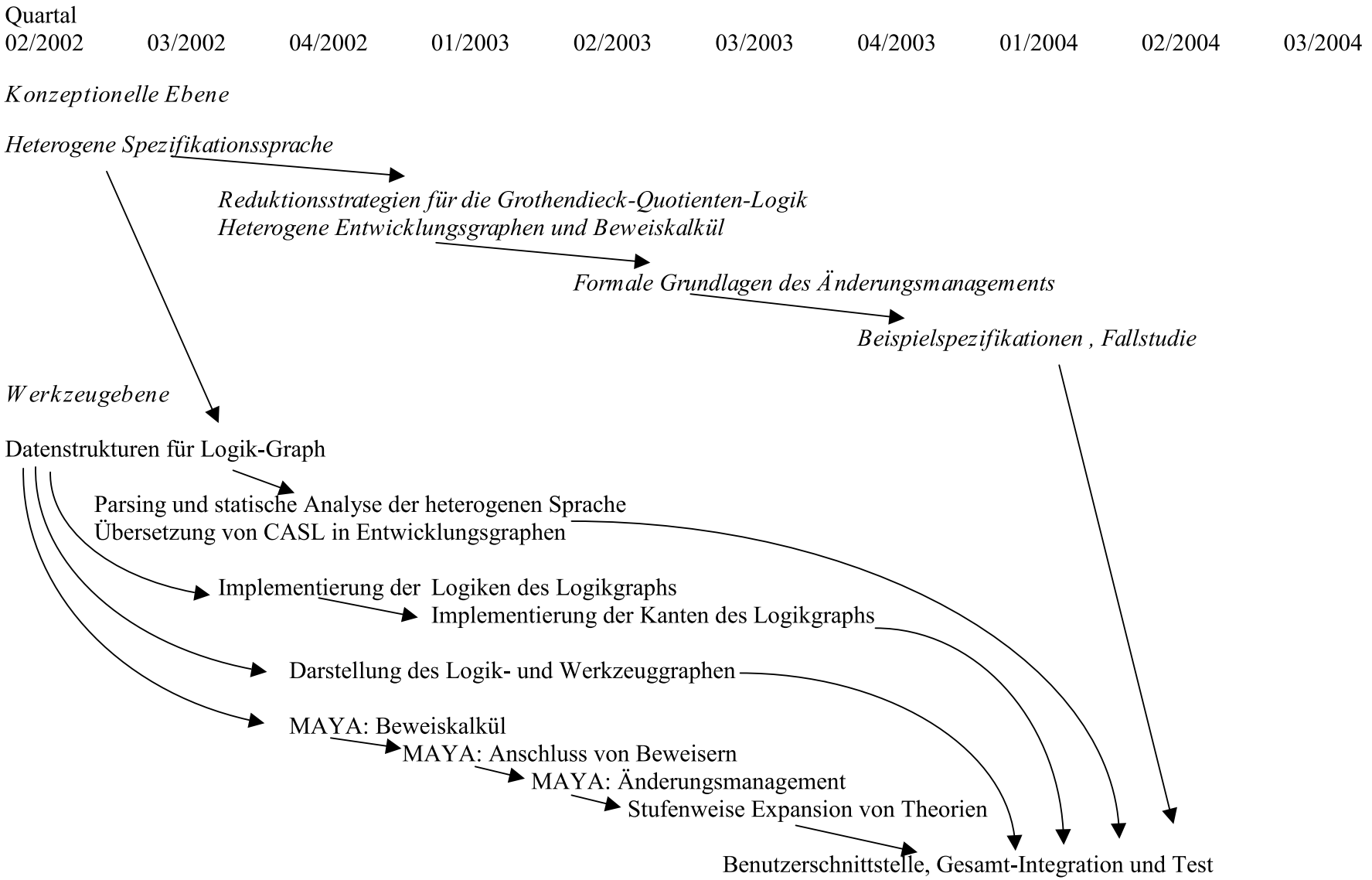
3.2.1 Heterogene Spezifikationssprache

Die theoretischen Grundlagen der heterogenen Spezifikationssprache sind mit Logik-Graph und Grothendieck-Quotienten-Logik bereits im ersten Teil des Projekts entwickelt worden. Es muss jedoch noch eine geeignete Syntax festgelegt werden, mit der man zwischen verschiedenen Logiken des Logik-Graphen wechseln kann. Außerdem bleibt eine wichtige semantische Fragestellung als naheliegende Verallgemeinerung zu bearbeiten: der Übergang von Spezifikations- zu Programmiersprachen. Dazu werden sogenannte Logik-Semi-Morphismen [ST88] benötigt, die im Gegensatz zu Logik-Morphismen keine Formeln übersetzen. Die Theorie der Grothendieck-Quotienten-Logik soll um diese noch erweitert werden, und auch im Logik-Graph soll zumindest eine Programmiersprache (gedacht ist an Haskell) oder programmiersprachen-nahe Logik (HasCasl) aufgenommen werden. Eine naheliegende Idee ist hier, Logik-Morphisen und Logik-Semi-Morphismen zu einem gemeinsamen Oberbegriff zu verallgemeinern: dies sind dann Logik-Morphismen mit einer partiellen Formelübersetzung.

3.2.2 Reduktionsstrategien für die Grothendieck-Quotienten-Logik

Die Grothendieck-Quotienten-Logik muss, um implementierbar zu sein, in einer geeigneten Form dargestellt werden. Dazu soll die Signatur-Kategorie dieser Logik als induzierte

Zeitplan MULTIPLE



Kategorie eines Kompositionsgraphen [Sch99] dargestellt werden (ein Kompositionsgraph ist eine syntaktische Präsentation einer Kategorie mit Generatoren und Relationen). Mittels der Kompositionsgraphen sollen geeignete Strategien für die Berechnung der Gleichheit von Signaturmorphismen untersucht werden; eine zu klärende Frage ist dabei, ob bzw. in welchen Spezialfällen Konfluenz und Terminierung gilt (auf diese Eigenschaften wird man jedoch wahrscheinlich nicht in allen Fällen bauen können).

3.2.3 Heterogene Entwicklungsgraphen und Beweiskalkül

Die im ersten Teil des Projekts entwickelten heterogenen Entwicklungsgraphen müssen auf signifikante Eigenschaften hin untersucht werden, um sie tatsächlich sinnvoll nutzen zu können. Dazu gehören einerseits schwache Amalgamierungseigenschaften, die im Kalkül für heterogene Entwicklungsgraphen mit Verstecken [Mos] vorausgesetzt werden. Diese Eigenschaften sollen in der Grothendieck-Quotienten-Logik über dem Logik-Graph untersucht werden.

Andererseits soll der Kalkül für heterogene Entwicklungsgraphen mit Verstecken auf Konfluenz und Terminierung hin untersucht und geeignete Reduktionsstrategien entwickelt werden, um Hinweise für die Implementierung im Werkzeug MAYA [AM02] zu bekommen. Dies ist also eine ähnliche Fragestellung wie in Abschnitt 3.2.2, mit dem Unterschied, dass es hier nicht um Term-, sondern um Graphersetzung geht. Hierbei soll auf die für Graph-Grammatiken entwickelten Theorien von Konfluenz und Terminierung zurückgegriffen werden, die ein zentraler Forschungsgegenstand der Arbeitsgruppe des Mitantragstellers Kreowski sind. Eventuell gelingt es dabei auch noch, die Voraussetzungen des Vollständigkeitssatzes in [Mos] abzuschwächen.

3.2.4 Formale Grundlagen des Änderungsmanagements

Das Werkzeug MAYA führt neben einem Beweis- auch ein Änderungsmanagement durch, das erlaubt, bei einer Änderung einer Spezifikation Teile von Beweisen wiederzuverwenden. In [AHMS00] ist eine entsprechende Methode beschrieben worden, die hier auf sowohl Verstecken als auch Heterogenität verallgemeinert werden soll.

Der erste Schritt besteht aus der Anwendung von Heuristiken, mit denen versucht wird, den einzelnen Symbolen einer Signatur entsprechende Symbole aus der geänderten Signatur zuzuordnen (sog. *tracking maps*). Hier kann eine Logik-Unabhängigkeit nur erreicht werden, in dem man zu *Logiken mit Symbolen* übergeht (vgl. die *Institutions with symbols* aus [Mos00]). Das Resultat ist ein Delta, das die Differenz zwischen der alten und der neuen Spezifikation ausdrückt.

Der zweite Schritt ist dann, dieses Delta in den bestehenden Entwicklungsgraphen (und seine gemäß der Kalkülregeln erfolgte Dekomposition in lokale Beweisziele) zu propagieren. Trickreich ist dabei die Entwicklung von geeigneten Strategien für den Durchlauf durch die Graphstruktur und ihre vorhandene Dekomposition (dieser Durchlauf wird durch das Vorhandensein von Verstecken deutlich komplizierter). Zudem muss hierbei in einer Weise vorgegangen werden, die Logik-unabhängige Korrektheit garantiert, d.h. es dürfen nur die von Meseguer cd[Mes89] für Logiken aufgestellten Axiome benutzt werden.

3.2.5 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. Interessant ist hier z.B. die Koppelung von datentyporientierten und Kontroll-orientierten Logiken. Als kleine Fallstudie sollen dazu u.a. Teile des Sicherheitssystems des Bremer autonomen Rollstuhls [RL00, LR01] (z.B. die virtuellen Sensoren) spezifiziert werden. Auf diese Weise wird eine Brücke zum Bremer DFG-Projekt Safe Robotics geschaffen.

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. Ziel ist ein Werkzeug-Paket, das im wesentlichen in Haskell vorliegt (basierend auf den umfangreichen Haskell-Erfahrungen im BMBF-Projekt UniForM [Kar99, Kri99, KPO+99]).

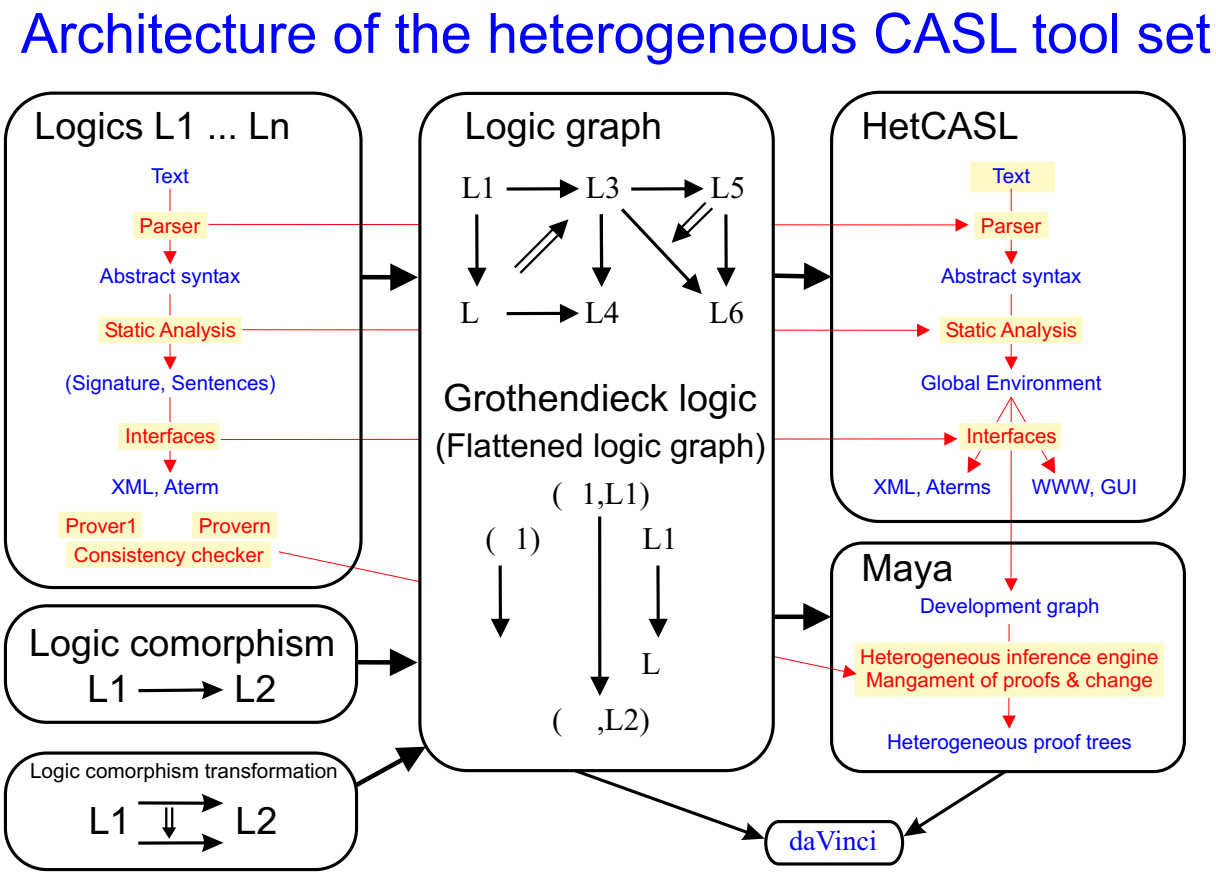


Abbildung 1: Übersicht über die zu entwickelnden Werkzeuge

3.2.6 Implementierung der Logiken des Logikgraphs

Die Logiken des Logikgraphs werden mittels einer Typklasse *Logic* in der Sprache Haskell implementiert. Die Typklasse umfasst Typen für die syntaktischen Bestandteile einer Meseguer-Logik [Mes89] (Signaturen, Signatur-Morphismen, Formeln, Formelübersetzungen) sowie weitere für abstrakte Syntax, Symbole und Teillogiken. Die Methoden der Typklasse umfassen u.a. Parsing, Pretty-Printing, statische Analyse von unstrukturierten Spezifikationen, Erkennen der Teillogiken sowie Aufruf von angeschlossenen Beweiswerkzeugen, wobei die Schnittstelle für letztere sich an dem in [AM02] entwickelten Beweiser-Interface orientiert. All diese Bestandteile müssen für die Logiken des Logik-Graphs bereitgestellt werden. Dabei soll soweit möglich auf existierende Werkzeuge zurückgegriffen werden, z.B. im Fall von CASL auf das CASL Tool Set, im Fall von HASCASL bzw. CSP-CASL auf die in den DFG-Projekten HasCASL und COOFL entwickelten Werkzeuge, für Z auf Z-Eves [CMS99]. Für andere Logiken, z.B. SB-CASL oder CASL-LTL, müssen noch Analysewerkzeuge entwickelt werden; diese können jedoch auf dem CASL Tool Set aufsetzen.

Die Werkzeuge sollen, sofern sie nicht in Haskell implementiert sind, über Haskells *foreign function interface* oder via Pipes über eine XML-Schnittstelle angeschlossen werden.

3.2.7 Implementierung der Repräsentationen und Repräsentations-Transformationen des Logikgraphs

Die Kanten des Logikgraphs entsprechen Logik-Repräsentationen (auch Comorphismen genannt) und -Projektionen; die *Kanten zwischen den Kanten* (in den Abbildungen im Zwischenbericht mit Doppelpfeilen dargestellt) entsprechen Repräsentations-Transformationen. Diese sollen als Haskell-Funktionen auf einem Teil der an der jeweiligen Quell- und Ziellogik beteiligten Datenstrukturen realisiert werden.

Ein schwieriges Problem ist hier eine geeignete Auswahl von partiellen Inversen der Signatur- und Formelübersetzung. Wenn ein Beweisziel entlang einer Repräsentation in eine andere Logik transportiert wird, möchte man in manchen Fällen dennoch das Beweisziel (und vor allem weitere, im Verlauf des Beweises entstehende Teilziele und Substitutions-Terme) in der Quelllogik dargestellt sehen. Da in der Regel die Signatur- und Formelübersetzung von Logik-Repräsentationen nicht vollständig, sondern eben nur partiell umkehrbar sind, stellt sich hier die Frage, ob gemischte Syntaxen entwickelt werden können (d.h. die rückübersetzbaren Teilterme bzw. -formeln werden in der Quelllogik dargestellt, die anderen dagegen in der Ziellogik). Noch besser ist natürlich die Verwendung von "Zero-distance"-Repräsentationen [Mes]; hier sind Syntax der Quell- und der Zielformel identisch (auch wenn sie in ihrer jeweiligen Logik durchaus für unterschiedliche Konzepte stehen können). Erfahrungsgemäß wird es jedoch nicht immer möglich sein, "Zero-distance"-Repräsentationen zu verwenden.

3.2.8 Darstellung des Logik- und Werkzeuggraphen

Der Logik-Graph soll zusammen mit den an die jeweiligen Logiken angeschlossenen Werkzeugen als ein kombinierter Logik- und Werkzeug-Graph mit dem in Bremen entwickelten Graph-Visualisierungs-Werkzeug daVinci [FW94] 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. Dabei sollen geeignete

Abstraktions-Sichten auf den Graphen entwickelt werden, um zu ermöglichen, aus dem ja recht großen Graphen relevante Teilinformationen darzustellen (vgl. auch die Abbildungen von Teilen des Logikgraphs im Zwischenbericht). 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]).

3.2.9 Parsing und statische Analyse der heterogenen Sprache

Für die heterogene Sprache soll ein Parser und eine statische Analyse entwickelt werden. Diese kümmert sich im Wesentlichen um die heterogene Strukturierung Im-Großen und ruft für die Logik-spezifischen Spezifikationen Im-Kleinen jeweils die in der Typklasse *Logic* vorhanden Logik-spezifischen Methoden auf. Für die heterogenen Strukturierungskonzepte wird die jeweilige Logik-Repräsentation aufgerufen.

Das heterogene Analysewerkzeug muss verschiedene Instanzen der Typklasse *Logic* gleichzeitig verwalten können, und zwar so, dass nicht für jede neue Logik der Code (etwa durch Erweiterung von entsprechenden Case-Expressions) wieder geändert werden muss. Diese Anforderung geht also über Generizität hinaus, wie sie etwa mit SML-Funktoren erreicht werden kann. Will man nicht eine Uniformität der Datenstrukturen aller Logiken erzwingen (was ein Verlust an Flexibilität bzw. zusätzlichen Konversions-Aufwand bedeuten würde), erfordert dies den Einsatz *heterogener Datenstrukturen*, die in Haskell mit existentiellen Typen realisiert werden können. So ist z.B. die Liste der Logiken eine heterogene Liste, deren Elemente unterschiedliche Typen haben können — die Typen sind jedoch alle Element der Typklasse *Logic* und haben damit die für eine Logik benötigten Funktionen verfügbar.

Die statische Analyse führt zu einem heterogenen *globalen Environment*. Diese ist die Basis sowohl für die Übersetzung in Entwicklungsgraphen zwecks Beweismanagement (siehe nächster Abschnitt), als auch für weitere Schnittstellen. U.a. soll es eine XML-Schnittstelle geben, die auch für das effiziente Abspeichern der Analyse-Informationen und für das Bibliotheksmanagement verwendet werden soll. Eine WWW-Schnittstelle soll den Aufruf des Analysetools aus dem Web heraus ermöglichen, ohne das man das Tool lokal installieren muss. Dies hat sich bereits beim CASL Tool Set als nützlich für die Gewinnung neuer Nutzer/innen herausgestellt.

3.2.10 Übersetzung von CASL in Entwicklungsgraphen

Die Übersetzung von CASL-strukturierten Spezifikationen in Entwicklungsgraphen aus [AHMS00] soll von Lisp nach Haskell portiert, um die Operation Verstecken erweitert und vor allem optimiert werden. Letzteres ist nötig, weil derzeit noch sehr viele prinzipiell eliminierbare interne Knoten erzeugt werden. Für die Übersetzung fehlt zudem noch ein Korrektheitsbeweis (dieser gehört natürlich genaugenommen zur konzeptionellen Ebene).

3.2.11 Heterogenes MAYA: Beweiskalkül

Das in Saarbrücken zum Beweismanagement mittels Entwicklungsgraphen entwickelte Werkzeug MAYA soll von Lisp nach Haskell portiert und gleichzeitig auf heterogene Entwicklungsgraphen mit Verstecken erweitert werden. Die Heterogenität ergibt sich über die Instantiierung mit der Grothendieck-Quotienten-Logik. Basierend auf den Ergebnissen aus Abschnitt 3.2.3 soll der Beweiskalkül für heterogene Entwicklungsgraphen imple-

mentiert werden. Zudem muss die Gleichheit auf Signaturmorphismen der Grothendieck-Quotienten-Logik implementiert werden, was keine triviale Angelegenheit ist (siehe 3.2.2).

3.2.12 Heterogenes MAYA: Anschluss von Beweisern

Für wichtige Logiken im Logik-Graph sollen gängige zur Verfügung stehende Beweiswerkzeuge mittels der in [AM02] beschriebenen XML-RPC-Schnittstelle an MAYA angeschlossen werden. Diese Schnittstelle erlaubt neben dem Senden von Theorien und Beweiszielen auch das (zunächst unstrukturierte) Abspeichern von Beweiser-spezifischen taktischen Informationen. Hier müssen möglicherweise auch noch Erweiterungen vorgenommen werden, so dass auch Beweiser-übergreifend Informationen über z.B. Rewriting-Strategien mit verwaltet werden.

Mittels des Beweiskalküls aus dem vorigen Abschnitt ergibt sich dann die Möglichkeit von heterogenen Beweisen. Natürlich wird die Ausstattung mit Beweiswerkzeugen sehr unterschiedlich sein: z.B. sollen für Logik erster und höherer Stufe verschiedene Beweiswerkzeuge angeschlossen werden, während für die meisten spezielleren Logiken, ja selbst CASL, kein eigenes Werkzeug bereitstehen wird, so dass hier auf Übersetzungen in Beweiser-unterstützte Logiken zurückgegriffen werden muss.

3.2.13 Heterogenes MAYA: Änderungsmanagement

Die Ergebnisse aus Abschnitt 3.2.4 sollen zu einer Implementierung eines Änderungsmanagements führen, das das bestehende Änderungsmanagement für homogene Entwicklungsgraphen ohne Verstecken auf den Fall der heterogenen Entwicklungsgraphen mit Verstecken erweitert.

3.2.14 Stufenweise Expansion von Theorien

MAYA erlaubt es, globale Beweisziele in lokale zu dekomponieren. Die lokalen Beweisziele sehen so aus, dass eine bestimmte Formel in einer Theorie bewiesen werden soll. In der Regel wird der Benutzer bzw. die Benutzerin die Theorie nicht zu einer unstrukturierten Theorie expandieren ("flachklopfen") wollen, da viele Beweiser Strukturierungsinformationen benutzen können. Jedoch ist die in MAYA zu einem Knoten des Entwicklungsgraphs assoziierte Theorie zu stark strukturiert: sie kann neben Umbenennungen auch Übersetzungen in andere Logiken enthalten. Beide Elemente werden von den meisten Beweisern aber nicht unterstützt. Deshalb soll eine Möglichkeit der stufenweisen Expansion ("Flachklopfen") einer Theorie implemeniert werden. So können zum Beispiel Umbenennungen und Logik-Übersetzungen expandiert werden, während Erweiterungen und Vereinigungen erhalten bleiben. Dagegen kann das Verstecken nicht einbezogen werden, weil hier keine semantik-erhaltende Expansion möglich ist — es muss stets mit den speziellen Entwicklungsgraphen-Kalkülregeln für Verstecken behandelt werden.

3.2.15 Gesamt-Integration, Benutzerschnittstelle und Test

Erfahrungsgemäß ist bei der doch recht komplexen Architektur des Gesamtsystems ein eigener Schritt notwendig, um die verschiedenen Teilsysteme zu testen und zu einem

funktionierenden Gesamtsystem zu integrieren. Zudem muss auch noch eine Benutzerschnittstelle entwickelt werden. Das Gesamtsystem soll dann mittels der in Abschnitt 3.2.5 beschriebenen 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 zwei Jahre auf zwei vollen BAT IIa-Stellen beantragt. Es ist vorgesehen, für eine dieser Stellen Herrn Dr. Till Mossakowski weiterzubeschäftigen, der den theoretischen und konzeptionellen Teil des Projekts und auch schwierige Teile der Werkzeugentwicklung weitgehend selbständig durchführen soll. Zudem soll er den anderen vorgesehenen Mitarbeiter, Herrn Klaus Lüttich, bei der Entwicklung der Werkzeuge anleiten, mit der Möglichkeit zur Promotion. Beide Mitarbeiter werden derzeit bereits im Rahmen des Projektes MULTIPLE beschäftigt.

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 hier nur für eine volle Stelle ein entsprechend qualifizierter Bewerber/Bewerberin zu bekommen.

Zusätzlich werden zur Unterstützung der Implementierungsarbeiten für die gesamte Dauer von zwei Jahren Mittel für zwei studentische Hilfskräfte im Umfang von 40 Stunden monatlich beantragt.

4.2 Wissenschaftliche Geräte

entfällt

4.3 Verbrauchsmaterial

entfällt

4.4 Reisen

Zur Zusammenarbeit mit den unter 5.2 genannten Wissenschaftlern werden Reisemittel in Höhe von insgesamt 4.500 Euro beantragt.

4.5 Publikationskosten

entfällt

4.6 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. Lutz Schröder und Dr. Christian Maeder (beide DFG-Projekt HasCasl), Dr. Markus Roggenbach (CSP-Casl), Dr. Christoph Lüth (Isabelle/HOL, generische grafische Benutzerschnittstelle) und Dr. George Russell (UniForM Workbench) an Fragestellungen, die eng mit dem beantragten Projekt zusammenhängen. Axel Lankenau arbeitet in DFG-Projekt SafeRobotics, das hier für die Fallstudie Bedeutung hat.

5.2 Zusammenarbeit mit anderen Wissenschaftlern

1. Prof. Dr. Andrzej Tarlecki, Polnische Akademie der Wissenschaften und Universität Warschau (heterogene Sprache)
2. Dr. Hélène Kirchner, LORIA-CNRS & INRIA Lorraine, Nancy (Termersetzung und Beweisen)
3. Prof. Dr. Tobias Nipkow, TU München (Isabelle)
4. Dr. Dieter Hutter, Serge Autexier, DFKI GmbH, Saarbrücken (Maya)

5.3 Arbeiten im Ausland und Kooperation mit ausländischen Partnern

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 Arbeitsgruppe ist mit leistungsfähigen Rechnern ausgestattet, die für das beantrage Projekt genutzt werden können.

5.5 Laufende Mittel für Sachausgaben

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

5.6 Sonstige Voraussetzungen

Schreib- und Organisationsarbeiten können von den Sekretariaten der Arbeitsgruppen der Antragsteller in angemessenem 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 Unterschriften

Bremen, den

Prof. Dr. B. Krieg-Brückner

Prof. Dr. H.–J. Kreowski

8 Verzeichnis der Anlagen

- Zwischenbericht für das Projekt MULTIPLE
- Dr. Till Mossakowski: Verzeichnis der wissenschaftlichen Veröffentlichungen seit 1997
- Dr. Till Mossakowski: Ausgewählte Publikationen: siehe Zwischenbericht
- Prof. Bernd Krieg-Brückner: Verzeichnis der wissenschaftlichen Veröffentlichungen seit 1997
- Prof. Hans-Jörg Kreowski: Verzeichnis der wissenschaftlichen Veröffentlichungen seit 1997

Literatur

[AHMS00] S. Autexier, D. Hutter, H. Mantel, and A. Schairer. Towards an evolutionary formal software-development using CASL. In C. Choppy and D. Bert, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France*, volume 1827 of *Lecture Notes in Computer Science*, pages 73–88. Springer-Verlag, 2000.

[AM02] S. Autexier and T. Mossakowski. Integrating HOL-CASL into the development graph manager MAYA. In A. Armando, editor, *Frontiers of Combining Systems, 4th International Workshop*, volume 2309 of *Lecture Notes in Computer Science*, pages 2–17. Springer-Verlag, 2002.

[CMS99] D. Craigen, I. Meisels, and M. Saaltink. Analysing Z specifications with Z/Eves. In J.P. Bowen and M.G. Hinchey, editors, *Inustrial-strength formal methods in practice*. Springer Verlag, 1999.

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

[FW94] M. Fröhlich and M. Werner. Demonstration of the interactive graph visualization system daVinci. In R. Tamassia and I. Tollis, editors, *Proceedings of the DIMACS workshop on graph drawing*, volume 894 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Kar99] E. Karlsen. *Tool Integration in a Functional Programming Language*. PhD thesis, Universität Bremen, 1999. Revised version, BISS Monographs Vol. 12.

[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.

[LR01] A. Lankenau and T. Röfer. The Bremen Autonomous Wheelchair – a versatile and safe mobility assistant. *IEEE Robotics and Automation Magazine, "Reinventing the Wheelchair"*, 7(1):29 – 37, Mar. 2001.

[Mes] J. Meseguer. A total approach to partial algebraic specification. Talk at the IFIP WG 1.3 meeting, L'Alpe d'Huez, January 2002.

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

[Mos] T. Mossakowski. Comorphism-based grothendieck logics. MFCS 2002, to appear.

[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.

[RL00] T. Röfer and A. Lankenau. Architecture and applications of the Bremen Autonomous Wheelchair. *Information Sciences*, 126(1-4):1 – 20, Jul. 2000.

[Sch99] L. Schröder. *Composition graphs and free extensions of categories*. PhD thesis, University of Bremen, 1999. in German; also: Logos, Berlin, 1999.

[ST88] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica*, 25:233–281, 1988.

Weitere Literatur siehe Zwischenbericht.