Programmierung, Spezifikation und Interaktives Beweisen

Auf dem Weg zu einer einheitlichen Sprache basierend auf Gleichungslogik, Termerzetzungslogik und Typtheorie

Mark-Oliver Stehr*
Universität Hamburg, Fachbereich Informatik,
Arbeitsbereich Theoretische Grundlagen der Informatik,
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany.
stehr@informatik.uni-hamburg.de

Abstract: Diese Dissertation mit dem englischen Titel "Programming, Specification, and Interactive Theorem Proving – Towards a Unified Language based on Equational Logic, Rewriting Logic, and Type Theory" beschäftigt sich mit dem Problem der Inflation von Formalismen in der Informatik im Kontext eines Spektrums formaler Methoden, das von Ausführung, über Analyse, bis zur formalen Verifikation reicht. Durch ihre Repräsentation in semantischen und logischen Rahmenwerken (semantic and logical frameworks), wie der Gleichungslogik (equational logic), der Termersetzungslogik (rewriting logic), oder der Typtheorie, wird ein Beitrag zum besseren Verständnis der Formalismen sowie ihrer Beziehungen untereinander geliefert. Konkret behandeln wir verschiedene Klassen von Petrinetzen, die UNITY-Temporallogik, das λ-Kalkül, Abadi und Cardellis ς-Kalkül, Milners π-Kalkül, sowie verschiedene logische Typtheorien. Gleichzeitig studieren wir interessante Verallgemeinerungen der repräsentierten Formalismen und weisen die Praxistauglichkeit des formalen Rahmens durch eine Reihe von Anwendungen nach. In einem weiteren Vereinheitlichungsschritt wird ein neues Rahmenwerk, das Kalkül der offenen Konstruktionen (open calculus of constructions), eingeführt, das die Ideen der Gleichungslogik, der Termersetzungslogik, und der Typtheorie in einer relativ einfachen Sprache zusammenführt. Der Einsatz als Programmier- und Spezifikationssprache, sowie als Formalismus zum interaktiven Beweisen, wird anhand eines Prototyps und zahlreicher Beispiele demonstriert.

1 Einleitung

Informatiker werden im Alltag mit einer zunehmend wachsenden Zahl von Beschreibungstechniken konfrontiert, die von informalen Diagrammen (Beispiele: Use-Case-Diagramme, Sequenz-Diagramme, Ereignisgesteuerte Prozessketten) über semi-formale Sprachen (Beispiele: C++, Java, Perl) bis hin zu formalen Programmier- und Spezifikationssprachen (Beispiele: ML, VHDL, Statecharts, Petrinetze, Z) reichen. Diese Beschreibungstechniken bilden die Grundlage für die heutzutage unverzichtbare Verwendung von rechnergestützten

*Zur Zeit tätig an der University of Illinois at Urbana-Champaign, Computer Science Department, Urbana, IL 61801, USA, e-mail: stehr@cs.uiuc.edu.

Werkzeugen, z.B. zur Unterstützung von Entwurf, Test, Simulation, Analyse und Verifikation von Programmen und Systemmodellen. Speziell beim Entwurf von Systemen mit einem hohen Grad an Nichtdeterminismus und Nebenläufigkeit ist bekannt, das herkömmliche Methoden des Testens nicht ausreichen, um einen akzeptablen Grad an Korrektheit zu erreichen. Andererseits ist eine vollständige Verifikation oft unmöglich, z.B. weil eine formalen Spezifikation des Gesamtsystems nicht existiert, oder nicht sinnvoll, z.B. weil nur wenige kritische Systemkomponenten oder -eigenschaften von Interesse sind.

Einen Kompromiß stellen die so genannten leichtgewichtigen formalen Methoden (leightweight formal methods) dar, die geeignet sind mit relativ wenig Aufwand fundamentale Fehler frühzeitig im Entwicklungsprozess aufzuspüren. Tatsächlich lässt sich das Entstehen eines ganzen Spektrums formaler Methoden beobachten, das von den leichtgewichtigen Methoden (light-weight methods), die zahlreiche Arten der statischen Analyse (z.B. Typprüfung) und der dynamischen Analyse (z.B. Zustandsraumexploration und Modellprüfung) einschliessen, bis hin zur schwergewichtigen Methoden (heavy-weight methods) reicht, wie die Verifikation mit Hilfe von deduktiven Methoden und Theorembeweisern. Idealerweise werden schwere Fehler früh und mit geringen Kosten mit Hilfe der schwächeren Methoden entdeckt, so das sich der Einsatz von stärkeren und aufwendigeren, z.B. deduktiven, Methoden nur auf relativ gut verstandene und mit weniger Fehlern behaftete Entwürfe beschränkt. Im gesamten Spektrum ist es ferner möglich, und aufgrund der hohen Komplexität üblicherweise notwendig, durch informale oder formale Abstraktionstechniken den Fokus auf spezielle Teilaspekte zu richten, die z.B. durch kritische Systemkomponenten oder kritische Systemeigenschaften charakterisiert werden. Ferner ist oft möglich die Verifikation relativ zu plausiblen Hypothesen zu strukturieren, die von der formalen Behandlung ausgeklammert werden, weil sie z.B. als mathematisches Basis- oder Hintergrundwissen gelten, als unkritisch empfunden werden, oder mit anderen Methoden behandelt werden sollen.

Das Ziel des Einsatzes formaler Methoden sollte es unserer Meinung nach sein, mit geringem Aufwand eine möglichst hohen Grad an Korrektheit zu erreichen anstatt eine vollständige und damit kostspielige Verifikation zu betreiben. Eine weitere, und über die Korrektheit hinausgehende, Motivation für die Verwendung von leichtgewichtigen formalen Methoden ist, daß bei heutigen Softwaresystem die Komplexität des Testens so große Maßstäbe erreicht hat, das es wirtschaftlich problematisch ist fundamentale Fehler erst spät in der Testphase zu erkennen.[1] Ein wichtiges Anliegen der aktuellen Forschung ist es deshalb Methoden und Werkzeuge zu entwickeln, die es erlauben möglichst viele Fehler früh zu erkennen, idealerweise bevor zeitraubende Tests begonnen werden.

Zusammenfassend ist es unserer Meinung nach erstrebenswert durch den gezielten und systematischen Einsatz des gesamten Spektrums formaler Methoden die Qualität von Informatiksystemen zu steigern. Eine wichtige Forschungsaufgabe ist konsequenterweise die Entwicklung von Techniken und Werkzeugen die es dem Informatiker erlauben mit möglichst wenig Aufwand fließend zwischen verschiedenen formalen Methoden zu wechseln. Da die Systeme in der Praxis heterogen und oft durch eine Kombination verschiedener Beschreibungstechniken entworfen werden, ist eine wichtige Anforderung die Verwen-

[1] Als aktuelles Beispiel sei hier das Betriebssystem Windows von Microsoft genannt, für das der komplette Testprozess schon mehrere Monate beansprucht [Ra03].

dung einer (Meta-)Beschreibungstechnik, die ausreichend allgemein ist, um die praktisch relevanten Beschreibungstechniken mit geringen Aufwand zu repräsentieren.

2 Übersicht

Das Ziel der Dissertation [St02b], die im folgenden übersichtsartig dargestellt werden soll, ist es die formalen Grundlagen einer solchen (Meta-)Beschreibungstechnik zu entwickeln und deren Anwendungen zu untersuchen.

Dabei wird zunächst der Einsatz bereits erfolgreicher Techniken, der Gleichungslogik (equational logic) [GM92, BJM00], der Termersetzungslogik (rewriting logic) [Me92], und der Typtheorie [Lu94, PSN90], als semantische und logische Rahmenwerke (semantic and logical frameworks) im Sinne von [Me96, MOM94, Me98, MOM96] in verschiedenen Anwendungsbereichen studiert. Es wird gezeigt wie die Anwendungsbereiche von diesem Einsatz profitieren, wie neue Bereiche abgedeckt werden können, aber auch wo die Beschränkungen der aktuellen Techniken liegen. Diese Beiträge sind Gegenstand der Kapitel 3 – 7 der Dissertation und werden zusammenfassend in den folgenden Abschnitten 3 – 7 dargestellt. Da sie aber neue und sehr allgemeine Repräsentationstechniken entwickeln, sind sie auch völlig unabhängig vom Hauptziel dieser Dissertation von Interesse.

Ausgehend von den in diesen Anwendungen gesammelten Erfahrungen wird schließlich in Kapitel 8 der Dissertation ein Formalismus entwickelt, der auf den Hauptcharakteristika der oben erwähnten recht unterschiedlichen Forschungsrichtungen, also der Gleichungslogik, der Termersetzunglogik, und der Typtheorie, basiert. Der resultierende Formalismus, den wir als offenes Kalkül der Konstruktionen (open calculus of constructions) bezeichnen, ist als erster Schritt in Richtung unseres langfristigen Ziels, der Entwicklung einer einheitlichen Sprache zur Programmierung, Spezifikation, und interaktivem Theorembeweisen, zu verstehen. Das Kalkül deckt wichtige wichtige Teile des Spektrums formaler Methoden ab und eignet sich wie die Ausgangsformalismen als semantisches und logisches Rahmenwerk. Zu einer Zusammenfassung dieses Teils der Dissertation kommen wir schließlich in Abschnitt 8 dieses Papiers.

3 Termersetzungslogik als semantisches Rahmenwerk: Repräsentation von höheren Petrinetzen

Unsere Arbeit beginnt mit der Untersuchung der Anwendbarkeit von Termersetzungslogik als semantisches Rahmenwerk (semantic framework) für Nebenläufigkeit. Hierzu geben wir eine einheitliche Behandlung verschiedener Petrinetz-Modelle, eines typischen und wichtigen Repräsentanten einer Klasse von Formalismen, die zur Modellierung und Spezifikation von nebenläufigen und verteilten Systemen benutzt werden und auf einer Multimengen-Repräsentation des verteilten Zustandsraumes basieren.

Speziell führen wir die Forschungsrichtung fort, die von Meseguer und Montanari un-

ter dem Motto “Petrinetze sind Monoide” [MM90, DMM96] initiiert wurde, indem wir eine Termersetzungssemantik für verschiedene Petrinetz-Klassen angeben. Insbesondere decken wir wichtige höhere (high-level) Petrinetz-Modelle, genauer algebraische Netzspezifikationen [Re91] und gefärbte Petrinetze [Je92] ab, und wir beweisen, daß die Modelle unserer Repräsentationen eine natürliche Isomorphie (im kategorientheoretischen Sinne) zur bekannten Prozess-Semantik von Best und Devillers [BD87] aufweisen. Zusätzlich zu ihrem Beitrag zu einer konzeptuellen Vereinheitlichung ist der wichtigste praktische Vorteil unserer Repräsentationen in Termersetzungslogik ihre Ausführbarkeit, die es uns erlaubt, moderne Termersetzungsmaschinen wie z.B. Maude [CDE$^+$99a, CDE$^+$00, EMS02] zur hocheffizienten symbolischen Ausführung von Systemmodellen sowie zu deren Analyse (Zustandsraumexploration, Modellprüfung, e.t.c) zu nutzen.

Die Resultate dieses Kapitels verallgemeinern die Resultate aus [MM90] und basieren auf unserer Arbeit [St03g] über die Repräsentation von algebraischen Netzspezifikationen. Diese Arbeit wurde verfeinert und fortgeführt in [SMÖ01b, SMÖ01a], um Petrinetze mit Test-Kanten und zeitbehaftete Petrinetze abzudecken. Eine sehr ausführliche Ausarbeitung mit unabhängigen Beweisen für den grundlegenden Fall der konventionellen Petrinetze findet sich in [COS03].

4 Metalogisches Beweisen im Kalkül der induktiven Konstruktionen: Eine formalisierte Verallgemeinerung von UNITY

Die nächste Anwendung, die in dieser Arbeit behandelt wird, betrifft die Verwendung der Typtheorie, genauer des induktiven Kalküls der Konstruktionen (inductive calculus of constructions) [BBC$^+$99] als logisches Rahmenwerk (logical framework) und für metalogische Beweise. In diesem Kapitel haben wir den COQ-Beweisassistenten [BBC$^+$99] in einer rigoros formalen Entwicklung einer Temporallogik im UNITY-Stil [CM88] (genauer im Stil von New UNITY [Mi94, Mi95]) eingesetzt, die den ursprünglichen Ansatz in wichtigen Punkten verallgemeinert.

Da die Inferenzregeln der Temporallogik als Theoreme in der Metalogik abgeleitet wurden, ist das Resultat dieser Entwicklung eine verifizierte Temporallogik-Bibliothek, die dank der Verwendung von allgemeinen, beschrifteten Transitionssystemen als semantische Basis für ein weites Spektrum von Systemmodellen (Petrinetze und Spezifikationen in Termersetzungslogik sind konkrete Beispiele) eingesetzt werden kann.

Unsere Entwicklung enthält u.a. eine neuartige Anwendung der Interpretation von logischen Formeln als Typen (propositions-as-types interpretation) [Ho80] im Kontext der kompositionalen Verifikation. Das Resultat ist eine neue Beweisregel für die Komposition von UNITY-Lebendigkeitseigenschaften (leads-to-Zusicherungen) in eng gekoppelten Systemen (loosely coupled systems). Die Eleganz mit der dieses Resultat erzielt wird, speziell die notwendige Formalisierung des syntaktisch relativ aufwendigen Begriffs der Interferenzfreiheit, lässt sich durch die typtheoretische Behandlung von Beweisen als Objekte erster Klasse (proofs-as-objects interpretation) besonders elegant realisieren.

Abgesehen von zusätzlichen Beweisregeln für relativierte Zusicherungen und zur kompo-

sitionalen Verifikation, unterscheidet sich unserer Ansatz von den UNITY-Formalisierungen anderer Autoren [ABN$^+$95, HC96, Pa00] in der Allgemeinheit des Systemmodells. Ferner verallgemeinert dieses Kapitel unsere frühere Formalisierung von UNITY [St98b]. Wir konnten trotz des Übergangs zu allgemeinen beschrifteten Transitionssystemen (mit Fairnessspezifikationen) die Eleganz von UNITY, die auf dessen induktiven Charakter beruht, weitgehend erhalten. Daß die Spezifikation und Verifikation auch in einem stark getypten Kontext (dank einer ausdrucksstarken Typtheorie) praktikabel ist, ist bemerkenswert in Hinblick auf die Kritik in [La92]. Besonders interessant ist ferner, daß die metatheoretischen Beweise, d.h. die Beweise der temporalen Inferenzregeln, völlig unabhängig von der unterliegenden (operationalen) Semantik sind.

Als Anwendung der Resultate dieses Kapitels verweisen wir auf [St03a, St98a], wo gezeigt wird wie sich unsere Temporallogik-Bibliothek zur Verifikation von Petrinetzen spezialisieren lässt. Weiterhin ist interessant zu erwähnen, daß sich die UNITY-Logik mit ihrer langen Geschichte, die auf u.a. auf [Ho69] und [DS90] zurückgeht, praktisch unverändert in dem neuen Ansatz zur Multiprogrammierung [Mi01] wiederfindet. Durch die Unabhängigkeit von der unterliegenden Programmiersprache bleibt unsere Formalisierung damit auch heute noch aktuell.

5 Repräsentation von Formalismen höherer Ordnung: Ein Kalkül der Namen und Substitutionen

Der Einsatz von Membership-Gleichungslogik (membership equational logic) [BJM00] oder Termersetzungslogik (rewriting logic) [Me92] als semantisches und logisches Rahmenwerk (semantic and logical framework) für Sprachen höherer Ordnung, oder allgemeiner für Sprachen mit Konstrukten zur Namensbindung, benötigt offensichtlich eine Behandlung von Namen und relevanten Operationen wie Substitutionen mit Mitteln erster Ordnung. Um solche Anwendungen systematisch anzugehen, haben wir CINNI entwickelt, ein neues Kalkül der Namen und Substitutionen, das Namen in dem Sinne respektiert, daß es nicht von ihnen abstrahiert, und das ferner generisch ist, in dem Sinne, daß es für nahezu beliebige Objekt-Sprachen instantiiert werden kann. Unsere Arbeit läst sich damit in den Bereich der expliziten Substitutionskalküle einordenen, eine inzwischen etablierte Forschungsrichtung, die u.a. durch die Arbeit [ACCL91] initiiert wurde.

Unser Kalkül vereinheitlicht die übliche Notation mit Namen und die namenslose Notation basierend auf de-Bruijn-Indizes [dB72], indem es eine Repräsentation benutzt, die ursprünglich von Berkling für das λ-Kalkül entwickelt wurde [Be76, BF82]. Ferner verallgemeinert CINNI das Kalkül $\lambda\upsilon$ der expliziten Substitutionen von Lescanne [Le94] und, wie wir zeigen, lassen sich die meisten metatheoretischen Resultate verallgemeinern. Schließlich beweisen wir ein sehr allgemeines Konfluenz-Resultat für die Komposition von CINNI mit Gleichungen und Regeln, die die Dynamik der Objektsprache widerspiegeln, und wir diskutieren, wie unserer Ansatz zur Repräsentation des λ-Kalküls, Abadi und Cardellis Objekt-Kalküls [AC96], auch als ς-Kalkül bezeichnet, und Milners π-Kalküls [Mi99] für kommunizierende und mobile Systeme verwendet werden kann.

Dieses Kapitel erweitert [St00] um weitere metatheoretische Resultate und beschreibt den Einsatz von CINNI in einer Anwendung aus der Praxis, der Spezifikation einer Programmiersprache für aktive Netzwerke [ST02c, MÖST02] in der auf Termersetzungslogik basierten Sprache Maude [CDE+99a, CDE+00]. Ferner wurde die Repräsentation des π-Kalküls mittels CINNI inzwischen von anderen Autoren zur Entwicklung von formalen Analysewerkzeugen eingesetzt [TSMO02]. Auch in einem ganz anderen Bereich, der Entwicklung von MuCAPSL, einer Spezifikationssprache für Multicast-Authentifizierungsprotokolle, wurde das CINNI-Kalkül bei der Implementierung erfolgreich verwendet [DM03]. Anwendungen in der Typtheorie und Logik sind Gegenstand der verbleibenden beiden Kapitel dieser Dissertation.

6 Termersetzungslogik als logisches Rahmenwerk: Repräsentation von reinen Typsystemen

Hier untersuchen wir die Verwendung von Membership-Gleichungslogik (membership equational logic) [BJM00] und Termersetzungslogik (rewriting logic) [Me92] als logisches Rahmenwerk erster Ordnung (first-order semantic framework), indem wir eine wichtige Klasse der reinen Typsysteme (pure type systems) [Be88, Te89, Ba92] repräsentieren. Reine Typsysteme verallgemeinern den λ-Würfel (λ-cube) [Ba92], eine wichtige Klasse von Typtheorien, die das Kalküls der Konstruktionen (calculus of constructions) [CH88] und seine bekannten Teilsysteme [Ch40, HHP87, Gi72, Re74] enthalten. Geeignete reine Typsysteme können ferner als Logiken höherer Ordnung entsprechend der Interpretation von logischen Formeln als Typen (propositions-as-types interpretation) [Ge93, Ho80] angesehen werden.

Unter Verwendung eines methodischen Ansatzes, der auf Meseguers allgemeinen Logiken (general logics) [Me89] in Kombination mit Termersetzunglogik als konkretes logisches Rahmenwerk basiert, haben wir Repräsentationen von reinen Typsystemen auf verschiedenen Abstraktionsebenen studiert, angefangen von einer abstrakten lehrbuchartigen Repräsentation bis hin zu einer konkreteren und ausführbaren Repräsentation einer wichtigen Unterklasse, die in sehr direkter Weise als Typinferenz- und Typprüfungsalgorithmus genutzt werden kann. Dabei wird die Korrektheit jedes einzelnen Verfeinerungsschrittes in dieser Kette bewiesen. Die letztere Repräsentation basiert auf einem neuen Begriff der einheitlichen reinen Typsysteme (uniform pure type systems), die dank der Verwendung des CINNI-Kalküls Namen respektieren, und gleichzeitig eine mögliche Lösung zum bekannten Problem der α-Abgeschlossenheit darstellen, auf das Pollack bereits in [Po93] hinwies.

Dieses Kapitel ist eine überarbeite Fassung von [St99, SM99] und erscheint in leicht erweiterter Form in [SM03]. Die Dissertation demonstriert darüber hinaus die Praktikabilität anhand der Validation von Beweisen, die mit dem LEGO-Beweisassistent in einer Erweiterung des Kalküls der Konstruktionen um Universen ausgeführt wurden. Damit zeigen wir, wie unser Ansatz unmittelbar zu einem ausführbaren Prototyp eines Beweisprüfers in der auf Termersetzungslogik basierenden Sprache Maude [CDE+99a, CDE+00] führt.

7 Klassische Logik durch Typtheorie: Ein beweistheoretischer Ansatz zur HOL/Nuprl-Verbindung

Als Einsatz der Typtheorie im Kontext des Beweisens in klassischer Logik studieren wir Howes HOL/Nuprl-Verbindung [Ho96, Ho98] in diesem Kapitel, das sich mit dem Problem der formalen Interoperabilität zwischen Beweisassistenten aus der Sicht von Meseguers allgemeinen Logiken (general logics) [Me89] beschäftigt. HOL und Nuprl sind grundverschiedene Systeme: HOL [GM93] basiert auf klassicher Logik höherer Ordnung und Nuprl [CAB+86] ist eine Weiterentwicklung von MartinLoefs polymorpher Typtheorie [PSN90]. Es ist deshalb sehr bemerkenswert, daß Howe in [Ho97] eine klassische mengentheoretische Semantik für Nuprl konstruieren konnte.

In diesem Kapitel ergänzen wir diese semantische Rechtfertigung für die Verbindung von HOL und Nuprl durch ein beweistheoretisches Korrektheitsargument, eine Arbeit die Beweisübersetzung als neue interessante Anwendung hat und damit über Howes urspüngliche HOL/Nuprl-Verbindung hinausgeht. Aus theoretischer Sicht fanden wir, daß die Kernidee der HOL/Nuprl-Verbindung, nämlich die Koexistenz einer intensionalen und einer extensionalen Logik in einem einzigen formalen System, nicht auf die speziellen Eigenschaften von Nuprl angewiesen ist, sondern ebenso in Martin-Löfs Typtheorie [PSN90] realisiert werden und ferner leicht an Typtheorien auf der Linie des Kalküls der Konstruktionen angepasst werden kann.

Dieses Kapitel basiert auf der theoretischen Arbeit [SNM01] und führte zu zwei interessanten Anwendungen, die ebenfalls kurz beschrieben werden: Zum einen demonstrierten wir den Einsatz von Maude [CDE+99a, CDE+00] als formales Meta-Werkzeug, eine wichtige Anwendung, über die bereits in [CDE+99b] berichtet wurde. Hierzu entwickelten wir eine formale Spezifikation der Abbildung zwischen den beteiligten Logiken unter Verwendung von Membership-Gleichungslogik (membership equational logic) [BJM00] als ausführbare Metalogik, und erhielten so einen praktikablen Logikübersetzer mit ähnlicher Funktionalität wie der von Howe entwickelte. Zum anderen beobachteten wir, daß es sich bei unserem metalogischen Korrektheitsbeweis in [SNM01] um einen rein konstruktiven Beweis handelt. Damit war die Idee geboren den berechnungsrelevanten Inhalt (computational contents) dieses Beweises zu implementieren. Diese Arbeit wurde basierend auf [SNM01] von Pavel Naumov durchgeführt und führte zu einer Erweiterung den Nurpl-Systems um einen Beweisübersetzer, d.h. um eine sehr praktische Funktion, die es erlaubt HOL-Beweise zu importieren und in Nurpl-Beweise zu überführen [NSM01].

8 Auf dem Weg zu einer einheitlichen Sprache: Das offene Kalkül der Konstruktionen

Der letzte und Hauptbeitrag dieser Arbeit ist die Entwicklung eines Formalismus, den wir als offenes Kalkül der Konstruktionen (open calculus of constructions, OCC) bezeichnen. Er basiert auf der überraschend mächtigen Interaktion zwischen seinen beiden Hauptcharakteristika, nämlich abhängigen Typen im Sinne von Martin-Löfs Typtheorie [PSN90]

und des Kalküls der Konstruktionen (calculus of constructions) [Lu94, CH88], und dem Berechnungssystem der Termersetzungslogik (rewriting logic) [Me92] und ihrer unterliegenden Membership-Gleichungslogik (membership equational logic) [BJM00], das auf bedingter Termersetzung modulo Gleichungstheorien (rewriting modulo) basiert. Die Anwendungen von Membership-Gleichungslogik, Termersetzungs-Logik und Typtheorie, die wir in dieser Arbeit studierten, waren nicht nur eine wichtige Inspiration für die Entwicklung dieses einheitlichen Formalismus, sondern werden selbst zu Anwendungen von OCC und profitieren wesentlich von seinem Einsatz.

Auf der theoretischen Seite führen wir OCC durch Angabe einer klassischen, mengentheoretischen Semantik und eines formalen Systems ein, für das wir Korrektheit und logische Konsistenz beweisen. Das formale System wird benutzt, um die ableitbaren Urteile und ihre operationale Semantik zu definieren und basiert auf den Ideen, die wir in den vorigen Kapiteln im Kontext der expliziten Substitutionen (CINNI) und einheitlichen, reinen Typsysteme (uniform pure type systems) entwickelt haben. Die modelltheoretische Semantik, die wir in dieser Arbeit entwickeln, ist eine sehr intuitive Semantik mit Beweis-Irrelevanz (proof-irrelevance semantics) für imprädikative Universen, aber anders als in existierenden Ansätzen ist sie direkter und kann unabhängig vom formalen System angegeben werden.

Unter Verwendung eines experimentellen Prototyps von OCC, den wir mit Hilfe des obigen Ansatzes zur Spezifikation von Typtheorien, also der Nutzung von Rewriting-Logik als logisches Rahmenwerk, in Kombination mit reflektiven Techniken in Maude [CDE$^+$99a, CDE$^+$00] entwickelt haben, wurde eine umfangreiche Sammlung von Beispielen erstellt, von denen viele eng mit den vorher diskutierten Anwendungen zusammenhängen. Die Beispiele vermitteln nicht nur die Pragmatik von OCC sondern zeigen gleichzeitig die Realisierbarkeit und den Nutzen seiner Konzepte. Unter den Themen, die beispielsartig behandelt werden, finden sich ausführbare, gleichungsbasierte und verhaltensorientierte Spezifikationen, Programmierung mit abhängigen Typen, symbolische Ausführung von Systemmodellen, Formalisierung von algebraischen und kategorientheoretischen Begriffen, induktives/coinduktives Theorembeweisen und Beweisen modulo Gleichungstheorien.

Interessanterweise findet sich die Idee Membership-Gleichungslogik, Rewriting-Logik, und Typtheorie zu integrieren — wenn auch mit einer anderen Motivation — bereits in [Jo98]. Allerdings basiert das in [BJO99] vorgeschlage Kalkül der algebraischen Konstruktionen (calculus of algebraic construction) immer noch auf relativ starken syntaktischen Restiktionen um starke Normalisierung zu erhalten, eine Eigenschaft die mit unseren Ansatz der allgemeinen ausführbaren Spezifikation kaum verträglich ist. Zusätzlich zu aktuellen Bestrebungen typtheoretische und algebraische Ansätze zu kombinieren, ist ein ganz anderer Aspekt unseres Ansatzes — und ebenfalls ein Gegenstand aktueller Forschung [DHK99, BKR01] — die Reduktion von Konzepten höherer Ordnung auf Konzepte erster Ordnung. Diese und weitere Bezüge zu anderen Ansätzen und Systemen finden sich in der Dissertation [St02b] und in dem Tutorial [St02a].

9 Zusammenfassung und Ausblick

Diese Dissertation beschäftigt sich mit dem systematischen Einsatz ausgewählter formaler Beschreibungstechniken, speziell der Gleichungslogik, der Rewriting-Logik, und der Typtheorie. Besondere Beachtung finden dabei zwei orthogonale Dimensionen: die Vielfältigkeit der Beschreibungsformalismen und das Spektrum von leichtgewichtigen zu schwergewichtigen Methoden. Neben zahlreichen methodologischen und anwendungsspezifischen Resultaten ist der Hauptbeitrag die Entwicklung und die theoretische sowie experimentelle Untersuchung des offenen Kalkül der Konstruktionen (open calculus of constructions), eine Sprache die wesentliche Aspekte der ausgewählten Beschreibungstechniken integriert.

Zwei sehr praxisnahe Projekte die unmittelbar Anwendungen der Verfahren und Resultate dieser Dissertation darstellen wurden bereits erwähnt: Die formale Spezifikation einer Programmiersprache für aktive Netzwerke in Maude [ST02c, MÖST02] und die Implementation eines Logik- und Beweisübersetzers von HOL in Nuprl [NSM01]. Eine weitere wichtige Anwendung unserer Resultate ist die systematische Implementierung von Typtheorien bzw. Logiken auf Basis ihrer formalen Systeme, was es uns beispielsweise ermöglichte einen Prototypen für das offene Kalkül der Konstruktionen zu erstellen.

Um einen Ausblick auf potentielle zukünftige Anwendungen der in dieser Dissertation eingesetzten und weiterentwickelten Techniken zu gegeben, seien kurz einige laufende Kooperationsprojekte mit anderen Forschern vorgestellt: Erwähnenswert ist zum einen die Entwicklung eines Protokolls zur dynamischen Konfiguration von Sicherheitsassoziationen (security associations) und Sicherheitsvorschriften (security policies) [GGM03], auf denen IPSec (die Sicherheitsarchitektur für der Internet) basiert. Auf Basis einer ausführbaren Spezifikation in Rewriting-Logik konnten mit Hilfe von Maude [CDE$^+$99a, CDE$^+$00, EMS02] frühzeitig unerwünschte Interferenzen lokalisiert werden und verschiedene Protokollvarianten entworfen und analysiert werden [St03d]. Eine ganz andere Situation liegt dagegen bei einem aktuellen Projekt vor, bei dem es um die formale Spezifikation von Middleware in Rewriting-Logik geht, genauer um Middleware zur Gruppenkommunikation. Hier soll die formale Spezifikation das Verhalten des bereits implementierten Systems “Spread” [AS98] beschreiben, und damit eine präzise formale Schnittstelle zwischen Anwendung und Middleware liefern [St03c]. Eine formale Spezifikation eines Protokolls zur sicheren Kommunikation in Cliquen, basierend auf einer Verallgemeinerung des Diffie-Hellmann-Protokolls [AST00] (und implementiert im Cliques-Toolkit), wurde ebenfalls in diesen Zusammenhang erstellt [St03e]. Die Bedeutung der Kompositionalität beim Entwurf komplexer, zuverlässiger Systeme, besonders im Hinblick auf das Spektrum formaler Methoden, ist ein aktueller Gegenstand der Forschung und wurde kürzlich anhand der Anwendung der sicheren und robusten Gruppenkommunikation in [St03b] erläutert. Eine Anwendung des offenen Kalküls der Konstruktionen zur Repräsentation der MSR-Cryptoprotokoll-Spezifikationssprache [Ce01a, Ce01b] ist schließlich ein weiteres laufendes Projekt. Hier wird das ausdrucksstarke Typsystem (speziell die abhängigen Typen) zur Spezifikation sicherheitsrelevanter Restriktionen verwendet [St03f]. Das Ziel dieses Projekts ist die Entwicklung einer Spezifikations- und Analyseumgebung für cryptographische Protokolle auf Basis eines ausführbaren Fragments der linearen Logik.

Werdegang und Danksagung Geboren in Hamburg im Jahre 1970, interessierte ich mich schon seit 1982 für die Informatik, speziell die Programmierung und die Konstruktion eigener Rechner. Bereits vor Abschluss des Abiturs im Jahre 1989, war ich von 1988 bis 1995 als freier Mitarbeiter in der Softwareindustrie tätig, hauptsächlich in den Bereichen Systemsoftware, Datenbanken und Compilerbau. Parallel dazu begann ich 1989 das Studium der Informatik mit Ergänzungsfach Physik an der Universität Hamburg, das ich mit Diplom im Jahre 1996 abschloss. Mein spezielles Interesse galt in dieser Zeit der theoretischen Informatik und den Arbeiten von Prof. h.c. Carl Adam Petri im Bereich der Netztheorie. Als wissenschaftlicher Mitarbeiter am Arbeitsbereich Theoretische Grundlagen der Informatik (Lehrstuhl Prof. Dr. Rüdiger Valk) arbeitete ich in Forschung und Lehre mit Schwerpunkt im Bereich der verteilten Systeme und des rechnergestützten Beweisens und begann gleichzeitig mit meiner Promotion. Insgesamt ca. drei Jahre verbrachte anlässlich diverser Forschungsaufenthalte in Kalifornien als International Fellow bei SRI International und teils als Visiting Scholar an der Stanford University. Meine Dissertation wurde unterstützt durch das EU-Projekt MATCH, eine Förderung durch den DAAD, und über zahlreiche Projekte von NSF, DARPA, ONR, und NASA. Nach meiner Promotion an der Universität Hamburg bin ich seit November 2002 als Postdoctoral Research Associate an der University of Illinois at Urbana-Champaign tätig, wo ich die Verfahren und Resultate aus meiner Dissertation in neuen Projekten, speziell im Bereich der Protokolle und der Sicherheit, anwende und weiterentwickle. Neben vielen Freuden und Kollegen möchte ich besonders Prof. Dr. Rüdiger Valk und Prof. Dr. José Meseguer danken, die beide mit Rat und Tat ungewöhnlich viel zum Gelingen meiner Arbeit beigetragen haben.

Literatur

[ABN+95] Andersen, F., Binau, U., Nyblad, K., Petersen, K. D., und Pettersson, J. S.: The HOL-UNITY verification system. In: Mosses, P. D., Nielsen, M., und Schwartzbach, M. I. (Hrsg.), *TAPSOFT'95: Theory and Practice of Software Development, 6th International Joint Conference CAAP/FASE, Aarhus, Denmark, May 22–26, 1995, Proceedings*. volume 915 of *LNCS*. S. 795–796. Springer. 1995.

[AC96] Abadi, M. und Cardelli, L.: *A Theory of Objects*. Monographs in Computer Science. Springer. 1996.

[ACCL91] Abadi, M., Cardelli, L., Curien, P.-L., und Lévy, J.-J.: Explicit substitutions. *Journal of Functional Programming*. 1(4):375–416. 1991.

[AS98] Amir, Y. und Stanton, J.: The Spread wide area group communication system. Technical Report CNDS-98-4. Center for Networking and Distributed Systems, Computer Science Department, Johns Hopkins University. 1998.

[AST00] Ateniese, G., Steiner, M., und Tsudik, G.: New multiparty authentication services and key agreement protocols. *IEEE Journal on Selected Areas in Communications*. 18(4):628–639. 2000.

[Ba92] Barendregt, H. P.: Lambda-calculi with types. In: Abramsky, S., Gabbay, D. M., und Maibaum, T. S. E. (Hrsg.), *Background: Computational Structures*. volume 2 of *Handbook of Logic in Computer Science*. Claredon Press, Oxford. 1992.

[BBC+99] Barras, B., Boutin, S., Cornes, C., Courant, J., Coscoy, Y., Delahaye, D., de Rauglaudre, D., Filliatre, J. C. ., Giménez, E., Herbelin, H., Huet, G., Laulhère, H., Muñoz, C., Murthy, C., Parent-Vigouroux, C., Loiseleur, P., Paulin, C., Saïbi, A., und Werner, B.: The Coq Proof Assistent Reference Manual, Version 6.3.1, Coq Project. Technical report. INRIA. 1999. `http://logical.inria.fr/`.

[BD87] Best, E. und Devillers, R.: Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*. 55:87–136. 1987.

[Be76] Berkling, K. J.: A symmetric complement to the lambda-calulus. Interner Bericht ISF-76-7. GMD, St. Augustin, Germany. 1976.

[Be88] Berardi, S.: Towards a mathematical analysis of the Coquand-Huet calculus of constructions and other systems in Barendregt's cube. Technical report. Carnegie Mellon University and Universita di Torino. 1988.

[BF82] Berkling, K. J. und Fehr, E.: A consistent extension of the lambda-calculus as a base for functional programming languages. *Information and Control*. 55:89–101. 1982.

[BJM00] Bouhoula, A., Jouannaud, J.-P., und Meseguer, J.: Specification and proof in membership equational logic. *Theoretical Computer Science*. 236:35–132. 2000.

[BJO99] Blanqui, F., Jouannaud, J.-P., und Okada, M.: The calculus of algebraic constructions. In: Narendran, P. und Rusinowitch, M. (Hrsg.), *Rewriting Techniques and Applications, 10th International Conference, RTA-99, Trento, Italy, July 2-4, 1999*. volume 1631 of *LNCS*. Springer. 1999.

[BKR01] Bonelli, E., Kesner, D., und Ríos, A.: From higher-order to first-order rewriting. In: *Proceedings of the 12th International Conference on Rewriting Techniques and Applications (RTA'01), Utrecht, The Netherlands, June 2001*. volume 2051 of *LNCS*. S. 47–62. Springer. 2001.

[CAB+86] Constable, R. L., Allen, S., Bromely, H., Cleveland, W., u. a.: *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall. 1986.

[CDE+99a] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., und Quesada, J.: *Maude: Specification and Programming in Rewriting Logic*. SRI International. January 1999. `http://maude.csl.sri.com`.

[CDE+99b] Clavel, M., Durán, F., Eker, S., Meseguer, J., und Stehr, M.-O.: Maude as a Formal Meta-Tool. In: Wing, J. M., Woodcock, J., und Davies, J. (Hrsg.), *FM'99 – Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 1999. Proceedings, Volume II*. volume 1709 of *LNCS*. Springer. 1999.

[CDE+00] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., und Quesada, J.: A tutorial on Maude. `http://maude.csl.sri.com`. March 2000.

[Ce01a] Cervesato, I.: A specification language for crypto-protocols based on multiset rewriting, dependent types and subsorting. In: Delzanno, G., Etalle, S., und Gabbrielli, M. (Hrsg.), *Workshop on Specification, Analysis and Validation for Emerging Technologies - SAVE'01 , pp. 1-22, Paphos, Cyprus, 1 December 2001*. 2001.

[Ce01b] Cervesato, I.: Typed MSR: Syntax and examples. In: Gorodetski, V., Skormin, V., und Popyack, L. (Hrsg.), *First International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security - MMM'01, St. Petersburg, Russia, 21-23 May 2001*. volume 2052 of *LNCS*. S. 159–177. Springer. 2001.

[Ch40] Church, A.: A formulation of the simple theory of types. *Journal of Symbolic Logic*. 5(1). 1940.

[CH88] Coquand, T. und Huet, G.: The calculus of constructions. *Information and Computation*. 76(2/3):95–120. 1988.

[CM88] Chandy, K. M. und Misra, J.: *Parallel Program Design. A Foundation*. Addison-Wesley. 1988.

[COS03] Coja-Oghlan, A. und Stehr, M.-O.: Revisiting the algebra of Petri net processes under the collective token philosophy. *Fundamenta Informaticae*. 54:151 – 164. 2003.

[dB72] de Bruijn, N. G.: Lambda calculus with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. In: *Proceedings Kninkl. Nederl. Akademie van Wetenschappen*. volume 75(5). S. 381–392. 1972.

[DHK99] Dowek, G., Hardin, T., und Kirchner, C.: HOL-lambda-sigma: An intentional first-order expression of higher-order logic. In: Narendran, P. und Rusinowitch, M. (Hrsg.),

Rewriting Techniques and Applications, 10th International Conference, RTA'99, Trento, Italy, July 2-4, 1999, Proceedings. volume 1631 of *LNCS*. S. 317–331. Springer. 1999.

[DM03] Denker, G. und Millen, J.: Design and implementation of multicast CAPSL and its intermediate language. Technical report. SRI International. 2003. In preparation.

[DMM96] Degano, P., Meseguer, J., und Montanari, U.: Axiomizing the algebra of net computations and processes. *Acta Informatica*. 33:641–667. 1996.

[DS90] Dijkstra, E. W. und Scholten, C. S.: *Predicate Calculus and Program Semantics*. Texts and Monographs in Computer Science. Springer. 1990.

[EMS02] Eker, S., Meseguer, J., und Sridharanarayanan, A.: The maude LTL model checker. `http://www.elsevier.nl/locate/entcs/volume71.html`. 2002.

[Ge93] Geuvers, H.: *Logics and Type Systems*. PhD thesis. University of Nijmegen. 1993.

[GGM03] Gunter, C. A., Goodloe, A., und McDougall, M.: Secure traceroute (sectrace). Draft, Version 1. March 2003.

[Gi72] Girard, J. Y.: *Interpretation fonctionelle et elimination des coupures dans l'arithmetique d'ordre superieur*. PhD thesis. Université Paris VII. 1972.

[GM92] Goguen, J. A. und Meseguer, J.: Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*. 105:217–273. 1992.

[GM93] Gordon, M. J. C. und Melham, T. F.: *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press. 1993.

[HC96] Heyd, B. und Crégut, P.: A modular coding of UNITY in Coq. In: von Wright, J., Grundy, J., und Harrison, J. (Hrsg.), *Theorem Proving in Higher Order Logics, 9th International Conference, TPHOLs'96, Turku, Finland, August 26-30, 1996, Proceedings*. volume 1125. Springer. 1996.

[HHP87] Harper, R., Honsell, F., und Plotkin, G.: A framework for defining logics. In: *Second Annual Symposium on Logic in Computer Science, Ithaca, New York, 22–25 June 1987, Proceedings*. S. 193–204. IEEE. 1987.

[Ho69] Hoare, C. A. R.: An axiomatic basis for computer programming. *Communications of the ACM*. 12(10):576–583. 1969.

[Ho80] Howard, W. A.: The formulae-as-types notion of construction. In: Hindley, J. und Seldin, J. (Hrsg.), *To H.B. Curry: Essays on Combinatory Logic*. Academic Press. 1980.

[Ho96] Howe, D. J.: Importing mathematics from HOL into Nuprl. In: J. Von Wright, J. Grundy, und J. Harrison (Hrsg.), *Theorem Proving in Higher Order Logics, 9th International Conference, TPHOLs'96, Turku, Finland, August 26-30, 1996, Proceedings*. volume 1125 of *LNCS*. S. 267–282. Springer. 1996.

[Ho97] Howe, D. J.: A classical set-theoretic model of polymorphic extensional type theory. Manuscript (submitted for publication). 1997.

[Ho98] Howe, D. J.: Toward sharing libraries of mathematics between theorem provers. In: *Frontiers of Combining Systems, FroCoS'98, ILLC, University of Amsterdam, October 2–4, 1998, Proceedings*. Kluwer Academic Publishers. 1998.

[Je92] Jensen, K.: *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*. volume 1 of *EATCS Monographs on Theoretical Computer Science*. Springer. 1992.

[Jo98] Jouannaud, J.-P.: Membership equational logic, calculus of inductive constructions, and rewrite logic. In: *International Workshop on Rewriting Logic and its Applications, Abbaye des Prémontrés at Pont-à-Mousson, France, September 1998, Proceedings*. volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier. 1998. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[La92] Lamport, L.: Types considered harmful. `http://research.microsoft.com/users/lamport/tla/notes.html`. December 1992.

[Le94] Lescanne, P.: From $\lambda\sigma$ to $\lambda\upsilon$, a journey through calculi of explicit substitutions. In: Boehm, H. (Hrsg.), *Conference Record of POPL'94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon, January 17–21, 1994*. S. 60–69. ACM. 1994.

[Lu94] Luo, Z.: *Computation and Reasoning: A Type Theory for Computer Science*. International Series of Monographs on Computer Science. Oxford University Press. 1994.

[Me89] Meseguer, J.: General logics. In: Ebbinghaus, H.-D. u. a. (Hrsg.), *Logic Colloquium'87, Granada, Spain, July 1987, Proceedings*. S. 275–329. North-Holland. 1989.

[Me92] Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*. 96:73–155. 1992.

[Me96] Meseguer, J.: Rewriting logic as a semantic framework for concurrency. A progress report. In: Montanari, U. und Sassone, V. (Hrsg.), *CONCUR'96: Concurrency Theory, 7th International Conference, Pisa, Italy, August 26–29, 1996*. volume 1119 of *LNCS*. S. 331–372. Springer. 1996.

[Me98] Meseguer, J.: Membership algebra as a logical framework for equational specification. In: Parisi-Presicce, F. (Hrsg.), *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Tarquinia, Italy, June 3–7, 1997, Selected Papers*. volume 1376 of *LNCS*. S. 18 – 61. Springer. 1998.

[Mi94] Misra, J.: New UNITY. Manuscript, `http://www.cs.utexas.edu/users/psp/newunity.html`. 1994.

[Mi95] Misra, J.: A logic for concurrent programming: Safety and progress. *Journal of Computer and Software Engineering*. 3(2):239–300. 1995.

[Mi99] Milner, R.: *Communicating and Mobile Systems: The Pi-Calculus*. Cambridge University Press. May 1999.

[Mi01] Misra, J.: *A discipline of multiprogramming: programming theory for distributed applications*. volume 18 of *Monographs in Computer Science*. Springer. 2001.

[MM90] Meseguer, J. und Montanari, U.: Petri nets are monoids. *Information and Computation*. 88(2):105–155. October 1990.

[MOM94] Martí-Oliet, N. und Meseguer, J.: General logics and logical frameworks. In: Gabbay, D. (Hrsg.), *What is a Logical System?* S. 355–392. Oxford University Press. 1994.

[MOM96] Martí-Oliet, N. und Meseguer, J.: Rewriting logic as a logical and semantic framework. In: *RWLW'96, First International Workshop on Rewriting Logic and its Applications, Asilomar Conference Center, Pacific Grove, CA, USA, September 3-6, 1996, Proceedings*. volume 4 of *Electronic Notes in Theoretical Computer Science*. Elsevier. 1996. `http://www.elsevier.nl/locate/entcs/volume4.html`. To appear in D. M. Gabbay, F. Guenthner, (eds.), *Handbook of Philosophical Logic* (2nd edition), Kluwer Academic Publishers.

[MÖST02] Meseguer, J., Ölveczky, P. C., Stehr, M.-O., und Talcott, C. L.: Maude as a wide-spectrum framework for formal modeling and analysis of active networks. In: *DARPA Active Networks Conference and Exposition (DANCE), San Francisco, May 29 – 31, 2002*. IEEE. May 2002.

[NSM01] Naumov, P., Stehr, M.-O., und Meseguer, J.: The HOL/NuPRL proof translator — A practical approach to formal interoperability. In: *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs'2001, Edinburgh, Scotland, UK, September 3–6, 2001, Proceedings*. volume 2152 of *LNCS*. S. 329 – 345. Springer. 2001.

[Pa00] Paulson, L. C.: Mechanizing UNITY in Isabelle. *ACM Transactions on Computational Logic*. 1(1):3–32. 2000.

[Po93] Pollack, R.: Closure under alpha-conversion. In: Barendregt, H. und Nipkow, T. (Hrsg.), *Types for Proofs and Programs: International Workshop TYPES'93, Nijmegen, May 1993, Selected Papers*. volume 806 of *LNCS*. S. 313–332. Springer. 1993.

[PSN90] Petersson, K., Smith, J., und Nordstroem, B.: *Programming in Martin-Löf's Type Theory. An Introduction*. International Series of Monographs on Computer Science. Oxford: Clarendon Press. 1990.

[Ra03] Rashid, R. F.: Discussion on the use of formal methods with vice president of Microsoft Research. Personal Communication. April 2003.

[Re74] Reynolds, J.: Towards a theory of type structure. In: *Programming Symposium, Paris*. volume 19 of *LNCS*. Springer. 1974.

[Re91] Reisig, W.: Petri nets and algebraic specifications. *Theoretical Computer Science*. 80:1–34. 1991.

[SM99] Stehr, M.-O. und Meseguer, J.: Pure type systems in rewriting logic. In: *LFM'99: Workshop on Logical Frameworks and Meta-languages, Paris, France, September 28, 1999, Proceedings*. 1999. `http://plan9.bell-labs.com/who/felty/LFM99/`.

[SM03] Stehr, M.-O. und Meseguer, J.: Pure type systems in rewriting logic. In: *Collection to the honor of O.J. Dahl*. volume 2635 of *LNCS*. Springer-Varlag. 2003.

[SMÖ01a] Stehr, M.-O., Meseguer, J., und Ölveczky, P. C.: Representation and execution of Petri nets using rewriting logic as a uniform framework. In: Ehrig, H., Ermel, C., und Padberg, J. (Hrsg.), *UNIGRA'2001, Uniform Approaches to Graphical Process Specification Techniques, Genova, Italy, March 31st and April 1st, 2001, Proceedings*. volume 44 of *Electronic Notes in Theoretical Computer Science*. 2001.

[SMÖ01b] Stehr, M.-O., Meseguer, J., und Ölveczky, P. C.: Rewriting logic as a unifying framework for Petri nets. In: Ehrig, H., Juhas, G., Padberg, J., und Rozenberg, G. (Hrsg.), *Unifying Petri Nets, Advances in Petri Nets*. volume 2128 of *LNCS*. S. 250–303. Springer. 2001.

[SNM00] Stehr, M.-O., Naumov, P., und Meseguer, J.: A proof-theoretic approach to HOL-Nuprl connection with applications to proof translation (full version). Manuscript, CSL, SRI-International, Menlo Park, CA, USA. Available at `http://www.uiuc.edu/~stehr`. March 2000.

[SNM01] Stehr, M.-O., Naumov, P., und Meseguer, J.: A proof-theoretic approach to HOL-Nuprl connection with applications to proof translation (extended abstract). In: *WADT/CoFI'01, 15th International Workshop on Algebraic Development Techniques and General Workshop of the CoFI WG, Genova, Italy, April 1 – 3, 2001, Proceedings*. 2001. See [SNM00] for the full version.

[St98a] Stehr, M.-O. Assertional reasoning and temporal logic for system verification. 1998. Lecture at the MATCH Advanced Summer School on System Engineering, September 14-22, 1998, Jaca, Spain. `http://www.cps.unizar.es/deps/DIIS/MATCH`.

[St98b] Stehr, M.-O.: Embedding UNITY into the calculus of inductive constructions. Fachbereichsbericht FBI-HH-B-214/98. University of Hamburg, Germany. September 1998.

[St99] Stehr, M.-O.: CINNI - A New Calculus of Explicit Substitutions and its Application to Pure Type Systems. Manuscript, CSL, SRI-International, Menlo Park, CA, USA. 1999.

[St00] Stehr, M.-O.: CINNI – A Generic Calculus of Explicit Substitutions and its Application to λ-, σ- and π-calculi. In: Futatsugi, K. (Hrsg.), *The 3rd International Workshop on Rewriting Logic and its Applications, Kanazawa City Cultural Hall, Kanzawa, Japan,*

September 18–20, 2000, Proceedings. volume 36 of *Electronic Notes in Theoretical Computer Science*. S. 71 – 92. Elsevier. 2000. http://www.elsevier.nl/locate/entcs/volume36.html.

[St02a] Stehr, M.-O.: Explicit substitutions and the open calculus of constructions. Tutorial at the 4th International Workshop on Rewriting Logic and its Applications, Pisa, Italy, September 19–21, 2002. Available at http://www.uiuc.edu/~stehr. 2002.

[St02b] Stehr, M.-O.: Programming, Specification, and Interactive Theorem Proving — Towards a Unified Language based on Equational Logic, Rewriting Logic, and Type Theory. Doctoral Thesis. Universität Hamburg, Fachbereich Informatik, Germany. 2002. http://www.sub.uni-hamburg.de/disse/810/.

[ST02c] Stehr, M.-O. und Talcott, C. L.: PLAN in Maude: Specifying an active network programming language. In: Gadducci, F. und Montanari, U. (Hrsg.), *The 4th International Workshop on Rewriting Logic and its Applications, Pisa, Italy, September 19–21, 2002, Proceedings*. volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier. 2002. http://www.elsevier.nl/locate/entcs/volume71.html.

[St03a] Stehr, M.-O.: Assertional reasoning. In: Girault, C. und Valk, R. (Hrsg.), *Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications*. Springer. 2003.

[St03b] Stehr, M.-O.: Composable formal models for high-assurance fault tolerance networks (joint work with C. L. Talcott). FTN PI Meeting, July 21-23, 2003, Honolulu, Hawaii. https://www.schafercorp-ballston.com/ftndccone2003. 2003.

[St03c] Stehr, M.-O.: Formal specification of group communication middleware (joint work with C. L. Talcott). Workshop on Context Sensitive Systems Assurance (Contessa'03), April 1, 2003, Philadelphia. http://formal.cs.uiuc.edu/contessa/. 2003.

[St03d] Stehr, M.-O.: Formal specification of sectrace (joint work with A. Sridharanarayanan). Workshop on Context Sensitive Systems Assurance (Contessa'03), April 1, 2003, Philadelphia. http://formal.cs.uiuc.edu/contessa/. 2003.

[St03e] Stehr, M.-O.: On the significance of formal methods in the development of high-assurance secure systems. Invited Lecture at the Naval Postgraduate School, Center for Information Systems Security Studies and Research, July 30, 2003, Monterey, California. http://www.cisr.nps.navy.mil/lecturearchive.html. 2003.

[St03f] Stehr, M.-O.: Relating the MSR crypto-protocol specification language to rewriting logic with dependent types. UMBC Protocol Analysis Meeting, June 10-11, 2003, Baltimore. http://theory.stanford.edu/~iliano/umbc/. 2003.

[St03g] Stehr, M.-O.: A rewriting semantics for algebraic nets. In: Girault, C. und Valk, R. (Hrsg.), *Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications*. Springer. 2003.

[Te89] Terlouw, J.: Een nadere bewijstheoretische analyse van GSTTs. Manuscript, University of Nijmegen, The Netherlands. 1989.

[TSMO02] Thati, P., Sen, K., und Martí-Oliet, N.: An executable specification of asynchronous π-calculus and may tesing in Maude 2.0. In: Gadducci, F. und Montanari, U. (Hrsg.), *The 4th International Workshop on Rewriting Logic and its Applications, Pisa, Italy, September 19–21, 2002, Proceedings*. volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier. 2002. http://www.elsevier.nl/locate/entcs/volume71.html.