Representing Logics and Logic Translations (Repräsentation von Logiken und Logik-Übersetzungen)

Florian Rabe

Jacobs University
Bremen
f.rabe@jacobs-university.de

1 Einführung

Logik [1] ist das Studium formaler Sprachen für Propositionen und Wahrheit. Logiken werden sowohl als foundation der Mathematik als auch als Spezifikationssprachen in der Mathematik und Informatik eingesetzt. Da Logik fundamental mit der Natur der Mathematik verwoben ist, ist die Frage, wie Logiken in unseren Gedanken zu repräsentieren sind, eine ständige Herausforderung für unser Verständnis. Und nur wenn dies verstanden ist, können wir anfangen die entsprechende Frage über Logik-Übersetzungen zu beantworten. Gleichzeitig werden Logiken in großem Umfang in der Informatik eingesetzt um sowohl über Mathematik als auch über Software-System zu schließen. Dies führt zu der Frage, wie Logiken und ihre Übersetzungen in einem Computer-System repräsentiert werden können. In diesem Text versuchen wir Antworten auf diese fundamentalen Forschungsfragen zu geben.

Das 20. Jahrhundert hindurch wurden bereits mehrere Antworten gegeben. Die meisten von ihnen können in zwei Arten gruppiert werden, die als Mengen-/Modell-Theorie Typ-/Beweis-Theorie bezeichnet werden können. Diese zwei Klassen repräsentieren verschiedene Forschungsgebiete mit in Konflikt stehenden philosophischen und mathematischen Hintergründen, die ihre Konzeptualisierungen ontologisch unterschiedlich machen. Jedoch haben beide Gebiete ausgefeilte und starke Lösungen bereitgestellt.

Daher gründen wir unsere Untersuchung auf dem Bestreben diese beiden Perspektiven zu vereinigen. Unser Fokus liegt darauf die bestehenden Konzepte von Logiken und Logik-Übersetzungen auf eine Weise zu erweitern, die ihr Wesen und das angesammelte Wissen über sie erhält, während sie sie in eine neue Richtung führt. Unser Ziel ist es beide Forschungsfelder zu bereichern, indem wir sie konsequenter aufeinander anwenden uns sie benutzbarer und verständlicher füreinander machen.

Wir wählen institutions als einen mengen-/modell-theoretischen und dependent type theo-

[1] Bei diesem Text handelt es sich um eine übersetzte und vereinfachte Fassung der offiziellen Zusammenfassung ([Rab08]) der Dissertation des Autors. Falls möglich, wurden wörtliche Übersetzungen bevorzugt. Fachbegriffe ohne weit verbreitetes deutsches Analogon sind im Englischen belassen worden.

ry als einen typ-/beweistheoretischen Vertreter der zwei Klassen. Wir stellen fest, daß beide komplementäre Vorteile haben, die ihre verschiedenen hintergründe reflektieren und erhalten unsere eigene Antwort, indem wir sie auf eine Weise kombinieren, die ihre jeweiligen Stärken konsequent kombiniert. Mathematisch können unsere Hauptresultate wie folgt zusammengefasst werden. Wir definieren Logiken, indem wir institutions um Konzepte von Beweiskategorien und beweistheoretischer Wahrheit erweitern, die sich parallel zu den existierenden Konzepten von Modellkategorien und satisfaction relation von institutions verhalten. Dann geben wir eine konkrete einfache Logik für dependent type theory, die Modelle verwendet, die von Kripke-Modellen für intuitionistische Logik inspiriert sind. Schließlich zeigen wir, wie diese Logik benutzt werden kann, um Logiken und Logik-Übersetzungen zu definieren und zu kodieren.

Während dies die Frage beantwortet, wie Logiken und Logik-Übersetzungen in unseren Gedanken repräsentiert werden können, ist es nicht adäquat für die speziellen Einschränkungen in und Anwendungen von Software-Systemen. Deshalb erforschen wir, in einer zweiten Untersuchung, wie unsere Repräsentationen konkreter und robuster gemacht werden können, um eine skalierbare mechanische Behandlung zu ermöglichen.

Wir wählen OMDoc als eine skalierbare, web-kompatible Repräsentationssprache für mathematisches Wissen. Weil wir finden, daß ihre Anwendbarkeit auf logisches Wissen begrenzt ist, revidieren wir sie, wobei wir die charakteristischen Anforderungen von Logik berücksichtigen. Während wir die Motivation und das Wesen von OMDoc bewahren, setzen wir eine anderen Methodologie ein, die besser zu Logik passt, wodurch wir OMDoc in der Folge neu erfinden. Mathematisch können unser Hauptresultate wie folgt zusammengefasst werden. Wir geben eine formale abstrakte Syntax für das modulare Entwickeln von Theorie und eine formale Semantik für sie. Unser Modulsystem behandelt logical frameworks, Logiken und logische Theorien wie auch Übersetzungen zwischen ihnen gleichartig als Theorien und Theorie-Morphismen, die durch die „ist Meta-Sprache für“-Relation verbunden sind. Und es unterscheidet konsequent zwischen logik-unabhängigen und logik-spezifischen Sprachkonstrukten, was es uns ermöglicht ein logik-unabhängiges Flattening-Theorem zu erhalten, das die Basis darstellt für logik-unabhängige Verwaltungsdienste für logisches Wissen (englisch: logic-independent logical knowledge management services).

Diese beiden Resultate zusammen betrachte ergeben ein Dreieck aus verschiedenen mathematischen Gemeinschaften (englisch: communities), Forschungszielen und Philosophien, das aus Mengen-/Modell-Theorie, Typ-/Beweis-Theorie und mathematischer Wissensverwaltung besteht. Unsere zentrale Kontribution ist es die Ecken dieses Dreiecks in ein kohärentes framework zu integrieren, das um Logik zentriert ist und ihre komparativen Vorteile ausnutzt.

Die Dissertation besteht aus drei Teilen. Teil I gibt einen Überblick über die historischen Arbeiten und den aktuellen Stand der Forschung im Bereich der Logik, wobei spezielle Aufmerksamkeit Logiken und Logik-Übersetzungen gewidmet wird. Er beschreibt detailliert die Beziehung zwischen Mengen-/Modell-Theorie und Type-/Beweis-Theorie und ihre jeweilige Beziehung zur mathematischen Wissensverwaltung und entwickelt schließlich die Forschungsobjekte „Kombination von Modell- und Beweis-Theorie“ (Combining

Model and Proof Theory) und „Logical Knowledge Management“ (Dieser Begriff ist absichtlich doppeldeutig – Verwaltung Logischen Wissens und Logische Wissensverwaltung –, was im Deutschen nicht exakt wiedergegeben werden kann.), die als Titel der folgenden Teile dienen. Teil I gibt auch eine formal rigorose Top-down-Einführung in die mathematischen Grundlagen von Logik, um die Dissertation in sich abgeschlossen zu machen, und Bottom-up-Einführung im Tutorium-Stil, um Leser bei der Lektüre der extrem abstrakten Materie zu unterstützen.

Teil II besteht aus drei Kapiteln. Das erste nimmt das modell-theoretische logical framework der institutions und entwickelt es zu einem, das ausgeglichener ist bezüglich Modell- und Beweis-Theorie. Wir nennen eine institution angereichert mit unserer beweistheoretischen Struktur eine *Logik*. Auf ähnliche Weise nimmt das zweite Kapitel das beweis-theoretische logical framework von Martin-Löf's dependent type theory und fügt ihm eine modell-theoretische Semantik hinzu. Dies ergibt eine Logik (in unserem Sinn) für dependent type theory. Wegen der fundamental unterschiedlichen Wesen von beweis- und modell-theoretischer Semantik von Logik können diese zwei Ansätze sich nicht einfach in der Mitte treffen. Wir vereinigen sie, indem wir die Logik aus dem zweiten Kapitel als Meta-Logik einsetzen, in der andere Logiken repräsentiert werden. Dies wird im dritten Kapitel ausgeführt, das auch Beispiel-Repräsentationen von Logiken und Logik-Übersetzungen gibt.

Teil III besteht auch aus drei Kapiteln. Im ersten entwickeln wir MMT, unser einfaches aber ausdrucksstarkes Modulsystem für mathematische Theorien. Vollkommen formal, foundation-unabhängig und skalierbar, ist es entworfen, um die Repräsentation von Logiken und Logik-Übersetzungen in der Praxis zu unterstützen. Im zweiten Kapitel wenden wir MMT auf die Repräsentation, die im zweiten Teil entwickelt wurden, an und zeigen, wie die foundations spezifischer Typ-Theorien und Mengen-Theorien in MMT wiedergewonnen werden können. Im letzten Kapitel beschreiben wir die skalierbare Infrastruktur und ihre Implementierung, die wir für MMT entwickelt haben.

This summary is structured along the above outline of Parts II and III of the thesis.

2 Teil II: Combining Model and Proof Theory

Seit der Grundlagenkrise der Mathematik war Logik ein wichtiges Forschungsthema in der Mathematik und der Informatik. Eine zentrale Frage war immer, was eine Logik eigentlich ist (geschweige denn eine Logik-Übersetzung). Während des letzten Jahrhunderts haben Forscher sehr unterschiedliche Antworten auf diese Frage gegeben, und Forschungsgebiete, die ursprünglich verbunden waren, haben sich auseinanderentwickelt. Während diese Spezialisierung zu sehr erfolgreichen Ergebnissen geführt hat, hat sie auch eine Trennung in der Logik-Forschung herbeigeführt, die manchmal schädlich ist.

Heute beobachten wir, daß es zwei Arten von logical frameworks gibt: diejenigen auf der Basis von mengen-theoretischen foundations, die Logik modelltheoretisch charakterisieren, und diejenigen auf der Basis von Typtheorie, die Logik beweis-theoretisch charakterisieren. Die ersteren gehen zurück auf Tarskis Verständnis von Konsequenz ([Tar33,

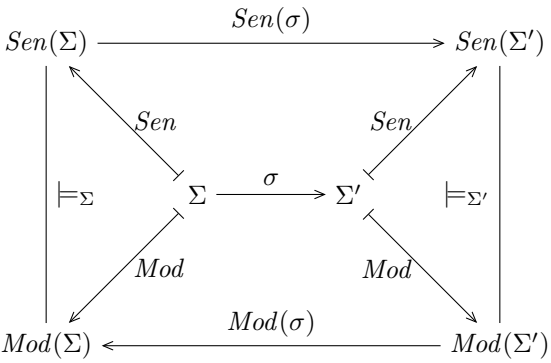
TV56]), und institutions ([GB92, GR02]) und general logics ([Mes89]) sind die wichtigsten Beispiele. Die letzteren basieren oft auf der Curry-Howard-Korrespondenz ([CF58, How80]), und Beispiele sind Automath ([dB70]), Isabelle ([Pau94]), und das Edinburgh Logical Framework (LF, [HHP93]).

Während einige von diesen die andere Seite integrieren, so wie die allgemeine Beweistheorie, die in [Mes89] entwickelt wird, neigen fast alle von ihnen zu einer der beiden Seiten. Oft sind diese Seiten nicht nur durch Forschungsfragen, sondern auch durch "conflicting cultures and a[t]titudes" getrennt, und "attempts to bridge these two cultures are rare and rather timid" (Zitat eines anonymen Gutachters eines Papiers). Dieser Teil der Dissertation macht einen solchen Versuch ein ausgeglichenes framework zu entwickeln, das beide Blickwinkel auf eine Weise integriert und subsumiert, die ihre jeweiligen Vorteile bewahrt und ausnutzt.

Vom Blickwinkel der Beweis-Theorie geben wir eine modell-theoretische Semantik für dependent type theory auf der Basis von institutions. Und vom Blickwinkel der Modell-Theorie geben wir ein Spezifikationssprache für institutions, indem wir dependent type theory verwenden. Von einem neutralen Blickwinkel ergeben diese Resultate ein logical framework, das Modell- und Beweis-Theorie verbindet.

Wir werden hier nicht auf die mathematisch sehr anspruchsvollen Details eingehen und nur kurz unsere Definition von Logik exemplarisch skizzieren. Institutions bieten abstrakte Definition von Syntax und modell-theorischer Semantik von Logik auf der Basis von Kategorien-Theorie ([Lan98]). Eine institution ist ein Tupel $(Sig, Sen, Mod, \models)$, wobei Sig eine Kategorie von Signaturen ist und $Sen : Sig \to \mathcal{SET}$ und $Mod : Sig \to \mathcal{CAT}^{op}$ jeder Signatur Propositionen und Modelle zuordnen. Dann werden die Menge der Propositionen und die Klasse der Modelle durch die satisfaction relation $\models_\Sigma$ in Beziehung gesetzt.

Für gegebene Signaturen Σ und Σ' und einen Signatur-Morphismus σ zwischen ihnen können die involvierten Ausdrücke wie folgt visualisiert werden:



Leser, die sich nicht mit institutions auskennen, mögen bei Σ und Σ' an die Signatur in Logik erster Stufe für Monoide (d.h., mit Deklarationen für Komposition und Einheit) beziehungsweise Gruppen (d.h., Monoide erweiternd um ein Symbol für das inverse Ele-

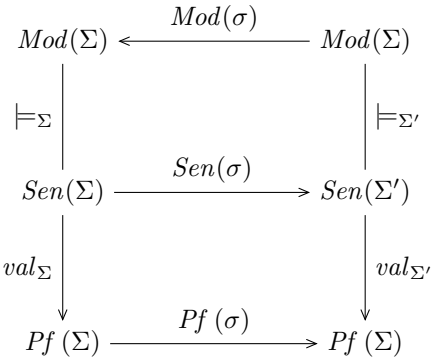
ment) denken und bei σ an die Inklusions-Abbildung von Σ nach Σ'. Dann sind $Sen(\Sigma)$ und $Sen(\Sigma')$ die Mengen der Propositionen erster Stufe über der jeweiligen Signatur, und $Sen(\sigma)$ ist the Abbildung die Σ-Propositionen auf Σ'-Propositionen abbildet. Im Monoid-Gruppen-Beispiel ist $Sen(\sigma)$ eine Inklusion, weil die Sprache erster Stufe der Monoide in der der Gruppen enthalten ist.

Auf ähnliche Weise sind $Mod(\Sigma)$ und $Mod(\Sigma')$ Kategorien von Modellen. Leser, die sich nicht mit Kategorientheorie auskennen, mögen bei $Mod(\Sigma)$ an die Klasse der Monoids und bei $Mod(\Sigma')$ an die Klasse der Gruppen denken. $Mod(\sigma)$ drückt die Modell-Reduktion aus, eine Übersetzung, die entgegen der Signatur-Übersetzung verläuft. Für unsere Beispiel-Signaturen ist $Mod(\sigma)$ die Abbildung, die jede Gruppe auf sich selbst qua Monoid abbildet.

Schließlich ist $\models_\Sigma$ die satisfaction relation: Für ein Model I und eine Propositon F, ist sie erfüllt, wenn F in I gilt.

Wir erweitern institutions zu Logiken. Eine *Logik* besteht aus einer Kategorie von Signaturen Sig, einem Propositions-Funktor Sen, einem Modell-Funktor Mod, einem Beweis-Kategorie-Funktor Pf , und modell- und beweis-theoretischen Definitionen von Wahrheit $\models$ und val. Pf bildet jede Signatur auf eine Kategorie von beweisen ab. Die Objekte dieser Kategorien sind Familien von judgments, wobei judgments z. B. Propositionen, Sequenzen, Tableaux-Äste, oder Klausel-Mengen sein können. Die Morphismen zwischen Familien von judgments sind die Beweise. Schließlich ist für jede Signatur Σ val_Σ eine Abbildung, die jeder Proposition das judgment zuordnet, das ihre Validität ausdrückt. Zum Beispiel, wenn die judgments Sequenzen sind, bildet $val_\Sigma\ A \in Sen(\Sigma)$ auf $\vdash A$ ab.

Dies kann durch das folgende Diagram visualisiert werden, wieder für einen Signatur-Morphismus $\sigma : \Sigma \to \Sigma'$, wobei $Pf\ (\sigma)$ Σ-judgements und -Beweise nach Σ' abbildet. Alle Pfeile repräsentieren Abbildungen zwischen Klassen, und die Kanten ohne Pfeilspitzen stellen Relationen zwischen Klassen dar. (Der Einfachheit in dieser Zusammenfassung halber ist dies kein Diagramm im strengen Sinne der Kategorientheorie.)



3 Teil III: Logical Knowledge Management

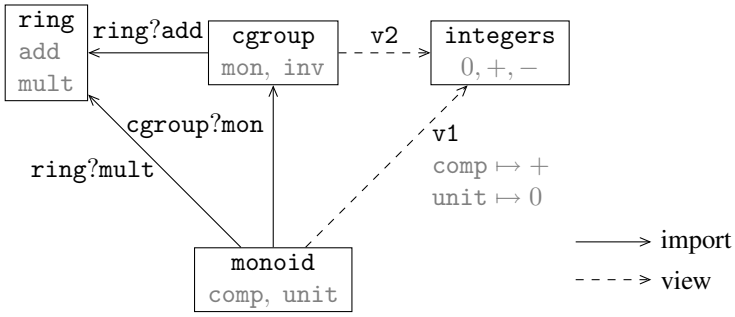
Die Entwicklung von MMT konzentriert sich auf ein Austauschformat zwischen Systemen, die auf mathematischem Wissen basieren. Es ist nicht durch die Wahl einer foundation eingeschränkt und erlaubt es die meta-theoretische foundation des mathematischen Wissens im gleichen Format zu repräsentieren und die foundations auf der meta-logischen Ebene zu verbinden. Insbesondere können die logischen Grundlagen der Repräsentationen selbst als Module repräsentiert werden. Dieser „Logiken-als-Theorien"-Ansatz macht das Systemverhalten wie auch das repräsentierte Wissen interoperabel und somit vergleichbar. Die explizite Repräsentation der epistemischen Grundlagen kommt auch den Systeme selbst zugute, deren mathematisches Wissen nur implizit in ihre Algorithmen eingebunden ist: Für diese kann die explizite Repräsentation als eine Dokumentation der Systemschnittstellen wie auch als Basis für Verifikation und Testen.

Natürlich ist Kommunikation durch Übersetzung auf den kleinsten gemeinsamen Nenner – der momentane Stand der Kunst – immer möglich. Aber solche Übersetzungen verlieren gerade die strukturellen Eigenschaften der Wissensrepräsentation, von denen die Wahl der Repräsentationssprache initial beeinflusst worden war. Deshalb enthält MMT eine Modulsystem, das darauf ausgelegt ist die flexible Wiederverwendung von Wissen via Theorie-Morphismen zu ermöglichen. Dieses Modulsystem ist die zentrale Komponente von MMT. Im Gegensatz zur Objektebene, die ontologisch nicht eingeschränkt werden sollte, muss das Modulsystem eine klare Semantik haben (relativ zur Semantik der Objektebene) and expressiv genug sein, um aktuelle Strukturierungspraxis zu subsumieren, so daß Systeme kommunizieren können, ohne die Struktur ihrer Wissensrepräsentationen zu verlieren.

Auf der praktischen Ebene muss das Austauschformat in dem Sinne skalierbar sein, daß es die Verteilung von Ressourcen (Theorien, Beweise etc.) über das Internet unterstützt, so daß sie kollaborativ bearbeitet werden können. In der aktuellen Web-Architektur bedeutet das, daß alle relevanten Ressourcen über URIs adressierbar sein müssen. Dabei ist zu beachten, daß bei einer komplexen Wiederverwendungs-Architektur auch Ressourcen adressierbar sein müssen, die nur virtuell existieren, da sie durch die Vererbungshierarchie induziert worden sind.

Schließlich sind Entwurf, Implementierung und Unterhaltung von „large scale logical knowledge management services" realistischerweise nur dann lohnend, wenn das gleiche framework für verschiedene foundations der Mathematik wiederverwendet werden kann. Deshalb wird eine Schnittstellenschicht benötigt zwischen dem logisch-mathematischen Kern einer mathematischen foundation und einem foundation-unabhängigen Wissensverwaltungs-Dienst, der die Semantik des verarbeiteten Wissens bewahren muss, ohne sie zu kennen.

Wir werden nur exemplarisch eine der Grundideen des MMT-Modul-Systems anhand der folgenden Abbildung eines Ausschnitts aus der algebraischen Hierarchie skizzieren.



Der untere Knoten repräsentiert die Theorie der Monoide, die Konstanten für Komposition und Einheit deklariert. (Wir lassen hier der Einfachheit halber die Axiome weg.) Die Theorie `cgroup` für kommutative Gruppen entsteht durch einen Import von `monoid` und das Hinzufügen einer Konstante `inv` für das inverse Element. Diese Theorie erbt Komposition und Einheit von Monoiden. Der Import induziert einen Theorie-Morphismus von Monoiden zu kommtuativen Gruppen.

Die Theorie der Ringe entsteht durch Importe von `monoid` für die multiplikative Struktur und von `cgroup` für die additive Struktur. Weil alle Importe Namen haben, können unterschiedliche Import-Pfade unterschieden werden: Die Theorie `ring` hat die Konstanten `ring?add/mon/comp` (Addition), `ring?add/mon/unit` (Null), `ring?add/inv` (additives Inverses), `ring?mult/comp` (Multiplikation), und `ring?mult/unit` (Eins).

MMT repräsentiert Modelle als Theorie-Morphismen. Der Knoten auf der rechten Seite repräsentiert eine Theorie für die ganzen Zahlen und deklariert Konstanten 0, $+$ und $-$. Die Tatsache, daß die ganzen Zahlen einen Monoid bilden, wird durch den view `v1` repräsentiert. Dieser ist ein Theorie-Morphismus, der explizit gegeben ist durch die Interpretation von `comp` als $+$ und von `unit` als 0. Der view `v2` ist besonders interessant: Er kann einerseits durch explizite Interpretationen aller Konstanten von `cgroup` gegeben werden: `inv` als $-$ und die beiden importierten Konstanten `mon/comp` und `mon/unit` als $+$ beziehungsweise 0. Andererseits kann der view `v1` wiederverwendet werden, indem `mon` als `v1` interpretiert wird. Beides führt zum gleichen Theorie-Morphismus, aber der zweite eliminiert Redundanz in der Wissensrepräsentation.

4 Konklusion

Unsere Arbeit war motiviert von den Zielen Logiken und Logik-Übersetzungen in unseren Gedanken und in Maschinen zu repräsentieren. Was das erste Ziel betrifft, leisten wir einen signifikanten Beitrag zu den Grundlangen von Informatik und Logik. Durch die Integration der oft konkurrierenden Ansätze von Beweis- und Modell-Theorie haben wir eine wegweisenden neue Perspektive auf Logik gewonnen. Und wir haben ein logical framework bereitgestellt, daß diese Perspektive formalisiert und ausnutzt.

Es gibt einige herausfordernde offene theoretische Problem in bezug auf unser framework, insbesondere die Untersuchung von allgemeinen Vollständigkeitsbeweisen in ihm. Jedoch

ist die wichtigste nächste Forschungsaufgabe praktischerer Natur, nämlich die Anwendung des frameworks auf Logiken und Logik-Übersetzungen in großem Maßstab. Mit großem Maßstab meinen wir ausgefeilte Software und Bibliotheken die erfolgreich im Software-Engineering eingesetzt werden, so wie PVS ([ORS92]) oder Isabelle ([Pau94]), für die aber Interoperabilität ein großes Problem darstellt.

Ein wichtiger Punkt, der aufkommen wird, ist die Notwendigkeit partielle Übersetzungen, die in Implementierungen verwendet werden, zu formalisieren und zu repräsentieren. In letzter Zeit sind solche Übersetzungen sehr erfolgreich eingesetzt worden, um Software-Unterstützung für das automatische Beweisen von anderen System zu borgen, z. B., in Leo ([BPTF07]) oder Isabelle. Diese Übersetzungen setzen oft nicht-triviale Kodierungs-Schritte ein, um die Logiken der beiden System oder Teile von ihnen aufeinander abzubilden. Wegen ihrer Ad-hoc-Natur, sind logical frameworks bisher noch nicht für sie verwendet worden.

Ein langfristiges Ziel ist es einen universellen Logik-Graphen zu bilden, der die wichtigsten Logiken und ihre Übersetzungen repräsentiert. Dieser wird als eine Dokumentations-Plattform für Forschung und Lehre dienen und als ein Netzwerk von Übersetzungen, das im menschlichen, interaktiven und automatischen Schließen verwendet werden kann. Ein kurzfristiger Schritt auf dieses Ziel hin ist die Integration unseres frameworks und des Hets-Systems ([MML07]).

Was das zweite Ziel betrifft, haben wir ein Forschungsgebiet untersucht, das wir logical knowledge management genannt haben. Hier haben wir die Grenzen erforscht zwischen Logik-Forschung und and -Anwendungen, für die Korrektheit und Zuverlässigkeit oberstes Gebot sind, auf der einen Seite, und Wissensverwaltung auf der anderen Seite, die auch Skalierbarkeit und Interoperabilität betrachtet. Mit MMT haben wir eine Schnittstellensprache an dieser Grenze eingeführt, die die Methoden, Annahmen und Prioritäten beider Seiten integriert.

MMT konzentriert sich auf die Wahl der richtigen primitiven Wissensverwaltungs-Konzepte und baut auf dieser Grundlage eine skalierbare Architektur. Hier bezieht sich Architektur sowohl auf die formalen Definitionen und die Implementierung. MMT erlaubt es alle Aspekte von Logiken und Logik-Übersetzungen in einem vereinigten framework darzustellen und bietet ein einfaches und ausdrucksstarkes Modul-System, um dies auf skalierbare Weise zu tun.

Eine zentrale Frage aktueller und zukünftiger Forschung ist, wie die Repräsentationen in MMT hinsichtlich nicht-struktureller und partieller Logik-Übersetzungen verbessert werden können. Spezialisierte frameworks wie Twelf ([PS99]) und Hets erreichen dies durch allgemein-verwendbare logische beziehungsweise funktionale Programmiersprachen, wohingegen MMT als deklarative Sprache konzipiert ist. Wir finden, daß die Entwicklung einer speziellen Logik-Übersetzungs-Sprache auf der Basis von MMT der richtige Kompromiss ist.

Ein langfristiges Ziel von MMT ist es, daß MMT eine evolutionäre Entwicklung hin zum Repräsentations-Format des oben erwähnten Logik-Graphen durchläuft. Weiterhin wird MMT die Schnittstellen-Schicht zwischen der formalen mathematischen Semantik des Graphs und der Wissensverwaltungs-Dienste sein, die eingesetzt werden, um den Graphen

zu editieren, unterhalten, durchsuchen und betrachten.

Literatur

[BPTF07] C. Benzmüller, L. Paulson, F. Theiss und A. Fietzke. The LEO-II Project. In *Automated Reasoning Workshop*, 2007.

[CF58] H. Curry und R. Feys. *Combinatory Logic*. North-Holland, Amsterdam, 1958.

[dB70] N. de Bruijn. The Mathematical Language AUTOMATH. In M. Laudet, Hrsg., *Proceedings of the Symposium on Automated Demonstration*, Jgg. 25 of *Lecture Notes in Mathematics*, Seiten 29–61. Springer, 1970.

[GB92] J. Goguen und R. Burstall. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GR02] J. A. Goguen und G. Rosu. Institution morphisms. *Formal Aspects of Computing*, 13:274–307, 2002.

[HHP93] R. Harper, F. Honsell und G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

[How80] W. Howard. The formulas-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, Seiten 479–490. Academic Press, 1980.

[Lan98] S. Mac Lane. *Categories for the working mathematician*. Springer, 1998.

[Mes89] J. Meseguer. General Logics. In H.-D. Ebbinghaus et al., Hrsg., *Proceedings, Logic Colloquium, 1987*, Seiten 275–329. North-Holland, 1989.

[MML07] T. Mossakowski, C. Maeder und K. Lüttich. The Heterogeneous Tool Set. In O. Grumberg and M. Huth, Hrsg., *TACAS 2007*, Jgg. 4424 of *Lecture Notes in Computer Science*, Seiten 519–522, 2007.

[ORS92] S. Owre, J. Rushby und N. Shankar. PVS: A Prototype Verification System. In D. Kapur, Hrsg., *11th International Conference on Automated Deduction (CADE)*, Seiten 748–752. Springer, 1992.

[Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*, Jgg. 828 of *Lecture Notes in Computer Science*. Springer, 1994.

[PS99] F. Pfenning und C. Schürmann. System Description: Twelf - A Meta-Logical Framework for Deductive Systems. *Lecture Notes in Computer Science*, 1632:202–206, 1999.

[Rab08] F. Rabe. *Representing Logics and Logic Translations (Summary)*. Dissertation, Jacobs University Bremen, 2008. Available at `http://kwarc.info/frabe/Research/phdthesis_summary.pdf`.

[Tar33] A. Tarski. Pojęcie prawdy w językach nauk dedukcyjnych. *Prace Towarzystwa Naukowego Warszawskiego Wydzial III Nauk Matematyczno-Fizycznych*, 34, 1933. English title: The concept of truth in the languages of the deductive sciences.

[TV56] A. Tarski und R. Vaught. Arithmetical extensions of relational systems. *Compositio Mathematica*, 13:81–102, 1956.

Florian Rabe wurde am 28.09.1979 in Wolfsburg geboren. Nach dem Abitur und dem Grundwehrdienst studierte er von Oktober 2000 bis Oktober 2004 Informatik an der Universität Karlsruhe. 2005 wechselte er für seine Promotion an die Jacobs University Bremen. Dort arbeitete er mit Prof. Dr. Michael Kohlhase an OMDoc und mit PD Dr. Till Mossakowski vom Deutschen Forschungszentrum Künstliche Intelligenz Bremen an institutions. Während seiner dreijährigen Promotion war er ein Jahr lang visiting scholar an der Carnegie Mellon University in Pittsburgh, USA, wo er mit Prof. Dr. Frank Pfenning an dependent type theory arbeitete. Im Juni 2008 schloss er seine Promotion mit Auszeichnung ab und ist seitdem post-doktoraler Mitarbeiter an der Jacobs University. Er wurde während seines Aufenthalts in Pittsburgh vom Deutschen Akademischen Austausch-Dienst und danach von der Studienstiftung des Deutschen Volkes gefördert.