Abschlussbericht zum Projekt
Effiziente Algorithmen zur induktiven Programmsynthese

1 Allgemeine Angaben

1.1 DFG-Geschäftszeichen: SCHM 1239/6-1, 6-2

1.2 Antragssteller

Prof. Dr. **Ute Schmid**, Hochschullehrerin

geboren am 24. Januar 1965, Nationalität deutsch

Privatadresse:
Balthasar-Neumann-Straße 2, 96047 Bamberg
Telefon: 0951/2973078

1.3 Institution

Fachgebiet „Angewandte Informatik, insbesondere Kognitive Systeme“

Fakultät Wirtschaftsinformatik und Angewandte Informatik
Otto-Friedrich Universität Bamberg
Feldkirchenstraße 21, 96045 Bamberg

Telefon: 0951/863-2860 (direkt), -2861 (Sekr.); Telefax: -2862

EMail: ute.schmid@uni-bamberg.de

Web: `http://www.uni-bamberg.de/kogsys/`

1.4 Thema

Effiziente Algorithmen zur induktiven Programmsynthese
Efficient Algorithms for Inductive Program Synthesis

1.5 Berichtszeitraum

März 2009 – September 2010 (Zwischenbericht: Oktober 2007 – Februar 2009, Antragszeitraum 1: Oktober 2007 – September 2009, Antragszeitraum 2: Oktober 2009 – September 2010)

Inhaltsverzeichnis

1.6 Liste der Publikationen

Aufgeführt sind alle Schriften, die im *gesamten* Projektzeitraum aus den Arbeiten des Projekts heraus entstanden sind. Alle bereits veröffentlichten Schriften sind als pdf-Dateien verfügbar unter: `http://www.cogsys.wiai.uni-bamberg.de/effalip/publications.html` Die sechs wichtigsten Publikationen sind mit (*) gekennzeichnet.

Literatur

Crossley, N., Kitzelmann, E., Hofmann, M., & Schmid, U. (2009a). Combining analytical and evolutionary inductive programming. In B. Goerzel, P. Hitzler, & M. Hutter (Eds.), *Proceedings of the Second Conference on Artificial General Intelligence (AGI-09), Arlington, Virginia, March 6-9 2009* (p. 19-24). Amsterdam: Atlantis. (Winner of the 2009 Kurzweil Best AGI Paper Prize)

Crossley, N., Kitzelmann, E., Hofmann, M., & Schmid, U. (2009b). Evolutionary programming guided by analytically generated seeds. In A. Dourado, A. C. Rosa, & K. Madani (Eds.), *Proceedings of the International Joint Conference on Computational Intelligence (International Conference on Evolutionary Computation, Madeira, Portugal, 5-7 October 2009)* (p. 198-203). Setubal, Portugal: INSTICC Press.

(*)Flener, P., & Schmid, U. (2009). An introduction to inductive programming. *Artificial Intelligence Review*, *29*(1), 45-62. (doi:10.1007/s10462-009-9108-7)

Flener, P., & Schmid, U. (to appeara). Inductive programming. In C. Sammut & G. Webb (Eds.), *Encyclopedia of Machine Learning (to appear november 2010).* Springer.

Flener, P., & Schmid, U. (to appearb). Programming by demonstration (short entry) (to appear november 2010). In C. Sammut & G. Webb (Eds.), *Encyclopedia of machine learning.* Springer.

Flener, P., & Schmid, U. (to appearc). Trace-based programming (short entry). In C. Sammut & G. Webb (Eds.), *Encyclopedia of Machine Learning (to appear november 2010).* Springer.

Hieber, T., Hofmann, M., Kitzelmann, E., & Schmid, U. (2008). Programming recursive functions by examples (abstract). In L. Urbas, T. Goschke, & B. Velichkovsky (Eds.), *Proceedings der 9. Jahrestagung der Gesellschaft für Kognitionswissenschaft (KogWis 2008, TU Dresden, 28.9.-1.10. 2008).* (Nominiert (Platz 2) für den Brain Products KogWis'08 Poster Award)

Hofmann, M. (2008). *Automated construction of XSL-templates: An inductive programming approach.* Saarbrücken: VDM. (Publikation auf Basis der Diplomarbeit)

Hofmann, M. (2010a). Data-driven detection of catamorphisms. In R. Page,

V. Zsok, & Z. Horvath (Eds.), *Proceedings of the Eleventh Symposium on Trends in Functional Programming (Eleventh Symposium on Trends in Functional Programming, Norman, Oklahoma, USA, 17.-19.Mai 2010)* (p. 62-76). Norman, Oklahoma, USA: University of Oklahoma Printing Services.

Hofmann, M. (2010b). Data-driven detection of catamorphisms — towards problem specic use of program schemes for inductive program synthesis. In J. Hage (Ed.), *Preproceedings of the 22nd Symposium on Implementation and Application of Functional Languages (IFL 2010) (22nd Symposium on Implementation and Application of Functional Languages (IFL 2010), Alphen aan den Rijn, 1. - 3. Sept. 2010)* (p. 25-39). Utrecht: Technical Report, Department of Information and Computing Sciences, Utrecht University.

(*)Hofmann, M. (2010c). IgorII - an analytical inductive functional programming system (tool demo). In J. Gallagher & J. Voigtländer (Eds.), *Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Madrid, 18.01.2010 - 22.01.2010)* (p. 29-32). ACM SIGPLAN.

Hofmann, M. (Abgabe Oktober 2010). *Signpost search spaces using type information applied to analytical inductive functional programming.* Unpublished doctoral dissertation, Fakultät Wirtschaftsinformatik und Angewandte Informatik, Otto-Friedrich Universität Bamberg.

Hofmann, M., & Hieber, T. (2010). Automated method induction – functional goes object oriented. In U. Schmid, E. Kitzelmann, & R. Plasmeijer (Eds.), *Approaches and Applications of Inductive Programming (3rd International Workshop, AAIP'09 (Edinburgh, UK, September 2009) Revised Papers* (Vol. LNCS 5812, pp. 159–173). Springer. (pervious version published in the AAIP'09 proceedings volume, 2009)

(*)Hofmann, M., & Kitzelmann, E. (2010). I/O Guided Detection of List Catamorphisms - Towards Problem Specific Use of Program Templates in IP. In J. Gallagher & J. Voigtländer (Eds.), *Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Madrid, 18.01.2010 - 22.01.2010)* (p. 93-100). New York: ACM SIGPLAN.

(*)Hofmann, M., Kitzelmann, E., & Schmid, U. (2008). Analysis and evaluation of inductive programming systems in a higher-order framework. In A. Dengel, K. Berns, T. M. Breuel, F. Bomarius, & T. R. Roth-Berghofer (Eds.), *KI 2008: Advances in Artificial Intelligence* (p. 78-86). Berlin: Springer, LNAI 5243.

Hofmann, M., Kitzelmann, E., & Schmid, U. (2009). A unifying framework for analysis and evaluation of inductive programming systems. In B. Goertzel, P. Hitzler, & M. Hutter (Eds.), *Proceedings of the Se-*

cond Conference on Artificial General Intelligence (AGI-09), Arlington, Virginia, March 6-9 2009 (p. 55-60). Amsterdam: Atlantis.

Hofmann, M., Kitzelmann, E., & Schmid, U. (2010). Porting IgorII from MAUDE to HASKELL — Introducing a System's Design. In U. Schmid, E. Kitzelmann, & R. Plasmeijer (Eds.), *Approaches and Applications of Inductive Programming (3rd International Workshop, AAIP'09 (Edinburgh, UK, September 2009) Revised Papers* (Vol. 5812, p. 140-158). Springer, LNCS 5812. (pervious version published in the AAIP'09 proceedings volume, 2009)

Hofmann, M., & Schmid, U. (2010). Data-driven detection of recursive program schemes. In H. Coelho, R. Studer, & M. Wooldridge (Eds.), *Proceedings of the 19th European Conference on Artificial Intelligence (19th European Conference on Artificial Intelligence, Lissabon, 16. - 20. August)* (Vol. 215, p. 1063-1064). Amsterdam: IOS Press.

Kitzelmann, E. (2008a). Analytical inductive functional programming. In M. Hanus (Ed.), *Pre-proceedings of the 18th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2008, Valencia, Spain)* (p. 166-180).

Kitzelmann, E. (2008b). Data-driven induction of functional programs. In M. Ghallab, C. Spyropoulos, N. Fakotakis, & N. N. Avouris (Eds.), *ECAI 2008, 18th European Conference on Artificial Intelligence, Proceedings* (p. 781-782). IOS.

(*)Kitzelmann, E. (2009). Analytical inductive functional programming. In M. Hanus (Ed.), *Proceedings of the 18th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2008, Valencia, Spain)* (Vol. LNCS 5438, p. 87-102). Springer.

Kitzelmann, E. (2010a). *A combined analytical and search-based approach to the inductive synthesis of functional programs.* Unpublished doctoral dissertation, Fakultät Wirtschaftsinformatik und Angewandte Informatik, Otto-Friedrich Universität Bamberg.

Kitzelmann, E. (2010b). Inductive programming – a survey of program synthesis techniques. In U. Schmid, E. Kitzelmann, & R. Plasmeijer (Eds.), *Approaches and Applications of Inductive Programming (3rd International Workshop, AAIP'09 (Edinburgh, UK, September 2009) Revised Papers* (Vol. LNCS 5812, p. 50-73). Springer. (pervious version published in the AAIP'09 proceedings volume, 2009)

Kitzelmann, E., & Hofmann, M. (2008). IGOR2: an inductive functional programming prototype. In M. Ghallab, C. Spyropoulos, N. Fakotakis, & N. N. Avouris (Eds.), *Proceedings of the System Demonstrations of the 18th European Conference on Artificial Intelligence* (p. 29-31). Patras, Greece.

Kitzelmann, E., & Schmid, U. (2007). Inducing constructor systems from example-terms by detecting syntactical regularities. *Electronic Notes in Theoretical Computer Science, 174*(1), 49-63. (Proceedings of

the 7th International Workshop on Rule Based Programming (RULE 2006))

Schmid, U. (2010). *Analytical functional inductive programming as cognitive rule acquisition device.* Vortrag beim Dagstuhl-Seminar 10302, Learning paradigms in dynamic environments (Barbara Hammer, Pascal Hitzler, Wolfgang Maass, Marc Toussaint; 25.07.10 - 30.07.10.

Schmid, U., Hofmann, M., & Kitzelmann, E. (2009a). Analytical inductive programming as a cognitive rule acquisition devise. In B. Goertzel, P. Hitzler, & M. Hutter (Eds.), *Proceedings of the Second Conference on Artificial General Intelligence, Arlington (AGI-09), Virginia, March 6-9 2009* (p. 162-167). Amsterdam: Atlantis.

Schmid, U., Hofmann, M., & Kitzelmann, E. (2009b). Inductive programming – example-driven construction of functional programs. *KI*(2/09), 38-41.

(*)Schmid, U., & Kitzelmann, E. (accepted). Inductive rule learning on the knowledge level. *Cognitive Systems Research.*

Schmid, U., Kitzelmann, E., & Plasmeijer, R. (Eds.). (2009). *Proceedings of the 3rd Workshop on Approaches and Applications of Inductive Programming (AAIP'09). (in conjunction with the 14th ACM SIGPLAN International Conference on Functional Programming (ICFP 2009) Edinburgh, Scottland September 4, 2009).* Bamberg: Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik.

Schmid, U., Kitzelmann, E., & Plasmeijer, R. (Eds.). (2010). *Approaches and Applications of Inductive Programming (3rd International Workshop, AAIP'09 (Edinburgh, UK, September 2009) Revised Papers.* Springer, LNCS 5812.

2 Arbeits- und Ergebnisbericht

Im Bericht wird im Wesentlichen auf die noch nicht im Zwischenbericht abgedeckten Arbeiten aus dem ersten Antragszeitraum sowie insbesondere auf die Arbeiten im zweiten Antragszeitraum eingegangen. Wo es sinnvoll scheint, nehme ich aber auf den gesamten Projektzeitraum von drei Jahren Bezug.

Neben Ute Schmid als Projektleiterin war Emanuel Kitzelmann (wiss. Mitarbeiter) als Grundausstattung sowie Martin Hofmann (wissenschaftlicher Mitarbeiter) am Projekt beteiligt. Beide Mitarbeiter haben ihre Dissertationsarbeiten im Zusammenhang mit dem Projektthema durchgeführt.

2.1 Ausgangsfragen und Zielsetzung des Projekts

Das Projekt liefert einen Beitrag zum Forschungsgebiet der Induktiven Programmsynthese. Ziel ist die Entwicklung von Algorithmen, die aus gegeben unvollständigen Spezifikationen – typischerweise Eingabe/Ausgabe-Beispiele – deklarative – typischerweise rekursive – Programme automatisch induzieren. Synthese-Algorithmen basieren im Wesentlichen auf einer der beiden folgenden Strategien: Erzeuge-und-Teste Ansätze generieren syntaktisch korrekte Hypothesen in einer gegebenen Hypothesensprache und testen die Hypothesen gegen die gegebenen Beispiele. Analytische Ansätze arbeiten dagegen beispielgetrieben, in dem Regularitäten in den Beispielen identifiziert werden und darüber generalisiert wird.

Erzeuge-und-Teste-Ansätze können prinzipiell beliebige Programme induzieren, wenn eine hinreichend mächtige Hypothesensprache gegeben ist. Allerdings benötigen solche Ansätze meist lange Synthesezeiten. Im Gegensatz dazu sind analytische Ansätze auf ein spezifisches rekursives Programmschema beschränkt, das die Suche nach Regularitäten steuert. Dafür ist die Synthese effizient und für die synthetisierten Programme können verschiedene Eigenschaften – wie Termination oder Minimalität der Generalisierung bezüglich vorgegebener Kriterien – garantiert werden.

Zentrales Thema des Projektes war die Weiterentwicklung des Ansatzes der *analytischen*, beispielgetriebenen induktiven Programmsynthese derart, dass die Effizienz des Syntheseprozesses sowie die Garantie spezifischer Charakteristika der synthetisierten Programme beibehalten wird und gleichzeitig die Mächtigkeit des Ansatzes hinsichtlich der Menge der synthetisierbaren Programme erhöht wird.

Ziele des Projektes waren:

- Die Entwicklung eines neuen, analytischen Kern-Algorithmus, der auf Konzepten der funktionalen Programmierung beruht und Redundanzen und Ineffizienzen des ursprünglich in der Gruppe entwickelten Ansatzes vermeidet.(vgl. G2 und P1 im Erstantrag)

- Die Erweiterung des Kern-Algorithmus um suchbasierten und wissensbasierte Komponenten, insbesondere zur Berücksichtigung von Hintergrundwissen. (vgl. G1 und P2 im Erstantrag)
- Die formale Charakterisierung der Klasse der mit dem Kern-Algorithmus und dem erweiterten Algorithmus synthetisierbarer Programme. (vgl. G3 und P3 im Erstantrag)
- Die empirische Evaluation unserer Algorithmen im Vergleich zu alternativen Systemen. (vgl. G4 und P4 im Erstantrag)
- Die Erschließung von Anwendungsbereichen, beispielsweise zur Unterstützung von Endnutzer-Programmierung. (vgl. G5 und P5 im Erstantrag)
- Die Portierung des in MAUDE implementierten Prototypen nach HASKELL, um unser System als Assistenzwerkzeug für HASKELL-Programmierer anbieten zu können. (vgl. A1 und Z3 im Fortsetzungsantrag)
- Die Erhöhung der Effizienz des Synthesesystems durch Einführungen von Heuristiken. (vgl. Z1 und A2 im Fortsetzungsantrag)
- Die Einführung von Funktionen höherer Ordnung um die Menge automatisch induzierbarer Programme zu erweitern. (vgl. Z2 und A3 im Fortsetzungsantrag)
- Die Exploration weiterer Anwendungen. (optional, vgl. Z4 und A5 im Fortsetzungsantrag)

2.2 Durchgeführte Arbeiten und Ergebnisse

2.2.1 Analytischer Kern-Algorithmus (erster Antragszeitraum)

Ein analytischer Kernalgorithmus als Ausgangspunkt für die Entwicklung des Systems IGORII wurde bereits zu Beginn der ersten Antragsphase in der Konstruktor-Term-Ersetzungssprache MAUDE implementiert. Dieser Algorithmus war zunächst weniger mächtig, als das Vorläufersystem IGORI, da keine automatische *function invention* umgesetzt war (siehe Zwischenbericht).

2.2.2 Erweiterung um such- und wissensbasierte Komponenten (erster Antragszeitraum)

Ebenfalls bis zum Zwischenbericht abgeschlossen war die Erweiterung des Kernalgorithmus derart, dass automatische *function invention* sowie die Nutzung von im Hintergrundwissen vorgegebenen Funktionen möglich wird. Der Algorithmus sowie die entsprechenden Publikationen sind im Zwischenbericht dokumentiert (Kitzelmann, 2008a, 2008b; Kitzelmann & Hofmann,

2008). Zudem haben wir, unterstützt durch eine studentische Projektarbeit und eine Masterarbeit (Neil Crossley) mit einer Kombination des evolutionären Systems ADATE und IGORII experimentiert und konnten zeigen, dass ADATE von durch IGORII generierten partiellen Programmen profitiert, das heißt der Syntheseprozess deutlich effizienter wird, da der Suchraum stärker eingeschränkt wird. Die erste Publikation dieser Ergebnisse auf der AGI (Crossley, Kitzelmann, Hofmann, & Schmid, 2009a) gewann einen *best paper award*. Weitere Resultate wurden dann auch einer Konferenz im Bereich evolutionäre Algorithmen präsentiert (Crossley, Kitzelmann, Hofmann, & Schmid, 2009b).

Im Februar 2009 hat Martin Hofmann begonnen, IGORII nach HASKELL zu portieren (siehe Abschnitt 2.2.6).

2.2.3 Formale Charakterisierung (erster Antragszeitraum)

Ein formaler Rahmen zur einheitlichen Beschreibung der verschiedenen Ansätze zur induktiven Programmierung (evolutionär, induktive logische Programmierung, analytisch) wurde gleichfalls im ersten Berichtszeitraum entwickelt (Hofmann, Kitzelmann, & Schmid, 2008, 2009). Spezifisch für die Entwicklung einer gemeinsamen Betrachtungsmöglichkeit von analytisch induktiven logischen versus funktionalen Ansätzen wurde gemeinsam mit Pierre Flener (Upsala) als Kooperationspartner gearbeitet (Flener & Schmid, to appeara, 2009). Diese Arbeiten lieferten eine sehr fruchtbare Grundlage für den empirischen Vergleich verschiedener Ansätze zur induktiven Programmierung (siehe Abschnitt 2.2.4). Allerdings ist es uns bislang nicht gelungen, eine sinnvolle semantische Charakterisierung für die durch verschiedene Synthesealgorithmen induzierbaren Programmklassen zu finden. Trotz Kontakten zu verschiedenen Wissenschaftlern im Bereich der theoretischen Informatik und der algorithmischen Lerntheorie, haben wir hier bislang keinen überzeugenden formalen Zugang gefunden. Dies ist allerdings ein Mangel des gesamten Forschungsgebiets der Induktiven Programmierung und nicht spezifisch für unsere Arbeiten.

Eine im Detail formal ausgearbeitete Darstellung unseres Induktionsalgorithmus zusammen mit Beweisen der Vollständigkeit und (extensionalen) Korrektheit findet sich in der Dissertationsschrift von Emanuel Kitzelmann (Kitzelmann, 2010a).

2.2.4 Empirische Evaluation (erster Antragszeitraum)

Im ersten Antragszeitraum wurden, basierend auf der MAUDE Implementation von IGORII umfangreiche empirische Vergleiche der Leistungsfähigkeit unseres Systems mit anderen Systemen durchgeführt. Im Rahmen dieser Arbeiten wurde auch ein Repository erarbeitet, das über die von uns betreute Internet-Plattform `www.inductive-programming.org` zur Verfügung ge-

stellt wird. Der Vergleich mit klassischen Systemen der induktiven logischen Programmierung GOLEM und FFOIL, dem System FLIP für funktional-logische Programmsynthese, dem evolutionären System ADATE sowie dem suchbasierten System MAGICHASKELLER zeigt, dass IGORII bezüglich des Umfangs der automatisch synthetisierbaren Programme und der Effizienz den meisten Systemen deutlich überlegen ist (Hofmann et al., 2008, 2009). IGORII kann, Hintergrundwissen berücksichtigen, kann Prädikate und Funktionen induzieren, kann vollautomatisch wechselseitig rekursive Programme induzieren und erlaubt die automatische Induktion von Unterfunktionen. Über die Beschäftigung mit MAGICHASKELLER haben wir sehr guten Kontakt zum Autor dieses Systems, Susumu Katayama (Miyzaki University, Japan) bekommen (siehe Abschnitt 2.4.4).

2.2.5 Erschließung von Anwendungsfeldern (erster Antragszeitraum/optional zweiter Antragszeitraum)

Bezüglich der Erschließung neuer Anwendungsfelder haben wir im ersten Berichtszeitraum eine empirische Untersuchung durchgeführt, um zu prüfen, inwieweit Nutzer in der Lage sind, die geeigneten Beispiele für IGORII anzugeben. Die Untersuchung wurde im Rahmen eines studentischen Projekts (Thomas Hieber) durchgeführt und als Poster auf der KogWis08 vorgestellt. Das Poster wurde für den Best Poster Award auf Platz 2 nominiert (Hieber, Hofmann, Kitzelmann, & Schmid, 2008). Zudem haben wir uns, unterstützt durch Thomas Hieber als studentische Hilfskraft, mit der Anwendung von IGORII auf das Lernen rekursiver XSL-Templates beschäftigt. Zwar zeigte sich, dass wir gegeben die richtigen Beispiele, korrekte Templates automatisch induzieren können. Allerdings gelang die Selektion der Beispiele nur halbautomatisch. Im Rahmen einer Bachelorarbeit (Thomas Hieber) wurde untersucht, inwieweit sich unser Ansatz zum *example-driven* Programming im Kontext der automatischen Induktion von Methoden für JAVA verwenden lässt (siehe Abschnitt 2.5), (Hofmann & Hieber, 2010).

Im ersten Antragszeitraum haben wir auch begonnen, induktive Programmierung als Zugang zum Lernen auf der Wissensebene zu untersuchen. Hier geht es uns um eine Anwendung im Bereich der Kognitiven Modellierung. Wir konnten zeigen, dass IGORII dazu geeignet ist, Problemlösestrategien aus beispielhaften Aktionsfolgen zu generalisieren, Inferenzregeln wie Transitivität aus Beispiel-Relationen zu lernen sowie einfache Phrasen-Struktur-Grammatiken aus Beispielen zu induzieren (Schmid, Hofmann, & Kitzelmann, 2009a; Schmid & Kitzelmann, accepted).

2.2.6 Portierung nach Haskell und Assistenz

Das System konnte erfolgreich von der eher unbekannten Termersetzungsprache MAUDE in die funktionale Sprache HASKELL portiert werden. HAS-

KELL ist eine der modernsten, state-of-the-art funktionalen Sprachen, mit einer über 20 Jahre alten Entwicklungsgeschichte und mit einer der größten, stets wachsenden Entwickler-Communities. Die Syntax von HASKELL findet sich in vielen verwandten Sprachen wieder. Dadurch konnte das System einer breiten, nicht nur auf die HASKELL-Gemeinde beschränkten, Forscheröffentlichkeit zugänglich gemacht werden (Hofmann, Kitzelmann, & Schmid, 2010).

Das System liegt als Paket für *Cabal*, dem Paketmanagementsystem für HASKELL vor, und kann somit einfach und schnell, ohne zusätzlichen Konfigurationsaufwand, mit nur einem Befehl installiert werden.

Bei der Portierung wurde ein interaktive Shell implementiert, deren Benutzung an gängige HASKELL Interpreter angelehnt ist. Als Beispielspezifikationen akzeptiert das System korrekte HASKELL Module und liefert als Ergebnis reinen, sofort ausführbaren HASKELL Code. Dadurch wurde die Schwelle zur Verwendung als Assistenzsystem für Entwickler funktionaler Sprache, insbesondere HASKELL äußerst niedrig gehalten.

2.2.7 Einführung von Heuristiken

Um die Auswirkung verschiedener Heuristiken auf die Entwicklung des Suchbaums im Hypothesenraum (unvollständiger) Programme besser evaluieren zu können, wurde von einer studentischen Hilfskraft (Olga Yanenko) ein XML-basiertes Tool zur Visualisierung entwickelt. Dieses Tool steht unter `http://igor.yanenko.de` zur Verfügung.

Der Versuch effektivere Heuristiken zur Auswahl unvollständiger Hypothese, führte lediglich zur geringen Effizienzsteigerungen. Intensive Test, die wir unterstützt durch eine studentische Hilfskraft (Thorsten Spieker) durchgeführt haben, zeigten, dass eine Veränderung der bestehenden Heuristik, nämlich solche Programme zu bevorzugen, die zur Lösung am wenigsten Fallunterscheidungen benötigen, kontraproduktive ist. In jedem Fall führte eine Lösung mit nicht-minimaler Anzahl von Fallunterscheidung früher oder später zu einer Art Overfitting.

Jedoch konnten die gewonnen Kenntnisse dazu verwendet werden, bei von Hypothesen mit gleicher Anzahl an Fallunterscheidungen, die eine oder andere anhand weiterer Kriterien vorzuziehen. Der dadurch gewonnen Effizienzgewinn war leider nicht so groß wie erhofft, da sich die Synthesezeiten mit der neuen Reimplementierung schon bereits im Bereich von Millisekunden befinden. Die Verwendung von Funktionen höherer Ordnung als Programmschemata führte aber zu erheblichen Steigerungen von Effizienz und Effektivität.

2.2.8 Funktionen höherer Ordnung als Programmschemata

Desweiteren wurde das System dahingehend erweitert, dass es die Verwendung Funktionen höherer Ordnung (FHO) als rekursive Programmschemata unterstützt (Hofmann, 2010b; Hofmann & Kitzelmann, 2010). FHOs sind in der funktionalen Programmierung wichtiges Sprachelement, vergleichbar mit objektorientierten Entwurfsmustern, um eine wiederverwendbare Vorlage zur Lösung eines wiederkehrenden Problems zu beschreiben.

Eine einfache FHO ist die Funktion `map`, die eine Argumentfunktion auf jedes Element einer Eingabeliste anwendet:

```
map :: (a -> b) -> [a] -> [b]
map _ []     = []
map f (x:xs) = (f x) : (map f xs)
```

Das Ergebnis des Aufrufs von `map (+1) [1,2,3]` wäre zum Beispiel `[2,3,4]`. Viele FHO lassen sich durch sogenannte *universelle Eigenschaften* charakterisieren.

Im Falle der Funktion `map` ließen sich die Universellen Eigenschaften wie folgt zusammenfassen:

```
g []     = []
g (x:xs) = (f x) : (g xs)   ⟺   g = map f
```

Jede Funktion `g` kann durch `map` definiert werden, wenn sie (*a*) auf der leeren Liste definiert ist und als Ergebnis ebenfalls die leere Liste liefert, und (*b*) sich das Ergebnis für jede nicht leere Liste dadurch berechnen lässt, dass das Ergebnis einer Funktion `f`, angewendet auf das erste Element der Eingabeliste, an den Anfang des Ergebnis eines rekursiven Aufrufs mit der Restliste eingefügt wird.

Derartige FHO mit entsprechenden universellen Eigenschaften lassen sich, bewiesenermaßen, für jeden induktive definierten Datentyp finden. Die Funktion `map` lässt sich zu sogenannten natürlichen Transformationen auf beliebigen Datentypen generalisieren. Natürliche Transformationen sind Spezielfälle sogenannter Catamorphismen, welche strukturelle Rekursion in einer FHO kapseln. Catamorphismen lassen sich zur Paramorphismen für primitive Rekursion erweitern.

Es konnte gezeigt werden, dass sich die universellen Eigenschaften dieser sogenannten Typmorphismen, in den Eingabe/Ausgabe-Beispielen überprüfen lassen und sie die Zielfunktion entsprechend durch eine FHO ausdrücken lässt. Dadurch reduziert sich das Syntheseproblem lediglich auf das generieren der weniger komplexen Argumentfunktion.

Dieser Aspekt der schema-geleiteten analytischen Programmsynthese wird ausführlich in der Dissertationsschrift von Martin Hofmann (Hofmann, Abgabe Oktober 2010) dargestellt.

2.3 Wirtschaftliche Verwertbarkeit

Die im Projekt durchgeführten Arbeiten sind im wesentlichen Grundlagenforschung. Allerdings wird bereits seit vielen Jahren das Potential solcher induktiven, beispielgetriebenen Ansätze für alternative Methoden der Programmassistenz diskutiert (Flener & Schmid, 2009). Inzwischen finden sich im Bereich der funktionalen Programmierung, insbesondere im Bereich Haskell, verschiedene Forschungsarbeiten, die auf die Unterstützung der automatischen Code-Generierung durch induktive Methoden abzielen. Einige dieser Arbeiten wurden auf dem 2009 von uns, gemeinsam mit Rinus Plasmeijer organisierten Workshop *Approaches and Applications of Inductive Programming* präsentiert (Schmid, Kitzelmann, & Plasmeijer, 2010).

2.4 Kooperationspartner

2.4.1 Workshop-Reihe AAIP

Der Workshop *Approaches and Applications of Inductive Programming*, den wir 2005 erstmalig auf der ICML durchgeführt haben, wurde auch 2007 (ECML, Warschau) sowie 2009 durchgeführt. Da es sich gezeigt hat, dass im Bereich Funktionales Programmieren derzeit das Interesse an induktiven Methoden wächst, haben wir uns mit dem Workshop auf der *14th ACM SIGPLAN International Conference on Functional Programming (ICFP 2009)* in Edinburgh beworben. Der Workshop wurde erfreulicherweise akzeptiert (siehe Zwischenbericht für Programmkommittee und eingeladene Referenten). Ergebnis waren neue Kontakte zu Wissenschaftlern im Bereich Funktionale Programmierung – insbesondere zu Pieter Koopman und Rinus Plasmeijer an der Univeristät Nijmegen. Ein Post-Proceedingsband mit revidierten Beiträgen zum Workshop wurde von Springer in der LNCS Reihe akzeptiert (Schmid et al., 2010). Der Workshop soll weiterhin im zweijährigen Turnus stattfinden.

2.4.2 Inductive Programming Webseite und Mailing List

Das bereits zum Start des Projekts von uns eingerichtete Webportal `www.inductive-programming.org` haben wir kontinuierlich weitergepflegt. Im zweiten Antragszeitraum haben wir, unterstützt durch eine studentische Hilfskraft (Sanne Grabisch) eine umfangreiche Bibliographie zum Thema induktive Programmierung online gestellt. Über die Mailing Liste informieren wir in unregelmäßigen Abständen zu Aktivitäten im Bereich induktive Programmierung. Insbesondere bewerben wir hier den AAIP Workshop.

2.4.3 Nationale Kooperationen

Direkt an der Universität Bamberg kooperieren wir mit Michael Mendler (Theoretische Informatik). Michael Mendler war im Promotionskommit-

tee von Emanuel Kitzelmann und ist ebenfalls im Promotionskommittee von Martin Hofmann. Gemeinsam haben wir ein Master-Seminar durchgeführt (siehe Zwischenbericht). Im Wintersemester 2009/2010 haben Michael Mendler, Gerald Lüdtken (Programmiersprachen und Compiler) und ich gemeinsam ein Colloquium durchgeführt, in dem auch Vorträge im Zusammenhang mit dem Projekt gehalten wurden.

Während des zweiten Antragszeitraums hat sich ein intensiver Kontakt zu Janis Voigtländer (Universität Bonn) ergeben, der aus Perspektive der Funktionalen Programmierung Interesse am Thema induktive Programmierung hat. Im Dezember 2010 wird Ute Schmid im Colloquium in Bonn vortragen, ein Gegenbesuch von Janis Voigtländer im Februar 2011 ist geplant.

Über die Dissertationsarbeit von Emanuel Kitzelmann hat sich verstärkter Kontakt zu Bernd Krieg-Brückner (Universität Bremen) ergeben, über die Dissertationsarbeit von Martin Hofmann zu Petra Hofstedt (TU Cottbus). Bernd Krieg-Brückner war Zweitgutachter der Dissertation von Emanuel Kitzelmann. Petra Hofstedt ist Zweitgutachterin der Dissertation von Martin Hofmann.

2.4.4 Internationale Kooperationen

Martin Hofmann war, finanziert über ein DAAD-Stipendium für zwei Monate zu Gast in Japan, bei Susumu Katayama an der Miyzaki Universität. Dort hat er sich intensiv mit dessen System MagicHaskeller befasst. Umgekehrt plant Susumu Katayama, Aspekte unseres Ansatzes in MagicHaskeller zu integrieren. Umgekehrt war Susumu Katayama im September 2009, im Anschluss an AAIP'09 in Bamberg zu einem Gastvortrag zu Besuch.

Ebenfalls über ein DAAD-Stipendium kam Oleg Monakhov (Russian Academy of Sciences, Siberian Branch) von September bis November 2009 nach Bamberg. Oleg Monakhov arbeitet an template-basierten evolutionären Algorithmen zur induktiven Programmierung. Unser Ziel war es, IgorII als analytischen Vorverarbeitungsschritt einzusetzen, um automatisch templates für das System von Monakhov zu generieren. Leider stellte sich dieses Vorhaben als deutlich aufwendiger als geplant heraus, da das System von Monakhov auf die Synthese iterativer Programme ausgerichtet ist.

Wie in Abschnitt 2.4.1 geschildert, ergaben sich Kontakte zu Rinus Plasmeijer und Pieter Koopman über die Organisation des AAIP'09 Workshops sowie die gemeinsame Herausgebe eines Spinger-LNCS Bandes.

Schließlich hat Ute Schmid sich mit Pierre Flener gemeinsam mit dem Vergleich analytischer Ansätze des induktiven Programmierens im Kontext der logischen Programmierung und der funktionalen Programmierung auseinandergesetzt. Das Ergebnis sind Beiträge zur im November 2011 bei Springer erscheinenden *Encyclopedia of Machine Learning* (Flener & Schmid, to appeara, to appearc, to appearb) sowie eines Überblicksartikels (Flener

& Schmid, 2009).

2.5 Qualifikation des wissenschaftlichen Nachwuchses

Informationen zu allen Lehrveranstaltungen, studentischen Projekten und Qualifikationsarbeiten sind über die Webseite `http://www.uni-bamberg.de/kogsys` verfügbar.

2.5.1 Lehre

Wie im Zwischenbericht dokumentiert, wurde (und wird) das Thema Induktive Programmierung in den Vorlesungen Maschinelles Lernen und Mensch-Computer Interaktion (Anwendung für die Endnutzerprogrammierung) behandelt. Zudem fanden Masterseminare und studentische Projekte zu ausgewählten Themen aus dem Bereich Induktives Programmieren statt.

2.5.2 Bachelor-, Master-, und Diplomarbeiten

Im Januar 2009 hat Neil Crossley seine Masterarbeit im Bereich Angewandte Informatik abgeschlossen. Das Thema der Arbeit lautet *Analytical preprocessing for evolutionary inductive programming*. Die Ergebnisse der Arbeit wurden als Tagungsbeitrag auf der AGI'09 akzeptiert. Der Beitrag gewann den *best paper award*.

Im Februar 2009 hat Thomas Hieber seine Bachelor-Arbeit im Bereich Angewandte Informatik begonnen. Das Thema der Arbeit lautet *Example-driven Programming – A Tool for Automated Method Induction in Rational Architect* und liefert einen Beitrag zur Erschließung von Anwendungsmöglichkeiten der induktiven Programmierung im Bereich Software-Entwicklung. Die Arbeit wurde beim AAIP'09 Workshop vorgestellt und ist in den Post-Proceedings publiziert.

2.5.3 Promotionen

Emanuel Kitzelmann, der am Projekt als Grundausstattung beteiligt war, war wesentlich für die Entwicklung des Kernalgorithmus zur analytischen induktiven Programmierung verantwortlich. Die Konzeption, Formalisierung und Umsetzung von IGORII waren Gegenstand seiner Dissertation. Die Arbeit wurde von Ute Schmid als Erstbetreuerin, von Michael Mendler als Mitglied der Promotionskommission sowie von Bernd Krieg-Brückner (Universität Bremen) als externer Gutachter betreut. Die Promotion war im Juli 2010 abgeschlossen und wurde als erste Disseration im Bereich Angewandte Informatik an der Universität Bamberg mit *summa cum laude* bewertet. Emanuel Kitzelmann arbeitet seit September 2010 als Post-Doktorand am ICSI und beschäftigt sich nun weiter mit den von uns gemeinsam begonnen Arbeiten zur Anwendung von IGORII im Bereich KI-Planung.

Martin Hofmann, der als wissenschaftlicher Mitarbeiter im Projekt beschäftigt war, wird im Oktober 2010 seine Dissertationsarbeit abgeben. Er hat sich insbesondere mit der Erweiterung von IGORII um Funktionen höherer Ordnung befasst. Seine Arbeiten zur automatischen Identifikation von Catamorphismen im Syntheseprozess haben zu einer weiteren Steigerung der Menge der durch IGORII vollautomatisch induzierbaren Programme beigetragen. Martin Hofmanns Arbeiten haben wesentlich dazu beigetragen, dass unsere vor dem Hintergrund von Forschungsarbeiten in der Künstlichen Intelligenz begonnen Arbeiten nun auch im Bereich der Funktionalen Programmierung sichtbar sind.

3 Zusammenfassung

Im Projekt *Effiziente Algorithmen zur induktiven Programmsynthese* befassen wir uns mit der Entwicklung eines analytischen, beispielgetriebenen Ansatzes zur induktiven funktionalen Programmierung. Zunächst wurde das System IGORII entwickelt, in MAUDE implementiert, empirisch – im Vergleich zu anderen Systemen – evaluiert und in den Bereichen Endnutzer-Programmierung sowie Kognitive Modellierung angewendet. Darauf aufbauend wurde IGORII in HASKELL portiert und um die Berücksichtigung von Funktionen höherer Ordnung erweitert.

IGORII erhält als Eingabe eine kleine Menge von Ein-/Ausgabe-Beispielen. Daraus können sehr effizient und vollautomatisch linear-, baum- sowie wechselseitig rekursive Funktionen induziert werden. IGORII ist in der Lage, Hilfsfunktionen zu induzieren (*necessary function invention*) und kann Hintergrundwissen in Form von beispielhaft spezifizierten Prädikaten und Funktionen in den Syntheseprozess einbeziehen. Die Synthesezeiten liegen meist deutlich unter einer Sekunde. Die induzierten Programme sind garantiert korrekt bezüglich der gegebenen Beispielmenge und stellen die minimale Generalisierung über diese Menge dar. IGOR2 ist über die Webseite des Projekts verfügbar. IGORII ist als *conditional higher-order term rewriting* System formalisiert und die Eigenschaften des Algorithmus wurden in diesem Rahmen charakterisiert. In umfangreichen empirischen Evaluationen konnte gezeigt werden, dass IGORII eine mit alternativen Systemen vergleichbare, häufig überlegene, Leistung erbringt. Im Vergleich zu suchbasierten Verfahren ist die Mächtigkeit von IGORII kaum eingeschränkt. Ein *Repository* von Problemen ist online verfügbar.

Die Anwendbarkeit von IGORII wurde im ECLIPSE Plug-in AUTOXSL erprobt. Hier werden aus Edit-Traces in XML-Dokumenten XSL Transformationen induziert, die dann auf das gesamte Dokument angewendet werden können. Die Portierung nach HASKELL erlaubt die Nutzung von IGORII als Programmierassistent für die beispielgetriebene Induktion von Funktionen. Zudem konnte gezeigt werden, dass analytische induktive Programmierung auch in den Bereichen Problemlösen, Schließen und Sprachverarbeitung angewendet werden kann um über Regularitäten zu generalisieren.

Summary

In the project *Efficient Algorithms for Inductive Program Synthesis* we have developed an analytical, example-driven approach to inductive functional programming. At first, we developed the IGORII system, realised it in MAUDE, conducted empirical comparisons with other systems and applied it to end-user programming and cognitive modelling. In a second step, IGORII was ported to HASKELL and extended to higher-order functions.

Input to IGORII are small sets of positive input/output examples. It can induce linear, tree and mutual recursive functions efficiently and without user assistance. IGORII can perform necessary function invention and can make use of background knowledge – specified in form of further examples for predicates and functions. Synthesis time for most problems is well under one second. Induced programs are guaranteed to be correct with respect to the given examples and are minimal generalisations over these examples. IGORII can be obtained from the project website. IGORII is based on higher-order term rewriting and the characteristics of the algorithm are formalised in this framework. In extensive empirical evaluations we could show that IGORII performs very well in comparison to other systems. The scope of synthesisable programs is comparable to search-based systems. A repository of problems can be obtained from the web page.

The applicability of IGORII was tested with the ECLIPSE plug-in AUTOXSL. XSL transformations are induced from edit traces in XML documents and can be applied to the complete document. Realising IGORII in HASKELL allows to use it as programming assistant for example-driven induction of HASKELL functions. Furthermore, we could show that analytical inductive programming can be applied to induce generalised rules over observed regularities in problem solving domains, reasoning, and natural language processing.