Kombinationsmethoden für spezielle Lösungsverfahren zur Deduktionsunterstützung

Abschlußbericht

Franz Baader
Theoretische Informatik
RWTH Aachen

Klaus U. Schulz
CIS
Universität München

Der folgende Arbeitsbericht gibt einen Überblick über die Arbeiten, die im Projekt mit dem Kennwort „Kombination spezieller Lösungsverfahren“ im Zeitraum Januar 1996 bis September 1998 stattfanden. Eine kurzer Bericht über die Ergebnisse des Gesamtprojekts wurde gleichfalls der DFG zugeleitet.

1 Allgemeine Angaben

1.1 Schwerpunktprogramm Deduktion (Az.: 322 698)

1.2 Antragsteller

- Franz Baader, Prof. Dr.-Ing.
- Universitätsprofessor
- Geschäftszeichen Ba 1122/2-2
- RWTH Aachen
  Fachgruppe Informatik

  Ahornstraße 55
  52074 Aachen
  Tel: 0241/80 21130
  Fax: 0241/8888 360
  `baader@informatik.rwth-aachen.de`

- Klaus U. Schulz,
  Prof. Dr. rer. nat.
- Universitätsprofessor
- Geschäftszeichen Schu 1026/1-2
- Universität München
  Centrum für Informations- und
  Sprachverarbeitung
  Oettingenstraße 67
  80538 München
  Tel: 089/2178 2700
  Fax: 089/2178 2701
  `schulz@cis.uni-muenchen.de`

1.3 Thema

Kombinationsmethoden für spezielle Lösungsverfahren zur Deduktionsunterstützung.

1.4 Kennwort

Kombination spezieller Lösungsverfahren.

1.5 Fachgebiet und Arbeitsrichtung

Theoretische Informatik: Deduktionssysteme.

1.6 Berichtszeitraum und Förderungszeitraum

Berichtszeitraum: Januar 1996 – September 1998.
Förderungszeitraum insgesamt: September 1994 – September 1998.

1.7 Qualifikation wissenschaftlichen Nachwuchses im Zusammenhang mit dem Projekt

Aus den Arbeiten im Rahmen des Projekts entstanden Dissertationen der Projektmitarbeiter Stephan Kepser [3] und Jörn Richts [4].

2 Publikationen

Die folgenden Liste enthält die Publikationen, die im Rahmen des Projekts im *Gesamtzeitraum* entstanden sind. Eine Liste aller anderen in den nachfolgenden Abschnitten zitierten Veröffentlichungen ist am Ende des Berichts beigefügt.

2.1 Bücher und Buchbeiträge

[1] Franz Baader und Klaus U. Schulz (Hrsg.). *Frontiers of Combining Systems.* Proceedings of the First International Workshop. Kluwer Academic Publishers, 1996.

[2] Franz Baader und Klaus U. Schulz. *Unification Theory.* In Wolfgang Bibel and Peter H. Schmitt (Hrsg.). Automated Deduction. A Basis for Applications. S. 225–263. Kluwer Academic Publishers, 1998.

2.2 Dissertationen

[3] Stephan Kepser. *Combination of Constraint Systems.* Dissertation am Centrum für Infomations- und Sprachverarbeitung, Universität München, 1998.

[4] Jörn Richts. *Effiziente Entscheidungsverfahren zur E-Unifikation.* Dissertation an der Mathematisch-Naturwissenschaftlichen Fakultät der RWTH Aachen, 1998.

2.3 Zeitschriftenartikel

[5] Franz Baader und Klaus U. Schulz. *Combination Techniques and Decision Problems for Disunification.* Theoretical Computer Science, 142:229–255, 1995.

[6] Franz Baader und Werner Nutt. *Combination Problems for Commutative/Monoidal Theories: How Algebra Can Help in Equational Reasoning.* Journal on Applicable Algebra in Engineering, Communication and Computing, 7(4):309–337, 1996.

[7] Franz Baader und Klaus U. Schulz. *Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures.* Journal of Symbolic Computation, 21:211–243, 1996.

[8] Franz Baader. *On the Complexity of Boolean Unification.* Information Processing Letters, 67(4):215–220:1998.

[9] Franz Baader und Klaus U. Schulz. *Combination of Constraint Solvers for Free and Quasi-Free Structures.* Theoretical Computer Science, 192:107–161, 1998.

[10] Klaus U. Schulz. *Tractable and Intractable Instances of Combination Problems for Unification and Disunification.* Journal of Logic and Computation. Erscheint demnächst.

[11] Klaus U. Schulz und Stephan Kepser. *Combination of Constraint Solvers II: Rational Amalgamation.* Theoretical Computer Science. Erscheint demnächst.

2.4 Konferenzbeiträge

[12] Franz Baader und Klaus U. Schulz. *Combination of Constraint Solving Techniques: An Algebraic Point of View.* In Jieh Hsiang (Hrsg.). Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA-95. Band 914 der Lecture Notes in Computer Science, S. 352–366. Springer-Verlag, 1995.

[13] Franz Baader und Klaus U. Schulz. *On the Combination of Symbolic Constraints, Solution Domains, and Constraint Solvers.* In Ugo Montanari und Francesca Rossi (Hrsg.). Proceedings of the International Conference on Constraint Programming, CP-95. Band 976 der Lecture Notes in Computer Science, S. 380–397. Springer-Verlag, 1995.

[14] Klaus U. Schulz. *On Existential Theories of List Concatenation.* In Leszek Pacholski und Jerzy Tiuryn (Hrsg.). Selected Papers of CSL'94. Band 933 der Lecture Notes in Computer Science, S. 294–308. Springer-Verlag, 1995.

[15] Stephan Kepser und Klaus U. Schulz. *Combination of Constraint Solvers II: Rational Amalgamation.* In Eugene Freuder (Hrsg.) Proceedings of the International Conference on Constraint Programming, CP-96. Band 1118 der Lecture Notes in Computer Science, S. 282–296. Springer-Verlag, 1996.

[16] Franz Baader. *Combination of Compatible Reduction Orderings that are Total on Ground Terms.* In G. Winskel (Hrsg.). Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science (LICS-97), S. 2–13. IEEE Computer Society Press, 1997.

[17] Franz Baader und Cesare Tinelli. *A New Approach for Combining Decision Procedures for the Word Problem, and Its Connection to the Nelson-Oppen Combination Method.* In William McCune (Hrsg.). Proceedings of the 14th International Conference on Automated Deduction (CADE-97). Band 1249 der Lecture Notes in Artificial Intelligence, S. 19–33. Springer-Verlag, 1997.

[18] Klaus U. Schulz. *A Criterion for Intractability of E-unification with Free Function Symbols and its Relevance for Combination of Unification Algorithms.* In Hubert Comon (Hrsg.). Rewriting Techniques and Applications, 8th International Conference (RTA-97), Band 1232 der LNCS, S. 284–298. Springer, 1997.

[19] Stephan Kepser. *Negation in Combining Constraint Systems.* In Maarten de Rijke and Dov Gabbay (Hrsg.). Frontiers of Combining Systems, Proceedings of the Second International Workshop. Kluwer Academic Publisher, 1998.

[20] Stephan Kepser und Jörn Richts. *Optimisation Techniques for Combining Constraint Solvers.* In Maarten de Rijke and Dov Gabbay (Hrsg.). Frontiers of Combining Systems, Proceedings of the Second International Workshop. Kluwer Academic Publisher, 1998.

3 Arbeits- und Ergebnisbericht

Im folgenden geben wir einen Überblick über die Arbeiten, die im Projekt im Zeitraum Januar 1996 bis September 1998 stattfanden. In Abschnitt 4 erfolgt eine kurze Zusammenfassung der wichtigsten Ergebnisse.

3.1 Ausgangsfragen und Zielsetzung des Projekts

Bevor wir auf die Zielsetzung und die in dieser Richtung erhaltenen Resultate näher eingehen, soll zunächst nochmals kurz der für das Projekt relevante Kontext erläutert sowie auf die in der ersten Phase des Projekts erzielten Ergebnisse eingegangen werden. (Eine ausführlichere Darstellung findet sich im Bericht über die erste Projektphase.)

Wir betrachten eine Situation, in der die Effizienz einer universellen Deduktionskomponente (z.B. eines Theorembeweisers, eines Knuth-Bendix-Vervollständigungsverfahrens oder einer Logischen Programmiersprache) durch den Einbau von Spezialverfahren zur effizienten Behandlung von Teilproblemen erhöht werden soll. Wichtige Beispiele für solche Spezialverfahren sind Algorithmen für Unifikation modulo Gleichungstheorien, durch die eine effizientere Gleichheitsbehandlung in Theorembeweisern erreicht werden kann, oder Constraint-Lösungsverfahren, durch die z.B. Datenstrukturen wie Zahlen, Listen und Mengen in Logische Programmiersprachen eingebaut werden können. In den meisten Anwendungen genügt allerdings der Einbau eines einzigen Spezialverfahrens nicht: man benötigt verschiedene Spezialverfahren, die in geeigneter Weise miteinander kombiniert werden müssen. Die Integration von Spezialverfahren (wie Unifikationsalgorithmen oder Constraint-Lösungsverfahren) in allgemeine Deduktionskomponenten würde also erheblich durch ein (möglichst allgemeines) Verfahren erleichtert, das es erlaubt, solche Spezialverfahren formal sauber und algorithmisch effizient miteinander zu kombinieren.

Für die disjunkte Kombination von Unifikationsalgorithmen, d.h. für Unifikation in der Vereinigung von Gleichungstheorien über disjunkten Signaturen, war dieses Kombinationsproblem aus theoretischer Sicht bereits vor Beginn des Projekts weitgehend gelöst: Schmidt-Schauß [30] beschreibt ein Verfahren zur Kombination von Algorithmen, welche vollständige Mengen von Unifikatoren berechnen, während das von uns entwickelte Verfahren [21] sowohl Entscheidungsverfahren, als auch Verfahren zur Berechnung vollständiger Mengen handhaben kann. Beide Verfahren beinhalten aber nicht-deterministische Schritte, welche bei einer naiven Implementierung zu unvertretbar großen Suchräumen führen. Während eine optimierte Version des Verfahrens von Schmidt-Schauß bereits zu Projektbeginn bekannt war [23], war das Problem der Optimierung von Verfahren zur Kombination von Entscheidungsverfahren zu diesem Zeitpunkt fast vollständig offen. Auch zu Erweiterungen der Kombinationsmethode auf allgemeinere Situationen gab es zu Beginn des Projekts nur sehr wenige Arbeiten.

Die Ziele für die erste Projektphase (von Herbst 1994 bis Herbst 1996), die in diesem Zeitraum auch weitgehend erreicht wurden, waren daher die folgenden:

1. *Implementierung* der von den Antragstellern entwickelten Kombinationsmethoden für Unifikation im Falle disjunkter Gleichungstheorien. Zum Austesten dieser Implementierung sowie der Optimierungen wurden außerdem für einige spezielle Theorien Lösungsverfahren für Unifikationsprobleme implementiert: es handelte sich hier um Unifikationsalgorithmen für die freie Theorie, die Theorie A eines assoziativen Symbols, die Theorie AC eines assoziativ-kommutativen Symbols und die Theorie ACI eines assoziativ-kommutativ-idempotenten Symbols. Die für diese Theorien bekannten Unifikationsalgorithmen mußten

zu Algorithmen für Unifikation mit linearen Konstantenrestriktionen (LKR) erweitert werden, da das in [21] beschriebene Kombinationsverfahren diese als Eingabe benötigt.

2. Entwicklung von *Optimierungsmethoden*, insbesondere für den Fall der Kombination von Entscheidungsverfahren. Diese sollten in die Implementierung eingebracht und für die betrachteten Theorien evaluiert werden. Neben einfachen Optimierungen des Basiskombinationsverfahrens wurden zwei orthogonale Optimierungsrichtungen verfolgt. Zum einen stellte sich heraus, daß bei der direkten Kombination von $n > 2$ Theorien spezielle Optimierungen möglich und nötig werden, die bei 2 Theorien nicht auftreten. Diese Optimierungen wurden in dem sogenannten „iterativen Verfahren“ realisiert, das eine geeignete Selektionsstrategie für die nächste zu treffende nicht-deterministische Entscheidung liefert. Zum anderen wurde in dem „deduktiven Verfahren“ versucht, durch sogenannte „Vortests“, welche von den zu kombinierenden Theorien bereitgestellt werden müssen, teilweise nicht-deterministische Entscheidungen ganz zu vermeiden. Implementiert wurden derartige Vortests zunächst für die freie Theorie, A, AC, und ACI sowie für die Klassen der regulären/kollapsfreien Theorien. Da beide Optimierungsansätze orthogonal zueinander sind, können sie gleichzeitig eingesetzt werden.

3. *Erweiterung* der Kombinationsmethoden auf nicht-disjunkte Theorien und allgemeinere Constraint-Systeme. In [12] wurde zunächst das Kombinationsverfahren auf allgemeinere Constraint-Sprachen erweitert: zusätzlich zu den Gleichheits-Constraints der Form $s = t$ in Unifikationsproblemen konnten nun auch relationale Constraints (z.B. Ordnungs-Constraints der Form $s \leq t$) behandelt werden. Als Lösungsstrukturen betrachtet man hierbei freie Strukturen anstelle der freien Algebren bei der Unifikation. In [13, 9] wurde dann eine Verallgemeinerung der freien Strukturen (sogenannte quasi-freie Strukturen) eingeführt, auf die unsere Kombinationsmethode weiterhin anwendbar ist. Interessante Beispiele für quasi-freie, aber nicht freie Strukturen aus dem Bereich der Constraint-Programmierung sind die Algebra der rationalen Bäume, Feature-Strukturen sowie Bereiche sogenannter nicht-fundierter Mengen und Listen. Voraussetzung für eine Verallgemeinerung des Basiskombinationsverfahrens auf quasi-freie Strukturen war zunächst, eine Form der Kombination zweier quasi-freier Lösungsstrukturen anzugeben, die die Bildung der gemeinsamen Quotientenalgebra im Falle der Unifikationsalgorithmen verallgemeinert. Dies leistet der in [13, 9] eingeführte Begriff des „freien amalgamierten Produkts“ zweier quasi-freier Strukturen.
   Die Behandlung der Kombination von Unifikationsalgorithmen für nicht-disjunkte Theorien stellte sich als wesentlich schwieriger heraus als erwartet, weshalb hierzu in der ersten Projektphase keine Ergebnisse erzielt wurden.

Die Einteilung in diese drei Hauptzielrichtungen blieb im Verlängerungsantrag im wesentlichen unverändert. Allerdings war zu diesem Zeitpunkt, wie oben erwähnt, das Basiskombinationsverfahren bereits verallgemeinert worden auf die Kombination von Constraint-Löseverfahren für quasi-freie Strukturen. Daraus ergab sich das Ziel, die erreichten Implementierungen und Optimierungen auf diese Verallgemeinerung zu übertragen. Als zusätzliche Komponentenverfahren sollten Boolesche Unifikation, *AC1*-Unifikation sowie Constraint-Löseverfahren für Feature-Strukturen und rationale Bäume implementiert werden. Darüberhinaus sollten für SETHEO eine inkrementelle Variante des Kombinationsverfahrens entwickelt und implementiert werden und für praktisch besonders relevante spezielle Kombinationen möglichst effiziente Implementierungen bereitgestellt werden.

Als Erweiterung sollten zum einen die wichtigsten Alternativen zur freien Amalgamierung von Lösungsstrukturen und Constraint-Löseverfahren untersucht werden sowie ein Verfahren zur Kombination von Constraint-Sprachen mit Negation entwickelt werden. Zum anderen sollte nochmals das offene Problem der Kombination nicht-disjunkter Theorien angegangen werden. Da sich dies für Unifikationsalgorithmen als sehr schwierig erwiesen hatte, sollten zunächst etwas einfachere Problemstellungen, wie die Kombination von Entscheidungsverfahren für das Wortproblem, betrachtet werden.

In den folgenden Abschnitten 3.2 bis 3.4 gehen wir zunächst auf die zu diesen drei Punkten erzielten Resultate und auf die Entwicklung der durchgeführten Arbeiten näher ein. In Abschnitt 3.5 erfolgt eine kurze Diskussion der wissenschaftlichen Fortschritte und der Anwendungsperspektive.

3.2 Implementierung

Nachdem die theoretischen Ergebnisse des ersten Projektzeitraums [13, 9] gezeigt hatten, daß eine einfache Erweiterung des Basiskombinationsverfahrens zur Kombination von Constraint-Löseverfahren über sogenannten quasi-freien Strukturen verwendet werden kann, wurde diese Erweiterung in die bestehende Implementierung aufgenommen.

Als Spezialverfahren zum Austesten dieses erweiterten Kombinationsverfahrens wurden Constraint-Löseverfahren für Feature-Strukturen und für rationale Bäume implementiert. Das Verfahren für rationale Bäume konnte durch eine relativ einfache Modifikation des bereits implementierten Unifikationsalgorithmus für die freie Theorie erhalten werden. Die theoretische Grundlage für die Behandlung von Feature-Strukturen war die Arbeit von Smolka und Treinen [31], da die hier betrachtete Variante von Feature-Strukturen – bei geeigneter Erweiterung auf den Nicht-Grundfall – in der Tat unsere Definition einer quasi-freien Struktur erfüllt. Das in [31] beschriebene Constraint-Löseverfahren stellte sich aber als zu ineffizient und schwer implementierbar heraus, weshalb unsere Implementierung auf dem in [34] beschriebenen praktikableren Verfahren beruht. Dieses mußte allerdings zu einem Verfahren erweitert werden, das mit linearen Konstantenrestriktionen umgehen kann.

Das Kombinationsverfahren wurde außerdem dahingehend erweitert, daß es inkrementell arbeiten kann. Dadurch muß bei Hinzunahme neuer Constraints zu einer bereits auf Erfüllbarkeit getesteten gemischten Constraint-Menge nicht nochmals der gesamte Suchbaum des nichtdeterministischen Kombinationsverfahrens durchlaufen werden. Durch eine geeignete Abspeicherung des beim Test der ursprünglichen Constraint-Menge durchlaufenen Suchbaums kann an der Stelle aufgesetzt werden, an der bei diesem Test Erfüllbarkeit festgestellt wurde. Die bereits als unerfüllbar festgestellten Alternativen werden also nicht nochmals durchsucht. Auf Seiten der Spezialverfahren wurden inkrementell arbeitende Varianten für die freie Unifikation sowie für Constraint-Löseverfahren über rationalen Bäumen und Feature-Strukturen erstellt.

Deutlich mehr Zeit als erwartet beanspruchte im zweiten Projektzeitraum die Implementierung eines Verfahrens zur Booleschen Unifikation mit LKR, da hier unerwartet auch noch interessante theoretische Fragestellungen zu lösen waren. Deshalb wurde auf die gleichfalls geplante Implementierung eines Verfahrens zur *AC1*-Unifikation mit LKR verzichtet. Aus der Literatur waren zwei Algorithmen zur Berechnung allgemeinster Unifikatoren für Boolesche Unifikation mit Konstanten bekannt, die auf Boole bzw. Löwenheim zurückgehen (siehe [26]). Zusammen mit dem in [30] beschriebenen Verfahren zur Konstantenelimination liefern diese Verfahren auch ein Entscheidungsverfahren für Booleschen Unifikation mit LKR. Im Rahmen des Projekts wurden beide Unifikationsalgorithmen sowie das Konstanteneliminationsverfahren implementiert. Das so erhaltene Entscheidungsverfahren für Booleschen Unifikation mit LKR erschien uns aber zunächst als unbefriedigend, da es auf der Berechnung allgemeinster Unifikatoren beruht, welche im schlimmsten Fall exponentielle Größe haben können. Es stellte sich daher die Frage, ob man direkt ein Entscheidungsverfahren erhalten kann, das keine derart großen Unifikatoren berechnet und damit von einer geringeren Komplexität ist. Es stellte sich hierbei heraus, daß – im Gegensatz zu allen anderen im Projekt untersuchten Gleichungstheorien – ein Entscheidungsverfahren für Boolesche Unifikation mit LKR nicht durch eine einfache Modifikation des Verfahrens für Boolesche Unifikation mit Konstanten erhalten werden kann. In [8] konnte die genaue Komplexität der verschiedenen Booleschen Unifikationsprobleme bestimmt werden: während elementare Boolesche Unifikation „nur" *NP*-vollständig ist und Boolesche Unifikation mit Konstanten auch noch relativ niedrig in der polynomiellen Hierarchie angesiedelt ist (Π_2^p-vollständig), ist Boolesche Unifikation mit LKR PSPACE-vollständig. Damit ist die Theorie Boolescher Algebren das erste Beispiel einer Gleichungstheorie, bei der ein solcher Komplexitätssprung beobachtet wurde.

3.3 Optimierungsmethoden

Erweiterung der Optimierungsansätze. Die beiden in der ersten Projektphase entwickelten Optimierungsansätze wurden auch auf das erweiterte Kombinationsverfahren für Constraints über quasi-freien Strukturen erweitert. Die für das deduktive Verfahren nötigen Vortests wurden für Feature-Strukturen und rationale Bäume entwickelt und implementiert.

Die im Verlängerungsantrag beschriebene Idee, bei den theoriespezifischen Vortests auch solche Informationen mitzuberücksichtigen, die nur Abhängigkeiten zwischen verschiedenen Entscheidungen angeben (ohne eine Entscheidung vollständig deterministisch zu machen), hat leider nicht zu einer Steigerung der Effizienz geführt. Mehrere Versuche, solche Informationen im Rahmen eines intelligenten Backtrackings auszunutzen, führten eher zu Effizienzverlusten infolge zusätzlichen Verwaltungsaufwands. Diese Optimierungsrichtung wurde daher nicht weiterverfolgt.

Erfolgreicher war die – ebenfalls im Verlängerungsantrag angedachte – Idee, bereits entwickelte Vortests für eine Theorie auf schwierigere handhabbare Teiltheorien anzuwenden. Als Beispiel boten sich hier die Theorie *ACI* und ihre Teiltheorie *A* an, da einerseits *A*-Unifikation sehr aufwendig ist und andererseits *ACI*-Unifikation sowie der zugehörige Vortest polynomiell sind. Die Verwendung des *ACI*-Vortests bei der *A*-Unifikation mit freien Funktionssymbolen führte zu einer deutlichen Effizienzsteigerung.

UniMoK-System und empirische Resultate. Um den Effekt der Optimierungen bewerten zu können, wurden umfangreiche Laufzeittests durchgeführt. Die im vorangegangenen Bericht erwähnten ersten Laufzeittests wurden erheblich ausgedehnt, wobei sich der erste Eindruck bestätigte, daß die durch die Optimierungen erreichten Laufzeitverbesserungen das Verfahren auch für praktische Aufgaben einsetzbar machen. Jede Zeile in der folgenden Tabelle enthält die Laufzeiten für eine Beispielmenge, die aus mehreren Unifikationsproblemen besteht.

Die ersten drei Zeilen enthalten alle Unifikationsprobleme, die der Beweiser Reveal beim Lösen eines Beweisproblems zu unifizieren hat; sie enthalten ein oder zwei *AC*-Symbole und mehrere freie Symbole. Bei diesen Beispielen zeigt sich, daß das iterative Verfahren keinen wesentlichen zusätzlichen Laufzeitgewinn zum deduktiver Verfahren bringt. Wenn die Vortests des deduktiven Verfahrens wie bei den hier verwendeten Theorien sehr viel Information liefern, macht sich das iterative Verfahren nur bei größeren Problemen mit mehr als drei Theorien bemerkbar, wie sie in

	Zeit in Sekunden					
Anzahl	i+d	ded	i+d-	ded-	it	orig
29	3.7	3.7	5.0	5.0	11.6	17.2
1002	154	155	1442	1488	∞	∞
404	109	108	∞	∞	∞	∞
98	36	48	∞	∞	∞	∞
100	135	141	1157	∞	∞	∞
195	395	1520	∞	∞	∞	∞
194	167	223	∞	∞	∞	∞
197	273	1286	∞	∞	∞	∞

Legende: Anzahl: Anzahl der Unifikationsprobleme; i+d: Integration von iterative und deduktiver Methode; ded: deduktive Methode; i+d-/ded-: wie i+d/ded, aber die *AC*-Komponente benutzt nur Kollapsfreiheit und Regularität; it: iterative Methode; orig: unoptimierter Algorithmus; ∞: das Verfahren wurde nach einer Stunde abgebrochen.

den letzten fünf Zeilen enthalten sind. Diese Beispiele wurden von einem Termgenerator erzeugt und enthalten zwei *AC*-Symbole, zwei *ACI*-Symbole und freie Symbole.

Die in diesem Abschnitt beschriebenen Optimierungen wurden von den Mitarbeitern entworfen und mit Unterstützung der Hilfskräfte implementiert. Alle Verfahren sind in COMMON LISP unter Verwendung von KEIM implementiert. Diese *Unification Modules for Keim* sind als UniMoK-System unter der URL `http://www-lti.informatik.rwth-aachen.de/Forschung/unimok.html` verfügbar. Für einen Teil der Programme wurden darüberhinaus PROLOG-Versionen implementiert. Nicht erreicht wurde das im Verlängerungsantrag genannte Ziel, eine inkrementelle Variante des Kombinationsverfahren in SETHEO zu integrieren. Das zunächst erhebliche Engagement wurde zuerst dadurch zurückgeworfen, daß der zuständige und auf das Problem eingearbeitete Mitarbeiter der Gruppe von Herrn Letz überraschend in die Industrie wechselte. Nachdem sich mit dem neu eingearbeiteten Nachfolger später genau dieselbe Situation nochmals wiederholte und darauf auch noch die für die PROLOG bzw. SETHEO-Implementierungen zuständige Hilfskraft am CIS-München die Arbeitsgruppe verließ, ohne daß ein Ersatz gefunden werden konnte, war es unmöglich, dieses Ziel im Projektzeitraum zu realisieren.

Grenzen der Optimierung. Um die Qualität der entwickelten Optimierungsansätze nicht nur empirisch, sondern auch analytisch zu untersuchen, stellten wir uns die Frage, was die Grenzen derartiger allgemeiner (d.h. nicht auf eine konkrete Kombination von Theorien ausgerichteter) Optimierungsansätze sind. In [18] konnte gezeigt werden, daß in fast allen interessanten Fällen eine polynomielle Optimierung des Verfahrens (unter der Annahme $P \neq NP$) prinzipiell unmöglich ist. Sobald etwa die freie Theorie mit einer regulären Gleichungstheorie E kombiniert wird, die ein assoziatives oder kommutatives Funktionssymbol enthält, kann es keinen polynomiellen Vortest für E geben, der für beliebige Input-Probleme zu einem polynomiell großen Suchbaum führt. Die Regularität der Theorie E spielt hierbei lediglich eine beweistechnische Rolle, es ist keinesfalls zu erwarten, daß sich im nicht-regulären Fall die Situation besser darstellt. Diese Resultate scheinen einen engen Rahmen für etwaige weitere Verbesserungen der Verfahren abzustecken.

Andererseits konnte eine (allerdings sehr eingeschränkte) Klasse von Theorien gefunden werden, deren Unifikationsverfahren mit nur polynomiellem Aufwand kombiniert werden können. In Anlehnung an [18] wurde hierzu in [10] ein hinreichendes Kriterium entwickelt, das die Existenz eines deterministisch-polynomiellen Kombinationsverfahrens für zwei Gleichungstheorien sicherstellt. Von den bekannten Theorieklassen erfüllen lediglich Theorien, die gleichzeitig unitär, regulär und kollapsfrei sind, dieses Kriterium. Eine Analyse des optimierten Kombinationsverfahrens zeigte [20], daß sich in diesem Fall bereits aus den von uns betrachteten Optimierungen ein solches optimales Verfahren ergibt, obwohl diese Optimierungen für den allgemeinen Fall entwickelt wurden. Auch hierdurch erhärtete sich die Vermutung, daß diese Optimierungsansätze nicht mehr wesentlich zu verbessern sind.

3.4 Erweiterungen

Alternative Kombinationen von Lösungsstrukturen. Im Bereich der Logik-Programmierung mit Constraints werden häufig Lösungsstrukturen verwendet, die sich als eine Kombination einfacherer Strukturen auffassen lassen (siehe z.B. [24, 29, 27]). Eine Analyse dieser Beispiele zeigte aber, daß bei der Kombination von Strukturen mit „unendlichen" Elementen (wie rationalen Bäume, nicht-fundierten Listen oder zyklischen Feature-Strukturen) eine Art der Kombination bevorzugt wird, bei der in den Elementen der kombinierten Lösungsstruktur unendlich viele „Signaturwechsel" auftreten können. Dies ist bei den Elementen des von uns definierten freien amalgamierten Produkts ausgeschlossen. In [15, 11] wurde daher in Form der sogenannten „rationalen Amalgamierung" eine allgemeine Methodologie zur Kombination sogenannter kollapsfreier quasi-freier Strukturen beschrieben, für die die erwähnten Beispiele aus der Literatur spezielle Instanzen darstellen. Darüberhinaus wird ein allgemeiner Rahmen für mögliche weitere Amalgamierungskonstruktionen umrissen. Auch im Fall der rationalen Amalgamierung läßt sich ein allgemeines Verfahren zur Kombination von Constraint-Lösern angeben [15, 11], das sehr ähnlich zu dem Kombinationsverfahren für die freie Amalgamierung ist. Es unterscheidet sich von diesem im wesentlichen nur dadurch, daß einer der drei nicht-deterministischen Schritte (Wahl der linearen Ordnung) vollständig entfallen kann, was einen Effizienzgewinn darstellt.

Ein weiterer alternativer Ansatz zur Kombination von Lösungsstrukturen wurde im Kontext von Reduktionsordnungen, die mit einer Gleichungstheorie kompatibel sind, betrachtet. Ein wichtiges Beispiel für die in [12] betrachteten freien Strukturen sind Termalgebren modulo einer Gleichungstheorie E_i, die mit einer E_i-kompatiblen Reduktionsordnung $<_i$ ausgestattet sind. Werden zwei derartige Stukturen E_1 und E_2 über disjunkter Signatur miteinander durch freie Amalgamierung kombiniert, so liefert dies eine kombinierte Struktur, deren Trägermenge die kombinierte Termalgebra modulo $E_1 \cup E_2$ ist. Diese Struktur besitzt zwei Relationen $<_1$ und $<_2$, die aber keine Reduktionsordnungen für diese Struktur sind. In [16] wurde gezeigt, wie man unter Verwendung der Reduktionsordnungen $<_i$ auf den Komponententermalgebren eine neue Ordnung $<$ auf der kombinierten Termalgebra modulo $E_1 \cup E_2$ definieren kann, welche ein $(E_1 \cup E_2)$-kompatible Reduktionsordnung ist und noch weitere angenehme Eigenschaften (wie Totalität auf Grundtermen) hat. Die Konstruktion derartiger kombinierter Ordnungen war lange Zeit sogar für Spezialfälle (wie der Kombination zweier AC-Symbole) ein offenes Problem.

Negationsbehandlung. Während die Kombination von Unifikationsalgorithmen bereits in [5] auf Disunifikation erweitert werden konnte, war zu Beginn der zweiten Projektphase zunächst nicht klar, ob auch bei allgemeineren Constraints über quasi-freien Strukturen negierte Constraints durch unseren Kombinationsansatz behandelt werden können. Der in [13, 9] verwendete algebraische Ansatz zum Nachweis der Korrektheit des Kombinationsverfahrens war direkt nicht auf negierte Constraints anwendbar, da er Abschlußeigenschaften *positiver* Formeln verwendete.

In [3, 19] konnte aber gezeigt werden, daß das Kombinationsverfahren auf negierte Constraints erweiterbar ist. Das dort geschilderte erweiterte Verfahren kann (implizit existentiell quantifizierte) Konjunktionen von positiven oder negierten Atomformeln behandeln. Außerdem wurde in [3, 19] die Modularität der sogenannten „Unabhängigkeitseigenschaft“ für negierte Constraints untersucht. Es wird ein hinreichendes Kriterium angegeben, wann die freie Amalgamierung von Strukturen mit Unabhängigkeitseigenschaft wieder zu einer Struktur mit dieser Eigenschaft führt. Für die rationale Amalgamierung wird in [3] gezeigt, daß man hier im Gengensatz zur freien Amalgamierung keine allgemeinen Resultate zur Negationsbehandlung erwarten kann.

Kombination bei nicht-disjunkter Signatur. Zu dem Problem der Kombination von Spezialverfahren für nicht-disjunkte Theorien wurden erste Ergebnisse zur Kombination von Entscheidungsverfahren für das Wortproblem erzielt. Das in [17] beschriebene Verfahren, welches gemeinsame Konstantensymbole zuläßt, konnte inzwischen auf ein Verfahren zur Behandlung gemeinsamer „Konstruktoren“ erweitert werden [22].

Zur Kombination von Unifikationsalgorithmen für nicht-disjunkte Theorien konnten bisher keine befriedigenden Ergebnisse erzielt werden. Zwar ist die abstrakte Definition des freien amalgmierten Produkts in [9] nicht auf disjunkte Signaturen eingeschränkt und es ist uns inzwischen auch relativ klar, unter welchen Bedingungen die explizite Amalgamierungskonstruktion auf den nicht-disjunkten Fall übertragen werden kann. Es ist aber nicht klar, wie ein allgemeines Kombinationsverfahren für diesen Fall auszusehen hat.

Eine spezielle Art der nicht-disjunkten Kombination von Unifikationsalgorithmen wurde in [6] betrachtet. Hier werden die Einzeltheorien (welche über disjunkten Signaturen gegeben sind) nicht einfach durch Vereinigung kombiniert, sondern es werden zusätzliche Interaktionsaxiome angegeben, welche beide Signaturen verwenden. Ein Beispiel ist die Kombination der Theorie $\mathit{AC1} := \{x * (y * z) = (x * y) * z,\ x * y = y * x,\ x * 1 = x\}$ mit der Theorie einer Involution $\mathit{Inv} := \{h(h(x)) = x\}$, wobei die Interaktionsaxiome durch $\mathit{IA} := \{h(x * y) = h(x) * h(y),\ h(1) = 1\}$ gegeben sind. Die Resultate in [6] zeigen, wie man aus einem Entscheidungsverfahren für $\mathit{AC1}$-Unifikation eines für Unifikation modulo $\mathit{AC1} \cup \mathit{IA} \cup \mathit{Inv}$ erhalten kann.

3.5 Wissenschaftliche Fortschritte und Anwendungsperspektive

Wie das Schaubild in Abb. 1 zu verdeutlichen sucht, betreffen die im *Gesamtprojekt* erzielten Ergebnisse im wesentlichen drei Anwendungsgebiete:

1. Auf dem Gebiet der automatischen Deduktion die Integration und Kombination von Gleichungstheorien in Resolutionsverfahren und Knuth-Bendix Vervollständigung.
2. Auf dem Gebiet der Logik-Programmierung mit Constraints das Problem der Kombination von Constraint-Systemen.
3. Auf dem Gebiet der Algebraischen Spezifikation die Kombination von Entscheidungsverfahren zum Wortproblem.

Man sollte bei einer Bewertung zwischen praktischen Ergebnissen (die vor allem das erste der genannten Gebiete betreffen) und theoretischen Ergebnissen (die alle drei Gebiete betreffen und darüberhinaus allgemeiner das grundsätzliche Problem der Kombination formaler Systeme) unterscheiden.

Aus praktischer Sicht stehen in Form des UniMoK-Systems erstmals allgemein kombinierte Entscheidungsverfahren zur Unifikation zur Verfügung, deren Effizienz ausreichend ist, damit ein Einbau in Theorembeweiser mit Gleichheitsbehandlung sinnvoll erscheint. Damit ist zunächst eine interessante Möglichkeit für Kopplungsexperimente gegeben. Freilich wird auf Seiten der

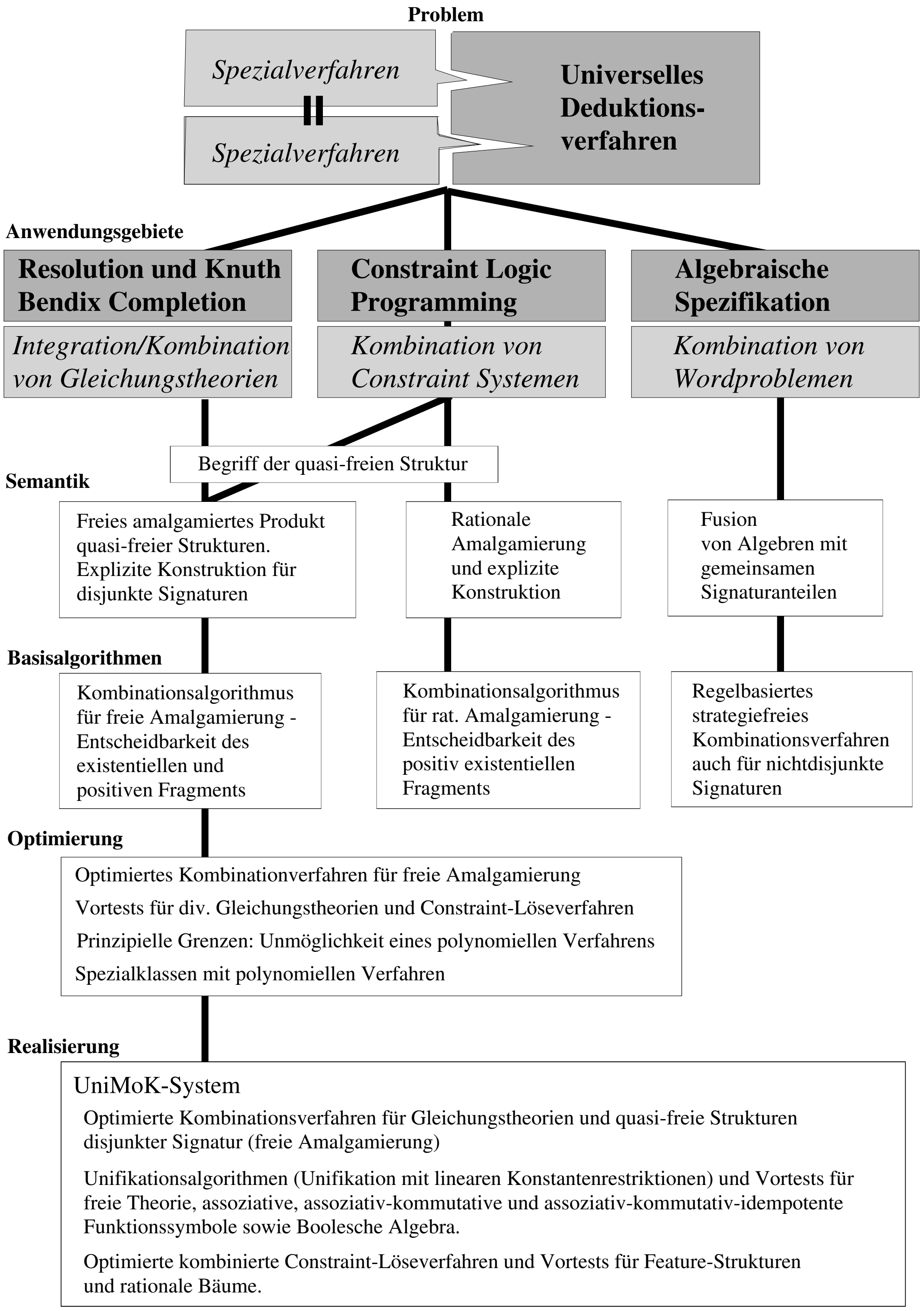

Abbildung 1: Darstellung der Projektergebnisse im Überblick

universellen Komponente noch genauer zu klären sein, wann und in welcher Form die Spezialverfahren in optimaler Weise aufzurufen sind, um den bestmöglichen Vorteil zu erzielen. Die in [18] beschriebenen prinzipiellen Grenzen der Optimierungsmöglichkeiten für Kombinationsverfahren stecken dagegen einen engen Rahmen für etwaige weitere Verbesserungen auf Seiten der Spezialverfahren ab.

Aus theoretischer Sicht haben die semantischen Begriffsbildungen (quasi-freie Strukturen, amalgamiertes Produkt) in [12, 13, 9] zweifellos zu einem qualitativ wesentlich verbesserten theoretischen Verständnis der Kombinationsproblematik geführt. Insbesondere wurde mit diesen Begriffen und durch die darauf aufbauenden rein algebraischen Korrektheitsbeweise der zugehörigen Kombinationsverfahren erstmals der logisch und algebraische Hintergrund des Problems geklärt. Die hierdurch gewonnenen Einsichten haben mittlerweile auch in Nachfolgearbeiten anderer Forschungsgruppen zur Kombinationsproblematik einen deutlichen Niederschlag gefunden [28, 32, 33].

Die Arbeiten [13, 9, 15] beschreiben erstmals für den Bereich der Logik-Programmierung mit Constraints allgemein anwendbare und formal fundierte Methoden zur Kombination von Constraint-Systemen. Freilich ist der dort betrachtete Rahmen sehr allgemein. Eine genaue Diskussion der Leistungsfähigkeit der optimierten Kombinationsverfahren für Constraints über den in der Praxis verwendeten speziellen Lösungsstrukturen, für die sehr effiziente Komponentenverfahren existieren, wäre noch zu leisten.

Im Bereich der nicht-disjunkten Kombination wurden erste Ansätze zur Kombination von Entscheidungsverfahren für das Wortproblem entwickelt. Das sehr schwierige Problem der nicht-disjunkten Kombination von Unifikationsalgorithmen ist noch weitgehend offen. Seit der Arbeit [25] gibt es hierzu keine wesentlichen neuen Fortschritte in Richtung auf ein möglichst allgemeines Kombinationsverfahren.

4 Kurzfassung der Ergebnisse des Gesamtprojekts

Durch die Integration spezieller Lösungsverfahren in eine universelle Deduktionskomponente kann die effiziente Behandlung spezieller Probleme mit der universellen Anwendbarkeit des allgemeinen Deduktionsverfahrens verbunden werden. Ziel des Projekts war es, die algorithmischen Probleme, die bei der Integration mehrerer Spezialverfahren in eine universelle Deduktionskomponente entstehen, systematisch zu studieren und für wohlverstandene Teilprobleme effiziente Kombinationsverfahren zu implementieren und optimieren. Daraus ergaben sich konkret drei Teilziele: Implementierung, Optimierung und Erweiterung der existierenden Kombinationsverfahren. Wir fassen zunächst die zu diesen Punkten geleisteten Arbeiten zusammen, ehe wir die wesentlichen Aspekte der erreichten Ergebnisse kurz kommentieren.

Implementierung. Als Ausgangspunkt wurde das in [21] beschriebene Verfahren zur Kombination von Unifikationsalgorithmen implementiert und auf eine Reihe von Spezialverfahren angewandt, für die ebenfalls Implementierungen bereitgestellt wurden. Das Kombinationsverfahren wurde dann dahingehend erweitert, daß eine Kombination von Constraint-Löseverfahren für allgemeinere (z.B. relationale) Constraints auch über nicht-freien Lösungsstrukturen möglich ist. Auch hier wurden entsprechende Spezialverfahren implementiert.

Optimierung. Als Hauptergebnis wurde ein optimiertes Verfahren für die direkte Kombination von $n \geq 2$ Theorien entwickelt, das mit Hilfe theoriespezifischer Vortests nicht-deterministische Verzweigungen soweit wie möglich vermeidet. Passende Vortests wurden für die freie Theorie, A, AC, ACI und die Klasse der regulären/kollapsfreien Theorien entwickelt. Die Optimierungen und Vortest wurden auf die oben genannten Erweiterungen übertragen. Die Algorithmen wurden in Gestalt der Toolbox „UniMoK“ in das KEIM-System integriert. Eine Evaluierung zeigt, daß mit dem erzielten Ergebnis erstmals eine Verwendung allgemein kombinierter Entscheidungsverfahren zur Unifikation praktisch möglich ist [20]. In [18] wurden die Grenzen der Optimierungsmöglichkeiten aufgezeigt. Die in [10] lokalisierte Klasse von Theorien, für die eine polynomielles Kombinationsverfahren existiert, wird auch durch unsere allgemeinen Optimierungsansätze optimal behandelt [20].

Erweiterungen. Das Ausgangsverfahren wurde dahingehend erweitert, daß nunmehr Constraint-Löseverfahren für sogenannte „quasi-freie" Strukturen über disjunkten Signaturen kombinierbar sind [13, 9], wobei auch Constraint-Sprachen mit Negation behandelbar sind [3, 19]. Der Nachweis der Korrektheit des Kombinationsverfahrens beruht auf einer algebraischen Konstruktion (der sog. freien Amalgamierung) der kombinierten Lösungsstruktur. Eine alternative Methode zur Kombination quasi-freier Strukturen über disjunkten Signaturen sowie entsprechende Constraint-Löseverfahren, welche im Kontext der Logischen Programmierung mit Constraints von Bedeutung sind, wurde in [15, 11] angegeben.

Wissenschaftliche Fortschritte und Anwendungsaspekte. Als praktisches Ergebnis des Projekts stehen in Form einer Toolbox nunmehr erstmals kombinierte Entscheidungsverfahren zur Unifikation zur Verfügung, deren Effizienz ausreichend für die praktische Verwendung in Theorembeweisern mit Gleichheitsbehandlung ist. Die in [18] beschriebenen prinzipiellen Grenzen der Optimierungsmöglichkeiten für Kombinationsverfahren scheinen einen engen Rahmen für etwaige weitere Verbesserungen abzustecken.

Die zentralen Arbeiten [13, 9, 15, 11] beschreiben erstmals für den Bereich der Logik-Programmierung mit Constraints allgemein anwendbare und formal fundierte Methoden zur Kombination von Constraint-Systemen. Die Arbeiten [12, 13, 9] haben zweifellos zu einem qualitativ wesentlich verbesserten theoretischen Verständnis der Kombinationsproblematik geführt. Insbesondere wurde dort erstmals der logisch und algebraische Hintergrund des Problems geklärt.

Ergänzende Literatur

[21] Franz Baader und Klaus Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In Deepak Kapur (Hrsg.) *Automated Deduction, Proceedings CADE-11*, Band 607 der *LNAI*, S. 50–65. Springer-Verlag, 1992.

[22] Franz Baader und Cesare Tinelli. *Deciding the word problem in the union of equational theories sharing constructors.* Technical Report, Department of Computer Science, University of Illinois, Urbana-Champaign, Illinois. Erscheint demnächst.

[23] Alexandre Boudet. Unification in a combination of equational theories: An efficient algorithm. In Mark E. Stickel (Hrsg.) *Automated Deduction, Proceedings CADE-10*, Band 449 der *LNAI*, S. 292–307, Springer–Verlag, 1990.

[24] Alain Colmerauer. An introduction to PROLOG III. *Communications of the ACM*, 33:69–90, 1990.

[25] Eric Domenjoud, Francis Klay und Christophe Ringeissen. Combination techniques for non-disjoint equational theories. In Alan Bundy (Hrsg.) *Automated Deduction, Proceedings CADE-12, Nancy, France*, Band 814 der *LNAI*, S. 267–281, Springer-Verlag, 1994.

[26] Ursula Martin und Tobias Nipkow, Boolean Unification – The Story So Far. *Journal of Symbolic Computation*, 7:275–293, 1989.

[27] K. Mukai. *Constraint Logic Programming and the Unification of Information*. PhD thesis, Department of Computer Science, Faculty of Engineering, Tokio Institute of Technology, Japan, 1991.

[28] Christophe Ringeissen. Cooperation of decision procedures for the satisfiability problem. In *Baader & Schulz* [1], S. 121–140, 1996.

[29] William C. Rounds. Set values for unification based grammar formalisms and logic programming. Technical Report CSLI-88-129, CSLI, Stanford University, 1988.

[30] Manfred Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8(1,2):51–99, 1989.

[31] Gert Smolka und Ralf Treinen. Records for Logic Programming. *Journal for Logic Programming*, 18(3): 229–258, 1994.

[32] Cesare Tinelli und Mehdi Harandi. A new correctness proof of the Nelson-Oppen combination procedure. In *Baader & Schulz* [1], S. 103–120, 1996.

[33] Cesare Tinelli und Christophe Ringeissen. Non-Disjoint Unions of Theories and Combinations of Satisfiability Procedures: First results. Technical Report no. UIUCDCS-R-98-2044, Department of Computer Science, University of Illinois, Urbana-Champaign, Illinois, April 1997 (ebenfalls erhältlich als INRIA Research Report no. RR-3402).

[34] Peter Van Roy, Michael Mehl und Ralf Scheidhauer. Integrating efficient records into concurrent constraint programming. In *8th International Symposium on Programming Languages, Implementations, Logic, and Programs (PLILP'96)*, Aachen, 1996.