Ein erweitertes CLP–Schema für eine hybride Wissensverarbeitung

Holger Wache, Panagiotis Tsarchopoulos
DFKI
Postfach 2080
67608 Kaiserslautern
Deutschland
e-mail:{*wache*, *tsarcho*}@dfki.uni-kl.de

28. Mai 1993

Zusammenfassung

In diesem Papier stellen wir einen Ansatz für ein hybrides Wissensrepräsenation und -verarbeitungssystem vor. Algebraische Constraints, wie sie in CLP($\mathbb{R}$) System formuliert werden können, als auch Constraints über endliche Wertemengen werden als Konkrete Bereiche in ein terminologisches System integriert. Terminologische Logik und Constraints werden unter Zuhilfenahme des erweiterten CLP–Schemas in Hornklauseln eingebettet. Der Ansatz besitzt eine wohl–fundierte Semantik und verspricht außerdem eine hohe Effizienz, da auf spezialisierte Algorithmen zurückgegriffen werden kann.

Die Integration wird an einem Beispiel aus der Konfigurierung verdeutlicht. Außerdem wollen wir weitere Entwicklungsmöglichkeiten für unseren Ansatz aufzeigen.

1 Motivation

Das Expertenwissen über eine Domäne läßt sich im allgemeinen nicht in einem uniformen Formalismus (z.B. Regeln) adäquat repräsentieren. Dies gilt insbesondere für komplexe Anwendungen wie zum Beispiel im Bereich der Konfiguration. In der Regel werden in einem uniformen Formalismus unterschiedliche Wissensarten wie zum Beispiel Wissen über Objekte, Wissen über deren (strukturelle und funktionale) Abhängigkeiten oder Reihenfolgewissen miteinander vermischt. Stattdessen sollten mehrere Repräsentationsformalismen zur Verfügung gestellt werden, um die unterschiedlichen Wissensarten deklarativ und adäquat beschreiben zu können. Aus diesem Grunde wurden verschiedene hybride Wissensrepräsentationssysteme wie BABYLON [CdPV89] oder COLAB [BHHM93] entwickelt. Die Systeme erlauben die Verwendung von verschiedenen Formalismen zur Repräsentation unterschiedlicher Teilaspekte eines komplexen Problems. Die meisten Systeme verwenden Taxonomien zur Repräsentation von Objekten, (Vorwärts– und/oder Rückwärts)Regeln, sowie Constraints.

Ein wichtiger Punkt bei hybriden Repräsentationssystemen ist neben der Darstellung des Wissens dessen effiziente Verarbeitung. Prinzipiell kann die Inferenz auf zwei Arten erfolgen. Zum einen kann man eine "universelle" Inferenzmethode für verschiedene Repräsenationsformalismen entwickeln. Im Projekt VEGA [BHH+92] wird unter anderem ein solcher Ansatz verfolgt. Zum anderen kann man die Methoden der verschiedenen Formalismen miteinander koppeln, indem man sie

über klar definierte Schnittstellen kommunizieren läßt. Dieser Ansatz hat den Vorteil, das die einzelnen Inferenzmethoden erhalten bleiben und daher auf effiziente Algorithmen, die bisher separat für die Methoden entwickelt wurden, zurückgegriffen werden kann. Das Metainterpreter–Konzept von BABYLON [CdPV89] oder das CLP–Schema [JL87] sind Beispiele für diesen Ansatz. Dabei hat sich gezeigt, das ein semantisch klares Konzept für die Integration der Methoden notwendig ist.

In diesem Papier werden wir einen Ansatz vorstellen, in dem terminologische Logiken zur Repräsentation der Begriffe (z.B. Objekte) einer Domäne, algebraische Constraints und Constraints über endliche Wertebereiche in Hornklauseln eingebettet werden. Die Inferenzmethoden sind auf der Basis einer logisch fundierten Semantik miteinander gekoppelt. An einem einfachen Beispiel, der Konfigurierung eines PC's, wollen wir veranschaulichen, wie die unterschiedlichen Wissensrepräsenationsformalismen verwendet werden können und welche weitere Arbeiten noch ausstehen.

2 Wissensrepräsentation auf der Basis terminologischer Logiken

Terminologische Systeme erlauben die Formulierung von Begriffen (Objekte, *Konzepten*) einer Domäne. Die Beschreibung der Konzepte erfolgt extensional über ihren strukturellen Beziehungen zu anderen Konzepten. Die terminologische Logiken lassen sich auf eine entscheidbare Teilmenge der Prädikatenlogik (Sortenlogik) zurückführen. Sie zeichnen sich daher gegenüber anderen Frame–Sprachen durch eine klare Semantik aus.

Terminologische Systeme[1] bestehen in der Regel aus zwei Teilen: einer T–Box und einer A–Box. In der T–Box wird das terminologische Wissen abgelegt. Das heißt, in der T–Box werden die Konzepte einer Domaine definiert. Dabei werden, ausgehend von einfachen Konzepten, mittels *Konstruktionsoperatoren* wie $C \sqcap D$ (AND), $C \sqcup D$ (OR) und $\neg C$ (NOT) neue, komplexere Konzepte gebildet (z.B. $E := C \sqcup D$). Konzepte können über zweistellige Relationen (Rollen R oder Attribute f) mittels $\exists\ R.C$ ($\exists\ f.C$) und $\forall\ R.C$ ($\forall\ f.C$) miteinander in Beziehung gesetzt werden.

Terminologische Systeme bieten den Vorteil, das aus den Konzeptbeschreibungen mit vollständigen und korrekten Algorithmen die Objekttaxonomie (Subsumptionsbeziehungen) automatisch generiert werden kann. Diese Dienstleistung kann zur Wissensaquisition herangezogen werden.

Neben der Definition können Begriffe für eine konkrete Problemstellung instanziiert werden (assertionales Wissen). Die Instanzen (*Individuen*) werden in Form von Zusicherungen (*assertionale Ausdrücke*) in die A–Box der terminologischen Systeme eingetragen. Ein Individuum läßt sich als Ausprägung einer Klasse (Konzept) oder — im Sinne der Sortenlogik — als ein Element einer Sorte (= Konzept) verstehen. Als Zusicherungen lassen sich Individuen explizit Konzepten zuordnen $(i : C)$[2], Relationen über Individuen $(R(j, i))$ definieren und Individuen miteinander gleichsetzen $(i = j)$.

Die A–Box stellt verschiedene Inferenzmethoden zur Verfügung:[3]

- *Konsistenztest*
  Die Konsistenz des in der A–Box enthaltenen assertionalen Wissens in Bezug zum terminologischen Wissen in der T–Box kann überprüft werden.

[1] Für eine genauere Beschreibung terminologischer Logiken siehe z.B. [HN90]

[2] Zur besseren Unterscheidung werden wir Individuen der A–Box klein und Konzepte der T–Box mit einem großen Anfangsbuchstaben schreiben.

[3] Die Inferenzmethoden werden in Wahrheit auf einen grundlegenden Mechanismus, dem Konsistenztest abgebildet.

- *Realisierung*
  Die Realisierung stellt eine Funktion dar, die ein Individuuum aufgrund des assertionalen Wissens einer Menge von Konzepten zuordnet. Diese Zuordnung ist nicht wie in anderenen Objektsystemen statisch, sondern ändert sich in Abhängigkeit zum Inhalt der A–Box. Es ist zwischen zwei Arten von Realisierungen zu unterscheiden:
  - *starke Realisierung*
    Es wird eine (minimale) Menge von speziellsten Konzepten zurückgeliefert, zu denen es bezüglich einer Open–World–Semantik *sicher* zugeordnet werden kann.
  - *schwache Realisierung*
    Es wird eine (minimale) Menge von speziellsten Konzepten zurückgeliefert, zu denen es zugeordnet werden kann. In der Regel sind dies terminale Konzepte.

Der Konsistenztest wird zur Einbettung in Hornklauseln als "Unifikationsmethode" benötigt. Auf die Realisierung werden wir bei der Diskussion des Beispiels in Abschnitt 5 zurückkommen.

3 Konkrete Bereiche

Terminologische Systeme besitzen eine wohl–definierte Semantik und ein hohes Maß an Deklarativität. Aber für reale Anwendungen ist ihre Ausdrucksmächtigkeit in der Regel zu schwach. Dieser Schwachpunkt tritt zum Beispiel in technischen Domänen zutage: Zahlen können nur als abstrakte Konzepte definiert werden; ihre inhärente Bedeutung geht verloren. In der terminologischen Logik werden Konzepten in der Semantik, die auf dem Interpretationsmodell der klassischen Logik basiert, keine semantische, sondern nur eine syntaktische Bedeutung beigemessen. Arithmetische Operationen $(+, -)$ über den reelen Zahlen oder Vergleichsprädikate $(=, \leq)$ lassen sich deshalb gar nicht darstellen.

Zur Beseitigung dieser Defizite erweiterten Baader und Hanschke terminologische Logiken um das Konzept der *Konkreten Bereiche* [BH91]. Der Konkrete Bereich stellt sogenannte *Konkrete Prädikate* zur Verfügung, die im terminologischen System gewissermaßen als "Built-In's" verwendet werden können. Über die Struktur und Beschaffenheit des Konkreten Bereichs werden aus der Sicht der terminologischen Logik nur wenige Aussagen gemacht. Es wird im wesentlichen für den Konsistenztest der A–Box ein (effizientes) Entscheidungsverfahren gefordert, das eine Menge von Konkreten Prädikate auf deren Konsistenz hin überprüft.

Das Konzept der Konkreten Bereiche ähnelt dem CLP–Ansatz. Im CLP–Schema [JL86] wird eine Schnittstelle definiert, über die ein (beliebiger) Constraintsolver in ein logisches Paradigma wie z.B. *PROLOG* integriert werden kann. Die gleiche Funktion übernehmen die Konkreten Bereiche für die terminologische Logik.

Über die Konkreten Bereiche lassen sich die verschiedene Methoden — auch gleichzeitig — in ein terminologisches System einbinden. Die Konkreten Prädikate fungieren als Methodenaufrufe. Als Beispiele für Methoden können OR-Verfahren oder algebraische Constraints genannt werden. Durch die Integration von OR-Verfahren kann auf effiziente Algorithmen zur Lösung eines speziellen (Sub–)Problems zurückgegriffen werden. Mit algebraischen Constraints lassen sich (reelle) Zahlen und algebraische Zusammenhänge $(X = Y + Z)$ in terminologischen Systemen darstellen. Analog

zu verschiedenen CLP($\mathbb{R}$)–Systemen (vgl. [JMSY92]) basiert der algebraische Constraintsolver auf der Simplex-Methode.

Es können aber auch Methoden verwendet werden, die eigenständiges Wissen benutzen. Datenbanken oder Constraints über endliche Wertebereiche können über die Schnittstelle der Konkreten Bereiche angeschlossen werden, sofern sie die Restriktionen der Konkreten Bereiche erfüllen. Insbesondere Constraints über endliche Wertebereiche helfen zum Beispiel in Anwendungen, wo eine Vielzahl von Objekten (> 1000) repräsentiert werden müssen, indem sie die Objekttaxonomie klein halten: Korreliert bei einem Konzept ein Attributwert mit anderen Attributwerten desselben Konzepts, so müssen in herkömmlichen terminologischen Systemen für jedes Korrelationstupel ein Subkonzept gebildet werden. Drückt man aber die Korrelation der Attributwerte als Constraint über endliche Wertebereiche aus, so reicht eine einzige Konzeptdefinition aus. Es brauchen keine Subkonzepte gebildet werden. Dieser Effekt kommt besonders dann zum Tragen, wenn mehrere Korrelationen der Attributwerte bei einem Objekt auftreten.

Konkrete Bereiche erweitern die Ausdrucksmächtigkeit von terminologische Logiken, indem sie die Einbindung spezieller Repräsentionsformalismen und deren effiziente Inferenzmethoden erlauben. Die Integration basiert auf einer klaren und wohl–fundierten Semantik [BH91].

4 Einbettung in die logische Programmierung

Terminologische Systeme sind nur im beschränkten Maße geeignet, das assertionale Wissen in der A–Box zu erweitern. Sie können im wesentlichen nur die Konsistenz des assertionalen Wissens überprüfen. Viele Anwendungen zeichen sich jedoch duch einen inkrementellen Lösungsgenerierungsprozeß aus (z.B. die Konfigurierung). Heuristisches Problemlösungswissen läßt sich als terminologisches Wissen in der T–Box nicht darstellen. Es ist daher notwendig, terminologische Systeme in eine größere Umgebung einzubetten.

In [Abe92] wird eine Integration von terminologischer Logik in Hornklauseln vorgeschlagen. Diesen Ansatz wollen wir hier aufgreifen.

Analog zu dem CLP-Schema, wo Constraints in Hornklauseln integriert werden, erweitern wir Hornklauseln um Zusicherungen an eine A–Box. Der Rumpf einer Hornklausel besteht somit aus zwei Teilen: einer Menge von Prädikaten und einer Menge von A–Box–Ausdrücken. A–Box–Ausdrücke können Zusicherungen an eine A–Box ($i : C$, $R(i, j)$, $i = j$), algebraische Constraints über reele Zahlen, wie sie aus CLP($\mathbb{R}$)[JMSY92] bekannt sind, Constraints über endliche Wertebereiche (vgl. ConTax [Mey92]), sowie allgemein Konkrete Prädikate ($P[i, j, k]$) für zusätzliche Inferenzmethoden sein. Die Constraintsolver sowie die zusätzlichen Inferenzmethoden werden als Konkrete Bereiche in das terminologische System eingebunden.

Es läßt sich zeigen, daß der Ansatz eine Instanz des allgemeinen Höhfeld/Smolka–Schemas [HS88] ist. Das Höhfeld/Smolka–Schema erweitert das CLP–Schema insofern, als das schwächere Annahmen über das eingebundene Constraintssystem gemacht werden. A–Box–Ausdrücke können in diesem Sinne als Constraints verstanden werden. Eine deklarative Semantik für diesen Integrationsansatz läßt sich somit direkt aus der Fixpunktsemantik des Schemas ableiten. Eine operationale Semantik wird in [Abe92] angegeben.

Die Integration der terminologische Logik und Constraints bietet mehrere Vorteile. Durch die Integration terminologischer Logiken steht ein Typenkonzept auf der Seite der Hornlogik zur Verfügung.

Zusammen mit den Constraints können so Einschränkungen an eine mögliche Lösung formuliert werden. Es besteht die begründete Hoffnung, das aufgrund der Einschränkungen inkonsistente Lösungen früher erkannt werden können und dadurch der Suchaufwand reduziert wird. Da die einzelnen Inferenzmethoden für die Repräsentationsarten weitgehend erhalten bleiben, kann auf effiziente, da spezialisierte Verfahren zurückgegriffen wird.

5 Ein Beispiel

Als Beispiel wollen wir die Konfigurierung eines PC's behandeln. In der Konfigurierungsaufgabe soll ein PC aus einer Menge von Einzelteile wie Motherboard oder Festplatten zusammengestellt werden. Bei der Zusammenstellung müssen Anforderungen eines Benutzers (z.B. langsamer oder schneller PC oder Festplattengröße) und die internen Abhängigkeiten (eine Festplatte muß mit dem Bussystems verträglich sein) erfüllt werden. Die Konzepte (Begriffe) in einer Konfigurierungsaufgabe sind also die Einzelteile, aus denen ein PC bestehen kann.

Ein PC besteht in unserem, vereinfachten Beispiel aus einem einer Menge von *Board*'s, die in ein Gehäuse gesteckt werden. *Board*'s können Motherboards, Festplatte, Diskettenstation oder Streamer sein. Die Boards kommunizieren über ein Bussystem, zu dem alle Boards kompatibel sein müssen. Die Konfigurierungsaufgabe besteht darin, die geeigneten Boards aufgrund von Anforderungen zu bestimmen.

Objekte des Diskurses sind die Boards eines PC's und werden daher als Konzepte in die T–Box eines terminologischen Systems modelliert. In unserem Beispiel wollen wir uns auf die Modellierung von Motherboards und Festplatten beschränken:

$$(attributes\ \ Speed,\ Bustype,\ CPU)$$

$$\begin{array}{lcl}
KindOfCPUs & := & (386 \sqcup 486) \\
KindOfBustypes & := & (AT \sqcup LocalBus \sqcup EISA) \\
 & & \\
Boards & := & (\exists\ Bustype.KindOfBustypes) \\
 & & \\
Motherboard & := & Board \sqcap \\
 & & (\exists\ Speed.(Slow \sqcup Fast)) \sqcap \\
 & & (\exists\ CPU.KindOfCPUs) \\
Motherboard386 & := & (\exists\ Speed.Slow) \sqcap (\exists\ Bustype.AT) \sqcap (\exists\ CPU.386) \\
Motherboard486 & := & (\exists\ Speed.Fast) \sqcap (\exists\ Bustype.EISA) \sqcap (\exists\ CPU.486)
\end{array}$$

Das (abstrakte) Konzept *Motherboard* wird durch drei Attribute *Speed*, *Bustype* und *CPU* beschrieben. Als "Werte" für die Attribute kommt nur eine Menge von (primitiven) Konzepten in Frage (Value Restriction). *Motherboard*386 und *Motherboard*486 sind konkrete Motherboards (Einzelteile).

Neben den *Motherboard*'s werden noch Festplatten repräsentiert:

$(attributes\ \ Size)$

$$
\begin{array}{lcl}
Harddisk & := & Board \sqcap \\
 & & (\exists\ >[Size,0]) \sqcap \\
Harddisk20MBAT & := & Harddisk\ \sqcap\ (\exists\ =[Size,20])\ \sqcap\ (\exists\ Bustype.LocalBus) \\
Harddisk20MBEISA & := & Harddisk\ \sqcap\ (\exists\ =[Size,20])\ \sqcap\ (\exists\ Bustype.EISA) \\
Harddisk40MBAT & := & Harddisk\ \sqcap\ (\exists\ =[Size,40])\ \sqcap\ (\exists\ Bustype.AT)
\end{array}
$$

Hervorzuheben ist das Attribute $Size$, welches auf den algebraischen Constraintsolver abgebildet wird. Ein PC läßt sich folgendermaßen darstellen:

$(roles\ \ HasBoards)$

$$
BusCompatible \Leftrightarrow \{(AT,\ AT);\ (AT,\ localBus);\ (EISA,\ EISA)\ \ldots\}
$$

$$
\begin{array}{lcl}
PC & := & (\forall\ HasBoards.Board) \\
 & & (\forall\ BusCompatible[(HasBoards.Bustype), \\
 & & \qquad\qquad (HasBoards.Bustype)])
\end{array}
$$

Der PC besteht also aus einer Menge von Boards (Role $HasBoards$). Die Verträglichkeit der Boards untereinander wird durch das Constraint $BusCompatibel$ gewährleistet. Durch die Darstellung als Constraints können Äquivalenzklassen über Attributewerte definiert werden. Allgemein lassen sich mittels Constraints in Konkreten Bereichen bidirektionale Korrelationen über Attributausprägungen formulieren.

Allgemein kann gesagt werden, das in einer Konfigurierungsaufgabe Komponenten und deren strukturellen Beziehungen in terminologischen Systemen dargestellt werden können. Spezielle Attributbeziehungen wie Korrelationen lassen sich auf Constraintmechanismen im Konkreten Bereich abbilden.

Neben der Modellierung der Konzepte in terminologischen Systemen muß das Problemlösungswissen repräsentiert werden. In der Konfigurierungsaufgabe sind damit die Konfigurierungsschritte gemeint. Allgemein kann man zwischen Methoden zur Auswahl, dem Parametrieren und dem Zusammenfügen von Komponenten unterscheiden [Hei93]. Die Auswahl von Komponenten kann durch den (schwachen) Realisierungsmechanismus der A–Box erfolgen, während das Parametrieren und Zusammenfügen in dem vorgestellten Schema auf der Ebene der Hornklauseln repräsentiert werden muß.

configure :- ?X:*PC*,
config–motherboard(?X),
config–mass-memory(?X).

config–motherboard(?PC) :- ?MB: *Motherboard*, *HasMotherboard*(?PC, ?MB),
parameter(speed, ?MB–SPEED),
Speed(?MB, ?MB–SPEED),
refine(?MB).

config–mass-memory(?PC) :- ?BD: *Harddisk*, *HasBoards*(?PC, ?BD),
Size(?BD, ?BD–SIZE),
parameter(memory–size, ?MEMORY–SIZE),
?MEMORY–SIZE <= ?BD–SIZE,
refine(?BD).

config-mass-memory(?PC) :- ?BD1: *Harddisk*, *HasBoards*(?PC, ?BD1),
?BD2: *Harddisk*, *HasBoards*(?PC, ?BD2),
Size(?BD1, ?BD1–SIZE),
Size(?BD2, ?BD2–SIZE),
parameter(memory–size, ?MEMORY–SIZE),
?MEMORY–SIZE <= ?BD1–SIZE + ?BD2–SIZE,
refine(?BD1),
refine(?BD2).

Anforderungen an einen PC werden in Form von Fakten (**parameter**) in die Faktenbasis eingetragen. Aufgrund der Anforderungen muß das Motherboard (**config–motherboard**) und die Festplatten (**config–mass-memory**) bestimmt werden. Dazu werden Einträge in die A–Box gemacht. Das Built-In **refine** erzwingt eine Verfeinerung eines Individuums bezüglich der Subsumptionshierarchie in der T–Box. Konkret wird mittels der schwachen Realisierung eine Menge von terminalen Konzepten (konkrete Boards) ermittelt und das Individuum einem Konzept hypothetisch zugeordnet.

Bei der Formulierung des **config–mass-memory** Prädikats wurde implizit heuristisches Vorgehenswissen verwendet. Zuerst soll versucht werden, mit einer Festplatte die Anforderung des Benutzer, ausgedrückt durch das algebraische Constraints ?MEMORY–SIZE <= ?BD–SIZE, zu erfüllen. Wenn die Anforderungen nicht erfüllt werden können, dann kann noch eine zweite Festplatte hinzugenommen werden.

Der Nachteil dieser Modellierung ist, das Vorgehenswissen und die deklarative Formulierung eines Konfigurierungsschritts miteinander vermischt werden. Das Schema sollte daher um eine Komponente erweitert werden, in der Vorgehenswissens explizit dargestellt werden kann. Dieses Wissen kann z.B. dann zur Auswahl der Regeln, die die einzelnen Konfigurierungsschritte repräsentieren, herangezogen werden.

In [Hei93] wird ein anderer Konfigurierungsansatz vorgestellt. Dieser Ansatz basiert auf der Erkenntnis, das die zu konfigurierenden Komponenten durch Funktionen beschrieben werden können, die sie entweder zur Verfügung stellen oder benötigen. Durch das Zusammenfügen von Komponenten wird die Gesamtfunktion eines zu konfigurierenden Geräts gewährleistet. In diesem Ansatz werden Beziehungen zwischen den Komponenten nicht explizit dargestellt, sondern es wird von strukturellen Beziehungen auf funktionale Abhängigkeiten abstrahiert.

Funktionelle Abhängigkeiten können direkt in terminologische Systemen nicht dargestellt werden. In [Wac93] werden allerdings erste Überlegungen vorgestellt, wie solche Abhängigkeiten behandelt werden können. Ein Ansatz beruht auf der Idee, Methoden zur Repräsentation funktionaler Abhängigkeiten in den Konkreten Bereich zu verlagern.

6 Zusammenfassung und Ausblick

Bisherige Systeme zur hybriden Wissenrepräsenation und –verarbeitung (z.B. [CdPV89, CGS90, BHHM93]) lassen eine klare Semantik und eine effiziente Integration vermissen.

Wir haben einen Ansatz vorgestellt, der auf der Basis des Höhfeld/Smolka–Schemas terminologische Logiken, algebraische und finite Constraints mit der logischen Programmierung verbindet. Die Constraintsolver wurden über die Konkreten Bereiche mit dem terminologischen System gekoppelt. Diese Mechanismen wurden in die Hornlogik eingebettet.

Unser Ansatz besitzt folgende Vorteile:

- Integration verschiedener Inferenzmechanismen über klare Schnittstellen
- eine wohl–fundierte Semantik der Methoden und der Schnittstellen
- hohe Deklarativität
- Effizienz, da spezialisierte Algorithmen für die unterschiedlichen Inferenzmechanismen erhalten bleiben.

Allerdings zeigte unser Beispiel auf, das noch weitere Arbeiten notwendig sind. Es stellt sich die Frage, wie Vorgehenswissen dargestellt und integriert werden kann. Hierzu könnte man das sture chronolgische Vorgehen des Hornklausel–Interpreters aufbrechen und Expertenwissen zur Abarbeitung einsetzen. Ein anderer Punkt ist, das terminologische Systeme primär nur strukturelle Abhängigkeiten darstellen können. Eine funktionale Kopplung von Komponenten, wie sie im Bereich der Konfigurierung auftreten, ist direkt nicht möglich. Erste Überlegungen zeigen aber, das eine funktionale Kopplung über spezielle Mechanismen in den Konkreten Bereichen möglich ist. Hier sind allerdings noch grundlegende Arbeiten erforderlich.

7 Danksagungen

Für die Unterstützung möchten wir dem gesamten ARC-TEC Team am DFKI danken. Hier sind insbesondere Andreas Abecker, Philipp Hanschke, Markus Junker und Michael M. Richter hervorzuheben. Diese Arbeit ist im Rahmen des TooCon–Projekts, einer Kooperation zwischen der Daimler–Benz AG und dem DFKI, entstanden.

Literatur

[Abe92] Andreas Abecker. TaxLog: Taxonomische Wissensrepräsentation und logisches Programmieren. In S. Hölldobler, editor, *Logische Programmierung, Beiträge zum 8. Workshop Logische Programmierung, Darmstadt.* AIDA-Report AIDA-92-10, 1992.

[BH91] F. Baader and P. Hanschke. A scheme for Integrating Concrete Domains into Concept Languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 1991.

[BHH+92] Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer, and Michael M. Richter. *VEGA – Knowledge Validation and Exploration by Global Analysis* . Project Proposal, DFKI Kaiserslautern, October 1992.

[BHHM93] H. Boley, P. Hanschke, K. Hinkelmann, and M. Meyer. COLAB: A Hybrid Knowledge Representation and Compilation Laboratory. Research Report RR-93-08, DFKI, Postfach 2080, 67608 Kaiserslautern, January 1993.

[CdPV89] Thomas Christaller, Franco di Primio, and Angi Voss, editors. *Die KI–Werkbank BABYLON: eine offene und portable Entwicklungsumgebung für Expertensysteme*. Künstliche Intelligenz. Addison–Wesley (Deutschland) GmbH, 1989.

[CGS90] R. Cunis, A. Günter, and H. Strecker, editors. *Das PLAKON–Buch*, volume 266 of *Subserie Künstliche Intelligenz*. Springer–Verlag, Berlin und Heidelberg, Germany, 1990.

[Hei93] Michael Heinrich. Ressourcenorientiertes Konfigurieren. *KI*, (1/93):11–16, 1993.

[HN90] B. Hollunder and W. Nutt. Subsumption Algorithms for Concept Languages. Research Report RR-90-04, DFIKI (German Research Center for Artifical Intelligence), Postfach 2080, 67608 Kaiserslautern, Germany, 1990.

[HS88] Markus Höhfeld and Gert Smolka. Definite Relations over Constraint Languages. Lilog-Report 53, IBM Deutschland, October 1988.

[JL86] J. Jaffar and J.-L. Lassez. Constraint Logic Programming. Technical report, Monash University, Department of Computer Science, June 1986.

[JL87] Joxan Jaffar and Jean-Louis Lassez. Constraint Logic Programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages (POPL), Munich, Germany*, pages 111–119. ACM, January 1987.

[JMSY92] Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, and Roland H. C. Yap. The CLP($\mathcal{R}$) Language and System. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 14(3):339–395, July 1992.

[Mey92] Manfred Meyer. Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment. In *Fifth International Symposium on Artificial Intelligence (ISAI), Cancun, Mexico*, pages 167–177. AAAI Press, December 1992.

[Wac93] H. Wache. Integration von ressourcen-basierten Ansätzen im Werkzeugkasten von TOOCON. internes Papier, June 1993.