Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-93-01

Terminological Reasoning with Constraint Handling Rules

Philipp Hanschke, Thom Frühwirth

February 1993

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Friedrich J. Wendl
Director

Terminological Reasoning with Constraint Handling Rules

Philipp Hanschke, Thom Frühwirth

DFKI-D-93-01

This paper describes some ideas carried out together with Thom Frühwirth (ECRC, Munich) and has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4).

Terminological Reasoning with Constraint Handling Rules

Thom Frühwirth
ECRC, Arabellastrasse 17, D-W-8000 Munich 81, Germany
thom@ecrc.de

Philipp Hanschke
DFKI, Postfach 2080, D-W-6750 Kaiserslautern, Germany
hanschke@dfki.uni-kl.de

February 4, 1993

Abstract

Constraint handling rules (CHRs) are a flexible means to implement 'user-defined' constraints on top of existing host languages (like Prolog and Lisp). Recently, M. Schmidt-Schauß and G. Smolka proposed a new methodology for constructing sound and complete inference algorithms for terminological knowledge representation formalisms in the tradition of KL-ONE. We propose CHRs as a flexible implementation language for the consistency test of assertions, which is the basis for all terminological reasoning services.

The implementation results in a natural combination of three layers: (*i*) a constraint layer that reasons in well- understood domains such as rationals or finite domains, (*ii*) a terminological layer providing a tailored, validated vocabulary on which (*iii*) the application layer can rely. The flexibility of the approach will be illustrated by extending the formalism, its implementation and an application example (solving configuration problems) with attributes, a new quantifier and concrete domains.

Contents

1 Introduction 3

2 Constraint Logic Programming with Constraint Handling Rules 4

3 Terminological Reasoning with Concrete Domains 6

4 Conclusions 10

1 Introduction

Constraint logic programming (CLP) [JaLa87, Sar89, Coh90, VH91] combines the advantages of logic programming and constraint solving. In logic programming, problems are stated in a declarative way using rules to define relations (predicates). Problems are solved by the built-in *logic programming engine* (LPE) using backtrack search. In constraint solving, efficient special-purpose algorithms are used to solve problems involving distinguished relations referred to as constraints. Constraint solving is usually 'hard-wired' in a built-in *constraint solver* (CS). While efficient, this approach makes it hard to extend or specialize a given CS, combine it with other CS's or build a CS over a new domain.

Constraint handling rules (CHRs) [Fru92] are a language *extension* providing the user (application-programmer) with a declarative and flexible means to introduce *user-defined* constraints (in addition to *built-in* constraints of the underlying host language). In this paper the host language is Prolog, a CLP language with equality over Herbrand terms as built-in constraint. CHRs define *simplification* of and *propagation* over user-defined constraints. Simplification replaces constraints by simpler constraints while preserving logical equivalence (e.g. `X>Y,Y>X <=> false`). Propagation adds new constraints which are logically redundant but may cause further simplification (e.g. `X>Y,Y>Z ==> X>Z`). When repeatedly applied by a CHR engine (CHRE) the constraints may become solved as in a CS (e.g. `A>B,B>C,C>A` is `false`).

CHIP was the first CLP language to introduce some constructs (demons, forward rules, conditionals) [D*88] for user-defined constraint *handling* (solving, simplification, propagation). These various constructs have been generalized into CHRs. CHRs are based on guarded rules, as can be found in concurrent logic programming languages [Sha89], in the Swedish branch of the Andorra family [HaJa90], Saraswats cc-framework of concurrent constraint programming [Sar89], and - with similar motivation as ours - in the 'Guarded Rules' of [Smo91]. However all these languages (except CHIP) lack features essential to define non-trivial constraint handling, namely handling conjunctions of constraints and defining constraint propagation. CHRs provide these two features by multiple heads and propagation rules.

Terminological formalisms based on KL-ONE [BS85] are used to represent the terminological knowledge of a particular problem domain on an abstract logical level. To describe this kind of knowledge, one starts with atomic concepts and roles, and defines new concepts using the operations provided by the language.

```
simple-device isa device and some connector is interface.
```

These intensionally defined concepts can be considered as unary predicates, and roles as binary predicates over individuals. The limited expressiveness of terminological formalisms allows for a number of interesting reasoning services like consistency of assertions and classification of concepts.

The key idea of [ScSm91] for constructing such inference algorithms is to reduce all inference services to a consistency test which can be regarded as a tuned tableaux calculus. We propose CHRs as a flexible implementation layer for this consistency test. These CHRs directly reflect the rules of the tableaux calculus.

In [BaHa91, Han92] we have shown how a terminological formalism can be parametrized by a *concrete domain*, e.g. constraints over rational numbers. This and other extensions carry over to the implementation with CHRs in a straightforward way. Concrete domains can be either also implemented by CHRs or provided as built-in constraints of the host language. In this way we obtain a fairly natural combination of three knowledge representation layers on a common implementational basis.

2 Constraint Logic Programming with Constraint Handling Rules

Syntax. A $\mathrm{CLP}_{\mathsf{CHRs}}$ program is a finite set of clauses from the CLP language and from the language of CHRs. Atoms and terms are defined as usual. There are two classes of distinguished atoms, built-in constraints and user-defined constraints.

A *CLP clause* is of the form

$$H \texttt{:-}\ B_1, \ldots B_n. \quad (n \geq 0)$$

where the head H is an atom but not a built-in constraint, the body $B_1, \ldots B_n$ is a conjunction of atoms called *goals*. There are two kinds of CHRs.

A *simplification* CHR is of the form

$$H_1, \ldots H_i \texttt{ <=> } G_1, \ldots G_j \mid B_1, \ldots B_k,$$

An *propagation* CHRs is of the form

$$H_1, \ldots H_i \texttt{ ==> } G_1, \ldots G_j \mid B_1, \ldots B_k, \quad (i > 0, j \geq 0, k \geq 0)$$

where the multi-head $H_1, \ldots H_i$ is a conjunction of user-defined constraints and the guard $G_1, \ldots G_j$ is a conjunction of atoms which neither are, nor depend on, user-defined constraints.

Semantics. Declaratively, CLP languages are interpreted as formulas in first order logic. A $\mathrm{CLP}_{\mathsf{CHRs}}$ program P is a conjunction of universally quantified clauses.

A CLP clause is an implication

$$H \rightarrow B_1 \wedge \ldots B_n.$$

A simplification CHR is a logical equivalence provided the guard is true

$$(G_1 \wedge \ldots G_j) \rightarrow (H_1 \wedge \ldots H_i \leftrightarrow B_1 \wedge \ldots B_k).$$

A propagation CHR is an implication provided the guard is true

$$(G_1 \land \ldots G_j) \rightarrow (H_1 \land \ldots H_i \rightarrow B_1 \land \ldots B_k).$$

The operational semantics of $\mathrm{CLP}_{\mathrm{CHR}_s}$ can be described by a transition system. In the following we do not distinguish between sets and conjunctions of atoms. A *constraint store* represents a set of constraints. Let UC and BC be two constraint stores for user-defined and built-in constraints respectively. Let GL be a set of goals. A *computation state* is a tuple $< GL, UC, BC >$. The *initial state* consists of a query GL and empty constraint stores, $< GL, \{\}, \{\} >$. A *final state* is either *successfull* (no goals left to solve), $< \{\}, UC, BC >$, or *failed* (due to an inconsistent constraint store), $< GL, \texttt{false}, BC >$ or $< GL, UC, \texttt{false} >$. The union of the constraint stores in a final state is called *conditional answer* for the query GL, written $answer(GL)$. The following *computation steps* are possible to get from one computation state to the next

Solve - Built-In CS

$< \{C\} \cup GL, UC, BC > \;\longmapsto\; < GL, UC, BC' >$

if $(C \land BC) \leftrightarrow BC'$

Simplify - CHRE with simplification CHRs

$< H' \cup GL, H'' \cup UC, BC > \;\longmapsto\; < GL \cup B, UC, BC >$

if $(H{<=>}\; G \mid B) \in P$, $(BC \rightarrow H = (H' \cup H'') \land answer(G))$

Propagate - CHRE with propagation CHRs

$< H' \cup GL, H'' \cup UC, BC > \;\longmapsto\; < GL \cup B, H' \cup H'' \cup UC, BC >$

if $(H{==>}\; G \mid B) \in P$, $(BC \rightarrow H = (H' \cup H'') \land answer(G))$

Nondeterministic Unfold - LPE with CLP clause

$< \{H'\} \cup GL, UC, BC > \;\longmapsto\; < GL \cup B, UC, \{H = H'\} \cup BC >$

if $(H{:-}\; B) \in P$

$< GL, \{H'\} \cup UC, BC > \;\longmapsto\; < GL \cup B, UC, \{H = H'\} \cup BC >$

if $(H{:-}\; B) \in P$

Implementation. An interpreter for CHRs has been implemented on top of ECRC's Eclipse Prolog utilizing its delay-mechanism and built-in meta-predicates to create, inspect and manipulate delayed goals. In such a sequential implementation, the transitions are tried in the above textual order. We wrote real-life constraint handlers for booleans, finite domains (a la CHIP), temporal reasoning (quantitative and qualitative constraints over points and intervals) and real closed fields (a la CLP(R)). Typically it took only a few days to produce a prototype, since one can directly express how constraints simplify and propagate without worrying about implementation details. If inefficient, once the handler has been tested and 'tuned' as required, it can be safely reworked in a low-level language.

3 Terminological Reasoning with Concrete Domains

In this section we will recall the concept language $\mathcal{ALC}$ and show its implementation in CHRs. We conclude with some extensions of this terminological logic (TL) showing the flexibility of the CHRs approach.

Terminology. A terminology (T-box) consists of a finite set of *concept definitions* "C `isa` s" where C is the newly introduced concept name and s is a concept term constructed from concept names and roles. Inductively a *concept term* is defined as follows:

1. Every concept name C is a concept term.
2. If s and t are concept terms and R is a role name then the following expressions are concept terms, too:
   s `and` t (conjunction), s `or` t (disjunction), `nota` s (complement),
   `every` R `is` s (value restriction), `some` R `is` s (exists-in restriction)

An interpretation $\mathcal{I}$ with a set $\mathcal{D}_\mathcal{I}$ as domain interprets a concept name C as a set $C^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I}$ and a role name R as a set $R^\mathcal{I} \subseteq \mathcal{D}_\mathcal{I} \times \mathcal{D}_\mathcal{I}$. It can be lifted to concept terms in a straight forward manner: conjunction, disjunction, and complement are interpreted as set intersection, set union, and set complement w.r.t. $\mathcal{D}_\mathcal{I}$, respectively, and
$a \in (\texttt{every } R \texttt{ is } s)^\mathcal{I}$ iff, for all $b \in \mathcal{D}_\mathcal{I}$, $(a, b) \in R$ implies $b \in s^\mathcal{I}$, and
$a \in (\texttt{some } R \texttt{ is } s)^\mathcal{I}$ iff, there is some $b \in \mathcal{D}_\mathcal{I}$ such that $(a, b) \in R^\mathcal{I}$, $b \in s^\mathcal{I}$. An interpretation is a *model* of a terminology T if $C^\mathcal{I} = s^\mathcal{I}$ for all "C `isa` s" $\in T$.

Example: The domain of a configuration application comprises at least devices, interfaces, and configurations. The following concept definitions express that these are disjoint sets.[1]

```
primitive(device).
interface isa nota device.
configuration isa nota (interface or device).
```

Let's assume that a simple device has at least one interface.

```
role(connector).
simple-device isa device and some connector is interface.
```
□

---

[1] It is convenient to introduce a short cut for this kind of definitions: `open-family(entities, [device, interface, configuration])`.

Assertional formalism. *Objects* are (Herbrand) constants or variables. Let a, b be objects, R a role, and C a concept term. Then $b : C$ is a *membership assertion* and $(a, b) : R$ *is a role-filler assertion. An A-box is a collection of membership* and role-filler assertions.

Example (contd): So we can introduce instances of devices and interfaces.

```
dev2:device, inter1:interface, (dev1,inter1):connector.
```

□

Reasoning services. An A-box A is *consistent (w.r.t. the terminology)* if there is a model $\mathcal{I}$ and a variable assignment σ : *objects* $\rightarrow \mathcal{D}_{\mathcal{I}}$ such that all assertions of A are satisfied, i.e., $(a\sigma^{\mathcal{I}}, b\sigma^{\mathcal{I}}) \in R^{\mathcal{I}}$ and $b\sigma^{\mathcal{I}} \in C^{\mathcal{I}}$, for all $(a, b) : R$ and $b : C$ in A. An object a is a member of a concept C iff in all models $\mathcal{I}$ of the terminology that satisfy the A-box by an assignment σ we have $a\sigma^{\mathcal{I}} \in C^{\mathcal{I}}$. A concept B *subsumes* a concept C if for all models $\mathcal{I}$ of the terminology $B^{\mathcal{I}} \supseteq C^{\mathcal{I}}$.

Note that subsumption (and similarly membership) queries can be reduced to the inconsistency problem of A-boxes: a concept A subsumes a concept B iff it is inconsistent to assume an object a that is a member of "B `and` `nota` A".

CLP$_{\mathsf{CHR}_s}$(TL). Roughly, the consistency test of A-boxes works as follows.

1. Use transformation rules to propagate the assertions in the A-box to make the knowledge more explicit.
2. Look for obvious contradictions (clashes) such as "`a:B, a:nota B`".

The transformation rules of the first step as well as the search for the obvious contradictions can be directly mapped to CHRs by regarding assertions as user-defined constraints (see Appendix). Nondeterministic transformations (due to disjunctions in the concept language and resulting from negation) are mapped into CLP clauses.

Extensions. In a number of papers the above technique has been applied successfully to variants of terminological logics (e.g., [HNS90, Hol90]). This flexibility carries over to extensions of our implementation.

Roles are interpreted as an arbitrary binary relation over $\mathcal{D}_{\mathcal{I}}$. *Attributes* (also called features) are functional roles, i.e., their interpretation is the graph of a partial function. Assuming declarations of attributes of the form `attribute(`F`)`, F a concept name, we just have to extend our implementation by

```
(I, J1):F, (I, J2):F <=>  attribute(F) | J1=J2,  (I, J1):F.
```

Example (contd): Now we are ready to define a simple configuration which consists of two distinguished devices.

```
attribute(component-1).
attribute(component-2).
simple-config isa
        configuration and
        some component-1 is simple-device and
        some component-2 is simple-device.
```

Extending the above A-box by

```
config1:simple-config,
(config1,dev1):component-1, (config1,dev2):component-2
```

the membership service can derive that `dev1` and `dev2` have connectors that are interfaces and are thus simple devices. □

A more local way to specify functionality of roles is provided through concept terms of the form "`exactly one` R", R a role name. An $a \in \mathcal{D}_{\mathcal{I}}$ is an element of `(exactly one` $R)^{\mathcal{I}}$ if there is exactly one R-role filler for a. This is implemented through

```
I:exactly one R, (I, J1):R, (I, J2):R   <=> role(R) |
                                             J1=J2,   (I, J1):R.
```

We have also to add a way of propagating the complement operator:

```
X:nota exactly one R :- X:every R is (S and nota S).
X:nota exactly one R :- (X,Y):R, (X,Z):R, Y≠Z.
```

The former says "there is no filler", the latter says "there are at least two fillers".

Example (contd):
`very-simple-device isa simple-device and exactly-one connector.` □

Concrete domains. In [Han92] restricted forms of quantification over predicates of a *concrete domain* D have been suggested as concept forming operators. Examples of concrete domains are Allen's temporal interval relations, rational (natural) numbers with comparison operators and real-closed fields (all of which have been implemented by CHRs). An *admissible* concrete domain has to be closed under complement (since we have to propagate the complement operator) and has to provide a satisfiability test for conjunctions of predicates. The syntax for the extension TL(D) of the concept language is as follows:

`every` w_0 `and` ...`and` w_n `is p`
`some` w_0 `and` ...`and` w_n `is p`

Where w_i is of the form "R_0 `of` ... `of` R_{k_i}", R_j are role/attribute names, $n > 0$, $k_i \geq 0$, $i = 1, \cdots, n$, and p is an n-ary concrete predicate (constraint) of D. For readability, we may use also infix notation for the predicates. These constructs are inspired by the value restriction and the exists-in restriction. Reading the expressions as natural language sentences should provide a good intuition about there semantics. See [Han92] for details.

Example (contd): Now we can associate price and voltage with a device and require that in an electrical configuration the voltages have to be compatible.

```
attribute(price).
attribute(voltage).
electrical-device isa very-simple-device and
      some voltage > 0 and some price > 1 .
low-cost-device isa electrical-device and every price < 200.
high-voltage-device isa electrical device and
      every voltage > 15.
electrical-configuration isa simple-configuration and
      every component-1 is electrical-device and
      every component-2 is electrical-device and
      every voltage of component-1 = voltage of component-2.
```

□

$\mathbf{CLP}_{\mathsf{CHR}_s}\mathbf{(TL(D))}$. The A-box of this extended concept formalism may also contain assertions of the form $p(a_1, \cdots, a_n)$. If we apply the CLP scheme of Höhfeld und Smolka [HS90] in a straight forward manner to these 'A-boxes', we obtain a CLP language with the three mentioned representation and reasoning layers.

Example (contd): The following CLP clauses specify the catalog of devices and describe possible configurations that are based on this catalog.

```
catalog(dev1) :- dev1:electrical-device,
      (dev1,10):voltage, (dev1,100):price.
catalog(dev2) :- dev2:electrical-device,
      (dev2,20):voltage, (dev2,1000):price.
possible-config(C) :-
      C:electrical-configuration,
      catalog(D1), (C,D1):component-1,
      catalog(D2), (C,D2):component-2.
```

The following queries enumerate possible configurations satisfying the requirements.

```
:-possible-config(C).
:-possible-config(C),
       (C,D1):component-1, D1:low-cost-device,
       (C,D2):component-2, D2:high-voltage-device.
```

The first query lists the configurations '(dev1,dev1)' and '(dev2,dev2)' whereas the second has no solution. ▭

4 Conclusions

Constraint handling rules (CHRs) are a language extension for implementing user-defined constraints. Rapid prototyping of novel applications for constraint techniques is encouraged by the high level of abstraction and declarative nature of CHRs.

In this paper we investigated the terminological reasoning formalism. Flexibility was illustrated by extending the formalism and its implementation with attributes, a special quantifier and concrete domains. Applicability was illustrated by sketching a generic terminology for solving configuration problems.

References

[BaHa91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In Proceedings of the 12^{th} International Joint Conference on Artificial Intelligence, 1991.

[BS85] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. Cognitive Science, 9(2):171–216, 1985.

[Coh90] J. Cohen, Constraint Logic Programming Languages, Communications of the ACM 33(7):52-68, July 1990.

[D*88] M. Dincbas et al., The Constraint Logic Programming Language CHIP, Fifth Generation Computer Systems, Tokyo, Japan, December 1988.

[Fru92] T. Frühwirth, Constraint Simplification Rules (former name for CHRs), Technical Report ECRC-92-18, ECRC Munich, Germany, July 1992 (revised version of internal Report ECRC-91-18i, October 1991).

[HaJa90] S. Haridi and S. Janson, Kernel Andorra Prolog and its Computation Model, Seventh Int Conference on Logic Programming, MIT Press 1990, pp. 31-46.

[Han92] P. Hanschke. Specifying role interaction in concept languages. In Third International Conference on Principles of Knowledge Representation and Reasoning (KR '92), October 1992.

[Hol90] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In 14th German Workshop on Artificial Intelligence (GWAI-90), volume 251, pages 38–47. Springer, 1990.

[HNS90] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In 9th European Conference on Artificial Intelligence (ECAI'90), pages 348–353. Pitman Publishing, 1990.

[HS90] M. Höhfeld and G. Smolka, Definite Relations over Constraint Languages. LILOG Report 53, IBM Deutschland, West Germany, October 1988.

[JaLa87] J. Jaffar and J.-L. Lassez, Constraint Logic Programming, ACM 14th POPL 87, Munich, Germany, January 1987, pp. 111-119.

[Sar89] V. A. Saraswat, Concurrent Constraint Programming Languages, Ph.D. Dissertation, Carnegie Mellon Univ., also TR CMU-CS-89-108, 1989.

[ScSm91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. In Journal of Artificial Intelligence, 47, 1991.

[Sha89] E. Shapiro, The Family of Concurrent Logic Programming Languages, ACM Computing Surveys, 21(3):413-510, September 1989.

[Smo91] G. Smolka, Residuation and Guarded Rules for Constraint Logic Programming, Digital Equipment Paris Research Laboratory Research Report, France, June 1991.

[VH91] P. van Hentenryck, Constraint Logic Programming, The Knowledge Engineering Review, Vol 6:3, 1991, pp 151-194.

Appendix - Basic Implementation

```
% Terminological Reasoning System with Constraint Handling Rules
% Hanschke and Fruehwirth 1992

% primitive clash
  I:nota S,I:S <=> false.

% negation
```

```
  I:nota (S or T) <=> I:(nota S and nota T).
  I:nota (S and T) <=> I:(nota S or nota T).
  I:nota nota S <=> I:S.
  I:nota every R is S <=> I:some R is nota S.
  I:nota some R is S <=> I:every R is nota S.

% conjunction
  I:S and T <=> I:S,I:T.

% quantifiers and attributes
  I:some R is S <=> role R | (I,J):R,J:S.
  I:every R is S, (I,J):R ==> role R | J:S.
  I:exactly_one R,(I,J):R,(I,K):R <=> role R | J=K, (I,J):R.
  (I,J):A,(I,K):A <=> attribute A | J=K, (I,J):A.

% concept unfolding
  I:C <=> (C isa S) | I:S.
  I:nota C <=> (C isa S) | I:nota S.

% CLP clauses expressing choices
% disjunction
  I:S or T :- I:S.
  I:S or T :- I:T.

% negation of exactly_one
  I:nota exactly_one R :- I:every R is (S and nota S).  % no R
  I:nota exactly_one R :- J=\=K, (I,J):R,(I,K):R. % two or more R
```

For space and presentation reasons we omit concrete domains in this abstract, our actual implementation will be presented in the full paper.

**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-14
Intelligent User Support in Graphical User Interfaces:
1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-16
Jochen Heinsohn, Daniel Kudenko, Berhard Nebel, Hans-Jürgen Profitlich: An Empirical Analysis of Terminological Representation Systems
38 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka: A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck: PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-23
Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohui Wu, Ansgar Bernardi, Christoph Klauck:
Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

RR-92-30
Rolf Backofen, Gert Smolka
A Complete and Recursive Feature Theory
32 pages

RR-92-31
Wolfgang Wahlster
Automatic Design of Multimodal Presentations
17 pages

RR-92-33
Franz Baader: Unification Theory
22 pages

RR-92-34
Philipp Hanschke: Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to Θ-Subsumption Based on Terminological Reasoning
9 pages

RR-92-40
Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 pages

RR-92-41
Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42
John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-47
Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 pages

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49
Christoph Klauck, Ralf Legleitner, Ansgar Bern
Heuristic Classification for Automated CAPP
15 pages

RR-92-50
Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55
John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-56
Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58
Franz Baader, Bernhard Hollunder:
How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59
Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
14 pages

RR-92-60
Karl Schlechta: Defaults, Preorder Semantics and Circumscription
18 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist:
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi:
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-04
Christoph Klauck, Johannes Schwagereit:
GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-05
Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer: COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-09
Philipp Hanschke, Jörg Würtz:
Satisfiability of the Smallest Binary Program
8 Seiten

DFKI Technical Memos

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann: Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Busemann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh:
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

DFKI Documents

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.):
Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.):
DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14
Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann:
Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

D-93-01
Philipp Hanschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

Terminological Reasoning with Constraint Handling Rules

Philipp Hanschke, Thom Frühwirth

D-93-01
Document