Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-40

Combining Terminological and Rule-based Reasoning for Abstraction Processes

Philipp Hanschke
Knut Hinkelmann

November 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

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

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
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, and Siemens. 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.

Dr. Dr. D. Ruland
Director

Combining Terminological and Rule-based Reasoning for Abstraction Processes

Philipp Hanschke, Knut Hinkelmann

DFKI-RR-92-40

Appears also in:

H.-J. Ohlbach (Hrsg.):
Proceedings German Workshop on Artificial Intelligence, GWAI-92
Springer-Verlag, 1992

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4).

Combining Terminological and Rule-based Reasoning for Abstraction Processes*

Philipp Hanschke and Knut Hinkelmann
DFKI, Postfach 2080
W-6750 Kaiserslautern, Germany
{hanschke,hinkelma}@dfki.uni-kl.de

Abstract

Terminological reasoning systems directly support the abstraction mechanisms generalization and classification. But they do not bother about aggregation and have some problems with reasoning demands such as concrete domains, sequences of finite but unbounded size and derived attributes. The paper demonstrates the relevance of these issues in an analysis of a mechanical engineering application and suggests an integration of a forward-chaining rule system with a terminological logic as a solution to these problems.

---

*Supported by BMFT Research Project ARC-TEC (grant ITW 8902 C4).

Contents

1 Introduction

In [Clancey, 1985] *heuristic classification* has been identified as a widespread problem-solving method underlying various expert systems. Heuristic classification is comprised of three main phases (cf. Fig. 1): *Abstraction* from a concrete, particular problem description to a problem class, *heuristic match* of a principal solution (method) to the problem class, and *refinement* of the principal solution to a concrete solution for the concrete problem.

In this paper we suggest a hybrid, declarative formalism for the abstraction phase. It is commonly agreed that *generalization* of concepts (*is-a* relation or subsumption), *classification* of entities (*instance-of* relation), and *aggregation* of objects (*part-of* relation) are the most important abstraction operations (see for example [Borgida *et al.*, 1984], and [Nixon *et al.*, 1989]). The presented formalism supplements the generalization, specialization and classification services of terminological knowledge representation languages by dealing with aggregation as an abstraction operation explicitly. This is achieved by a tight coupling of terminological and rule inferences.

As described in [Bernardi *et al.*, 1991] production planning can be determined as an instance of the inference structure of heuristic classification (Fig. 1). The objective of production planning is to derive a working plan describing how a given workpiece can be manufactured. The input to a production planning system is a very 'elementary' description of a workpiece as it comes from a CAD system. Geometrical descriptions of the workpiece's surfaces, topological neighbourhood relations, and technological data are the central parts of this product model representation. If possible at all, production planning with these input data starting from (nearly) first principles would require very complex algorithms. Thus, planning strategies on such a detailed level are neither available nor do they make sense. Instead, human planners have a library of *skeletal plans* in their minds [Schmalhofer *et al.*, 1991]. Each of these plans is accessed via a more or less abstract description of a characteristic (part of a) workpiece, which is called a *workpiece feature* [Klauck *et al.*, 1991]. A feature thus associates a workpiece model with corresponding manufacturing methods. Therefore, the first step in produc-

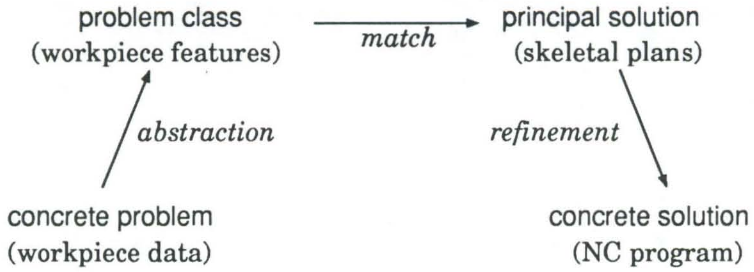

Figure 1: Heuristic Classification Inference Structure

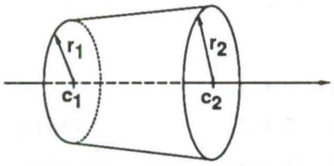

Figure 2: A truncated cone

tion planning is the generation of an abstract feature description from the elementary workpiece data. In the second step, skeletal plans are associated with each of the features before they are merged and instantiated in the third step, ruling out unsuitable combinations.

In the following section we shall define concepts comprising (a part of) the terminological knowledge of our sample application in a terminological formalism and illustrate how terminological systems support generalization, specialization, and classification. Section 3 will then show how aggregation can be managed by incorporating rules, which at the same time supports the solution of some other representation and reasoning problems in terminological systems. Section 4 makes the formalism more precise and discusses the operational semantics of the proposed hybrid system.

2 Abstraction in Terminological Languages

In the T-box formalism of terminological systems concepts are defined intensionally. This is done by the use of concept terms that partially describe entities. The language we are using provides in addition to the usual concept operators concrete domains such as predicates over rational numbers [Baader and Hanschke, 1991]. This is especially useful in our technical domain where we deal with geometric entities. The geometry, as the main ingredient of a CAD drawing, is given in our application as a collection of rotational-symmetric surfaces that are fixed to the symmetry axis of the lathe work. An important geometric element is the truncated cone. Since the surfaces are fixed to an axis, they can be characterized by four rational numbers r_1, r_2, c_1, and c_2 (Fig. 2). But not all quadruples represent a truncated cone. So we have to restrict their values such that the radii are positive and the quadruples do not correspond to a line, a circle, or even a point. These restrictions are expressed by the four place predicate truncone-condition over the concrete domain of rational numbers

$$\mathsf{truncone} \quad = \quad \mathsf{truncone\text{-}condition}(\mathsf{r}_1, \mathsf{r}_2, \mathsf{c}_1, \mathsf{c}_2).$$

This definition can be specialized to a cylinder by further restricting the radii as being equal using equality on rational numbers and the *conjunction* operator ⊓. Similarly, the definitions of ascending and descending truncated cones, rings, etc. can be obtained by specialization.

$$\begin{array}{lcllcl}
\mathsf{cylinder} & = & \mathsf{truncone} \sqcap (\mathsf{r}_1 = \mathsf{r}_2), & \mathsf{ring} & = & \mathsf{truncone} \sqcap (\mathsf{c}_1 = \mathsf{c}_2), \\
\mathsf{asc} & = & \mathsf{truncone} \sqcap (\mathsf{r}_1 < \mathsf{r}_2), & \mathsf{ascring} & = & \mathsf{ring} \sqcap \mathsf{asc}, \\
\mathsf{desc} & = & \mathsf{truncone} \sqcap (\mathsf{r}_1 > \mathsf{r}_2), & \mathsf{descring} & = & \mathsf{ring} \sqcap \mathsf{desc}
\end{array}$$

Conversely, the truncone generalizes the concepts obtained through specialization. Generalization can also be expressed using the *disjunction* operator $\sqcup$. For example, truncated cones that are not cylinders can be defined as the most specific generalization of ascending and descending truncated cones: $\mathsf{asc\text{-}desc} = \mathsf{asc} \sqcup \mathsf{desc}$.

Since features comprise in general more than one surface, it is necessary to aggregate the primitive surfaces. For instance, a biconic is comprised of two neighbouring truncated cones.

$$\mathsf{biconic} = \exists \mathsf{left.truncone} \sqcap \exists \mathsf{right.truncone} \sqcap (\mathsf{left}\ \mathsf{c}_2 = \mathsf{right}\ \mathsf{c}_1) \sqcap (\mathsf{left}\ \mathsf{r}_2 = \mathsf{right}\ \mathsf{r}_1)$$

Here the attributes left and right play the role of *part-of* attributes linking a biconic to its components. The semantics of the *exists-in restriction* in $\exists$left.truncone is that an object is a member of this concept iff it has a truncated cone as a filler for left. The expression ($\mathsf{left}\ \mathsf{c}_2 = \mathsf{right}\ \mathsf{c}_1$) forces the right center of the left truncated cone to be equal to the left center of the right truncated cone.

Specializations of biconic are defined using the *value restriction* operator $\forall$. An object belongs to $\forall$left.cylinder if it has no attribute filler or a cylinder as attribute filler for left.

$$\begin{array}{lcl}
\mathsf{ascasc} & = & \mathsf{biconic} \sqcap \forall \mathsf{left.asc} \sqcap \forall \mathsf{right.asc}, \\
\mathsf{hill} & = & \mathsf{biconic} \sqcap \forall \mathsf{left.asc} \sqcap \forall \mathsf{right.desc}, \\
 & & \ldots \\
\mathsf{rshoulder} & = & \mathsf{biconic} \sqcap \forall \mathsf{left.cylinder} \sqcap \forall \mathsf{right.ascring}, \\
\mathsf{lshoulder} & = & \mathsf{biconic} \sqcap \forall \mathsf{right.cylinder} \sqcap \forall \mathsf{left.descring}, \\
\mathsf{shoulder} & = & \mathsf{lshoulder} \sqcup \mathsf{rshoulder}
\end{array}$$

Terminological reasoning systems provide an interesting service called *concept classification* that arranges the introduced concepts in a subsumption graph (Fig. 3).

To represent a particular lathe workpiece in a terminological system, the assertional formalism, called A-box, is employed. It allows to instantiate the concepts with objects and to fill in their attributes. A single truncated cone could for example be represented as:

$$\mathsf{truncone(tc)}, \quad \mathsf{c}_1(\mathsf{tc}) = 0, \quad \mathsf{r}_1(\mathsf{tc}) = 10, \quad \mathsf{c}_2(\mathsf{tc}) = 5, \quad \mathsf{r}_2(\mathsf{tc}) = 10.$$

The *object classification*[1] of the A-box computes the most specific concept(s) an object belongs to. In the example it would detect that tc is a cylinder. Now we consider a second truncated cone neighbouring the first one:

$$\mathsf{truncone(tc')}, \quad \mathsf{c}_1(\mathsf{tc'}) = 5, \quad \mathsf{r}_1(\mathsf{tc'}) = 10, \quad \mathsf{c}_2(\mathsf{tc'}) = 5, \quad \mathsf{r}_2(\mathsf{tc'}) = 15.$$

---

[1] This service is sometimes also called *realization* [Nebel, 1990]

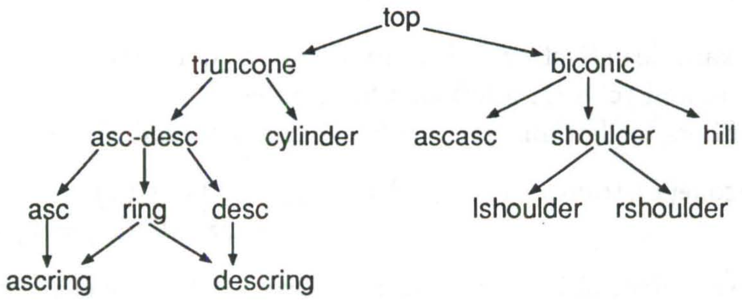

Figure 3: A subsumption graph

The object classification would derive that tc' is an ascending ring. But it **cannot** detect that they both form a 'biconic'—unless tc and tc' are aggregated to a **single** instance. Once there is an object bi with assertions

$$\mathsf{left(bi) = tc}, \quad \mathsf{right(bi) = tc'}$$

bi can be classified as a rshoulder. But this generation of a new instance is not a standard operation in terminological reasoning systems. The selection of instances that are composed to a new object does not depend on terminological knowledge. On the contrary, knowledge about aggregation of instances is part of the assertional box. This can easily be seen in the case that the aggregation is not unique. To illustrate this, let us consider a simple configuration example. Let a terminal be defined as a keyboard connected to a screen. Suppose there are two keyboards k_1 and k_2 and two screens s_1 and s_2. If and how screens and keyboards are put together is not part of the terminological but of the assertional component. So there must be a rule which describes under which particular circumstances (for example because of customer requirements) k_1 and s_2 are connected to form a terminal t_1.

3 Abstraction with Rules

Analyzing the above examples we see that terminological systems directly support generalization, specialization, and classification operations. These operations are based on the subsumption relation between concept definitions. Aggregation, however, is not directly supported by terminological reasoning systems. Instead of enhancing the terminological system with an additional aggregation service it is coupled with a general-purpose rule system, which also allows to overcome other restrictions of terminological reasoning.

3.1 Aggregation

Remember the example in Section 2: it is impossible to derive that the two neighbouring truncated cones **tc** and **tc'** form a left shoulder, unless there is an object **bi** with **tc** and **tc'** as attribute fillers for **left** and **right**. This suggests to add an aggregation rule:

$$\begin{aligned}\forall x, y : [(\mathsf{truncone}(x), \mathsf{truncone}(y), \mathsf{c}_2(x) = \mathsf{c}_1(y), \mathsf{r}_2(x) = \mathsf{r}_1(y))\\ \rightarrow \exists z : (\mathsf{left}(z) = x, \mathsf{right}(z) = y)]\end{aligned}$$

Aggregation rules collect objects or values to form a new object if certain conditions hold for the constituent parts: "If there are two neighbouring truncated cones, then aggregate them using the attributes **left** and **right**". The truncated cones x and y are left and right parts of a new object z, which depends on its constituents. This becomes clear when looking at the skolemized version of the rule (all occurring variables are universally quantified, see Section 4):

$$\begin{aligned}(\mathsf{truncone}(x), \mathsf{truncone}(y), \mathsf{c}_2(x) = \mathsf{c}_1(y), \mathsf{r}_2(x) = \mathsf{r}_1(y))\\ \rightarrow (\mathsf{left}(f(x, y)) = x, \mathsf{right}(f(x, y)) = y)\end{aligned} \tag{1}$$

Note that it is not necessary to repeat the definition of a rule for every concept in the terminology which describes an aggregate. The automatically computed subsumption graph helps the knowledge engineer to find the most general level on which he can formulate a rule. For example, instead of defining aggregation rules for **hill**, **lshoulder**, **rshoulder** etc. separately, it is sufficient to do so only for a **biconic**, the most general composition of two neighbouring truncated cones.

3.2 Derived Attribute and Role Fillers

Now we turn to a further difficulty associated with assertional reasoning in terminological systems. Consider the following concept definitions for regular, tall, and flat shoulders:

$$\begin{aligned}\mathsf{rshoulder\text{-}with\text{-}hw} &= \mathsf{rshoulder} \sqcap (\mathsf{height} = \mathsf{right}\ \mathsf{r}_2 - \mathsf{right}\ \mathsf{r}_1)\\ &\quad \sqcap (\mathsf{width} = \mathsf{left}\ \mathsf{c}_2 - \mathsf{left}\ \mathsf{c}_1)\\ \mathsf{regular\text{-}rshoulder} &= \mathsf{rshoulder\text{-}with\text{-}hw} \sqcap (\mathsf{height} = \mathsf{width})\\ \mathsf{tall\text{-}rshoulder} &= \mathsf{rshoulder\text{-}with\text{-}hw} \sqcap (\mathsf{height} > \mathsf{width})\\ \mathsf{flat\text{-}rshoulder} &= \mathsf{rshoulder\text{-}with\text{-}hw} \sqcap (\mathsf{height} < \mathsf{width})\end{aligned}$$

Note that $(\mathsf{height} = \mathsf{right}\ \mathsf{r}_2 - \mathsf{right}\ \mathsf{r}_1)$ is the application of a three-place predicate to the attribute chainings **height**, **right** r_2, and **right** r_1. The object classification cannot identify the aggregate $f(\mathsf{tc}, \mathsf{tc'})$ of the above example as a regular shoulder. This is only possible if the attribute fillers for **height** and **width** are available. This could be achieved by the following rule:

$$\begin{aligned}\mathsf{rshoulder}(x) \rightarrow\ & (\mathsf{height}(x) = \mathsf{r}_2(\mathsf{right}(x)) - \mathsf{r}_1(\mathsf{right}(x)),\\ & \mathsf{width}(x) = \mathsf{c}_2(\mathsf{left}(x)) - \mathsf{c}_1(\mathsf{left}(x)))\end{aligned}$$

3.3 Varying Size Aspects

Integrating a terminological reasoning system with a rule-based system can also eliminate a third restriction. Because terminological systems provide decision procedures for their reasoning problems, it cannot be avoided that they have a restricted expressiveness. For example, in general it is not possible to deal with concrete domains (e.g. real numbers in the truncated-cone condition of Section 2) and varying size aspects (e.g. sequences) in *one* concept language in a reasonable way, without having an undecidable subsumption problem [Baader and Hanschke, 1992]. Our way out of this problem is to exclude varying size aspects from the terminological formalism and deal with them in the rule language.

Example 3.1 To represent an ascending sequence of neighbouring truncated cones, the varying length of the sequence together with the restrictions on the attributes over rational numbers have to be considered. In the combined formalism we define a sequence of truncated cones as a rule relying on the terminology:

$$\begin{array}{lcl} \mathsf{truncone}(x) & \rightarrow & \mathsf{asc\text{-}list}([x]) \\ \mathsf{truncone}(x), \mathsf{asc\text{-}list}([y|r]), (\mathsf{c}_2(x) = \mathsf{c}_1(y)), (\mathsf{r}_2(x) = \mathsf{r}_1(y)) & \rightarrow & \mathsf{asc\text{-}list}([x, y|r]) \end{array}$$

The predicate asc-list is not a concept treated by the terminological inferences, it is solely defined by these two rules much as it would have been done in Prolog ($[x, y|r]$ denotes a list of two elements x, y, and a rest r). This rule allows us to detect sequences of neighbouring truncated cones in the incoming elementary geometric data. Please note that we do not intend to solve an undecidable problem here. The knowledge engineer has to make sure that the intended inferences involving the rules terminate.□

4 The Rule Formalism

In this section we are going to make the syntax and the semantics of the overall formalism more precise.

4.1 Syntax

An *A-box expression with terms* is a conjunction of expressions, each of which is of one of the following forms:

1. *Membership assertion:* $C(t)$, where C is a concept (possibly not defined in the terminology such as asc-list above) and t is a (Herbrand) term possibly containing variables.

2. *Predicate assertion:* $P(u_1(t_1), \cdots, u_n(t_n))$, where the u_i are possibly empty compositions of attributes, the t_i are terms, and P is an n-ary predicate from the concrete domain.
3. *Role-filler assertion:* $R(s, t)$, where R is a role and s, t are terms.
4. *Attribute-filler assertion:* $F(s) = t$, where F is an attribute or a chaining of attributes, and s, t are terms.
5. *Atom:* $P(t_1, \cdots, t_n)$, where P is a predicate only defined by rules and the t_i are terms.

We shall refer to sets of these assertions as (generalized) A-boxes, too. The expression $G \rightarrow H$ is a *rule* if G and H are A-box expressions with terms. The variables that occur only in H are considered as existentially quantified, whereas all other variables are considered as universally quantified at the rule level. An expression is *ground* iff it does not contain any variable.

4.2 Semantics

The extension of the abstract concept language by a concrete domain is formally presented in [Baader and Hanschke, 1991], and it is shown that if the concrete domain is *admissible*, then there exist sound and complete reasoning algorithms for the reasoning problems of the terminological formalism, in particular, for concept classification and object classification. The model-theoretic semantics of the extended concept language given there induces a mapping ψ from concept definitions and A-box assertions into logical formulas. It maps concepts to unary predicates, roles and attributes to binary predicates, where the latter are restricted to be functional in their first argument, and predicate symbols from the concrete domains to fixed interpretations determined by the concrete domain. This mapping easily extends to the rule formalism if the arrow "$\rightarrow$" is interpreted as logical implication, the "," as logical conjunction, and the quantifiers as logical quantifiers.

Let an A-box expression G and a ground A-box $\mathcal{A}$ be given. Then $\mathcal{A}$ *constructively implies* G by (the substitution) σ iff *(i)* $G\sigma$ is ground, and *(ii)* $G\sigma$ is logically implied by $\mathcal{A}$ and the current terminology. This kind of implication can be effectively tested by a generalized membership test.

For the existentially quantified variables in a head H of a rule $G \rightarrow H$ new objects should be generated. For that purpose we substitute skolem terms $f(x_1, \cdots, x_n)$ for these variables. By choosing a new function symbol per existentially quantified variable, and by using all universally quantified variables of the respective rule as arguments, different ground instantiations of a rule lead to different new objects. We assume without loss of generality that from now on all rules are skolemized and, thus, all variables occurring in the head also occur in the body.

The *operational semantics* of a set of rules $\mathcal{R}$ can now be defined by a monotonic operator T (depending on $\mathcal{R}$) that maps a ground A-box $\mathcal{A}$ to an enlarged A-box

$$\begin{aligned} T(\mathcal{A}) = \mathcal{A} \cup \{h_j\sigma;\ & \text{there is a rule } G \rightarrow H \text{ in } \mathcal{R} \text{ such that} \\ & \mathcal{A} \text{ constructively implies } G \text{ by } \sigma, \\ & H = h_1, \ldots, h_n, \text{ and } 1 \leq j \leq n\}. \end{aligned}$$

Given a ground A-box $\mathcal{A}_0$ we call $T^{\omega}(\mathcal{A}_0) := \bigcup_{i=0,1,2,\ldots} T^i(\mathcal{A}_0)$ the *output.* It is straightforward to show that this semantics is correct with respect to the model-theoretic semantics. I.e., each element of the output is logically implied by $\mathcal{R}$ and $\mathcal{A}$. The situation is more complicated for the converse direction as the following discussion shows.

(1) Hidden objects in the A-box: Consider a rule $C(x) \rightarrow B(x)$ and an A-box $(\exists R.C)(a)$. Here, no substitution σ can be found such that $C(x\sigma)$ is implied. But, according to the semantics of the exists-in restriction there is an (unknown) object, say b, satisfying the premise of the rule. So, the A-box can be transformed into the logically equivalent A-box $R(a,b), C(b)$ and the rule fires.

(2) Case distinctions in the A-box: Consider the A-box $((\exists R.C) \sqcup (\exists S.C))(a)$ and the rule $C(x) \rightarrow B(x)$. Here also an anonymous object satisfying the premise must exist. Unfortunately, whether this object is related to a via R or S depends on the particular interpretation. A similar situation may also occur in the absence of explicit disjunctions as the following example shows: Consider the concept definitions $\mathsf{male} = \neg\mathsf{female}$, and $\mathsf{ma\text{-}of\text{-}boy} = \mathsf{female} \sqcap \exists\mathsf{child.male}$ and an A-box consisting of $\mathsf{female(mary)}$, $\mathsf{male(bob)}$, $\mathsf{child(mary, x)}$, and $\mathsf{child(x, bob)}$.[2]

None of the objects can be classified as being a ma-of-boy. But it is not possible that both x and mary are at the same time not members of ma-of-boy. Hence, it could be logically concluded that there is a ma-of-boy, but no single instance can be found until the sex of x is known. This is one of the main reasons why we have chosen 'constructive implication' to test the premises. If the concept language would contain attribute agreements and disagreements it would even be undecidable whether an A-box $\mathcal{A}$ implies $\exists x C(x)$ [Baader *et al.*, 1991].

(3) Constructive character of rules: A classical implication $A \rightarrow B$ can be inverted: $\neg B \rightarrow \neg A$. These hidden rules are not captured by the operational semantics above. This source of incompleteness can be avoided by careful use of the rules. For example, the aggregation rules, the derived-parameter rules, and the rules for varying size aspects above, contain only 'positive' expressions in the head, and in the abstraction process their negative counterparts do not occur in the A-box.

A fourth source of incompleteness comes from (2) and (3), together. Consider the A-box $(A \sqcup B)(a)$, and the rules $A(x) \rightarrow C(x)$ and $B(x) \rightarrow C(x)$. Logically $C(a)$ holds, but it is not in the output.

---

[2] The idea to this example is due to Maurizio Lenzerini.

4.3 Refining the Operational Semantics

An implementation of the system can refine the above operational semantics in several directions.

4.3.1 Optimize Premise Instantiation

The semi-naive strategy known from deductive databases can be adopted. This strategy fires a rule only if new assertions participate in its instantiation. In the process of instantiating a single rule shared variables lead to partially instantiated expressions which constrain the search for the remaining part of the premise. Finding a good sideway information passing strategy [Beeri and Ramakrishnan, 1991] can optimize rule instantiation. It finds a substitution σ for a rule $G \rightarrow H$ incrementally by ordering G.

Other optimizations rely on the terminological structure of the abstraction knowledge. Each object in the A-box is classified and the resulting information is used to build an index structure. During the instantiation of a rule's premise this structure provides quick retrieval of objects for membership expressions.

Example 4.1 At compile time rule (1) can be rearranged to

$$\begin{aligned}(\mathsf{truncone}(x), \mathsf{c}_2(x) = \mathsf{c}_1(y), \mathsf{r}_2(x) = \mathsf{r}_1(y), \mathsf{truncone}(y))& \\ \rightarrow (\mathsf{left}(f(x,y)) = x, \mathsf{right}(f(x,y)) = y).&\end{aligned} \tag{2}$$

At runtime the index structure is used to find an instance tc of $\mathsf{truncone}(x)$. Sideway information passing yields a 'neighbouring' object tc' such that $\mathsf{c}_2(\mathsf{tc}) = \mathsf{c}_1(\mathsf{tc'})$, $\mathsf{r}_2(\mathsf{tc}) = \mathsf{r}_1(\mathsf{tc'}))$. If, finally, tc' is a truncone, the rule fires and a new instance $f(\mathsf{tc}, \mathsf{tc'})$ is created, which has tc and tc' as attribute fillers for left and right, respectively. The aggregate $f(\mathsf{tc}, \mathsf{tc'})$ will then be classified. An analysis of the premise of the rule reveals that instantiations of $f(x, y)$ will always be a member of a specialization of biconic.

If the truncated cones tc and tc' of Section 2 are used to instantiate the rule then $f(\mathsf{tc}, \mathsf{tc'})$ will be classified as an rshoulder (which is subsumed by biconic), because tc is a cylinder and tc' is an ascending ring (cf. the definition of rshoulder in Section 2). Thus, $f(\mathsf{tc}, \mathsf{tc'})$ can trigger any rule with a membership expression $C(x)$ in its premise where C subsumes rshoulder.

4.3.2 Optimizing the Object Classification

Each time assertions are added to the A-box all "affected" objects—in particular newly generated objects—are (re)classified and the index structure is updated. In general, determining which objects are affected and may have a more specific classification and computing the classification is very expensive [Nebel, 1990]. This section explores the characteristics of an 'abstraction' to reduce the costs of the (re)classifications.

Our goal is to understand how the classification of an object depends on axioms that are added to the A-box by a rule and which subsets of an A-box are necessary to compute a classification.

Our first observation is that the terminological formalism is *directed*: All concepts in $\mathcal{T}$ are inductively defined in terms of the operators

$\neg\_$	*(negation)*
$\_\sqcap\_$	*(conjunction)*
$\_\sqcup\_$	*(disjunction)*
$\forall\_.\_$	*(value restrictions)*
$\exists\_.\_$	*(exists-in restriction)*
$P(\_,\ldots,\_)$	*(concrete domain predicate restriction)*

So, essentially, an object a qualifies as a member of a concept only by belonging to simpler concepts and by properties of its attribute and role fillers. It does not matter whether a is a role or attribute filler by its own, say $R(\mathsf{b},\mathsf{a})$. Only if the R-role fillers of b are constrained, the assertion $R(\mathsf{b},\mathsf{a})$ may influence the classification of a. For instance, the object a has classification $\top$ w.r.t. the A-box $\{\top(\mathsf{a})\}$. It remains the same if we add $R(\mathsf{b},\mathsf{a})$ to the A-box. Whereas w.r.t. $\{(\forall R.Q)(\mathsf{b}),\ \top(\mathsf{a})\}$ its classification will change from $\top$ to Q if we add $R(\mathsf{b},\mathsf{a})$, and Q is defined in the terminology.

In general, adding a predicate assertion with a concrete domain predicate can change a realization, too. For example, let a terminology be given by $C = (f < 5)$, f an attribute, and a definition of $\top$. Then w.r.t. the A-box $\{f(\mathsf{b}) = \mathsf{a}\}$ the object b has classification $\top$ whereas w.r.t. $\{f(\mathsf{b}) = \mathsf{a},\ \mathsf{a} < \mathsf{c},\ \mathsf{c} = 5\}$ it has classification C. But consider the case of an object a that is already constrained to a single element of the concrete domain by an A-box. So, a has become a constant and there is no consistent extension of the A-box that further constrains the interpretation of a.

What do these observations imply for the (re)classifications in an abstraction process? The abstraction process starts with a concrete description. The corresponding A-box $\mathcal{A}$ can be split into A-boxes of the form $\{C(\mathsf{a}),\ R_i(\mathsf{a},\mathsf{b}_i),\ P_i(\mathsf{b}_i);\ i = 1,\ldots,n\}$ containing the membership and role/attribute assertions of a. R_i are attributes and roles and the P_i restrict the b_i to constants.[3] Thus, the classification of an a w.r.t. $\mathcal{A}$ can be reduced to a classification w.r.t one of the above small A-boxes—provided the A-box is consistent.

An *aggregation rule* asserts only axioms $R_i(n,\mathsf{a}_i)$ where the R_i are roles or attributes and the premise of the rule only refers to objects that can be reached from one of the a_i through a directed path of role/attribute assertions in the premise. For these rules it suffices to (re)classify the aggregated object n, if the following side conditions hold:

C1. The A-box remains consistent,

---

[3] To increase readability, in the examples in the previous sections $P_i(\mathsf{b}_i)$ has always been omitted and b_i has directly been written as a constant.

C2. there is no role/attribute assertion $R(\mathsf{c}, n)$, and

C3. there is no membership assertion $C(n)$ constraining the R_i-role fillers of n.

A similar observation can be made for the rules dealing with the varying-size aspect.

A rule for *derived fillers* only asserts axioms of the form $P(u(\mathsf{a}), v_1(\mathsf{a}), \ldots, v_n(\mathsf{a}))$ and, analogue to aggregation rules, the premise of the rule only refers to objects that can be reached from an a in its head through a directed path of role/attribute assertions. The predicates P also have the property that for each tuple $(\mathsf{a}_1, \ldots, \mathsf{a}_n)$ of the concrete domain there is an element b such that $P(\mathsf{b}, \mathsf{a}_1, \ldots, \mathsf{a}_n)$. So, $u(\mathsf{a})$ is the derived filler and only a has to be reclassified, if

C4. the A-box remains consistent,

C5. there is no role/attribute assertion $R(\mathsf{c}, \mathsf{a})$, and

C6. the attribute fillers $v_1(\mathsf{a}), \ldots, v_n(\mathsf{a})$ already exist.

How can the side conditions be ensured? To satisfy (C2) and (C5) we choose a particular strategy of the forward chaining. Note that possible rule applications to the initial A-box satisfy these two side conditions. In successive rule firings the strategy does not fire a rule that adds a new object a with an assertion $R(\mathsf{a}, \mathsf{b})$ if it can fire another rule that just adds new *role* or *attribute fillers* (e.g. for b). This means that the strategy applies derived-filler rules before aggregation rules.

To ensure consistency the initial A-box is checked for consistency. This is sufficient because the aggregation rules do not cause inconsistencies and if only one rule is responsible for one derived attribute or role they do not produce inconsistencies either. The other side conditions are satisfied without additional requirements on the abstraction process.

Note that we have assumed that only rules of the mentioned kinds participate in the abstraction and that the abstraction starts with an A-box of a particular form. Together with the above strategy this enables a further optimization. The (re)classification of an a w.r.t $\mathcal{A}$ can be reduced to a classification w.r.t. $\mathcal{A}_\mathsf{a} \subseteq \mathcal{A}$ where, roughly, $\mathcal{A}_\mathsf{a}$ is the set of assertions that can be reached from a through a directed path of attributes or roles. Formally, $\mathcal{A}_\mathsf{a}$ can be defined as follows: $\mathcal{A}_\mathsf{a}$ is the smallest subset of $\mathcal{A}$ such that for every object b occurring in $\mathcal{A}_\mathsf{a}$

1. if $R(\mathsf{b}, \mathsf{c}) \in \mathcal{A}$, R a role or attribute, then $R(\mathsf{b}, \mathsf{c}) \in \mathcal{A}_\mathsf{a}$,
2. if $C(\mathsf{b}) \in \mathcal{A}$, then $C(\mathsf{b}) \in \mathcal{A}_\mathsf{a}$, and
3. if $P(\cdots, \mathsf{b}, \cdots) \in \mathcal{A}$, then $P(\cdots, \mathsf{b}, \cdots) \in \mathcal{A}_\mathsf{a}$.

If we drop the requirement of the initial A-box that the $P_i(\mathsf{b}_i)$ restrict the b_i to a single constant, the restriction to $\mathcal{A}_\mathsf{a}$ is no longer possible, but still no more (re)classifications are necessary.

Discussion In this subsection we observed that terminological languages as presented here are 'directed' from objects to fillers — from abstract to concrete. For rules that respect this 'directionality' several optimizations concerning the (re)classification are possible. More precisely, the optimizations explore the fact that the rules describe abstraction, i.e., they preserve the existing, more concrete part of a (problem) description and extend it towards a more abstract description without constraining the initial description modulo the 'directionality' of the concept language.

5 Conclusions

The paper has shown how a terminological system can be integrated in a rule formalism for which we have presented an operational semantics based on forward chaining and terminological inferences. It turned out that this hybrid formalism is suitable to deal with aggregation (in addition to generalization and classification), derived attributes, and varying length aspects in the abstraction phase of heuristic classification. The decision procedures for the terminological inferences together with the conceptual simplicity of forward chaining have led to a powerful representation formalism with transparent inferences.

Our approach is based on a decidable concept language. This enables us to automatically compute the generalization hierarchy from intensional concept definitions. This is different from related work in logic programming (LOGIN [Ait-Kaci and Nasr, 1986]) and query languages for databases [Kifer and Lausen, 1989]. In these formalisms the rule inferences take a fixed taxonomy given as a semi lattice of sorts into account. LOGIN's feature logic provides no relational roles and a query is answered by strict top-down, left-to-right reasoning, which is less appropriate for the abstraction application compared to our data-driven approach.

Probably the most closely related work is the query language presented in [Abiteboul and Kanellakis, 1989]. They deal with bottom-up execution of rules, too, and in particular, they also generate new objects (they call it object identities) for variables that occur only in the head of a rule. But there are certain differences: they employ the closed-world assumption, they have no quantification over role and attribute fillers, they deal with a fixed taxonomy, they do not have concrete domains, and they do not identify an decidable subformalism such as our concept formalism.

LOOM [MacGregor, 1988], MESON [Edelmann and Owsnicki, 1986], and CLASSIC [Brachman *et al.*, 1991] are terminological reasoning systems that provide rules of the following restricted form: $C(x) \rightarrow D(x)$, where C and D are concepts, which are not expressive enough to represent aggregation.

A reasoning system along the lines of this paper that also deals with the other phases of heuristic classification has been implemented in Common Lisp and has been tested with the mentioned prototypical production planning application [Boley *et al.*, 1991].

References

[Abiteboul and Kanellakis, 1989] S. Abiteboul and P. C. Kanellakis. Object identity as a query language primitive. In *ACM SIGMOD*, pages 159–173, 1989.

[Ait-Kaci and Nasr, 1986] H. Ait-Kaci and R. Nasr. LOGIN: A logic programming language with built-in inheritance. *JLP*, 3:185–215, 1986.

[Baader and Hanschke, 1991] F. Baader and Ph. Hanschke. A scheme for integrating concrete domains into concept languages. Research Report RR-91-10, DFKI / Kaiserslautern, 1991.

[Baader and Hanschke, 1992] F. Baader and Ph. Hanschke. Extensions of concept languages for a mechanical engineering application. In *GWAI-92*, 1992.

[Baader *et al.*, 1991] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. Research Report RR-91-01, DFKI, 1991.

[Beeri and Ramakrishnan, 1991] C. Beeri and R. Ramakrishnan. On the power of magic. *Journal of Logic Programming*, 10:255–299, 1991.

[Bernardi *et al.*, 1991] A. Bernardi, H. Boley, K. Hinkelmann, Ph. Hanschke, C. Klauck, O. Kühn, R. Legleitner, M. Meyer, M.M. Richter, G. Schmidt, F. Schmalhofer, and W. Sommer. ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge. In *Expert Systems and their Applications: Tools, Techniques and Methods*, Avignon, France, 1991.

[Boley *et al.*, 1991] H. Boley, Ph. Hanschke, K. Hinkelmann, and M. Meyer. COLAB: A hybrid konwoledge compialtion laboratory. submitted for publication, January 1991.

[Borgida *et al.*, 1984] A. Borgida, J. Mylopoulos, and H. K. T. Wong. Generalization/spezialization as a basis for software specification. In *On Conceptual Modelling.* Springer, 1984.

[Brachman *et al.*, 1991] R. Brachman, D. McGuinness, P. Pate-Schneider, and L. Resnick. Living with CLASSIC: When and how to use a KL-ONE-like language. In *Principles of Semantic Networks.* Morgan Kaufmann, 1991.

[Clancey, 1985] J. Clancey, W. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.

[Edelmann and Owsnicki, 1986] J. Edelmann and B. Owsnicki. Data models in knowledge representation systems: A case study. In *GWAI-86 and 2. Österreichische Artificial-Intelligence Tagung*, pages 69–74. Springer, 1986.

[Kifer and Lausen, 1989] M. Kifer and G. Lausen. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD*, pages 134–146, 1989.

[Klauck *et al.*, 1991] Ch. Klauck, R. Legleitner, and A. Bernardi. FEAT-REP: Representing features in CAD/CAM. In *4th International Symposium on Artificial Intelligence: Applications in Informatics*, 1991.

[MacGregor, 1988] R. MacGregor. A deductive pattern matcher. In *AAAI*, pages 403–408, 1988.

[Nebel, 1990] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Springer, 1990.

[Nixon *et al.*, 1989] A. Nixon, B., L. Chung, K., and D. Lauzon. Design of a compiler for a semantic data model. In *Foundations of Knowledge Base Management.* Springer, 1989.

[Schmalhofer *et al.*, 1991] F. Schmalhofer, O. Kühn, and G. Schmidt. Integrated knowledge acquisition from text, previously solved cases, and expert memories. *Applied Artificial Intelligence*, 5:311–337, 1991.

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 publisched 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-91-35
Winfried Graf, Wolfgang Maaß: Constraint-basierte Verarbeitung graphischen Wissens
14 Seiten

RR-92-01
Werner Nutt: Unification in Monoidal Theories is Solving Linear Equations over Semirings
57 pages

RR-92-02
Andreas Dengel, Rainer Bleisinger, Rainer Hoch, Frank Hönes, Frank Fein, Michael Malburg:
Π_{ODA}: The Paper Interface to ODA
53 pages

RR-92-03
Harold Boley:
Extended Logic-plus-Functional Programming
28 pages

RR-92-04
John Nerbonne: Feature-Based Lexicons: An Example and a Comparison to DATR
15 pages

RR-92-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Michael Schulte, Rainer Stark:
Feature based Integration of CAD and CAPP
19 pages

RR-92-06
Achim Schupetea: Main Topics of DAI: A Review
38 pages

RR-92-07
Michael Beetz:
Decision-theoretic Transformational Planning
22 pages

RR-92-08
Gabriele Merziger: Approaches to Abductive Reasoning - An Overview -
46 pages

RR-92-09
Winfried Graf, Markus A. Thies:
Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe
15 Seiten

RR-92-10
M. Bauer: An Interval-based Temporal Logic in a Multivalued Setting
17 pages

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler:
Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger:
Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

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

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

DFKI Technical Memos

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 Bussmann: 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
27pages

DFKI Documents

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

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 Wissensmodel-lierung 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

Combining Terminological and Rule-based Reasoning for Abstraction Processes

Philipp Hanschke, Knut Hinkelmann

RR-92-40

Research Report