Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-36

Extensions of Concept Languages for a Mechanical Engineering Application

Franz Baader, Philipp Hanschke

August 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, Philips, SEMA Group Systems, 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
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

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.

Prof. Dr. Gerhard Barth
Director

Extensions of Concept Languages for a Mechanical Engineering Application

Franz Baader, Philipp Hanschke

DFKI-RR-92-36

This report appears also in:

Proceedings of GWAI'92, Springer, Lecture Notes in Artificial Intelligence

This work has been supported by grants from The Federal Ministry for Research and Technology (FKZ ITW-8902 C4 and FKZ ITW-8903 0)

© Deutsches Forschungszentrum für Künstliche Intelligenz 1992

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

Extensions of Concept Languages for a Mechanical Engineering Application

Franz Baader
baader@dfki.uni-sb.de

Philipp Hanschke
hanschke@dfki.uni-kl.de

German Research Center for AI (DFKI)
Kaiserslautern, Germany

Abstract

We shall consider an application in mechanical engineering, and shall show that the adequate modeling of the terminology of this problem domain in a conventional concept language poses two main representation problems. The first requires access to concrete domains, such as real numbers, while the second asks for a construct which can be used to represent sequences of varying length. As shown in recent papers by the authors there exist extended concept languages—equipped with sound and complete reasoning algorithms—that satisfy the respective representation demands separately.

The main result presented in this paper is that the combination of both extensions leads to undecidable terminological inference problems. In particular, the important subsumption problem is undecidable. It should be noted that the need for these extensions is not particular to the considered problem domain; similar representation demands are likely to occur in other non-toy applications.

Contents

1 Introduction 3

2 Motivation and Problem Domain 4

3 The Concept Language ALCF 5

4 Integrating Concrete Domains 7

5 Adding Transitive Closure 9

6 Combining the Extensions 10

7 How to Live with this Undecidability Result 13

1 Introduction

Concept languages based on KL-ONE [Brachman and Schmolze, 1985] are mostly 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. Concepts can be considered as unary predicates which are interpreted as sets of individuals, and roles as binary predicates which are interpreted as binary relations between individuals. Examples for atomic concepts may be Human and Female, and for roles child. If the logical connective conjunction is present as language construct, one may describe the concept Woman as "humans who are female", and represent it by the expression Human $\sqcap$ Female. Many languages provide quantification over role fillers which allows for example to describe the concept Mother by the expression Woman $\sqcap$ $\exists$child.Human.

KL-ONE was first developed for the purpose of natural language processing [Brachman *et al.*, 1979], and some of the existing systems are still mostly used in this context (see e.g., SB-ONE [Kobsa, 1989]). However, its success in this area has also led to applications in other fields (see e.g., MESON [Edelmann and Owsnicki, 1986] which is used for computer configuration tasks, CLASSIC [Borgida *et al.*, 1989] which is e.g. used for retrieval in software information systems, or K-REP [Mays *et al.*, 1987; Mays *et al.*, 1988] which is used in a financial marketing domain).

In this paper we shall investigate how concept languages can be used in a mechanical engineering domain. More precisely, we shall consider the representation of concepts that are related to lathe workpieces. We shall describe this application in more detail in Section 2. There it will be pointed out that the adequate formalization of these concepts demands two substantial extensions of conventional concept languages. On the one hand, reference to concrete notions such as real numbers is mandatory to represent, for example, the geometric aspects of a lathe workpiece. On the other hand, the abstraction in this domain requires to describe classes of lathes which are sequences of geometric primitives. These sequences have a finite, but varying and not a priori bounded length.

The extensions needed to satisfy these demands have separately been considered in recent papers by the authors. Section 4 summarizes the schematic extension by concrete domains proposed in [Baader and Hanschke, 1991]. An instantiation of this scheme appropriate for representing the geometric aspects in our problem domain is obtained by taking the concrete domain "real numbers." In Section 5 we recall the extension of a conventional concept language by a transitive closure operator as proposed in [Baader, 1991]. This can be used to deal with the varying length aspect. Both papers not only introduce the extended formalisms, but also describe decision procedures for the common terminological inference problems such as subsumption.

In contrast to these positive results we shall show in the present paper (Section 6)

that the subsumption problem in a concept language combining both extensions is undecidable. The paper concludes with some remarks on how this algorithmic problem can be circumvented in a hybrid knowledge representation architecture. As the common base for both extensions the concept language $\mathcal{ALCF}$ is introduced in Section 3. It provides abstract concept-forming operators as used in the introductory examples above.

2 Motivation and Problem Domain

As already mentioned, the domain we want to consider is production planning for CNC lathe machines. More precisely, our work has been motivated by the following application:

> Given the geometry of a rotational-symmetric workpiece, generate abstract NC macros for turning the workpiece on a CNC lathe machine.

Reasoning in this application follows a scheme (Figure 1) that is inspired by William J. Clancey's *heuristic classification*: The input to the system is a CAD drawing describing the workpiece in terms of primitive surfaces and basic technological data. The *abstraction* phase generates a schematic description of the workpiece in terms of *(CAD/CAM) features* [Klauck *et al.*, 1991]. Such features are often associated with parts of the workpiece that are characteristic with respect to how these parts (or the whole lathe) may be manufactured. The second phase associates *skeletal (production) plans* to the features (i.e. to the nodes in the feature DAG). Finally, the third phase *refines* and merges the skeletal plans to a complete numerical control (NC) program.

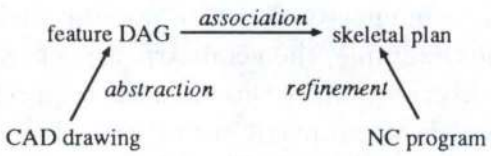

Figure 1: Planning Scheme

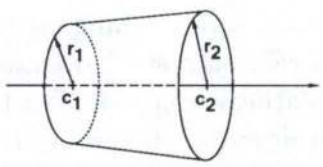

Figure 2: A Truncated Cone

This problem domain requires, among other things, the representation of geometric primitives used in the CAD drawing and of technological data of the workpiece. It also contains the features which characterize the workpiece. If this could be done with a concept language, the abstraction phase could be mapped naturally into a terminological framework:

- Arrange the features represented as concepts in a generalization hierarchy using the subsumption service of the terminological system.
- Represent a particular CAD drawing of the workpiece with its geometric and technological information as instances of appropriate concepts.

- Employ the so-called realization service [Nebel, 1990] to compute the most specific concepts that apply to the particular lathe. The features corresponding to these concepts are then the output of the abstraction phase to which the skeletal plans can be associated.

However it is easy to see that conventional concept languages cannot be used to adequately represent this problem domain. Consider for example the concept of a truncated cone (see Figure 2). Since we consider geometric objects as fixed to an axis, a truncated cone can be characterized by four real numbers, two for its radii and two for the corresponding centers. But of course not all quadruples of rational numbers represent a truncated cone. So we have to restrict the values such that the radii are positive. In addition, one has to exclude cases where the truncated cone degenerates to a line, a circle, or even a point. It seems to be impossible to represented this using only "abstract" concept terms without reference to predicates over, for example, real numbers, which shows the need for an integration of concrete domains.
Another problem is due to the fact that one has to describe classes of lathes which are sequences of geometric primitives. The problem is that these sequences have a finite, but varying and not a priori bounded length. It is quite simple to define concepts for features such as

'1 truncated cone', '2 truncated cones', ...

Assume, for example, that the following two concepts are already defined: Truncone, which stands for the class of truncated cones, and Neighboring, which characterizes when two truncated cones fit together. We shall see later how these concepts could be defined using our first extension. Then the mentioned sequence of concepts can be defined as follows:

$$\begin{aligned} \mathsf{One} &= \exists \mathsf{head}.\mathsf{Truncone} \sqcap \forall \mathsf{tail}.\mathsf{Bottom} \\ \mathsf{Two} &= \exists \mathsf{head}.\mathsf{Truncone} \sqcap \exists \mathsf{tail}.\mathsf{One} \sqcap \mathsf{Neighboring} \\ &\ldots \end{aligned}$$

But it remains the problem to represent the most specific generalization (union) of these *infinitely* many features (concepts). The resulting concept could be termed a 'sequence of neighboring truncated cones'. It should be noted that its specialization 'ascending sequence of truncated cones' (see the definition in Section 6) is essential for characterizing the production classes of lathes. In Section 6 we shall give a formal representation of the kind of sequences exemplified in the figure.

3 The Concept Language ALCF

This section introduces the language $\mathcal{ALCF}$ as a prototypical conventional concept language. It will be the starting point for the two extensions described in

the next two sections.

Definition 3.1 (syntax of $\mathcal{ALCF}$) Concept terms *are built from concept, role, and attribute*[1] *names using just the concept-forming operators*

negation $(\neg C)$, disjunction $(C \sqcup D)$, conjunction $(C \sqcap D)$,
exists-in restriction $(\exists R.C)$, *and* value restriction $(\forall R.C)$.

Here C and D are syntactic variables for concept terms and R is a role or attribute name.
Let A be a concept name and let D be a concept term. Then $A = D$ is a terminological axiom. *A* terminology (T-box) *is a finite set $\mathcal{T}$ of terminological axioms with the additional restrictions that* (i) *no concept name appears more than once as a left hand side of a definition, and* (ii) *$\mathcal{T}$ contains no cyclic definitions.*[2]

Please note that the exists-in and the value restrictions are not only defined for roles but also for attributes. The next definition gives a model-theoretic semantics for the language introduced in Definition 3.1.

Definition 3.2 (semantics of $\mathcal{ALCF}$) *An* interpretation *$\mathcal{I}$ for $\mathcal{ALCF}$ consists of a set $dom(\mathcal{I})$ and an interpretation function. The interpretation function associates with each concept name A a subset $A^{\mathcal{I}}$ of $dom(\mathcal{I})$, with each role name R a binary relation $R^{\mathcal{I}}$ on $dom(\mathcal{I})$, i.e., a subset of $dom(\mathcal{I}) \times dom(\mathcal{I})$, and with each attribute name f a partial function $f^{\mathcal{I}}$ from $dom(\mathcal{I})$ into $dom(\mathcal{I})$.*
For such a partial function $f^{\mathcal{I}}$ the expression $f^{\mathcal{I}}(x) = y$ is sometimes written as $(x, y) \in f^{\mathcal{I}}$. The interpretation function—which gives an interpretation for atomic terms—can be extended to arbitrary concept terms as follows: Let C and D be concept terms and let R be a role or attribute name. Assume that $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ are already defined. Then

1. $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, *and* $(\neg C)^{\mathcal{I}} = dom(\mathcal{I}) \setminus C^{\mathcal{I}}$,
2. $(\forall R.C)^{\mathcal{I}} = \{x \in dom(\mathcal{I});$ *for all y such that $(x, y) \in R^{\mathcal{I}}$ we have $y \in C^{\mathcal{I}}\}$*
   and
   $(\exists R.C)^{\mathcal{I}} = \{x \in dom(\mathcal{I});$ *there exists y such that $(x, y) \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$*

An interpretation $\mathcal{I}$ is a model *of the T-box $\mathcal{T}$ iff it satisfies $A^{\mathcal{I}} = D^{\mathcal{I}}$ for all terminological axioms $A = D$ in $\mathcal{T}$.*

An important service terminological representation systems provide is computing the subsumption hierarchy, i.e., computing the subconcept-superconcept relationships between the concepts of a T-box. This inferential service is usually called classification. The model-theoretic semantics introduced above allows the following formal definition of subsumption.

[1] To avoid confusion with the 'CAD/CAM features' of the application domain we refer to the functional roles as attributes and not as features as, e.g., in [Baader and Hanschke, 1991].

[2] See [Nebel, 1989; Baader, 1990] for a treatment of cyclic definitions in concept languages.

Definition 3.3 (subsumption) *Let $\mathcal{T}$ be a T-box and let C, D be concept terms. Then D* subsumes *C with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.*

In addition to the formalism defined so far, terminological systems usually provide for an assertional component, and corresponding reasoning services [Nebel, 1990]. Because of the space limitations we shall not address this aspect in the present paper.

4 Integrating Concrete Domains

In this section we introduce a formalism that is capable to deal with the first representational problem mentioned in the introductory sections. Before we can define this extended language, we have to formalize the notion "concrete domain" which has until now only been used in an intuitive sense.

Definition 4.1 *A* concrete domain *$\mathcal{D}$ consists of a set $dom(\mathcal{D})$, the domain of $\mathcal{D}$, and a set $pred(\mathcal{D})$, the predicate names of $\mathcal{D}$. Each predicate name P is associated with an arity n, and an n-ary predicate $P^{\mathcal{D}} \subseteq dom(\mathcal{D})^n$.*

An important example for our application is the concrete domain $\mathcal{R}$ of real arithmetic. The domain of $\mathcal{R}$ is the set of all real numbers, and the predicates of $\mathcal{R}$ are given by formulae which are built by first order means (i.e., by using logical connectives and quantifiers) from equalities and inequalities between integer polynomials in several indeterminates.[3] For example, $x + z^2 = y$ is an equality between the polynomials $p(x, z) = x + z^2$ and $q(y) = y$; and $x > y$ is an inequality between very simple polynomials. From these equalities and inequalities one can e.g. build the formulae $\exists z(x + z^2 = y)$ and $\exists z(x + z^2 = y) \vee (x > y)$. The first formula yields a predicate name of arity 2 (since it has two free variables), and it is easy to see that the associated predicate is $\{(r, s);\ r \text{ and } s \text{ are real numbers and } r \leq s\}$. Consequently, the predicate associated to the second formula is $\{(r, s);\ r \text{ and } s \text{ are real numbers}\} = dom(\mathcal{R}) \times dom(\mathcal{R})$.
To get inference algorithms for the extended concept language which will be introduced below, the concrete domain has to satisfy some additional properties. For technical reasons we have to require that the set of predicate names of the concrete domain is *closed under negation*, e.g., if P is an n-ary predicate name in $pred(\mathcal{D})$ then there has to exist a predicate name Q in $pred(\mathcal{D})$ such that $Q^{\mathcal{D}} = dom(\mathcal{D})^n \setminus P^{\mathcal{D}}$. In addition, we need a unary predicate name which denotes the predicate $dom(\mathcal{D})$.

[3] For the sake of simplicity we assume here that the formula itself is the predicate name. In applications, the user will probably take his own intuitive names for these predicates.

The property which will be formulated now clarifies what kind of reasoning mechanisms are required in the concrete domain. Let P_1, ..., P_k be k (not necessarily different) predicate names in $pred(\mathcal{D})$ of arities n_1, ..., n_k. We consider the conjunction

$$\bigwedge_{i=1}^{k} P_i(\underline{x}^{(i)}).$$

Here $\underline{x}^{(i)}$ stands for an n_i-tuple $(x_1^{(i)}, ..., x_{n_i}^{(i)})$ of variables. It is important to note that neither all variables in one tuple nor those in different tuples are assumed to be distinct. Such a conjunction is said to be *satisfiable* iff there exists an assignment of elements of $dom(\mathcal{D})$ to the variables such that the conjunction becomes true in $\mathcal{D}$.
For example, let $P_1(x, y)$ be the predicate $\exists z(x + z^2 = y)$ in $pred(\mathcal{R})$, and let $P_2(x, y)$ be the predicate $x > y$ in $pred(\mathcal{R})$. Obviously, neither the conjunction $P_1(x, y) \wedge P_2(x, y)$ nor $P_2(x, x)$ is satisfiable.

Definition 4.2 *A concrete domain $\mathcal{D}$ is called* admissible *iff* (i) *the set of its predicate names is closed under negation and contains a name for $dom(\mathcal{D})$, and* (ii) *the satisfiability problem for finite conjunctions of the above mentioned form is decidable.*

The concrete domain $\mathcal{R}$ is admissible. This is a consequence of Tarski's decidability result for real arithmetic [Tarski, 1951; Collins, 1975]. However, for the linear case (where the polynomials in the equalities and inequalities have to be linear) there exist more efficient methods (see e.g. [Weispfenning, 1988; Loos and Weispfenning, 1990]). We are now ready to define the extension $\mathcal{ALCF}(\mathcal{D})$ of $\mathcal{ALCF}$ which is parametrized by an admissible concrete domain $\mathcal{D}$.

Definition 4.3 ($\mathcal{ALCF}(\mathcal{D})$) *The concept formalism of $\mathcal{ALCF}$ is extended by one new construct, called predicate restriction. Assume that $f_1, \cdots, f_m$ are $m > 0$ attributes. Then $f_1 \cdots f_m$ is called an attribute chaining. If $u_1, \cdots, u_n$ are attribute chainings and P is an n-ary concrete predicate then $P(u_1, \cdots, u_n)$ (*predicate restriction*) is a concept term.*
The only differences between interpretations $\mathcal{I}$ of $\mathcal{ALCF}(\mathcal{D})$ and $ALCF$ are:

- *The* abstract domain *$dom(\mathcal{I})$ is required to be disjoint to $dom(\mathcal{D})$.*
- *Attributes f are interpreted as partial functions $f^{\mathcal{I}} : dom(\mathcal{I}) \longrightarrow dom(\mathcal{I}) \cup dom(\mathcal{D})$. They establish the link between the abstract and the concrete domain.*

The semantics of the predicate restriction is defined as

$$P(u_1, ..., u_n)^{\mathcal{I}} = \{x \in dom(\mathcal{I}); \text{ there exist } r_1, ..., r_n \in dom(\mathcal{D}) \text{ such that } u_1^{\mathcal{I}}(x) = r_1, ..., u_n^{\mathcal{I}}(x) = r_n \text{ and } (r_1, ..., r_n) \in P^{\mathcal{D}}\}.$$

Here the application of the interpretation function to the attribute chainings u_i is the composition of the respective partial functions.

Under the assumption that the concrete domain is admissible [Baader and Hanschke, 1991] describes sound and complete algorithms for all reasoning services for this extended language that are usually provided for terminological systems. In particular, for two concept terms C and D and a terminology $\mathcal{T}$ over $\mathcal{ALCF}(\mathcal{D})$ it is decidable whether C subsumes D w.r.t. $\mathcal{T}$. The basic algorithm relies on a form of tableaux calculus and is an extension of the procedures in [Schmidt-Schauß and Smolka, 1991; Hollunder *et al.*, 1990; Hollunder, 1990; Donini *et al.*, 1991] that have been designed for concept languages without a concrete domain. The following sample terminology shows how a part of the problem domain of lathes can be formalized using $\mathcal{ALCF}(\mathcal{R})$.

$$
\begin{aligned}
\mathsf{Bottom} &= \mathsf{A} \sqcap \neg\mathsf{A} \\
\mathsf{Truncone} &= \mathsf{truncone\text{-}condition}(\mathsf{r}_1, \mathsf{r}_2, \mathsf{c}_1, \mathsf{c}_2) \\
\mathsf{Ascend} &= \mathsf{Truncone} \sqcap (\mathsf{r}_1 \leq \mathsf{r}_2) \\
\mathsf{Cylinder} &= \mathsf{Truncone} \sqcap (\mathsf{r}_1 = \mathsf{r}_2) \\
\mathsf{Neighboring} &= \exists \mathsf{head}.\mathsf{Truncone} \sqcap \exists \mathsf{tail}.\exists \mathsf{head}.\mathsf{Truncone} \sqcap \\
&\quad (\mathsf{head}\ \mathsf{r}_2 = \mathsf{tail}\ \mathsf{head}\ \mathsf{r}_1) \sqcap (\mathsf{head}\ \mathsf{c}_2 = \mathsf{tail}\ \mathsf{head}\ \mathsf{c}_1) \\
\mathsf{Last} &= \forall \mathsf{tail}.\mathsf{Bottom} \\
\mathsf{One} &= \exists \mathsf{head}.\mathsf{Truncone} \sqcap \mathsf{Last} \\
\mathsf{Two} &= \exists \mathsf{head}.\mathsf{Truncone} \sqcap \exists \mathsf{tail}.\mathsf{One} \sqcap \mathsf{Neighboring}
\end{aligned}
$$

In this terminology r_1, r_2, c_1, c_2, head, and tail are attributes. The A is an arbitrary concept name used to define the empty concept Bottom. The concrete predicate truncone-condition restricts the attribute fillers for r_1, r_2, c_1, and c_2, according to the requirements mentioned in Section 2. For the concrete predicates $=$ and $\leq$ we have used infix notation to increase readability. The expression

$$(\mathsf{head}\ \mathsf{r}_2 = \mathsf{tail}\ \mathsf{head}\ \mathsf{r}_1) \sqcap (\mathsf{head}\ \mathsf{c}_2 = \mathsf{tail}\ \mathsf{head}\ \mathsf{c}_1)$$

in the definition of Neighboring ensure that the 'right' co-ordinates of the 'left' truncated cone coincide with the 'left' co-ordinates of the 'right' truncated cone.

5 Adding Transitive Closure

This section introduces an extension $\mathcal{ALCF}^*$ of $\mathcal{ALCF}$ that can satisfy the demands of the problem domain for representing concepts which contain objects that are sequences of finite but previously unknown length (varying length aspects). The basic idea of this extension is to allow role and attribute *terms* in value-restrictions and exists-in restrictions instead of just allowing role and attribute *names* as in $\mathcal{ALCF}$.

Definition 5.1 (syntax of $\mathcal{ALCF}^*$) *The* role/attribute terms *are built from role and attribute names with*

union *($R \sqcup S$),* composition *($R \circ S$), and* transitive closure *($\mathsf{trans}(R)$)*

of roles and attributes. Concept terms in $\mathcal{ALCF}^*$ are defined as in $\mathcal{ALCF}$ with the only difference that role/attribute terms can be used in value-restrictions and exists-in restrictions.

For example, a sequence of truncated cones can be defined as follows:

$$\mathsf{Sequence} \;=\; \exists \mathsf{head}.\mathsf{Truncone} \sqcap \forall \mathsf{trans}(\mathsf{tail}).\exists \mathsf{head}.\mathsf{Truncone}$$

Since $\mathcal{ALCF}^*$ provides no concrete domains Truncone is a primitive (i.e., not further defined) concept in this terminology and we have not expressed that the truncated cones are neighboring.

Definition 5.2 (semantics of $\mathcal{ALCF}^*$) *The interpretation can be easily extended to attribute/role terms in the obvious way: $R \sqcup S^{\mathcal{I}} = R^{\mathcal{I}} \cup S^{\mathcal{I}}$, $R \circ S^{\mathcal{I}} = \{(x,y); \exists z : (x,z) \in R^{\mathcal{I}}$ and $(z,y) \in S^{\mathcal{I}}\}$, and $\mathsf{trans}(R)^{\mathcal{I}} := \bigcup_{n \geq 1} (R^{\mathcal{I}})^n$.*

In [Baader, 1991] it is shown that for $\mathcal{ALC}^*$ (i.e., $\mathcal{ALCF}^*$ *without* attributes) the subsumption problem is decidable. A close look at the algorithm for $\mathcal{ALC}^*$ (which is much too complex to be sketched here) reveals that the result also holds for $\mathcal{ALCF}^*$, that means, for concept terms C, D and a terminology $\mathcal{T}$ over $\mathcal{ALCF}^*$ (with attributes and roles) it is decidable whether C subsumes D.

6 Combining the Extensions

Up to now we have considered two different extensions of our base language $\mathcal{ALCF}$: The extension by concrete domains and the extension by role/attribute terms involving transitive closure. Now we consider the language $\mathcal{ALCF}^*(\mathcal{D})$ which we obtain if we combine both extensions.
As mentioned in the introductory sections we should like to have both representational facilities available to solve our representation problems. With $\mathcal{R}$ as the concrete domain this language is expressive enough to define concepts that are of great importance in our application domain, such as a 'sequence of neighboring truncated cones' (Seq-tc) and its specialization 'ascending sequence of truncated cones' (Aseq-tc):

$$\begin{aligned} \mathsf{Seq\text{-}tc} \;&=\; \mathsf{Sequence} \sqcap (\mathsf{Last} \sqcup \mathsf{Neighboring}) \sqcap \\ &\quad\; \forall \mathsf{trans}(\mathsf{tail}).(\mathsf{Last} \sqcup \mathsf{Neighboring}) \\ \mathsf{Aseq\text{-}tc} \;&=\; \mathsf{Seq\text{-}tc} \sqcap \forall \mathsf{head}.\mathsf{Ascend} \sqcap \forall \mathsf{trans}(\mathsf{tail}) \circ \mathsf{head}.\mathsf{Ascend} \end{aligned}$$

where the other concepts are as in the sample terminology of Section 4.
The price we have to pay for this expressiveness is that we cannot decide in general whether a concept C subsumes a concept D in this language.
This will be shown by reducing the Post Correspondence Problem to the subsumption problem for this language. The reduction will use only very simple predicates from real arithmetic, namely equalities between linear polynomials in at most two variables.
First, we recall the definition of the Post Correspondence Problem. Let Σ be a finite alphabet. A *Post Correspondence System* (PCS) over Σ is a nonempty finite set $S = \{(l_i, r_i); i = 1, ..., m\}$ where the l_i, r_i are words over Σ. A nonempty sequence $1 \leq i_1, ..., i_n \leq m$ is called a *solution* of the system S iff $l_{i_1} \cdots l_{i_m} = r_{i_1} \cdots r_{i_m}$. It is well-known that the *Post Correspondence Problem*, i.e., the question whether there exists a solution for a given system, is in general undecidable if the alphabet contains at least two symbols [Post, 1946].
A solution of a PCS is a sequence of pairs of words with a previously unknown size. The varying size is represented with the help of the transitive closure on the abstract level, whereas, the words and their concatenation is modeled by predicates of the concrete domain $\mathcal{R}$ over real numbers.
The words are encoded into $\mathcal{R}$ as follows. For $B := |\Sigma| + 1$ we can consider the elements of Σ as digits $1, 2, ..., B-1$ of numbers represented at base B. For a given nonempty word w over Σ we denote by $\overline{w}$ the nonnegative integer (in ordinary representation at base 10) it represents at base B. We assume that the empty word ε represents the integer 0. Obviously, the mapping $w \mapsto \overline{w}$ is a 1−1-mapping from Σ^* into the set of nonnegative integers. Concatenation of words is reflected on the corresponding numbers as follows. Let v, w be two words over Σ. Then we have $\overline{vw} = \overline{v} \cdot B^{|w|} + \overline{w}$, where $|w|$ denotes the length of the word w.
We are now ready to define names for the predicates of the concrete domain $\mathcal{R}$ we shall use in our reduction. For $i = 1, ..., m$,

$$\begin{aligned} C_l^i(x, y, z) &\iff y = \overline{l_i} \wedge z = y + x \cdot B^{|l_i|}, \\ C_r^i(x, y, z) &\iff y = \overline{r_i} \wedge z = y + x \cdot B^{|r_i|}, \\ E(x, y) &\iff x = y, \quad \text{and} \quad L(x) \iff x = 0. \end{aligned}$$

Let l, r, w_l, w_r, and f be attribute names. The concept term $C(S)$ corresponding to the Post Correspondence System S is now defined as follows:

$$\begin{aligned} C(S) \;=\; & \bigsqcup_{i=1}^{m} \left(C_l^i(w_l, l, f\, w_l) \sqcap C_r^i(w_r, r, f\, w_r) \right) \sqcap \\ & L(w_l) \sqcap L(w_r) \sqcap \\ & \forall \mathsf{trans}(f). \left(\bigsqcup_{i=1}^{m} \left(C_l^i(w_l, l, f\, w_l) \sqcap C_r^i(w_r, r, f\, w_r) \right) \right) \sqcap \\ & \exists \mathsf{trans}(f). E(w_l, w_r). \end{aligned}$$

A concept term D is satisfiable iff there is a interpretation such that $D^{\mathcal{I}}$ is not empty. Obviously, a concept term D is satisfiable iff Bottom does not subsume D. Thus, undecidability of satisfiability implies undecidability of subsumption.

Theorem 6.1 *The concept term $C(S)$ is satisfiable if and only if the Post Correspondence System S has a solution. Consequently, satisfiability is in general undecidable for concept terms which may contain transitive closure of attributes and predicate restrictions of an admissible concrete domain.*

Proof. Assume that S has a solution $i_1, ..., i_n$ of length n. We extend this sequence to an infinite sequence $i_1, ..., i_n, i_{n+1}, i_{n+2}, ...$ by choosing arbitrary indices $1 \leq i_{n+1}, i_{n+2}, ... \leq m$. This new sequence is used to define an interpretation $\mathcal{I}$ as follows:

$$\begin{aligned} dom(\mathcal{I}) &:= \{k; k \geq 1\}, \text{ and for all } k \geq 1, \\ f^{\mathcal{I}}(k) &:= k+1, \\ l^{\mathcal{I}}(k) &:= \overline{l_{i_k}} \text{ and } r^{\mathcal{I}}(k) := \overline{r_{i_k}}, \\ w_l^{\mathcal{I}}(k) &:= \overline{l_{i_1} \cdots l_{i_{k-1}}} \text{ and } w_r^{\mathcal{I}}(k) := \overline{r_{i_1} \cdots r_{i_{k-1}}}. \end{aligned}$$

Please note that for $k = 1$, the word $l_{i_1} \cdots l_{i_{k-1}}$ is the empty word, and thus $\overline{l_{i_1} \cdots l_{i_{k-1}}} = 0$. It is now easy to show that $1 \in C(S)^{\mathcal{I}}$. Obviously, this implies that $C(S)$ is satisfiable.
On the other hand, assume that $C(S)$ is satisfiable, and let $\mathcal{I}$ be an interpretation such that $C(S)^{\mathcal{I}} \neq \emptyset$. This interpretation can be used to find a solution of S. Consider an arbitrary element c of $C(S)^{\mathcal{I}}$. Obviously, $c \in (L(w_l) \sqcap L(w_r))^{\mathcal{I}}$ yields $w_l^{\mathcal{I}}(c) = 0 = w_r^{\mathcal{I}}(c)$. Since

$$c \in \left(\bigsqcup_{i=1}^{m} \left(C_l^i(w_l, l, f\, w_l) \sqcap C_r^i(w_r, r, f\, w_r) \right) \right)^{\mathcal{I}},$$

we know that there exists an index between 1 and m, say i_1, such that

$$c \in \left(C_l^{i_1}(w_l, l, f\, w_l) \sqcap C_r^{i_1}(w_r, r, f\, w_r) \right)^{\mathcal{I}}.$$

By the definition of the concrete predicates we get that $l^{\mathcal{I}}(c) = \overline{l_{i_1}}$ and $r^{\mathcal{I}}(c) = \overline{r_{i_1}}$, and $(f\, w_l)^{\mathcal{I}}(c) = \overline{l_{i_1}}$ and $(f\, w_r)^{\mathcal{I}}(c) = \overline{r_{i_1}}$.
Similarly, one can show by induction on k that for all $k \geq 0$ there exists an index i_{k+1} between 1 and m such that

$$\begin{aligned} (f^k\, l)^{\mathcal{I}}(c) &= \overline{l_{i_{k+1}}}, & (f^k\, r)^{\mathcal{I}}(c) &= \overline{r_{i_{k+1}}}, \\ (f^{k+1}\, w_l)^{\mathcal{I}}(c) &= \overline{l_{i_1} \cdots l_{i_{k+1}}}, & (f^{k+1}\, w_r)^{\mathcal{I}}(c) &= \overline{r_{i_1} \cdots r_{i_{k+1}}}. \end{aligned}$$

From $c \in (\exists \mathsf{trans}(f).E(w_l, w_r))^{\mathcal{I}}$ we can now deduce that there exists a positive integer n such that $(f^n\, w_l)^{\mathcal{I}}(c) = (f^n\, w_r)^{\mathcal{I}}(c)$, and thus we have $\overline{l_{i_1} \cdots l_{i_n}} = \overline{r_{i_1} \cdots r_{i_n}}$. Consequently, $l_{i_1} \cdots l_{i_n} = r_{i_1} \cdots r_{i_n}$, which shows that the sequence $i_1, ..., i_n$ is a solution of S. □

7 How to Live with this Undecidability Result

Since terminological systems are only subsystems in larger representation architectures which usually have incomplete reasoners or provide only semidecision procedures, there are two possible ways to avoid undecidable inference problems in the concept language:

- Take $\mathcal{ALCF}(\mathcal{D})$ and deal with the varying length aspects in the surrounding formalism.
- Take $\mathcal{ALCF}^*$ and provide for concrete domains in the surrounding system.

The first approach has been taken in the ARC-TEC project at DFKI, where the varying size aspects are shifted to a rule formalism [Hanschke and Baader, 1991; Hanschke and Hinkelmann, 1991].

References

[Baader and Hanschke, 1991] 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. A long version is available as DFKI Research Report RR-91-10.

[Baader, 1990] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 2, pages 621–626. AAAI, 1990. A long version is available as DFKI Research Report RR-90-01.

[Baader, 1991] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the* 12^{th} *International Joint Conference on Artificial Intelligence*, 1991. A long version is available as DFKI Research Report RR-90-13.

[Borgida *et al.*, 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *International Conference on Management of Data*. ACM SIGMOD, 1989.

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

[Brachman *et al.*, 1979] R. J. Brachman, R. J. Bobrow, P. R. Cohen, J. W. Klovstad, B. L. Webber, and W. A. Woods. Research in natural language understanding, annual report. Tech. Rep. No. 4274, Cambrige, MA, 1979. Bolt Beranek and Newman.

[Collins, 1975] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *2nd Conference on Automata Theory & Formal Languages*, volume 33 of *LNCS*, 1975.

[Donini *et al.*, 1991] Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantificatoin in concept languages. Research Report DFKI-RR-91-02, DFKI / Kaiserslautern, january 1991.

[Edelmann and Owsnicki, 1986] J. Edelmann and B. Owsnicki. Data models in knowledge representation systems: a case study. In GWAI-86 *und 2. Östereichische Artificial-Intelligence-Tagung*, volume 124 of *Informatik-Fachberichte*, pages 69–74. Springer, 1986.

[Hanschke and Baader, 1991] Philipp Hanschke and Franz Baader. Terminological reasoning in a mechanical-engineering domain. In *Workshop on Terminological Logics, Berlin*. KIT-Back research report, 1991.

[Hanschke and Hinkelmann, 1991] Philipp Hanschke and Knut Hinkelmann. Transforming CAD-like geometries into NC programs. In *Workshop on Terminological Logics, Berlin*. KIT-Back Research Report, 1991. description of system demo.

[Hollunder *et al.*, 1990] 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, 1990.

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

[Klauck *et al.*, 1991] Christoph Klauck, Ralf Legleitner, and Ansgar Bernardi. FEAT-REP: Representing features in CAD/CAM. In *4th International Symposium on Artificial Intelligence: Applications in Informatics*, Cancun, Mexiko, 1991. An extended Version is also available as Research Report RR-91-20, DFKI GmbH.

[Kobsa, 1989] A. Kobsa. The SB-ONE knowledge representation workbench. In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, 1989. Two Harbors, Cal.

[Loos and Weispfenning, 1990] R. Loos and V. Weispfenning. Applying linear quantifier elimination. Technical report, Wilhelm Schickard-Institut für Informatik, Universität Tübingen, Germany, 1990.

[Mays *et al.*, 1987] E. Mays, C. Apté, J. Griesmer, and J. Kastner. Organizing knowledge in a complex financial domain. IEEE *Expert*, 2(3):61–70, 1987.

[Mays *et al.*, 1988] E. Mays, C. Apté, J. Griesmer, and J. Kastner. Experience with K-Rep: an object centered knowledge representation language. In *Proceedings of* IEEE CAIA-88, pages 62–67, 1988.

[Nebel, 1989] B. Nebel. Terminological cycles: Semantics and computational properties. In *Proceedings of the Workshop on Formal Aspects of Semantic Networks*, 1989. Two Harbors, Cal.

[Nebel, 1990] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *LNAI*. Springer, 1990.

[Post, 1946] E. M. Post. A varian of a recursively unsolvable problem. *Bull. Am. Math. Soc.*, 52:264–268, 1946.

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

[Tarski, 1951] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. U. of California Press. Berkley, 1951.

[Weispfenning, 1988] V. Weispfenning. The complexity of linear problems in fields. *J. Symbolic Computation*, 5:3–27, 1988.

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-91-24
Jochen Heinsohn: A Hybrid Approach for Modeling Uncertainty in Terminological Logics
22 pages

RR-91-25
Karin Harbusch, Wolfgang Finkler, Anne Schauder:
Incremental Syntax Generation with Tree Adjoining Grammars
16 pages

RR-91-26
M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Koehler, G. Merziger:
Integrated Plan Generation and Recognition
- A Logic-Based Approach -
17 pages

RR-91-27
A. Bernardi, H. Boley, Ph. Hanschke, K. Hinkelmann, Ch. Klauck, O. Kühn, R. Legleitner, M. Meyer, M. M. Richter, F. Schmalhofer, G. Schmidt, W. Sommer:
ARC-TEC: Acquisition, Representation and Compilation of Technical Knowledge
18 pages

RR-91-28
Rolf Backofen, Harald Trost, Hans Uszkoreit:
Linking Typed Feature Formalisms and Terminological Knowledge Representation Languages in Natural Language Front-Ends
11 pages

RR-91-29
Hans Uszkoreit: Strategies for Adding Control Information to Declarative Grammars
17 pages

RR-91-30
Dan Flickinger, John Nerbonne:
Inheritance and Complementation: A Case Study of *Easy* Adjectives and Related Nouns
39 pages

RR-91-31
H.-U. Krieger, J. Nerbonne:
Feature-Based Inheritance Networks for Computational Lexicons
11 pages

RR-91-32
Rolf Backofen, Lutz Euler, Günther Görz:
Towards the Integration of Functions, Relations and Types in an AI Programming Language
14 pages

RR-91-33
Franz Baader, Klaus Schulz:
Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures
33 pages

RR-91-34
Bernhard Nebel, Christer Bäckström:
On the Computational Complexity of Temporal Projection and some related Problems
35 pages

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-92-33
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 H-Subsumption Based on Terminological Reasoning
9 pages

DFKI Technical Memos

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

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 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

DFKI Documents

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

D-92-01
Stefan Bussmann: Simulation Environment for Multi-Agent Worlds - Benutzeranleitung
50 Seiten

D-92-02
Wolfgang Maaß: Constraint-basierte Plazierung in multimodalen Dokumenten am Beispiel des Layout-Managers in WIP
111 Seiten

D-92-03
Wolfgan Maaß, Thomas Schiffmann, Dudung Soetopo, Winfried Graf: LAYLAB: Ein System zur automatischen Plazierung von Text-Bild-Kombinationen in multimodalen Dokumenten
41 Seiten

D-92-04
Judith Klein, Ludwig Dickmann: DiTo-Datenbank - Datendokumentation zu Verbrektion und Koordination
55 Seiten

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 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-92-15
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

Extensions of Concept Languages for a Mechanical Engineering Application

Franz Baader, Philipp Hanschke

RR-92-36
Research Report