Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-37

Specifying Role Interaction in Concept Languages

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

Specifying Role Interaction in Concept Languages

Philipp Hanschke

DFKI-RR-92-37

This report appears also in:

B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*, Morgan Kaufmann, San Mateo, CA.

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

Specifying Role Interaction in Concept Languages

Philipp Hanschke
German Research Center for AI (DFKI)
Project ARC-TEC Postfach 2080
W-6750 Kaiserslautern
Germany

August 18, 1992

Abstract

The KL-ONE concept language provides role-value maps (RVMs) as a concept forming operator that compares *sets* of role fillers. This is a useful means to specify structural properties of concepts. Recently, it has been shown that concept languages providing RVMs together with some other common concept-forming operators induce an undecidable subsumption problem. Thus, RVMs have been restricted to chainings of *functional* roles as, for example, in CLASSIC. Although this restricted RVM is still a useful operator, one would like to have additional means to specify interaction of general roles. The present paper investigates two concept languages for that purpose. The first one provides concept forming operators that generalize the restricted RVM in a different direction. Unfortunately, it turns out that this language also has an undecidable subsumption problem. The second formalism allows to specify structural properties w.r.t. roles without using general equality and is equipped with (complete) decision procedures for its associated reasoning problems.

Contents

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 friend and enemy. Many terminological formalisms concentrate on the following three categories of operators to build a terminology:

- *Boolean connectives* ($\sqcap$, $\sqcup$, and $\neg$) that allow concepts to be combined without any direct reference to their internal structure. For example, if the logical connective conjunction is present as a language construct, one may describe the concept Woman as "humans who are female", and represent it by the expression Human $\sqcap$ Female.

- *Role-forming operators* that allow new roles to be defined. For example the composition ($\circ$) allows the role "friend of enemy" to be represented by enemy $\circ$ friend.

- Operators on *role fillers* that allow the 'internal' structure of the concepts to be operated on. Many languages provide quantification over role fillers which allows, for example, the concept "human with a friend" (resp., "human with only female friends") to be described by the expression Human $\sqcap$ $\exists$friend.Human (resp., Human $\sqcap$ $\forall$friend.Female). An interesting subclass of operators on role fillers are the operators for *role interaction*. The frequently used *number restrictions* can be seen as a degenerated form to specify role interaction (on one role). For example, the concept Lucky-Human could be defined as $\exists^{>100}$friend $\sqcap$ $\exists^{<2}$enemy. As soon as an individual belonging to this concept has two role fillers for enemy, it can be deduced that they are equal.

The kind of models that can be specified by the operators considered so far is quite restricted. If a concept C is satisfiable, then it is satisfiable by an interpretation that arranges its individuals in a tree structure. For example, it is possible to require that the members of a concept have role fillers for a role R, say an individual a, and a role S, say b. But it is not possible to specify that a equals b or that a and b have any common (transitive) role-filler, or that their respective role-fillers are in any relation to each other.

So there is a need for additional means to specify role interaction. The classical prototypes of this kind of operators are the *structural descriptions* and *role-value*

maps (RVMs; see Section 3 for a definition) that are discussed and motivated, e.g., in [Brachman and Schmolze, 1985].

An RVM would allow one to specify that the set of all friends of an individual is equal to the set of all enemies (which may be true for some people if one looks at some never ending soap operas): **enemy** $=_{\text{RVM}}$ **friend** where **enemy** and **friend** are roles.

In [Schmidt-Schauß, 1989; Patel-Schneider, 1989] it has been shown that a concept language with RVMs and a few other common operators has an undecidable subsumption problem. As a reaction on this disappointing negative result, RVMs have been restricted in existing systems to attribute agreements, see, e.g., [Borgida *et al.*, 1989]. *Attributes* are functional roles and are sometimes also called features. I.e., they have at most one role filler per object. Let **best-friend** and **main-enemy** be attributes. Then an individual belongs to the concept **main-enemy** $=_{\text{RVM}}$ **best-friend** if it does not have a main enemy, or if it does not have a best friend, or if its best friend is at the same time its main enemy.[1]

In this paper several other operators for specifying interaction of role and attribute fillers are investigated. The *existential role/attribute agreement* can be used to specify that there is at least one enemy that is also a friend: $\exists(\textbf{enemy} = \textbf{friend})$. If this operator is restricted to attribute chainings it is just the same-as operator in CLASSIC.

The expression $\exists(\textbf{enemy} \circ \textbf{best-friend} = \textbf{friend} \circ \textbf{best-friend})$ represents that there is at least one **enemy** and one **friend** who have the same **best-friend**. The *universal* agreement is used in the expression $\forall(\textbf{enemy} \circ \textbf{best-friend} = \textbf{friend} \circ \textbf{best-friend})$ to formalize that the **best-friend**s of all friends and enemies are the same. On attribute chainings this construct agrees with the RVMs.

The *existential role/attribute disagreement* can express that there is at least one enemy and one friend that are not identical: $\exists(\textbf{enemy} \neq \textbf{friend})$. The expression $\forall(\textbf{enemy} \neq \textbf{friend})$ says that each member has only true friends and true enemies—there is no filler that is both a friend and an enemy.

Although it is at least not obvious how RVMs (on roles) can be simulated by this group of operators, it turns out that the existential and universal agreements lead to an undecidable subsumption problem (Section 3), too.

Section 4 introduces a new concept language which is able to relate fillers of role/attribute chainings. The main idea is to replace the general "=" (resp., "≠") above, by abstract, not further defined predicates or by predicates of a *concrete domain*. In [Baader and Hanschke, 1991a] we already proposed an extension scheme with concrete domains, but there, the predicates are only applied to chainings of attributes. The present paper generalizes this extension scheme considerably.

[1] Actually, in CLASSIC the **same-as** operator requires the existence of one **main-enemy** and one **best-friend**.

As an example, consider the classic (toy) domain of families. Let **age**, **spouse**, and **husband** be attributes, **child** a role, and **Male**, **Human** not further defined concepts. Then a family could be represented by

$$\begin{array}{lcl} \mathsf{Woman} & = & \mathsf{Human} \sqcap \mathsf{Female} \\ \mathsf{Man} & = & \mathsf{Human} \sqcap \neg\mathsf{Female} \\ \mathsf{Family} & = & \exists \mathsf{husband.man} \sqcap \exists \mathsf{spouse.woman} \sqcap \\ & & \forall \mathsf{child.human} \end{array}$$

The specification can be further refined by enforcing that there is a marriage certificate and that children are younger than their parents.

$$\begin{array}{l} \mathsf{Normal\text{-}family} = \mathsf{Family} \sqcap \\ \quad \forall(\mathsf{child} \circ \mathsf{age} > \mathsf{spouse} \circ \mathsf{age}) \sqcap \\ \quad \exists \mathsf{husband}, \mathsf{spouse.marriage\text{-}certificate} \end{array}$$

Here the concrete predicate ">" and an abstract binary predicate **marriage-certificate** are used to formulate the additional requirements.

Section 5 sketches sound and complete reasoning algorithms (see Section 6 for a proof) for this expressive concept language with attribute (dis)agreements and the new structural description operators that are based on predicates.

The concept language $\mathcal{ALCF}$ is the basis for the two extensions and is defined in the next section.

2 THE BAISC LANGUAGE

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

Definition 2.1 (T-box syntax) Concept terms *are built from concept, role, and attribute names using concept-forming operators. If C and D are syntactic variables for concept terms and R is a role or attribute name, then*

$$\begin{array}{ll} C \sqcap D & \text{(conjunction)}, \\ C \sqcup D & \text{(disjunction)}, \\ \neg C & \text{(negation)} \\ \exists R.C & \text{(exists-in restriction)}, \textit{ and} \\ \forall R.C & \text{(value restriction)} \end{array}$$

are concept terms.

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 no concept name appears more than once as a left hand side of a definition, and $\mathcal{T}$ contains no cyclic definitions.[2]

A concept name that does not occur on the left side of a concept definition is called primitive. □

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 2.1.

Definition 2.2 (T-box semantics) *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. $a \in (C \sqcup D)^{\mathcal{I}}$ *iff* $a \in C^{\mathcal{I}}$ *or* $a \in D^{\mathcal{I}}$,
   $a \in (C \sqcap D)^{\mathcal{I}}$ *iff* $a \in C^{\mathcal{I}}$ *and* $a \in D^{\mathcal{I}}$,
   $a \in (\neg C)^{\mathcal{I}}$ *iff* $a \in dom(\mathcal{I}) \setminus C^{\mathcal{I}}$,

2. $a \in (\forall R.C)^{\mathcal{I}}$ *iff*
   for all y with $(x, y) \in R^{\mathcal{I}}$ we have $y \in C^{\mathcal{I}}$, and
   $a \in (\exists R.C)^{\mathcal{I}}$ *iff*
   there exists y with $(x, y) \in R^{\mathcal{I}}$ 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 and satisfiability.

Definition 2.3 (T-box services) *Let $\mathcal{T}$ be a T-box and let C, D be concepts. Then D* subsumes *C with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$. A concept C is* satisfiable *if there is a model $\mathcal{I}$ of $\mathcal{T}$ that satisfies C, i.e., $C^{\mathcal{I}}$ is not empty.* □

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

All extensions of $\mathcal{ALCF}$ in the present paper involve attribute/role chainings, which are built from role and attribute names with the binary, associative infix operator $\circ$ which is interpreted according to

$(a, b) \in (R_1 \circ R_2)^{\mathcal{I}}$ iff
there is a c with $(a, c) \in R_1^{\mathcal{I}}$ and $(c, b) \in R_2^{\mathcal{I}}$

The special attribute name ϵ is always interpreted as identity.

In addition to the formalism defined so far, there is an assertional component (A-box) to draw terminological inferences about individuals.

Definition 2.4 (A-box syntax) *Let* OB *be an alphabet of individuals. If C is a concept and R is a role or attribute, and a, b are individuals, then*

$a = b$ (equality),
$a \neq b$ (negated equality),
$a : C$ (membership assertion), *and*
$(a, b) : R$ (role-filler assertion)

are assertional axioms. *An* A-box *is a finite set of assertional axioms.* □

An interpretation of an A-box is an interpretation $\mathcal{I}$ of $\mathcal{ALCF}$ that in addition assigns to each individual $a \in \mathsf{OB}$ an element $a^{\mathcal{I}} \in dom(\mathcal{I})$.

An interpretation $\mathcal{I}$ *satisfies an equality (a negated equality)* $a = b$ ($a \neq b$) if $a^{\mathcal{I}} = b^{\mathcal{I}}$ ($a^{\mathcal{I}} \neq b^{\mathcal{I}}$), it *satisfies a membership assertion* $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it *satisfies a role- or attribute-filler assertion* $(a, b) : R$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. It *satisfies an A-box* $\mathcal{A}$ if it satisfies all assertional axioms in $\mathcal{A}$.

An interpretation $\mathcal{I}$ is called a *model* of $\mathcal{A}$ w.r.t. a terminology $\mathcal{T}$ if it is a model of $\mathcal{T}$ and satisfies $\mathcal{A}$.

In particular, there is no unique name assumption (UNA) ($a \neq b$ does not imply $a^{\mathcal{I}} \neq b^{\mathcal{I}}$). Since inequality assertions are allowed, the UNA can be easily simulated—if needed for selected individuals. Finally, the model-theoretic semantics is the basis for the formal specification of the reasoning services of the assertional component.

Definition 2.5 (A-box services) *Let a T-box $\mathcal{T}$ be given. An A-box is called* consistent *if it has a model w.r.t. $\mathcal{T}$. An object a is a* member *of a concept C w.r.t. an A-box $\mathcal{A}$ if all models $\mathcal{I}$ of $\mathcal{A}$ satisfy $a : C$, too.* □

Note that a is a member of C w.r.t. $\mathcal{A}$ iff $\mathcal{A} \cup \{a : \neg C\}$ is not consistent.

3 EQUALITY BASED OPERATORS

In this section a concept language based on $\mathcal{ALCF}$ with additional concept forming operators, called existential and universal role/attribute (dis)agreements, is formally defined. These concept forming operators are based on equality and negated equality. This is quite different to the operators introduced in Section 4. As for RVMs, the subsumption problem in a concept language with these additional language constructs is undecidable.

Let $u =_{\text{RVM}} v$ be the original RVM construct, where u and v are two, possibly empty, chainings of roles and attributes. An individual a belongs to the concept $u =_{\text{RVM}} v$ iff the two sets of (transitive) role-fillers of u and v are identical. Formally, an interpretation extends to the RVMs according to:[3]

$$a \in (u =_{\text{RVM}} v)^{\mathcal{I}} \quad \text{iff} \quad a^{\mathcal{I}}u^{\mathcal{I}} = a^{\mathcal{I}}v^{\mathcal{I}}$$

Note that each of the following constructs is different from the RVM construct.

Definition 3.1 (equality-based operators) *The new concept forming operators based on equality and negated equality are defined as follows. Let u and v be two role chainings. Then*

$$\begin{array}{ll} \forall(u = v) & \text{(universal agreement)} \\ \forall(u \neq v) & \text{(universal disagreement)} \\ \exists(u = v) & \text{(existential agreement)} \\ \exists(u \neq v) & \text{(existential disagreement)} \end{array}$$

are concept terms with the following semantics:

$a \in \forall(u = v)^{\mathcal{I}}$ *iff*
for all b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ we have $b^{\mathcal{I}} = c^{\mathcal{I}}$

$a \in \forall(u \neq v)^{\mathcal{I}}$ *iff*
for all b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ we have $b^{\mathcal{I}} \neq c^{\mathcal{I}}$

$a \in \exists(u = v)^{\mathcal{I}}$ *iff*
there exists b with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in u^{\mathcal{I}}$

$a \in \exists(u \neq v)^{\mathcal{I}}$ *iff*
there exist b, c with $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in v^{\mathcal{I}}$ and $(a^{\mathcal{I}}, c^{\mathcal{I}}) \in u^{\mathcal{I}}$ and $b^{\mathcal{I}} \neq c^{\mathcal{I}}$ □

If u and v are attribute agreements, $u =_{\text{RVM}} v$ and $\forall(u = v)$ are equivalent concepts. This is not the case if u and/or v would contain a role. Moreover, it is at least not obvious how RVMs with roles can be simulated by the equality-based

[3]For a binary relation r and an object a the expression ar is defined as the set $\{b;\ r(a, b)\}$.

operators. Unfortunately, $\mathcal{ALCF}$ together with the constructs of the previous definition has an undecidable subsumption problem, too.

This will be shown by a reduction of the *word problem for semi-groups* to the subsumption problem in the concept language. First, the definition of the word problem is recalled. Let Σ be a finite alphabet, let Σ^* be the set of finite, possibly empty words over Σ, and let ϵ be the empty word. Then a set $S = \{l_i = r_i;\ l_i, r_i \in \Sigma^*, i = 1, \ldots, m\}$ is called a *finite presentation of a semi-group*. This set induces a binary relation $\rightarrow_S$ on Σ^*:

$u \rightarrow_S v$ iff
there are words w_1, $w_2 \in \Sigma^*$, and an $l = r \in S$ such that $u = w_1 l w_2$ and $v = w_1 r w_2$.

By $\sim_S$ we denote the reflexive, transitive, and symmetric closure of $\rightarrow_S$. It is well known that a finite presentation S exists consisting of seven equations over a two-element alphabet, $\Sigma = \{a, b\}$ say, such that it is undecidable for two words u and v whether $u \sim_S v$ holds or not (see, for instance [Boone, 1959]).

Now let this system S be given. For the two elements a, $b \in \Sigma$ two attributes a, b are introduced, respectively. Let start, left, right be additional attributes, let back, forth be additional role names, and let A be a fresh concept name. Then for two given words u, $v \in \Sigma^*$ the following concept definition schema is introduced

$$\begin{aligned}\mathsf{Eq}_{u,v} = \ & \exists(\mathsf{left} = \mathsf{start} \circ u) \ \sqcap \ \exists(\mathsf{right} = \mathsf{start} \circ v) \sqcap \\ & \exists(\mathsf{forth} = \mathsf{start})\end{aligned}$$

For any model $\mathcal{I}$ (of the terminology up to this point) satisfying $\mathsf{Eq}_{f_1 \cdots f_m, g_1 \cdots g_n}$ there are (not necessarily distinct) objects, c, a_0, $a_1, \cdots, a_m$, b_0, $b_1, \cdots, b_n$ such that

1. $(a_{i-1}, a_i) \in f_i^{\mathcal{I}}$, for $0 < i \leq m$, and $(b_{i-1}, b_i) \in g_i^{\mathcal{I}}$, for $0 < i \leq n$
2. $a_0 = b_0$, $(c, a_0) \in \mathsf{start}$, $(c, a_0) \in \mathsf{forth}$, $(c, a_m) \in \mathsf{left}$, and $(c, b_n) \in \mathsf{right}$.

This attribute/role structure is depicted in Figure 1.

The concept definition $\mathsf{Top} = \mathsf{A} \sqcup \neg\mathsf{A}$ introduces a name for the universal concept. A single equation $l = r \in S$ can be modeled by a concept $\mathsf{Equation}_{l=r}$ defined by the following schema:

$$\begin{aligned}\mathsf{Equation}_{l=r} = \ & (\exists l.\mathsf{Top} \Rightarrow \exists(l = r)) \sqcap \\ & (\exists r.\mathsf{Top} \Rightarrow \exists(l = r))\end{aligned}$$

where the expressions of the form $A \Rightarrow B$ are abbreviations for $(\neg A) \sqcup B$. The concept $\exists l.\mathsf{Top}$ is satisfied by an element $x \in dom(\mathcal{I})$ iff $l^I(x) \in dom(\mathcal{I})$, i.e., the partial funtion l^I is defined on x.

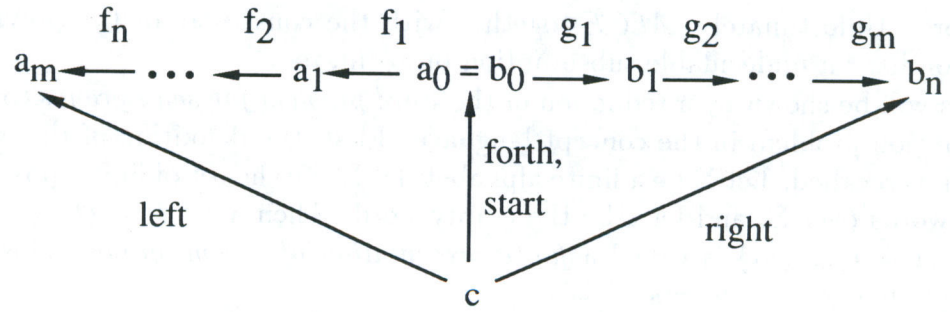

Figure 1: Representing u and v

Assume that the model $\mathcal{I}$ satisfies $\mathsf{Equation}_{l=r}$ and that $a_0 \in \mathsf{Equation}^{\mathcal{I}}_{l=r}$. Then it is easy to show that $a_0 \in \mathsf{Equation}_{l=r}$, $lw \in \Sigma^*$, and $(lw)^{\mathcal{I}}(a_0) \in dom(\mathcal{I})$ implies $(rw)^{\mathcal{I}}(a_0) \in dom(\mathcal{I})$.

The presentation $S = \{e_1, \ldots, e_7\}$ can now be easily represented as

$$\mathsf{LocalS} = \mathsf{Equation}_{e_1} \sqcap \ldots \mathsf{Equation}_{e_7}.$$

But how can this restriction be imposed on each element x for which there is a $w \in \Sigma^*$ such that $w^{\mathcal{I}}(a_0) = x$? Since the concept language provides no transitive closure or cyclic definitions, the element c in Figure 1 is used as a 'relay that refreshes' the restriction. Consider, the following concept definition schema:

$$\begin{aligned} \mathsf{Loop}_\sigma = \; & \forall \mathsf{forth}.\forall\sigma.\exists(\mathsf{back} \circ \mathsf{forth} = \epsilon) \sqcap \\ & \forall(\mathsf{forth} \circ \sigma \circ \mathsf{back} = \epsilon) \end{aligned}$$

Any model of the concept $\mathsf{Eq}_{u,v} \sqcap \mathsf{Loop}_\mathsf{a} \sqcap \mathsf{Loop}_\mathsf{b}$ leads to a role/attribute structure

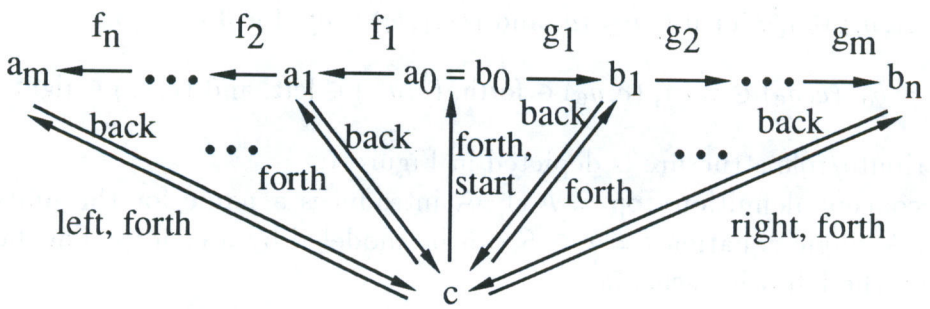

Figure 2: Repeating back and forth

as depicted in Figure 2. More precisely, c has all the x as role fillers for forth that can be reached from a_0 by a word $w \in \Sigma^*$. Now it is easy to impose the requirements of S on each of these elements: $\mathsf{GlobalS} = \forall\mathsf{forth}.\mathsf{LocalS}$

Proposition 3.2 *Given two words* $u, v \in \Sigma^*$

$$\begin{array}{ll} & \mathsf{Eq}_{u,v} \sqcap \mathsf{Loop_a} \sqcap \mathsf{Loop_b} \sqcap \mathsf{GlobalS} \textit{ subsumes} \\ & \exists(\mathsf{left} = \mathsf{right}) \\ \textit{iff} & u \sim_S v. \end{array}$$

Proof.
1) *Assume that* $u \sim_S v$*:*
Let $\mathcal{I}$ be a model of the above concept definitions, and let c be in $(\mathsf{Eq}_{u,v} \sqcap \mathsf{Loop_a} \sqcap \mathsf{Loop_b} \sqcap \mathsf{GlobalS})^{\mathcal{I}}$. Relying on the above construction it is easy to prove the following:

> If $w^{\mathcal{I}}$ is defined on $\mathsf{start}^{\mathcal{I}}(\mathsf{c})$, and $w \rightarrow_S w'$ or $w' \rightarrow_S w$ then $w^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c})) = w'^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c}))$.

By definition there is a finite derivation of $u \sim_S v$ in terms of the symmetric closure of $\rightarrow_S$ and thus, $u^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c})) = v^{\mathcal{I}}(\mathsf{start}^{\mathcal{I}}(\mathsf{c}))$ and $\mathsf{left}^{\mathcal{I}}(c) = \mathsf{right}^{\mathcal{I}}(c)$. By definition: $c \in \exists(\mathsf{left} = \mathsf{right})$.

2) *Assume that not* $u \sim_S v$*:*
It is easy to verify that the interpretation constructed below is a model of the above concept definitions and a counter example to the subsumption relation in question.

Let $dom(\mathcal{I}) = \Sigma^*/_{\sim_S} \cup \{c\}$ where $\Sigma^*/_{\sim_S}$ is the set of equivalence classes induced by the congruence relation $\sim_S$. The partial functions $\mathsf{a}^{\mathcal{I}}$ and $\mathsf{b}^{\mathcal{I}}$ are defined as left multiplications for all $[x] \in \Sigma^*/_{\sim_S}$:

$$\mathsf{a}^{\mathcal{I}}([x]) = [ax] \text{ and } \mathsf{b}^{\mathcal{I}}([x]) = [bx].$$

The other roles and attributes are defined as suggested by the construction:

1. $\mathsf{left}^{\mathcal{I}}(c) = [u]$, $\mathsf{right}^{\mathcal{I}}(c) = [v]$, and $\mathsf{start}^{\mathcal{I}}(c) = [\epsilon]$,
2. $(c, x) \in \mathsf{forth}^{\mathcal{I}}$, for every $x \in \Sigma^*/_{\sim_S}$, and
3. $(x, y) \in \mathsf{back}^{\mathcal{I}}$ if $(y, x) \in \mathsf{forth}^{\mathcal{I}}$. □

Corollary 3.3 *The subsumption problem in a concept language based on* $\mathcal{ALCF}$ *and extended by the* equality-based operators *universal and existential agreement is undecidable.* □

This result shows that, as long as equality is involved, it is wise to restrict oneselfs to attribute (dis)agreements.

4 OPERATORS WITH PREDICATES

It is easy to see (for example by a comparison with CLASSIC) that the subsumption problem remains decidable if the equality-based operators are restricted to chainings of attributes. The reduction in the undecidability proof in the previous section relied heavily on the possibility to specify cyclic role structures (for instance, $\exists(\mathsf{back} \circ \mathsf{forth} = \epsilon)$).

In this section two ideas are developed that remove the capability to specify this kind of cyclic structure from the concept language. The first idea is to replace the equality in the equality-based operators by uninterpreted, possibly negated n-ary predicate symbols. The second idea is to split the interpretation domain into two separate domains: the *abstract* and the *concrete domain* [Baader and Hanschke, 1991a]. Role and attribute fillers can now be restricted by predicates of the concrete domain, too. But concepts are always subsets of the concrete domain.

Together with attribute (dis)agreements the abstract and the concrete predicate based operators are a powerful, still decidable, means to specify structural properties.

4.1 CONCRETE DOMAINS

Before the concept forming operators are introduced the notion "concrete domain" has to be formalized.

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 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.[4] 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 $\{(r, s);\ r$ and s are real numbers and $r \leq s\}$. Consequently, the predicate associated to the second formula is $\{(r, s);\ r$ and s are real numbers$\}$ $= dom(\mathcal{R}) \times dom(\mathcal{R})$.

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

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 the set of predicate names of the concrete domain is required to be *closed under negation*, e.g., if p is an n-ary predicate name in $pred(\mathcal{D})$ then a predicate name q in $pred(\mathcal{D})$ has to exist such that $q^{\mathcal{D}} = dom(\mathcal{D})^n \setminus p^{\mathcal{D}}$. In addition, a unary predicate name is needed which denotes the predicate $dom(\mathcal{D})$.

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. 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]. 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]).

4.2 THE ADDITIONAL OPERATORS

With the above formalization of concrete domains the extension $\mathcal{ALCFP}(\mathcal{D})$ of $\mathcal{ALCF}$ which is parametrized by an admissible concrete domain $\mathcal{D}$ can be defined. The new concept forming operators can be seen as generalizations of the value restriction and the exists-in restriction.

Definition 4.3 (syntax of $\mathcal{ALCFP}(\mathcal{D})$) *The concept formalism of $\mathcal{ALCF}$ is extended by the following operators. Let $u_1, \cdots, u_n$ be role/attribute chainings. Then*

$\forall u_1, \cdots, u_n.\rho$ (generalized value restriction)
$\exists u_1, \cdots, u_n.\rho$ (generalized exists-in restriction)

are concept terms in each of the following cases: The term ρ which is called restrictor

1. *is a predicate of the concrete domain with arity n,*
2. *is of the form p or $\neg p$, where p is an abstract predicate of arity n,*
3. *is a concept term and $n = 1$, or*
4. *is "$=$" or "$\neq$", n is 2, and u_1, u_2 are chainings of* **attributes**. □

In addition to defining the interpretation of the new operators, the interpretation function has to take care of the concrete domain. This somehow makes the definition complicated at first glance.

Definition 4.4 (semantics of $\mathcal{ALCFP}(\mathcal{D})$) *The differences of interpretations of $\mathcal{ALCF}$ and the extended language are as follows:*

The set $dom(\mathcal{I})$, which is called abstract domain *for this language, is required to be disjoint to $dom(\mathcal{D})$.*

Because attributes and roles link the abstract with the concrete domain their interpretation is liberated: An attribute f is interpreted as a partial function

$$f^{\mathcal{I}} : dom(\mathcal{I}) \longrightarrow dom(\mathcal{I}) \cup dom(\mathcal{D})$$

and a role r as a binary predicate

$$r^{\mathcal{I}} \subseteq dom(\mathcal{I}) \times (dom(\mathcal{I}) \cup dom(\mathcal{D})).$$

An abstract predicate p of arity n is interpreted as $p^{\mathcal{I}} \subseteq dom(\mathcal{I})^n$ and $(\neg p)^{\mathcal{I}}$ as $dom(\mathcal{I})^n \setminus p^{\mathcal{I}}$, and a concrete predicate p is interpreted as $p^{\mathcal{I}} = p^{\mathcal{D}}$.

It remains to define how the new operators are interpreted:

$a \in (\forall u_1, \cdots, u_n.\rho)^{\mathcal{I}}$ *iff*
for all $y_1, \cdots, y_n$ with $(x, y_1) \in u_1^{\mathcal{I}}$, $\cdots$, $(x, y_n) \in u_n^{\mathcal{I}}$ we have $(y_1, \cdots, y_n) \in \rho^{\mathcal{I}}$

$a \in (\exists u_1, \cdots, u_n.\rho)^{\mathcal{I}}$ *iff*
there exists $y_1, \cdots, y_n$ with $(x, y_1) \in u_1^{\mathcal{I}}$, $\cdots$, $(x, y_n) \in u_n^{\mathcal{I}}$ and $(y_1, \cdots, y_n) \in \rho^{\mathcal{I}}$

□

The assertional component of the extended formalism allows additional forms of assertional axioms. These are the *predicate assertions* $\rho(a_1, \cdots, a_n)$ where ρ can be an abstract, a negated abstract, or a concrete predicate of arity n. They are satisfied by an interpretation $\mathcal{I}$ iff $(a_1^{\mathcal{I}}, \cdots, n^{\mathcal{I}}) \in \rho^{\mathcal{I}}$.

As already mentioned the membership problem can be reduced to a consistency test. Since a concept term C subsumes a concept term D iff $a : \neg C \sqcap D$ is inconsistent, the subsumption problem can also be reduced to a consistency test. The next section presents a decision procedure that answers in finite time the consistency problem of this extended language. So, the following theorem holds:

Theorem 4.5 *Assume that an admissible concrete domain $\mathcal{D}$ is given. Then there exist decision procedures for the consistency, the subsumption, and the membership problem in $\mathcal{ALCFP}(\mathcal{D})$.* □

5 THE BASIC ALGORITHM

This section presents an algorithm that decides in finite time whether a given A-box $\mathcal{A}_0$ is consistent or not. The algorithm is a generalization of the technique that was introduced in [Schmidt-Schauß and Smolka, 1991] and further elaborated, e.g., in [Baader and Hanschke, 1991b; Baader, 1991; Hollunder *et al.*, 1990]

Roughly, the algorithm proceeds as follows. It starts with a given A-box $\mathcal{A}$, and applies transformation rules to $\mathcal{A}$ that make the knowledge represented by the assertions more explicit. Ultimately, one of the following two situations occurs:

1. The A-box becomes "obviously contradictory", or
2. all knowledge has been made explicit.

In the latter case the A-box is called complete and describes directly a model of the original $\mathcal{A}$. In the other case $\mathcal{A}$ is inconsistent.

Sometimes it is necessary to make a case distinction during the transformation process, since disjunctions occur (implicitly and explicitly) in the formalism. So, a transformation step may transform a single A-box $\mathcal{A}$ in two new A-boxes $\mathcal{B}_1$ and $\mathcal{B}_2$. In this case $\mathcal{A}$ is inconsistent if both $\mathcal{B}_1$ and $\mathcal{B}_2$ are inconsistent. For that reason, the algorithm operates with sets of A-boxes rather than a single A-box. If the consistency of an A-box $\mathcal{A}$ has to be checked, the algorithm is initialized with the singleton set $\mathcal{M}_0 = \{\mathcal{A}_0\}$ where $\mathcal{A}_0$ is the unfolded (see below) version of $\mathcal{A}$.

5.1 UNFOLDING

Let a terminology $\mathcal{T}$ and an A-box $\mathcal{A}_0$ be given. To simplify the presentation, the A-boxes are first normalized by the *unfolding rule*. It replaces a concept name C by its definition t if the concept definition $C = t$ is in $\mathcal{T}$. Because terminologies do not contain cycles this rule can only be applied finitely many times. After all the defined concepts have been replaced, the terminology is not needed any more for the consistency test.

5.2 TRANSFORMATION RULES

This section presents the transformation rules that operate on the set $\mathcal{M}_0$. They generate a finite sequence (see the next section for a proof of the finiteness) of sets $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, ... $\mathcal{M}_k$ of A-boxes.

5.2.1 Pushing Negation

The *negation rules* propagate negation ("$\neg$") towards the leaves of the concept terms. Recall that $\neg$ is a complement operator w.r.t. $dom(\mathcal{I})$ and that attributes and roles link the abstract domain with the concrete domain. So it is convenient to introduce a global complement operator $\sim$. It is defined by $\sim\rho^{\mathcal{I}} = (dom(\mathcal{I}) \cup dom(\mathcal{D})^n \setminus \rho^{\mathcal{I}}$ where ρ is a restrictor with arity n (Definition 4.3).

$$\frac{\neg\neg s}{s}, \quad \frac{\neg(s \sqcap t)}{\neg s \sqcup \neg t}, \quad \frac{\neg(s \sqcup t)}{\neg s \sqcap \neg t}, \quad \frac{\sim\sim\rho}{\rho}$$

$$\frac{\neg\forall v_1, \cdots, v_n.\rho}{\exists v_1 \cdots v_n.\sim\rho}, \quad \frac{\neg\exists v_1, \cdots, v_n.\rho}{\forall v_1 \cdots v_n.\sim\rho}$$

5.2.2 Transformation Rules

Whereas the previous rules are rewriting rules that operate on (sub)terms the following rules operate on the level of assertions. For these remaining rules the expressions of the form

$$\frac{\textit{premises}}{\textit{consequences}}$$

have to be read as follows: if there is an A-box $\mathcal{A}$ in the current $\mathcal{M}_i$ that fulfills the premises, then the successor $\mathcal{M}_{i+1}$ is obtained by adding the consequences to $\mathcal{A}$. A rule must not be applied in an A-box $\mathcal{A}$ with a particular instantiation of the premises if the rule has already been applied with the same instantiation of the premises in that A-box. Neither must it be applied if the A-box contains an obvious contradiction (see Section (5.2.4).

If vertical bars "|" occur in the consequence of a rule, this means that the A-box $\mathcal{A} \in \mathcal{M}_i$ to which the rule is applied has to be replaced with new A-boxes for each of the alternatives that are separated by the bar(s). So, in these cases $\mathcal{M}_{i+1}$ contains more A-boxes than its predecessor $\mathcal{M}_i$.

5.2.2.1 The Operator Rules These rules split concept terms into its immediate subterms and generate new assertions.

$$\textbf{(R}\sqcap\textbf{)} \quad \frac{a : s \sqcap t}{a : s, a : t}$$

$$\textbf{(R}\sqcup\textbf{)}\quad \frac{s \sqcup t : a}{a : s \mid a : t}$$

This rule transforms the affected A-box into two A-boxes.

$$\textbf{(R}\forall\textbf{)}\quad \frac{(a, b_1) : v_1, \cdots, (a, b_n) : v_n, a : \forall v_1 \cdots v_n.\rho}{(b_1, \cdots, b_n) : \rho}$$

A premise $(a, b) : v$ is fulfilled if

1. $a = b$ and v is ϵ or
2. there is a $(a, c) : R$ in the A-box, v splits into Rv' where R is an attribute or role, and, recursively, $(c, d) : v'$ is fulfilled.

$$\textbf{(R}\exists\textbf{)}\quad \frac{a : \exists v_1 \cdots v_n.\rho}{(a, b_1) : v_1, \cdots, (a, b_n) : v_n, (b_1, \cdots, b_n) : \rho}$$

Here the b_i are fresh individuals.

5.2.2.2 The Role and Attribute Rules The (R$\exists$) rule may generate new assertions of the form $(a, b) : v$ where v is a chaining of attributes or roles. It may also cause *forks*. These are pairs of attribute-filler assertions $(a, b) : f$, $(a, c) : f$ with $b \neq c$. These configurations are treated by the following rules:

$$\textbf{(R}\circ\textbf{)}\quad \frac{(a, b) : R \circ v}{(a, c) : R, (c, b) : v}$$

Here c is a fresh individual.

$$\textbf{(R}\epsilon\textbf{)}\quad \frac{(a, b) : \epsilon}{a = b}$$

$$\textbf{(R}\mapsto\textbf{)}\quad \frac{(a, b_1) : f, (a, b_2) : f}{b_1 = b_2} \text{ if } f \text{ is an attribute.}$$

5.2.2.3 The $\sim$ Rules The following rules deal with the global complement operator if it occurs at the top level in an assertion.

$$\textbf{(R}\sim\mathcal{D}\textbf{)}\quad \frac{(a_1, \cdots, a_n) : \sim\rho}{a_1 : \top \mid \cdots \mid a_n : \top \mid (a_1, \cdots, a_n) : \overline{\rho}}$$

if ρ is a concrete predicate and $\overline{\rho}$ is the complement of ρ w.r.t. $dom(\mathcal{D})$ (since $\mathcal{D}$ is admissible $\overline{\rho}$ is also a predicate of the concrete domain), and $\top$ is a specific concept name that is always interpreted as $dom(I)$.

$$\textbf{(R}\sim\mathcal{P}\textbf{)}\quad \frac{(a_1, \cdots, a_n) : \sim\rho}{a_1 : \mathcal{D} \mid \cdots \mid a_n : \mathcal{D} \mid (a_1, \cdots, a_n) : \neg\rho}$$

if ρ is a concept term and $n = 1$ or its is an abstract predicate.

$$(\mathbf{R}{\sim}{=}) \quad \frac{(a_1, a_2) : {\sim}{=}}{(a_1, a_2) : {\neq}}$$

$$(\mathbf{R}{\sim}{\neq}) \quad \frac{(a_1, a_2) : {\sim}{\neq}}{(a_1, a_2) : {=}}$$

5.2.2.4 The Domain Rules The abstract and the concrete domain are disjoint. This may lead to obvious contradictions. The domain rules try to make explicit the domain to which an individual belongs.

$$(\mathbf{R}\mathcal{P}\top) \quad \frac{(a_1, \cdots, a_n) : \rho}{a_1 : \top, \cdots, a_n : \top}$$

if ρ is a primitive concept or an abstract predicate.

$$(\mathbf{R}R\top) \quad \frac{(a_1, a_2) : R}{a_1 : \top}$$ if R is a role or attribute.

$$(\mathbf{R}\mathcal{D}\top) \quad \frac{(a_1, \cdots, a_n) : \rho}{a_1 : \mathcal{D}, \cdots, a_n : \mathcal{D}}$$

if ρ is a concrete predicate different from $\mathcal{D}$.

5.2.2.5 The Identification Rule The attribute agreements and the functional character of the attributes may lead to equality assertions. These are treated by the following rule:

$$(\mathbf{R}{=}) \quad \frac{(a, b) : =}{\text{replace } a \text{ by } b \text{ in the affected A-box}}$$

5.2.3 The Strategy

In order to get a terminating algorithm the order in which the rules may be executed has to be restricted. Identifications of individuals have to take place as soon as possible. So the "role and attribute rules" and the identification rule are executed with the highest priority.

If a transformation rule has been applied to some assertions and later some individuals in these assertions are replaced during applications of the (R=) rule, the transformation rule must not be applied again to these assertions (although the premises are not exactly the same).

5.2.4 Obvious Contradictions

A single A-box $\mathcal{A}$ is *obviously contradictory* in each of the following cases:

Primitive Clash: The A-box contains a pair of assertions of the form $\underline{a} : \rho, \underline{a} : \neg\rho$ where ρ is an abstract predicate (resp., a concept term) and $\underline{a}$ is a tuple of individuals (resp., an individual).

Domain Clash: The A-box contains $a : \top$, $a : \mathcal{D}$.

Equality Clash: The A-box contains $a \neq a$.

Concrete Domain Clash: The A-box contains predicate assertions $\underline{a}_1 : p_1, \ldots, \underline{a}_n : p_n$ where the p_i are concrete predicates and the satisfiability test of the concrete domain says that the above conjunction is not satisfiable.

5.3 SUMMARY OF ALGORITHM

The following procedure, written in a pseudo programming language, summarizes the consistency test of A-boxes of $\mathcal{ALCFP}(\mathcal{D})$.

Algorithm 5.1 (consistency test) *The procedure takes an A-box $\mathcal{A}$ as an argument and checks whether it is consistent or not.*

```
define procedure check-consistency(A)
  A_0 := unfold(A)
  r := 0
  M_0 := {A_0}
  while 'a transformation rule is applicable to M_r'
    do
      r := r + 1
      M_r := apply-a-transformation-rule(M_{r-1})
    od
  if 'there is an A ∈ M_r that is not obviously contradictory'
    then return consistent
    else return inconsistent
```

□

6 THE PROOF

In this section termination, soundness, and completeness of the consistency test (Algorithm 5.1) are proved. Together, these facts imply that the algorithm is a decision procedure for the consistency of an A-box $\mathcal{A}$.

Proposition 6.1 *Assume that Algorithm 5.1 is applied to $\mathcal{A}$. Then*

1. *the algorithm always computes in finite time a set $\mathcal{M}_r$ of A-boxes each of which is complete or obviously contradictory, and*

2. *the initial A-box is inconsistent iff all A-boxes $\mathcal{A} \in \mathcal{M}_r$ contain a clash.*

Proof. The proposition is a consequence of the four lemmata (6.2, 6.3, 6.4) stated and proved below. □

As already mentioned, unfolding terminates, because terminologies are acyclic. Since unfolding does not change the satisfiability of an A-box, this preparatory step is neglected in the proof.

The **while** loop of Algorithm 5.1 reduces the semantic problem of consistency for the A-box $\mathcal{A}_0$ to a simple syntactic problem for a finite set $\mathcal{M}_r$ of A-boxes. This syntactic problem is to check whether there is an A-box in $\mathcal{M}_r$ that is not obviously contradictory. In order to show the correctness of the reduction, termination is proved first.

Assume that a computation using the algorithm is given and that in a single execution of the loop body the A-boxes $\mathcal{B}_1,\ldots,\mathcal{B}_n$, $n > 0$, have been derived by an application of one of the transformation rules to an A-box $\mathcal{A}$. Then the $\mathcal{B}_i$ are called *descendants* of $\mathcal{A}$.

Lemma 6.2 (termination) *The algorithm always computes a complete set of A-boxes $\mathcal{M}_r$ in finite time.*

Proof. Assume that a possibly infinite computation is given. In order to show termination it suffices to prove that there is no infinite sequence of A-boxes $\mathcal{A}_0, \mathcal{A}_1, \ldots$ where $\mathcal{A}_{i+1}$ is a descendant of $\mathcal{A}_i$.

This sequence with the associated applications of transformation rules defines a sequence of trees $\delta_0, \delta_1, \ldots$ as follows:

1. The initial tree δ_0 consists just of the edges $\beta \to \nu(a : C)$ where $a : C$ is a membership assertion in $\mathcal{A}_0$ and β is an additional root.

2. For an individual a let $a^{(k)}$ denote the individual name that stands in place of a after all replacements of the rule (R=) up to $\mathcal{A}_k$ have been performed.

   Each time a transformation rule is applied to an A-box, $\mathcal{A}_k$ say, generating new membership assertions $b_j : B_j$, $j = 1,\ldots,l$, the tree δ_{i+1} is constructed from δ_i. This is done by adding edges $\nu(a : A) \to \nu(b_j : B_j)$, $j = 1,\ldots,l$, where the $\nu(b_j : B_j)$ are new nodes and $a^{(k)} : A$ occurs in the (instantiated) premise of the transformation rule (there is always exactly one such assertion).

If individuals are replaced, this is not done in the tree. So δ_i conservatively extends δ_{i+1}.

Note that not every application of a transformation rule leads to a new δ_i (consider for example the rule (R↦). But, it is easy to observe that the computation of an infinite sequence of descending A-boxes (as the one above) leads to an infinite sequence of trees with an increasing number of nodes.

If the δ's are considered as sets of edges,

$$\Delta = \bigcup_{i=0,1,2,\ldots} \delta_i$$

is a tree, too. If it can be shown that Δ is finite, this yields a contradiction and the proof is done.

Assume that Δ is infinite.

1) The mapping $|.|$ from nodes to naturals is inductively defined as

1. $|\nu(b : B)| := |B|$
2. $|\rho| := 1$, if ρ is an abstract or a concrete predicate, a primitive concept, $=$, or $\neq$.
3. $|B \sqcap C| := |A| + |B|$,
   $|B \sqcup C| := |A| + |B|$,
   $|\exists v_1, \ldots, v_n.\rho| := |\rho| + 1$,
   $|\forall v_1, \ldots, v_n.\rho| := |\rho| + 1$,
   $|{\sim}\rho| := |\rho| * 2 + 1$,
   $|\neg\rho| := |\rho| * 2$

has the following nice property: $\nu(a : A) \rightarrow \nu(b : B)$ implies $|\nu(a : A)| > |\nu(b : B)|$.

This implies that the depth of Δ (defined as the number of edges in the longest directed path) is bounded by

$$\max\{|\nu(a : C)|;\ \nu(a : C) \text{ occurs in } \delta_0\} + 1.$$

2) It remains to show that the tree is finitely branching. Let a node $\nu(a : C)$ be given. If C is not a generalized value restriction, the node has exactly one descendant. This holds because the transformation rule applied to the assertion related to this node has exactly on assertion in its premise, and, by definition, rules are only applied once per premise.

If C is a generalized-value restriction $\forall u_1, \ldots, u_n.\rho$, where ρ is not a concept term then the node does not have any successor, because only new membership assertions lead to new nodes.

So a node $\nu(a : C)$ where C is a value restriction $\forall R.C$ is the last kind of node that could have infinitely many immediate successors. The rule (R$\forall$) is applied once to this assertion per attribute-filler or role-filler assertion $(a, b) : R$.

Since the "role and attribute rules" are executed with a priority higher than the priority of (R$\forall$), the node has only one descendant, too, if R is an *attribute*.

Let a node ν of the form $\nu(a_0 : \forall R.C)$ be given where R is a *role* and C is a concept. Note that this is the last remaining case. All other kinds of nodes have already been proved to have only finitly many successors. Assume that ν has infinitely man descendants.

Observation 1: To get these infinitely many descendants the computation has to generate infinitely many role-filler assertions of the form $(a_0, b) : R$. These come

1. either from an assertion $c : \exists u_1, \ldots, u_n.\rho$ or

2. an assertion $(a', b) : R$ has been generated from an assertion $c : \exists u_1, \ldots, u_n.\rho$ and, later, the individuals $'a$ and a_0 have been identified.

In both cases c is linked to a_0 through a directed path labeled with attributes. Let N be the infinite set of nodes belonging to the generating assertions $c : \exists u_1, \ldots, u_n.\rho$, and let F be the infinite set of generated R role fillers of a_0.

Note that there can be only finitely many exceptions $(a_0, b) : R$ that are not generated from a node in N.

Observation 2: Let $b \in F$. Now observe that all individuals d that can be reached from b through a directed path of attribute/role assertions cannot be reached from another $b' \in F$, $b' \neq b$.

Observation 3: If $\nu(a : A)$ is any node in Δ and $\nu(b : B)$ is any other node below $\nu(a : A)$ then a equals b or there is a directed path from a to b.

Let $(a_0, b) : R$, $b \in F$ be one of the generated assertions. Then there does not exist an attribute/role path from b to a_0.

Together with the contraposition of Observation 3 this implies that the infinitely many nodes in N are not descendants of ν in Δ.

These infinitely many nodes must be descendants of the finitely many nodes in $\mathcal{A}_0$. Consider the subtree Δ' of Δ that is obtained by taking all pathes from the nodes in D to the root β.

Since Δ has finite depth, Δ' has finite depth. Now assume that there is an infinite branch in Δ' at a node $\nu(b : B)$. Then, as above, B is a value restriction $\forall R.'C'$ with a role R', and there are infinitely many role-fillers for b. Only finitely many are not generated by the rules (R$\exists$) and (R$\circ$).

So, there are at least two nodes in the infinite N that are in different subtrees of $\nu(b : B)$ belonging to *generated* role-fillers b_1 and b_n of b w.r.t. R'.

Analog to Observation 2, the set of individuals reachable from b_1 and b_2, respectively, are disjoint. But a_0 is reachable from both b_1 and b_2 because of observations 3 and 1: contradiction. So, Δ' is finite which contradicts the infinity of N. So, ν has only finitely many descendants and Δ is finite,too. □

To prove the second part of Proposition 6.1, the notion of *contradictory A-boxes* is introduced. It is the syntactic equivalent to inconsistent A-boxes. The definition is by induction on the relation "descendant" which has just been proved noetherian. An A-box $\mathcal{A}$ occurring in the computation is *contradictory* with respect to a computation iff

- $\mathcal{A}$ does not have descendants and is obviously contradictory, or

- all descendants of $\mathcal{A}$ are contradictory.

Please note that according to this definition $\mathcal{A}_0$ is contradictory iff after the loop in Algorithm 5.1 all A-boxes in $\mathcal{M}_r$ are obviously contradictory.

Lemma 6.3 (soundness) *An A-box that is contradictory with respect to a given computation is inconsistent.*

Proof. The proof is by induction on the definition of *contradictory*, with a case analysis according to the transformation rule applied. Assume that a contradictory A-box $\mathcal{A}$ is given. It has to be shown that it does not have a model.

1) If $\mathcal{A}$ does not have a descendant, it must be obviously contradictory and cannot have a model.

2) For the induction step, assume to the contrary that $\mathcal{A}$ has a model $\mathcal{I}$. It has to be shown that at least one of the descendants of $\mathcal{A}$ has a model. This will be a contradiction to the induction hypothesis, because all descendants of contradictory A-boxes are contradictory.

This shall only be demonstrated for the case of the (R$\forall$) rule. The other cases can be treated similarly.

Assume that the rule has been applied to the axioms $(a, b_1) : v_1, \cdots, (a, b_n) : v_n$, $a : \forall v_1 \cdots v_n.\rho$ generating the descendant $\mathcal{B}$. Please note that $\mathcal{B}$ is a superset of $\mathcal{A}$ and that the only axiom in $\mathcal{B}$ that is not in $\mathcal{A}$ is $(b_1, \cdots, b_n) : \rho$. Hence, it suffices to show that $\mathcal{I}$ satisfies $b : C$. This is an immediate consequence of the definition of the generalized value restriction. □

Lemma 6.4 (completeness) *If the initial A-box A_0 is not contradictory with respect to a given computation, it has a model.*

Proof. If $\mathcal{A}_0$ is not contradictory then there is an A-box $\mathcal{B} \supseteq \mathcal{A}_0$ in $\mathcal{M}_r$ that is not obviously contradictory. Next an interpretation $\mathcal{I}$ of $\mathcal{B}$ is defined:

1. Because the clash rule related to the concrete domain is not applicable, there is a variable assignment α that satisfies the conjunction of all occurring axioms of the form $P(x_1, \ldots, x_n)$. The interpretation $\mathcal{I}$ interprets all x with $x : \mathcal{D}$ in $\mathcal{B}$ as $\alpha(x)$.

2. The domain $dom(\mathcal{I})$ consists of all the objects x with $x : \top$ in $\mathcal{B}$.

3. Let ρ be a primitive concept or an abstract predicate. Then $(a_1, \ldots, a_n) \in \rho^{\mathcal{I}}$ iff $(a_1, \ldots, a_n) : \rho$ occurs in $\mathcal{B}$. The domain rules ensure that all a_i belong to $dom(\mathcal{I})$.

4. Let R be a role or attribute. Then $(a, b) \in R^I$ iff $(a, b) : R$ is in $\mathcal{B}$. This is well defined even if R is an attribute, because of the transformation rule (R$\mapsto$), which is not applicable to $\mathcal{B}$. The domain rules ensure that a belongs to $dom(\mathcal{I})$.

It is straightforward, but tedious, to show by induction on the size of the axioms that $\mathcal{I}$ is not only an interpretation but also a model of $\mathcal{B}$.

Here only the case of the generalized value restriction is demonstrated:

Assume $a : \forall v_1 \cdots v_n.\rho$ is in $\mathcal{B}$. Let any objects $b_1, \ldots, b_n$ be given. If $(a, b_1) \in v_1^{\mathcal{I}}, \cdots, (a, b_n) \in v_n^{\mathcal{I}}$ the transformation rule (R$\forall$) ensures that $(b_1, \ldots, b_n) : \rho$ is in $\mathcal{B}$. By induction hypothesis, $\mathcal{I}$ satisfies this assertion.

Since the b_i were arbitrary, by definition, $\mathcal{I}$ satisfies the generalized value restriction.

Finally, $\mathcal{A}_0 \subseteq \mathcal{B}$ is used to deduce that $\mathcal{I}$ is also a model for $\mathcal{A}_0$ □

7 CONCLUSIONS

In [Schmolze, 1989] a family of concept languages is presented which is also based on n-ary predicates. One motivation for this formalism is that some concepts are more naturally expressed in terms of n-ary predicates. The terminological formalisms of the present paper are more "object centered" and use predicates only to specify role interaction. It is not clear for which members of the family of concept languages presented in [Schmolze, 1989] decision procedures for the common reasoning services exist.

In the present paper concept forming operators to specify interaction of roles and attributes have been studied. It has been shown that universal and existential role/attribute (dis)agreements lead in general to an undecidable subsumption problem. Nevertheless, an expressive concept language with sound and complete reasoning algorithms has been presented that allows to specify interactions on the basis of abstract and concrete predicates as well as attribute (dis)agreements.

The TAXON system implements a superset of the concept forming operators of $\mathcal{ALCFP}(\mathcal{D})$. The system is written in CommonLisp and has been implemented in the ARC-TEC project (Acquisition and Representation and Compilation of TEChnical Knowledge) [Bernardi *et al.*, 1991]. It is mainly used in an application in mechanical engineering (ARC-TEC) dealing with the production planning of lathe workpieces and a project TOOCON (TOOls for model-based CONfiguration) that develops a configuration system for low-voltage switch boards.

References

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

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

[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, 1990.

[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.

[Bernardi *et al.*, 1991] A. Bernardi, H. Boley, K. Hinkelmann, Ph. Hanschke andh 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*, 1991.

[Boone, 1959] W. W. Boone. The word problem. *Ann. of Mat.*, 1959.

[Borgida *et al.*, 1989] A. Borgida, R. Brachman, D. McGuinness, and L. Resnick. CLASSIC: A structural data model for objects. In *International Conference on Managment on 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), 1985.

[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, 1975.

[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)*, 1990.

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

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

[Patel-Schneider, 1989] P.F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2), 1989.

[Schmidt-Schauß and Smolka, 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Journal of Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *First International Conference On Principles of Knowledge Representation and Reasoning*, 1989.

[Schmolze, 1989] J. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[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. *Journal of Symbolic Computation*, 5, 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

Specifying Role Interaction in Concept Languages

Philipp Hanschke

RR-92-37
Research Report