Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-92-34

Terminological Reasoning and Partial Inductive Definitions

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

Terminological Reasoning and Partial Inductive Definitions

Philipp Hanschke

DFKI-RR-92-34

This report appears also in:

Lars-Henrik Eriksson, Lars Hallnäs, Peter Schroeder-Heister (eds.): *Proceedings of the second workshop on extensions of logic programming*. ELP '91, SICS, Stockholm, Sweden, January 1991, Springer, Lecture Notes in Artificial Intelligence, 1992.

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

Terminological Reasoning and Partial Inductive Definitions

Philipp Hanschke
DFKI, Project ARC-TEC
Kaiserslautern, Germany
hanschke@dfki.uni-kl.de

Abstract

There are two motivations for this paper.

(*i*) In terminological systems in the tradition of KL-ONE the taxonomic and conceptual knowledge of a particular problem domain can be represented by so called concepts. The intensional definitions of these concepts can be analyzed and checked for plausibility using certain reasoning services (e.g. subsumption) that make the user conscious of some of the consequences of his definitions. A hybrid knowledge base can then rely on these checked definitions. In this paper a terminological formalism is embedded into the formalism of partial inductive definitions (PID) such that a flexible environment for experimenting with this kind of hybrid systems and the terminological formalism itself is obtained.

(*ii*) Terminological formalisms provide (terminating) decision procedures for their reasoning services dealing with a restricted kind of quantification. Mapping these algorithms to PID improves the understanding of control and explicit quantification in PID.

Contents

1 Introduction

The family of terminological knowledge representation (TR) languages [6, 5, 10] originated with Brachman's KL-ONE [7]. In these languages the taxonomic and conceptual knowledge of a particular problem domain can be represented by so called concepts and roles. Concepts can be viewed as unary predicates and roles as binary predicates. Concepts are constructed from already defined concepts and roles by *concept forming operators* such as conjunction ($\sqcap$), disjunction ($\sqcup$), and two operators for a restricted form of quantification over role fillers, called exists-in restriction ($\exists$), and value-restriction ($\forall$). For instance the concept Woman, "humans who are female", could be defined as Human $\sqcap$ Female or the concept Mother as Woman $\sqcap$ $\exists$child.Human.

A major advantage of TR in the style of KL-ONE compared to frames and semantic networks is the precise Tarsky-style semantics, which can be defined (in most cases) by a mapping into first-order logic.[1]

The restricted expressiveness of these formalisms allows algorithms that decide in finite time some interesting problems related to concept definitions (T-box reasoning) and instantiations of concepts and roles (A-box reasoning). These reasoning services are the means to analyze and to check the definitions for plausibility. An example of a T-box reasoning service is *subsumption.* A concept A is said to subsume another concept B, if B implies A as a logical formula. For example one would expect that in this partial ordering (commonly referred to as the *subsumption hierarchy*) the concept Woman is above or more general than the concept Mother. If not, this indicates that there is something wrong with the concept definitions.

For a real application it is necessary (because of the restricted expressiveness) and suitable (for adequate knowledge representation) to combine a terminological formalism with other languages covering different knowledge classes. There are a lot of proposals how this combinations should look like, varying from systems where the concepts are used as sorts in a first-order system (full theorem prover or Horn logic) to demon architectures with attached procedures.

The theory of PID has been developed by Lars-Hallnäs [9] and is the

[1] Some systems provide operators that result in predicates not expressible in first-order logic, e.g. transitive closure of roles [3]

basis for the GCLA II system [2]. A main design goal in this approach is the strict separation of logic and control which is also a design goal for Logic Programming (Robert A. Kowalski).

"Algorithm = Logic + Control"

But in Prolog programs this principle is often violated: Control primitives are mixed into Horn clauses thereby violating this principle. Therefore, GCLA II provides two languages:

1. An object-level language for the logic
2. A meta-level language for control

If the syntax of the object-level is restricted to Horn logic and the appropriate control is specified, the pure Horn logic kernel of Prolog is obtained.

The ideal balance between expressive power and efficiency in both the object and the meta-level language is still subject of research. In this paper it is shown how TR formalism with their well understood decision procedures can be embedded in PID. This is an interesting exercise contributing to the mentioned research topics. More precisely, specifying the control for the terminating, logically sound and complete reasoning algorithms (i.e. decision procedures) of TR in a system based on PID will lead to a better understanding of control in this kind of systems. The quantification in the TR language maps to an infinite definition in the formalism of PID. To get a finite one, variables and a restricted form of quantification is introduced. Thus, a subclass of PID is obtained for which explicit quantification inclusive control can be handled.

Since several other programming paradigms, such as object oriented programming and functional programming, have already been embedded in GCLA II [2], a flexible environment is obtained in which combinations of TR reasoning with these paradigms (and Horn logic) can be investigated.

2 The Terminological Formalism

This Section continues the introduction to TR and defines formally a TR formalism, called $\mathcal{ALC}$ [11].

As already mentioned, terminological systems usually split in two parts, called A-box (box for assertional knowledge) and T-box (box for terminological knowledge). The concepts defined in the T-box can be viewed as the natural laws of the application domain. Thus T-box reasoning is reasoning about the natural laws independent of a specific case. Whereas the knowledge items of the A-box correspond to specific cases (observations) in the world that instantiate the natural laws.

A T-box usually consists of a set of *concept definitions* $C = t$ where C is the newly introduced concept name and t is a concept term constructed from *primitive* (i.e. not further defined) concepts, already defined concepts, and primitive roles using concept forming operators. The semantics is obtained by mapping the concept terms to first-order formulas.

Definition 2.1 (terminology) *There are two finite alphabets $\mathcal{C}$ and $\mathcal{R}$ for concept names and role names respectively. Every concept name is a* concept term. *If s and t are concept terms and r is a role name then the following expressions are concept terms:*

$$s \sqcap t \text{ (conjunction)}, s \sqcup t \text{ (disjunction)}, \neg s \text{ (complement)},$$
$$\exists r.s \text{ (exists-in restriction)}, \forall r.s \text{ (value restriction)}$$

The family of mappings ϕ_x, indexed by variables, from concept terms into formulas of first-order logic provides a precise model-theoretic semantics. Let C denote a concept name, r a role name, s,t concept terms, and x a variable. Then the ϕ_x are inductively defined as follows:

1. $\psi_x : C \longmapsto C(x)$
2. (a) $\psi_x : s \sqcap t \longmapsto \psi_x s \wedge \psi_x t$
   (b) $\psi_x : s \sqcup t \longmapsto \psi_x s \vee \psi_x t$
   (c) $\psi_x : \neg t \longmapsto \neg \psi_x t$
   (d) $\psi_x : \exists r.s \longmapsto \exists y : (r(x,y) \wedge \psi_y s)$; *with y a variable different from x*
   (e) $\psi_x : \forall r.s \longmapsto \forall y : (r(x,y) \Rightarrow \psi_y s)$; *with y a variable different from x*[2]

[2]This shows that the first-order formula that corresponds to a concept can be written using at most two variable symbols. Hence this formulas belong to the Gödel class of first order formulas.

A concept definition $C = t$ *is mapped to* $\forall x : (\psi_x C \Leftrightarrow \psi_x t)$ *and a set of concept definitions is called a* terminology. □

The following is a terminology on families.

$$\begin{aligned}
\mathsf{Female} &= \neg\mathsf{Male} \\
\mathsf{Both} &= \mathsf{Male} \sqcap \mathsf{Female} \\
\mathsf{Parent} &= \mathsf{Human} \sqcap \exists \mathsf{child}.\mathsf{Human} \\
\mathsf{Woman} &= \mathsf{Human} \sqcap \mathsf{Female} \\
\mathsf{Man} &= \mathsf{Human} \sqcap \mathsf{Male} \\
\mathsf{Mother\text{-}of\text{-}only\text{-}sons} &= \mathsf{Parent} \sqcap \mathsf{Woman} \sqcap \forall \mathsf{child}.\mathsf{Male}
\end{aligned}$$

In this toy terminology Male and Human are primitive concepts, child is a role, and the rest are defined concept names.

An interesting service provided by TR systems is *subsumption*. A concept term s *subsumes* a concept t iff $\forall x : (\phi_x t \rightarrow \phi_x s)$ is a theorem in the logical theory generated by the terminology. A concept t is *satisfiable* iff $\phi_x t$ is satisfiable w.r.t. the current T-box.

For instance, in the sample terminology Parent subsumes Mother-of-only-sons, but Man does not subsume Woman nor does Woman subsume Man. In this example, all concepts, but Both, are satisfiable.

For each concept term t we can obtain a new concept term logically equivalent to t w.r.t. the involved concept definitions by iteratively unfolding concept names to their definitions. This new term contains only primitive concepts and roles. For example, by iteratively unfolding Mother-of-only-sons we obtain

$$\begin{aligned}
&\mathsf{Human} \sqcap \exists \mathsf{child}.\mathsf{Human} \\
\sqcap\ &\mathsf{Human} \sqcap \neg\mathsf{Male} \\
\sqcap\ &\forall \mathsf{child}.\mathsf{Male}
\end{aligned}$$

Note that none of the remaining concepts appears in the above terminology on a left hand side of a definition. To ease the presentation, without losing generality, in the remainder only concept terms are considered that are constructed by the operators from primitive roles and concepts, and do not contain defined concept names.

Reasoning about a certain family in the example corresponds to assertional reasoning. An *A-box* (assertional box) is a *finite* collection of assertional axioms each of which is of one of the following forms:

$$(i,j) : r \text{ (role assertion)}, \quad i : t \text{ (member-ship assertion)}$$

where i and j are individual names from an infinite alphabet $\mathcal{I}$, $r \in \mathcal{R}$ is a role name, and t is a concept term. Assertional axioms map as follows to first-order formulas:

$$(i,j) : r \longmapsto r(i,j), \quad i : t \longmapsto \phi_i t$$

For the interpretation of the individual names the *unique-name assumption* (UNA) is adopted, i.e.: For all interpretations M and all names $i, j \in \mathcal{I}$ with $i \neq j$ $i^M \neq j^M$.

An A-box is *consistent* if it is consistent as a set of logical axioms. An individual i is a *member* of a concept (term) t, if $\phi_i t$ is a logical consequence of the current A-box (as set of logical axioms).

An algorithm that can be used to build decision procedures for subsumption, satisfiability, member ship and consistency is introduced in [11]. The idea of this algorithm (which is actually a consistency test for an A-box, performed by a specialized tableaux calculus) has been applied to various other concept languages and has been the first sound and complete algorithm for a non-trivial concept language.

3 TR and Infinite Partial Inductive Definitions

In the formalism of PID [9], a query takes the form of a *sequent*:

$$\Gamma \vdash_D C$$

It is answered with respect to a set of definitions[3] D. A possible intuition behind this is that the definitions correspond to the natural laws of the world, that the Γ represents observations in the world, and that the C is a

[3]If the definition D is clear from the context, one just writes $\Gamma \vdash_D C$.

condition which should be deduced from the observations using the natural laws D. This is analogue to the concept languages as the following table shows:

	partial inductive definitions	*concept languages*
natural laws	definition	terminology
observations	antecedent	assertional knowledge

This correspondence together with the reasoning algorithms for $\mathcal{ALC}$ introduced in [11] is the basis of embedding TR in PID. As a next step the infinite formalism of PID without variables is defined more formally (Subsection 3.1) before an embedding of TR in this formalism is shown (Subsection 3.2).

3.1 Infinite PID

Let $\mathcal{U}$ be an universe of atoms including the special atoms falsity $\bot$ and truth $\top$. Then *conditions* can be inductively defined as follows: Every atom $a \in \mathcal{U}$ is a condition. If $C_k, k \in K$, are conditions then the *vector* $(C_k)_{k \in K}$ is a condition. If C and D are conditions then the *conditional* $C \rightarrow D$ is a condition.

A *definition* is a family of equations $a_k = A_k$. The *completion* $\widehat{\mathcal{D}}$ of a definition $\mathcal{D}$ given by $\{a_k = A_k\}_{k \in K}$ is $\mathcal{D} \cup \{a = \bot |\ a \notin \{a_k |\ k \in K\}\}$. The *definiens* $\mathcal{D}(a)$ of an atom a with respect to a definition $\mathcal{D}$ is the set $\{A |\ (a = A) \in \widehat{\mathcal{D}}\}$

We are now ready to define the *condition relation* $\vdash$ on conditions that is generated by a definition $\mathcal{D}$. Let Γ (possibly with subscripts) denote collections of conditions and A, B, C (possibly with subscripts) denote conditions. Then the *condition relation* is the smallest relation satisfying the following properties:

$$\Gamma \vdash \top \qquad \text{(truth property)}$$

$$\Gamma, \bot \vdash C \qquad \text{(falsity property)}$$

$$\Gamma, a \vdash a \qquad \text{(reflexivity)}$$

$$\frac{\{\Gamma \vdash C_k\}_{k \in K}}{\Gamma \vdash (C_k)_{k \in K}} \qquad \text{(vector-right property)}$$

$$\frac{\Gamma, \{C_k\}_{k \in K} \vdash C}{\Gamma, (C_k)_{k \in K} \vdash C} \qquad \text{(vector-left property)}$$

$$\frac{\Gamma, B \vdash C}{\Gamma \vdash B \to C} \qquad \text{(arrow-right property)}$$

$$\frac{\Gamma \vdash B \quad \Gamma, C \vdash a}{\Gamma, (B \to C) \vdash a} \qquad \text{(arrow-left property)}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash a} A \in \mathcal{D}(a) \qquad \text{(D-closed right property)}$$

$$\frac{\{\Gamma, A \vdash C\}_{A \in \mathcal{D}(a)}}{\Gamma, a \vdash C} \qquad \text{(D-closed left property)}$$

In addition, it is assumed that the left hand side of a sequent is unordered and provides implicit *contraction* and *weakening*.

A sequent $\Gamma \vdash C$ can be *proved using* D if the sequent is in the condition relation generated by D. For a formal treatment of this theory see [9].

3.2 Embedding

The union of the following sets is the universe $\mathcal{U}_{TR}$ of atoms in the translation of TR to PID.

- $\{\mathrm{tr}(s,i) |\ s$ a concept term, $i \in \mathcal{I}$ an individual$\}$
- $\{r(i,j) |\ r$ a role name and $i,j \in \mathcal{I}$ individuals$\}$

Intuitively $\mathrm{tr}(s,i)$ means that the individual i is a member of s. The definition D_{TR} is given by the following definition schemata:

- (Dr) $r(i,j) = r(i,j)$
  (Dq) $\mathrm{tr}(q,i) = \mathrm{tr}(q,i)$
  for concept names q, role names r, and individuals i, j. These schemata reflect that by a terminology it is not specified in advance to which (pairs of) individuals a primitive concept (role) applies.

- (D$\sqcap$) $\mathrm{tr}(s \sqcap t, i) = (\mathrm{tr}(s,i), \mathrm{tr}(t,i))$
  (D$\sqcup$1) $\mathrm{tr}(s \sqcup t, i) = \mathrm{tr}(s,i)$
  (D$\sqcup$2) $\mathrm{tr}(s \sqcup t, i) = \mathrm{tr}(t,i)$
  for concept terms s,t and individuals i.

- (D$\neg$) $\mathrm{tr}(\neg q, i) = (\mathrm{tr}(q,i) \rightarrow \bot)$
  (D$\neg\sqcap$) $\mathrm{tr}(\neg(s \sqcap t), i) = \mathrm{tr}(\neg s \sqcup \neg t, i)$
  (D$\neg\sqcup$) $\mathrm{tr}(\neg(s \sqcup t), i) = \mathrm{tr}(\neg s \sqcap \neg t, i)$
  (D$\neg\neg$) $\mathrm{tr}(\neg\neg s, i) = \mathrm{tr}(s,i)$
  (D$\neg\exists$) $\mathrm{tr}(\neg\exists r.s, i) = \mathrm{tr}(\forall r.\neg s, i)$
  (D$\neg\forall$) $\mathrm{tr}(\neg\forall r.s, i) = \mathrm{tr}(\exists r.\neg s, i)$
  for concept terms s,t, (primitive) concept names q, and individuals i.

- (D$\exists$) $\mathrm{tr}(\exists r.s, i) = (r(i,j), \mathrm{tr}(s,j))$
  (D$\forall$) $\mathrm{tr}(\forall r.s, i) = (r(i,k) \rightarrow \mathrm{tr}(s,k))_{k \in \mathcal{I}}$
  for concept terms s, role names r, and individuals i, j. Note: since the set of individuals $\mathcal{I}$ is infinite, $\mathrm{tr}(\exists r.s, i)$ is defined by infinitely many right hand sides in schema (D$\exists$) and $\mathrm{tr}(\forall r.s, i)$ is defined by an infinite right hand side in (D$\forall$).

These schemata provide an infinite realization of TR in PID. The realization is sound and complete with respect to the first order-semantics of TR. The following proposition states this more precisely. To facilitate the formulation of this result a certain class of sequents and a mapping ψ from this class to first-order formulas is defined.

Definition 3.1 (simple TR sequents) *The* simple TR conditions *are a subclass of conditions that is inductively defined.*

- *Every condition of one of the following forms is a simple* TR *condition:*

  $$\bot,\ \top,\ tr(s,i),\ r(i,j),\ (r(i,j) \rightarrow tr(s,j)),\ \text{and}\ (tr(q,i) \rightarrow \bot),$$

  with $i,j \in \mathcal{I}$, s *a concept term, and* q *a (primitive) concept name.*

- *Every vector of simple* TR *conditions is a simple* TR *condition.*

Simple conditions are mapped to first order formulas according to:

$$\begin{array}{lll}
\psi: & \bot & \longmapsto \{\text{false}\}, \\
\psi: & \top & \longmapsto \{\text{true}\}, \\
\psi: & tr(s,i) & \longmapsto \{\phi_i s\}, \\
\psi: & tr(q,i) & \longmapsto \{\phi_i q\} \\
\psi: & tr(q,i) \rightarrow \bot & \longmapsto \{\phi_i \neg q\} \\
\psi: & r(i,j) \rightarrow tr(s,j) & \longmapsto \{r(i,j) \Rightarrow \phi_j s\} \\
\psi: & (v_k)_{k \in K} & \longmapsto \bigcup_{k \in K} \psi v_k
\end{array}$$

A simple TR sequent *is a sequent* $\Gamma \vdash \bot$, *where* Γ *is a collection of simple* TR *conditions.* □

Please note that every A-box $\mathcal{A}$ can be transformed into a simple TR condition C_A such that ψC_A is logically equivalent to $\mathcal{A}$ as a set of logical formulas. In the remainder depending on the context an A-box $\mathcal{A}$ is viewed as a set of assertional axioms, as a simple TR condition, as an antecedent of a simple TR sequent, or as a set (or a conjunction) of logical formulas.

Proposition 3.2 *Let* $\mathcal{A}$ *be a (finite) A-box. Then* $\mathcal{A}$ *is inconsistent iff there is a proof for* $\mathcal{A} \vdash \bot$ *using* D_{TR}.

Proof. Only the ideas of the proof will be sketched. The interested reader is referred to [4] for a detailed proof of a similar proposition.

Assume first that $\Gamma \vdash w$ can be proved using D_{TR}. It is straightforward to show by induction on the properties that if $\mathcal{A} \vdash \bot$ can be proved then $\mathcal{A}$

is inconsistent.[4] Conversely, it is possible to show that proving the original sequent can be reduced by the use of the properties of $\vdash$ to proving a set of 'completed' sequents. If at least one of these 'completed' sequents is not provable the original sequent is not provable and the antecedent of the critical 'completed' sequent resembles a model of the original A-box. □

4 TR and a Finitary Calculus for PID

In order to obtain a finite calculus for the condition relation generated by the definitions D_{TR} variables and explicit quantification are introduced.

The inductive definition of conditions with variables (*X-conditions*[5]) is now based on atoms possibly containing variables (*X-atoms*): Every X-atom is an X-condition. If A and B are X-conditions then $(A \rightarrow B)$ is an X-condition. If v_k are X-conditions, $k \in K, K$ finite, then the vector $(v_k)_{k\in K}$ is an X-condition. If $C(x)$ is an X-condition containing a variable x then $\Pi x : C(x)$ (*product*) and $\Sigma x : C(x)$ (*sum*) are X-conditions.

An *X-definition* D^X is a finite set of equations $a = A$, a an X-atom and A an X-condition. The set of *X-definiens* $D^X(a)$ of an X-atom a is the set $\{B\sigma |\ (b = B) \in D^X, a = b\sigma\}$. A substitution σ is called *a-sufficient* with respect to an X-definition D^X iff for all substitutions τ the set of atoms $D^X(a\sigma\tau)$ equals $D^X(a\sigma)\tau$.

The following inference rules for sequents possibly containing variables shall provide a finite calculus for the condition relation introduced in the previous section. It operates on finite collections of sequents and the left-hand sides of the sequents are considered as sets. Thus, duplications do not occur and the X-conditions are not ordered. In addition the vector-left rule is assumed to be implicit, i.e. it is applied as soon as possible. This view of sequents facilitates the formulation of the control. An inference rule

$$\frac{\{\Gamma_k \vdash C_k\}_{k\in K} \quad \theta}{\Gamma \vdash C \quad \theta\sigma}$$

[4] More generally it can be shown that $\vdash$ is correct with respect to classical logic if $\vdash$ and $\rightarrow$ are interpreted as logical implications, vectors as conjunctions, and the set of definitions D as a set of bi-implications $\{a \Leftrightarrow \bigvee D(a) | a \in \mathcal{U}\}$.

[5] The prefix "X-" is a reminder that the condition may contain variables.

is read as follows: Let $\mathcal{M}$ denote a finite set of sequents. If

$$\mathcal{M}\sigma \cup \{\Gamma_k \vdash C_k\}_{k\in K}$$

can be proved with answer substitution θ then

$$\mathcal{M} \cup \{\Gamma \vdash C\}$$

can be proved with answer substitution $\theta\sigma$. By abuse of notation Γ, A denotes a set $\Gamma \cup A$ or $\Gamma \cup \{A\}$ wherever suitable.

- *Truth rule:* $\dfrac{\theta}{\Gamma \vdash \top \quad \theta}$

- *Falsity rule:* $\dfrac{\theta}{\Gamma, \bot \vdash C \quad \theta}$

- *Reflexivity:* $\dfrac{\theta}{\Gamma, a \vdash b \quad \theta\sigma}$
  Here σ is the most general unifier (mgu) of a and b.

- *Vector-right rule:* $\dfrac{\{\Gamma \vdash C_k\}_{k\in K} \quad \theta}{\Gamma \vdash (C_k)_{k\in K} \quad \theta}$
  Here K has to be finite.

- *Arrow-right rule:* $\dfrac{\Gamma, B \vdash C \quad \theta}{\Gamma \vdash B \to C \quad \theta}$

- *Arrow-left rule:* $\dfrac{\Gamma, (B \to C) \vdash B \quad \Gamma, (B \to C), C \vdash a \quad \theta}{\Gamma, (B \to C) \vdash a \quad \theta}$

- Π-*left rule:* $\dfrac{\Gamma, \Pi x : C(x), C(y) \vdash B \quad \theta}{\Gamma, \Pi x : C(x) \vdash B \quad \theta}$
  Here y denotes a fresh variable not occurring elsewhere.

- Σ-*left rule:* $\dfrac{\Gamma, \Sigma x : C(x), C(c) \vdash C \quad \theta}{\Gamma, \Sigma x : C(x) \vdash B \quad \theta}$
  Here c denotes a fresh constant not occurring elsewhere. In general this formulation of the Σ-left rule is unsound. The newly introduced constant c should rather be a *parameter* which could be unified with other constants, but not with variables occurring in $\Gamma, \Sigma x : C(x) \vdash B$.

For a more detailed formulation, which is also sound in the general case, the reader is referred to [8]. Here, soundness of the formulation for the TR application for the TR application depends on two facts:

- The sequents to which the D-left rule is applied is always ground.
- The equations defining $C(x)$ contain always a variable in the position where the c occurs. Similarly, the axiom rule is only applied, if c has to be unified with a variable.

- *D-right rule:* $\dfrac{\Gamma\sigma \vdash B\sigma \quad \theta}{\Gamma \vdash a \quad \theta\sigma}$
  Here $(b = B) \in D^X$ and $\sigma = mgu(a, b)$

- *D-left rule :* $\dfrac{\{\Gamma\sigma, A\sigma \vdash C\sigma\}_{A \in D^X(a\sigma)} \quad \theta}{\Gamma, a \vdash C \quad \theta\sigma}$
  Here σ is an a-sufficient substitution. If it can be assumed that a is a ground atom and that for all $(c = C) \in D^X$ every free variable in C also occurs in c then the conditions A are the right-hand sides $B\tau$ of the equations $b = B$ in D^X with τ a most general substitution satisfying $a = (b\tau)$. For a more general discussion of a-sufficient substitution see [12].

TR reasoning systems usually provide decision procedures for their reasoning services. For the embedding this means that, after the declarative part has been mapped to the definitions, a control for the application of the inference rules has to be specified that guarantees termination while preserving completeness. In principle the control is known from research in TR. The main problem is the value restriction. If $i : \forall r.s$ is present and $r(i, j)$ is present, then $j : s$ has to be derived in order to get completeness. But to get termination this should only be done once.

There seem to be at least three approaches to this control problem:

1. Introducing an additional argument position for the atoms that represent $i : \forall r.s$ is one possibility. The individuals j, for which $i : \forall r.s$ has been applied to an atom $r(i, j)$, could then be stored in a list in this argument position. Although this approach might work, it would violate the design goal to separate the declarative aspects from the control by coding control information at the declarative "object level".

2. If the control language is expressive enough, the algorithms developed for TR could be coded directly at this level using the object level sequents just as a data structure. This would violate the principle of separating declarative aspects from control again because the definition no longer specifies the logic. In addition, this approach would make it more difficult to extend or modify TR reasoning in this framework or to integrate it closely with other formalisms embedded in PID.

3. The third approach, which will be carried out in the remainder of this paper, tries to find a general control paradigm that applies to a larger class of languages, still provides termination, and allows of a straightforward translation of D_{TR} to an X-definition. Although this approach probably results not in the most efficient algorithms, it preserves flexibility and is open for further efficiency enhancements.

The next step is to turn the infinite definition D_{TR} given in the previous section into a finite one: For the equation schemata (Dr), (Dq), (D$\sqcap$), (D$\sqcup$1), (D$\sqcup$2), (D$\neg$), (D$\neg\sqcap$), (D$\neg\sqcup$), (D$\neg\neg$), (D$\neg\exists$), and (D$\neg\forall$) this is simply done by viewing the i, j, s, t as variables.

The translation of the remaining schemata (D$\exists$), (D$\forall$) involves the new condition forming operators.

$$\begin{array}{lrcl} (\mathrm{D}\exists) & \mathrm{tr}(\exists r.s, x) & = & \Sigma y : (r(x,y), \mathrm{tr}(s,y)) \\ (\mathrm{D}\forall) & \mathrm{tr}(\forall r.s, x) & = & \Pi y : (r(x,y) \rightarrow \mathrm{tr}(s,y)) \end{array}$$

Up to this point, the declarative part of TR reasoning has been specified using infinite PID. This specification has then been translated into a finite calculus for PID introducing variables and a kind of explicit quantification. It remains to specify the appropriate control for the finite calculus.

5 Control

In GCLA II [1] the left-hand side of a sequent is viewed as an ordered list of conditions. In contrast to this in our finite calculus for PID it is considered as a set. Consequently, no duplications of conditions are allowed. This feature is the main means to restrict the applicability of inference rules to a goal. Roughly speaking, the control will be specified such that a rule is not applicable if the newly generated conditions are already present in the left-hand side of the goal sequent. To strengthen the effect of this restriction the "left" rules of the finite calculus (arrow-left rule, Π-left rule, Σ-left rule,

and D-left rule) have been designed such that they preserve the condition to which they have been applied.

For example assume that the presence of a conditional in an antecedent inhibits the application of the D-left rule to an atom that would generate the conditional a second time. Since the arrow-left rule has been designed such that it reproduces the conditional it is applied to, the D-left rule will still be inhibited if the arrow-left rule has been applied to the conditional. For example consider the definition $D = \{a = B \to C\}$ and the following derivation step:

$$\frac{a, (B \to C) \vdash B, \quad a, (B \to C), C \vdash \bot}{a, (B \to C) \vdash \bot}$$

The D-left rule is still not applicable to a.

Subsection 5.1 shows in more detail how this principal strategy, which will be referred to as the *non-duplication strategy*, works for simple TR sequents $\Gamma \vdash \bot$. Subsection 5.2 discusses how subsumption and member-ship queries can be answered using the embedding.

5.1 Non-duplication Strategy

Recall that the vector-left rule is assumed to be implicit in the sequents. For a condition C and a set of conditions Γ the phrase "C is contained in Γ" means that, after the vectors in C and all $B \in \Gamma$ have been splitt into their components, each condition coming from C is an instance of a condition coming from Γ.

The rules with an empty antecedent (*truth rule, falsity rule, reflexivity*) remain unchanged as well as the *vector-right rule,* and the *arrow-right rule.* The Π-*left* rule is only applied to a sequent $\Gamma, \Pi x : B(x) \vdash C$ if there is not already a $B(y)$ in Γ, where y is a variable that does not occur elsewhere.

> In our case, this means that a new condition $r(i, y) \to \mathrm{tr}(s, y)$ is only generated from $\Pi x : (r(i, x) \to \mathrm{tr}(s, x))$ if there is no other $r(i, y') \to \mathrm{tr}(s, y')$ already in Γ.

The Σ-*left rule* is treated similar to the Π-left rule: It is only applied to a sequent $\Gamma, (\Sigma x : B(x)) \vdash C$ if there is not already an $B(c)$ in Γ, with c any term.

What happens when the *arrow-left rule* is applied to a sequent $\Gamma, (A \rightarrow B) \vdash C$?

$$\dfrac{\dfrac{\dfrac{\dots}{\Gamma,(A\rightarrow B)\vdash A} \quad \dfrac{\dots}{\Gamma,(A\rightarrow B),B\vdash,A}}{\Gamma,(A\rightarrow B)\vdash A} \quad \dfrac{\dfrac{\dots}{\Gamma,(A\rightarrow B)\vdash A} \quad \dfrac{\dots}{\Gamma,(A\rightarrow B),B\vdash C}}{\Gamma,(A\rightarrow B),B\vdash C}}{\Gamma,(A\rightarrow B)\vdash C}$$

There are two possibilities to deal with these potential loops:

- The first possibility is to apply the arrow-left rule only iff both generated sequents are new. But, in the above situation the arrow-left rule would then again be applicable as soon as any other rule has been applied. Thus the rule could be applied very frequently without a real progress.

- The idea of the second approach is to delay the application of the arrow-left rule until A can be proved very easily from Γ. Here "easily" means that a very restrictive proof strategy is sufficient to find a proof. An example of a very restrictive strategy, which would be sufficient for the arrow conditions occurring in simple TR sequents, is the strategy which just tries to apply the reflexivity rule.

  If the meta-level is a formalism based on PID and the structure of a proof is represented by so called *proof terms* as in [1, 2] then this proviso could be formulated as a meta-level condition as follows:

  $$\text{reflexivity}(S) \rightarrow (\Gamma \vdash r(i, y))$$

  or more generally

  $$\textit{proof term} \rightarrow (\Gamma \vdash C)$$

  Here reflexivity(S) is used to express that reflexivity is the first (and only) rule to be used.

  According to the non-duplication paradigm the rule is also not applied if B is already present in Γ.

It remains to consider the D-rules. To avoid obvious loops the *D-right rule* is not applied when the involved equation is of the form $a = a$ and the related mgu is just a renaming w.r.t. the variables of the current sequent(s).

In our case this means that equations of the form $r(x, y) = r(x, y)$ and $\mathrm{tr}(q, y) = \mathrm{tr}(q, y)$ are not used together with the D-right rule.

The *D-left rule* is restricted in a similar way. If this rule is applied to a sequent $\Gamma, a \vdash C$, the new goal sequents are $\Gamma\sigma, a\sigma, A\sigma \vdash C\sigma$ where $A \in D^X(a\sigma)$ and σ denotes an a-sufficient substitution. If one of the X-conditions A is already contained in Γ and σ is just a renaming w.r.t. Γ, a, C then to find a proof for the sequent with this A is as hard as to find a proof for the original goal sequent. Consequently, it is reasonable not to apply the rule as the non-duplication paradigm suggests.

In our case this means that equations of the form $r(x, y) = r(x, y)$ and $\mathrm{tr}(q, y) = \mathrm{tr}(q, y)$ are not used together with the D-left rule, either.

As a consequence of the restriction of D-rules an equation $a = a$ can be used to declare that the ground instances of the X-atom a have an unknown truth value as far as the natural-laws D are concerned. Speaking more operationally, instances of an X-atom a are declared not to be subject to the D-rules by an equation $a = a$ in D.

With the control specified above the proof search for a simple TR sequent $\Gamma \vdash \bot$ terminates always. But a further efficiency enhancement is possible, without losing completeness.

Since it has been assumed that simple TR sequents are always ground, the effects of the application of one of the "left rules" to a condition is completely local such that the ordering in which they are applied does not matter. Hence, it is sufficient to proceed as follows:

1. Assume an ordering on the conditions in the left hand side of the sequents (say: "from left to right").

2. Try to find the left-most condition to which the restricted versions of the falsity, the D-left, the arrow-left, the Σ-left, or the Π-left rule is applicable and commit the proof search to this application. I.e. it is not necessary to reconsider another possibility at this point when backtracking takes place. To express this in a control language requires something like an "if-then-else" or a Prolog-like cut or a meta-meta language (if a restricted PID formalism is used as the meta-level as suggested in the already mentioned [2, 1]).

The effect of this strategy is illustrated in the following small example. Consider the definition

$$\begin{aligned} a_1 &= b_1 \\ a_2 &= b_2 \end{aligned}$$

and the sequent $a_1, a_2 \vdash c$. Without the above restriction one would first apply the D-left rule to a_1 and then to a_2. If backtracking takes place, the next attempt would be to apply the D-left rule to a_2 and then to a_1. With the above strategy only the former attempt would be made. This is sufficient, if the effect of the D-left rule is "context free".

5.2 Reasoning Services

The reader might wonder why only $\bot$ is allowed in the right hand side of a simple TR sequent and how subsumption or member-ship queries could be answered using the realization of TR in this paper. Consider for example the concept terms Human and Tall $\sqcup \neg$Tall where Human and Tall are (primitive) concept names. Although, the latter term corresponds to a logical tautology, it is not possible to find a proof for the sequent

$$\mathrm{tr}(\mathsf{Human}, \mathsf{Bill}) \vdash \mathrm{tr}(\mathsf{Tall} \sqcup \neg\mathsf{Tall}, \mathsf{Bill}).$$

Hence, the condition relation is not complete for *all* sequents over U_{TR} w.r.t. the classical semantics of TR. The condition relation (or the finite calculus of it) can only be used to check the consistency of an A-box, coded in the antecedent of a simple TR sequent $\Gamma \vdash \bot$.

The explicit disjunction in the above sequent seems to be the main reason for the problem with the above sequent. But there are similar, less obvious problems which will be illustrated with the member-ship test. Consider the following query[6], which is first presented in English:

> Mary has a child together with her son Ödipus, this child has a daughter, called Jane. Does Mary have a son that has a daughter?

The answer is yes (?!) since the anonymous child of Mary and Ödipus is

[6]The idea of this example is due to M. Lenzerini and has been pointed out to the author by F. Baader.

either male or female. At a first glance the query could be mapped to

$$\begin{array}{r} \mathrm{tr}(\mathsf{Male}, \ddot{\mathsf{O}}\mathsf{dipus}), \\ \mathsf{child}(\mathsf{Mary}, \ddot{\mathsf{O}}\mathsf{dipus}), \\ \mathsf{child}(\mathsf{Mary}, \mathsf{Xxx}), \\ \mathsf{child}(\ddot{\mathsf{O}}\mathsf{dipus}, \mathsf{Xxx}), \\ \mathsf{child}(\mathsf{Xxx}, \mathsf{Jane}), \\ \mathrm{tr}(\neg\mathsf{Male}, \mathsf{Jane}) \end{array} \vdash \mathrm{tr}(\exists\mathsf{child}.(\mathsf{Male} \sqcap \exists\mathsf{child}.\neg\mathsf{Male}), \mathsf{Mary})$$

To deal with the condition

$$\Sigma x : (\mathsf{child}(\mathsf{Mary}, x), \mathrm{tr}(\mathsf{Male} \sqcap \exists\mathsf{child}.\neg\mathsf{Male}, \mathsf{Mary}))$$

occurring at the right hand side of the sequent the (obvious?) Σ-right rule

$$\frac{\Gamma \vdash C(y) \quad \theta}{\Gamma \vdash \Sigma x : C(x) \quad \theta}$$

has to be introduced, where y denotes a fresh variable. But still there is no proof to this query because no substitution for the variable introduced by the first application of the Σ-right rule can be found, since the son in question is either Xxx (if Xxx is male) or Ödipus (if Xxx is not male).

These examples show that subsumption and member-ship queries have to be treated in a special way. Two new predicate symbols *subsumes* and *member* are introduced. They are defined by the equations

$$\begin{array}{lrcl} \text{(Sub)} & \mathrm{subsumes}(s,t) & = & ((\Sigma x : r(\neg s \sqcap t, j)) \rightarrow \bot) \text{ and} \\ \text{(Mem)} & \mathrm{member}(x,t) & = & \mathrm{tr}(\neg t, x) \rightarrow \bot. \end{array}$$

Atoms with this predicate symbols are not allowed in the antecedent.

Then s subsumes t iff $\top \vdash \mathrm{subsumes}(s,t)$ has a proof. Similarly, an individual i is a member of the concept s w.r.t. an A-box A iff $\Gamma \vdash \mathrm{member}(i,s)$ has a proof. As an example consider the derivation for $\mathrm{tr}(\mathsf{Human}, \mathsf{Bill}) \vdash$

member(Bill, Tall $\sqcup \neg$Tall) which becomes:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,(tr(Tall,Bill)\rightarrow\bot),tr(Tall,Bill)\vdash tr(Tall,Bill)} \quad \dfrac{}{\Gamma,(tr(Tall,Bill)\rightarrow\bot),\bot,tr(Tall,Bill)\vdash\bot}}{\Gamma,(tr(Tall,Bill)\rightarrow\bot),tr(Tall,Bill)\vdash\bot}}{\Gamma,(tr(Tall,Bill)\rightarrow\bot),tr(\neg\neg Tall,Bill)\vdash\bot}}{\Gamma,tr(\neg Tall,Bill),tr(\neg\neg Tall,Bill)\vdash\bot}}{\Gamma,tr(\neg Tall\sqcap\neg\neg Tall,Bill)\vdash\bot}}{\Gamma,tr(\neg(Tall\sqcup\neg Tall),Bill)\vdash\bot}}{\Gamma\vdash tr(\neg(Tall\sqcup\neg Tall),Bill)\rightarrow\bot}}{\Gamma\vdash member(Bill,Tall\sqcup\neg Tall)}$$

6 Conclusions

The embedding of TR in the (infinite) theory of PID shows that sound and complete reasoning using PID is possible for a restricted class of sequents that have a semantics based on classical first-order logic and includes a restricted form of universal and existential quantification. Introducing variables and the operators Σ, Π leads to a finite calculus for PID which has been further specialized to a decision procedure for TR using the non-duplication paradigm to specify control. Finally, the examples in Subsection 5.2 illustrate the limits of a constructive formalism, such as PID, if it is used to reason about definitions with classical first-order semantics.

During control specification the demand for the following features of the meta-level formalism showed up:

- The meta-level should provide a means to avoid redundant conditions such as a predicate to test whether one condition is an instance of another.
- The meta-level should have the ability to reason about the success or failure of (a part) of a proof (similar to Kowalski's demo predicate).
- The meta-level should have the ability to commit further proof search to a particular choice.

It may also be a good idea to provide efficient data structures in an implementation that allows both to consider the antecedent as a set or as an

ordered sequence. The former would support the non-duplication paradigm whereas the latter is convenient to specify left to right search.

An implementation of TR in GCLA II along the lines of this paper has been carried out by Martin Aronsson, Per Kreuger and the author.

References

[1] M. Aronsson. The GCLA users's manual. Technical Report T91:21, SICS, 1991.

[2] M. Aronsson, L.-H. Eriksson, L. Hallnäs, and P. Kreuger. A survey of GCLA: A definitional approach to logic programming. In *Proceedings of the International International Workshop on Extensions of Logic Programming, Tübingen, FRG*, 1989. Springer, LNCS 475.

[3] 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.

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

[5] 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.

[6] R. J. Brachman, V. P. Gilbert, and H. J. Levesque. An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 532–539, 1985.

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

[8] L.-H. Eriksson. A finitary version of the calculus of partial inductive definitions. In *this volume*, 1992.

[9] L. Hallnäs. Partial inductive definitions. *Theoretical Computer Science*, to appear.

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

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

[12] P. Schröder-Heister and L. Hallnäs. A proof-theoretic approach to logic programming. Research Report R88005, SICS, 1988. A revised version (in two parts) appears in Journal of Logic and Computation, 2:1, 1990 and n.n.

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

Terminological Reasoning and Partial Inductive Definitions

Philipp Hanschke

RR-92-34
Research Report