A Description Logic Based Approach For Product Knowledge Reasoning[1]

Nizamuddin Channa[1,2], Shanping Li[1], Xiangjun Fu[1]
[1]College of Computer Science, Zhejiang University,Hangzhou, P.R. China 310027
[2]Institute of Business Administration, University of Sindh,Jamshoro, Pakistan 71000
nchanna68@yahoo.com; shan@cs.zju.edu.cn; fuxiangjun@hotmail.com

Abstract

In the context of the semantic web; product knowledge is represented as product ontologies in it's conceptual level. Product ontology construction, integration, and evolution greatly depend on the availability of well-defined semantics and powerful reasoning tools, as to facilitate machine understandability of web resources. In order to capture the semantic of a complicated product data model, the expressive language ALCNHR+K(D) is introduced not only can represent knowledge about concrete domain and constraints, but also, rule in some sense of a closed world semantic model hypothesis. This paper investigates an extension to description logic based knowledge reasoning, by means of decomposing and rewriting complicated hybrid concepts into partitions. We present an approach that automatically decomposes the whole knowledge base into a description logic compatible and constraints solver. Our argument is two-fold. First, complex description logics with powerful representation ability and second is how to reason with the combination of inferences from distributed heterogeneous reasoner.

1. Introduction

Description logics (DLs) [1] are a family of logical formalisms that originated in the field of artificial intelligence as a tool for the representation of conceptual knowledge. Since then, DLs have been successfully used in a wide range of application areas such as knowledge representation, reasoning pertaining to class based formalisms, (e.g conceptual database models and UML diagrams) and ontology engineering in the context of semantic web [2]. The basic syntactic entities of description logics are concepts which are constructed from concept names (unary predicates) and role names (binary relations). Furthermore, a set of axioms (also called Tbox) are used for modeling the terminology of an application Knowledge about specific individuals and their interrelationships is modeled with a set of additional axioms (so-called ABox). Using different constructors defined with a uniform syntax and unambiguous semantics, complex concept definitions and axioms can be built from simple components. Therefore, DLs are particularly appealing both to represent ontological knowledge and to reason with it. Unfortunately, due to the inherent complexity with the product knowledge, the expressive power needed to model complex real-world product ontologies is quite high. Practical product ontology not only needs to represent abstract concept in the application, but also the concrete domain and constrains roles [3]. Even in some system, such as expert system, procedural rules also need to be considered. During the last few years, much research has been devoted to the development of more powerful representation system in DL family [4] [5] [6]. Despite the diversity of their representations, most of them are based on ALC [7] and its expressive successors SHIQ [8], extend the original tableau-based algorithm in different ways. It has been proved that by the reasoning extension of ALC with concrete domains is generally intractable. This problem can be moderated only if suitable restrictions are introduced by way of combining concept constructors [9]. Homogeneous reasoning systems (or systems with homogeneous inference algorithms) have encountered the difficulty of finding the right 'trade-off' between expressiveness and computational complexity. To take advantage of the DLs popularity and flexibility in the context of semantic web we argue that; consistent DLs representation pattern is necessary. However, for reasoning ability, we need to decompose the product ontology into partitions, so that different reasoning paradigms can be jointly used. The benefits of such an approach in the context of ontology sharing through the articulation of ontology interdependencies are highlighted in [10].

The rest of this paper is organized as follows. Section 2 presents the overview of the expressive description logic, section 3 decomposes $ALCNHR+K(D)$ knowledge base into partitions, Section 4 describes System architecture for product knowledge reasoning in detail. Section 5 expresses our conclusion and future work.

[1]*The research, is funded by the Natural Science Foundation of China (No. 60174053,No. 60473052).*

2. The expressive description logic

In this section, we introduce the DLs language $ALCNHR^{+}(D)$[11], which support practical modeling requirements and had been implemented in the RACER (Reasoner for ABoxes and Concept Expression Reasoner) system [12]. Based on $ALCNHR^{+}(D)$, we further extend it by epistemic operator to capture rule knowledge in product data. The following is it's main syntax and semantics explanation:

2.1. Syntax and semantics of $ALCNHR^{+}(D)$

We briefly introduce the syntax and semantics of the DLs language $ALCNHR+(D)$. We assume five disjoint sets: a set of concept names c, a set of role names R, a set of feature names F, a set of individual names O and a set of concrete objects O_C. The mutually disjoint subsets P and T of R denote non-transitive and transitive roles, respectively $(R = P \cup T)$. For presenting the syntax and semantics of the language $ALCNHR+(D)$, a few definitions are required.

Defination 1(Concrete Domain) a concrete domain D is a pair (Δ_D, Φ_D), where Δ_D is a set called the domain and Φ_D is a set of predicate names. The interpretation name function maps each predicate name P_D from Φ_D with arity n to a subset P^I of Δ_D^n. Concert objects from O_C are mapped to an element of Δ_D. A concrete domain D is called admissible iff the set of predicate names Φ_D is closed under negation and Φ_D contains a name T_D for Δ_D and the satisfiability problem $P_1^{n1}(x_{11},....x_{1n1}) \Lambda ... \Lambda P_m^{nm}(x_{m1},...x_{mnm})$ is decidable (m is finite, $P_i^{ni} \in \Phi_D$, ni is the arity of P and x_{jk} is a name for concrete object from Δ_D). We assume that $\perp_D$ is the negation of the predicate T_D.Using the definitions from above, the syntax of concept terms in $ALCNHR+(D)$ is defined as follows.

Definition 2 (Concept Terms) Let C be a set of concept names with is disjoint form R and F. Any elements of C is a concept term. If C and D are concept terms, $R \in$ R is an arbitrary role, $S \in$ S is a simple role, $n, m \in \square$, $n \geq 1$ and $m \geq 0$, $P \in \Phi_D$ is a predicate of the concrete domain, $f, f_1, ..., f_k \in$ F are features, then the following expressions are also concept terms:

$C \cap D$ (conjunction)
$C \cup D$ (disjunction)
$\neg C$ (negation)
$\forall R.C$ (concept value restriction)
$\exists R.C$ (concept exists restriction)
$\exists_{\leq m} S$ (at most number restriction)
$\exists_{\geq n} S$ (at least number restriction)
$\exists f, f_1, ..., f_k.P$ (predicate exists restriction)
$\forall f. \perp_D$ (no concrete domain filler restriction).

Definition 3 (Terminological Axiom, TBox) If C and D are concept terms, then $C \subseteq D$ is a terminological axiom. A terminological axiom is also called generalized concept inclusion or GCI. A finite set of terminological axioms is called a terminology or *TBox*.

The next definition gives a theoretical model semantics to the language introduced above. Let $D = (\Delta_D, \Phi_D)$ be a concrete domain.

Definition 4 (Semantics) an interpretation $I_D = (\Delta_I, \Delta_D, \cdot^I)$ consists of a set Δ_I (the abstract domain), a set Δ_D (the domain of the 'concrete domain' D) and an interpretation function $\cdot^I$. The interpretation function $\cdot^I$ maps each concept name C to a subset C^I of Δ_I, each role name R from R to a subset R^I of $\Delta_I \times \Delta_I$. Each feature f from F is mapped to a partial function f^I from Δ_I to Δ_D where $f^I(a) = x$ will be written as $(a, x) \in f^I$. Each predicate name P from Φ_D with arity n is mapped to a subset P^I of Δ_D^n. Let the symbols C, D be concept expressions, R, S be role names, $f, f_1, ... f_n$ be features and let P be a predicate name. Then, the interpretation function is extended to arbitrary concept and role terms as follows ($\|\bullet\|$denotes the cardinality of a set):

$$(C \cap D)^I := C^I \cap D^I, (C \cup D)^I := C^I \cup D^I, (\neg C)^I := \Delta_I \setminus C^I$$

$$(\exists R.C)^I := \{a \in \Delta_I \mid \exists b : (a,b) \in R^I, b \in C^I\}$$

$$(\forall R.C)^I := \{a \in \Delta_I \mid \forall b : (a,b) \in R^I, b \in C^I\}$$

$$(\exists_{\geq n} R)^I := \{a \in \Delta_I \mid \| \{b \mid (a,b) \in R^I\} \| \geq n\}$$

$$(\exists_{\leq m} R)^I := \{a \in \Delta_I \mid \| \{b \mid (a,b) \in R^I\} \| \leq m\}$$

$$(\exists f_1, ..., f_n.P)^I := \{a \in \Delta_I \mid \exists x_1, ..., x_n \in \Delta_D$$
$$: (a, x_1) \in f_1^I, ..., (a, x_n) \in f_n^I, (x_1, ..., x_n) \in P^I\}$$

$$(\forall f. \perp_D)^{\mathrm{I}} := \{a \in \Delta_I \mid \neg \exists x_1 \in \Delta_D : (a, x_1) \in f^I\}$$

An interpretation I_D is a model of a concept C iff $C^{I_D} \neq \varnothing$. An interpretation I_D satisfies a terminological axiom $C \subseteq D$ iff $C^I \subset D^I$. I_D is a model of a TBox iff it satisfies all terminological axioms $C \subseteq D$ in TBox. An interpretation I_D is a

model of an RBox iff $R^I \subseteq S^I$ for all role inclusions $R \subseteq S$ in R and, in addition, $\forall transtive(R) \in R : R^I = (R^I)^+$

Definition 5 (Assertional Axioms, ABox) Let $O = O_O \cup O_N$ be a set of individual names (or individuals), where the set O_O of "old" individuals is disjoint with the set O_N "new"individuals. Old individuals are those names that explicitly appear in an ABox given as input to an algorithm for solving an inference problem, (i.e. the initially mentioned individuals must not be in O_N). Elements of O_N will be generated internally. Furthermore, let O_C be a set of names for concrete objects $(O_C \cap O = \varnothing)$. If C is a concept term, $R \in R$, a role name, $f \in F$ a feature, $a, b \in O_O$, are individual names and $x, x_1, ...x_n \in O_C$, are names for concrete objects, then the following expressions are assertional axioms or ABox assertions:

$a : C$ (concept assertion), $(a, b) : R$ (role assertion), $(a, x) : f$ (concrete domain feature assertion) and $(x_1..x_n) : P$ (concrete domain predicate assertion).

For example, part of the product model, illustrated in figure 1, can be represented as following:

$$PC \subseteq \forall has\_part.HD \cap \forall has\_part.FD \cap \forall has\_part.Mother\_board \cap \forall has\_part.OS \cap \exists has\_part.HD.storag\_space,$$
$$has\_part.OS.storag\_space\_req.more$$
$$HD \subseteq \forall storage\_space.integer$$
$$OS \subseteq \forall storage\_space\_requirment.integer .$$

2.2. Epistemic Operation K

In a product knowledge system, such as computer aided process planning (CAPP), in addition to terminologies and world descriptions, guidelines are used to express knowledge, especial heuristic rules and default rules [13]. The simplest variant of such rules are expressions of the form $C \Rightarrow D$, where C, D are concepts. The definition, "if an individual is proved to be an instance of C, then derive that it is also an instance of D". Operationally, a forward process can describe the semantics of a finite set of rules. Starting with an initial knowledge base K, a series of knowledge bases $K^{(0)}, K^{(1)}, K^{(2)}, \ldots\ldots$ is constructed, where $K^{(0)} = K$ and $K^{(i+1)}$ is obtained from $K^{(i)}$ by adding a new assertion $D(a)$ whenever there exists a rule $C \Rightarrow D$ such that $K^{(i)} \models C(a)$ holds, but $K^{(i)}$ does not contain $D(a)$. These processes eventually halt if the initial knowledge base contains infinite rules. The difference between the rule $C \Rightarrow D$ and the inclusion axiom $C \subseteq D$ is that the rule is not equivalent to its contrapositive $\neg D \Rightarrow \neg C$. In addition, when applying rules one does not make a case analysis. For example, the inclusions $C \subseteq D$ and $\neg C \subseteq D$ imply that every object belongs to D, whereas none of the rules $C \Rightarrow D$ and $\neg C \Rightarrow D$ applies to an individual a for which neither $C(a)$ nor $\neg C(a)$ can be proven. In order to capture the meaning of rules in a declarative way, we must augment description logics by an operator K [14], which does not refer to objects in the domain, but to what the knowledge base knows about the domain. Therefore, K is an epistemic operator.

To introduce the K-operator, we enrich both the syntax and the semantics of description logic languages. Originally, the K-operator has been defined for ALC [15]. First, we add one case to the syntax rule that allows us to construct epistemic concepts: $C, D \to \mathrm{K}C$ (epistemic concept). Intuitively, the concept $\mathrm{K}C$ denotes those objects for which the knowledge base knows that they are instances of C. Next, using K, we translate rules $C \Rightarrow D$ into inclusion axioms $\mathrm{K}C \subseteq D$.

For example, rules like this: "in a computer, if the motherboard type is B1, then the CPU is only limited to 386 types and the operation system is only limited to Linux can be represented as:

$K(\forall has\_part.B1) \Rightarrow \forall has\_part.linux$. And it can be translated into:

$K(\forall has\_part.B1) \subseteq \forall has\_part.linux$.

Intuitively, the K operator in front of the concept C has the effect that the axiom is only applicable to individuals that appear in the ABox and for which ABox and TBox imply that they are instances of C. Such a restricted applicability prevents the inclusion axiom from influencing satisfiability or subsumption relationships between concepts. In the sequel, we will define a formal semantics for the operator K that has exactly this effect.

A rule knowledge base is a triple $K = (T, A, R)$, where T is a TBox, A is an ABox, and R is a set of rules written as inclusion axioms of the form as $\mathrm{K}C \subseteq D$. The procedural extension of such a triple is the knowledge base $\bar{K} = (T, \bar{A})$ that is obtained from (T, A) by applying the trigger rules as described above. We call the extended knowledge base ALCNHR$^+$K(D) knowledge base, because it extended by the operator K.

The semantics of epistemic inclusions will be defined only to individuals in the knowledge base that provably are instances of C, but not to arbitrary domain elements, which would be the case if we dropped K. The semantics will go beyond first-order logic because we not only have to interpret concepts, roles and individuals, but also have to model the knowledge of a knowledge base. The fact

that a knowledge base has knowledge about the domain can be understood in such a way that it considers only a subset W of the set of all interpretations as possible states of the world. Those individuals that are interpreted as elements of C under all interpretations in W are then "known" to be in C. To make this formal, we modify the definition of ordinary (first-order) interpretations by assuming that: There is a fixed countable infinite set Δ that is the domain of every interpretation (Common Domain Assumption); There is a mapping from the individuals to the domain elements that fixes the way individuals are interpreted (Rigid Term Assumption). The Common Domain Assumption guarantees that all interpretations speak about the same domain. The Rigid Term Assumption allows us to identify each individual symbols with exactly one domain element. These assumptions do not essentially reduce the number of possible interpretations. As a consequence, properties like satisfiability and subsumption of concepts are the same independently of whether we define them with respect to arbitrary interpretations or those that satisfy the above assumptions. Now, we define an *epistemic interpretation* as a pair (I,W), where I is a first-order interpretation and W is a set of first-order interpretations, all satisfying the above assumptions. Every epistemic interpretation gives rise to a unique mapping $\cdot^{I,W}$ associating concepts and roles with subsets of Δ and $\Delta\times\Delta$, respectively. For T, $\bot$ for atomic concepts, negated atomic concepts, and for atomic roles, $\cdot^{I,W}$ agrees with $\cdot^{I}$. For intersections, value restrictions, and existential quantifications, the definition is similar to the one of $\cdot^{I}$.

$$(C\cap D)^{I,W} = C^{I,W}\cap D^{I,W}$$

$$(\forall R.C)^{I,W} = \{a\in\Delta \mid \forall b.(a,b)\in R^{I,W} \rightarrow b\in C^{I,W}\}$$

$$(\exists R.\mathrm{T})^{I,W} = \{a\in\Delta \mid \exists b.(a,b)\in R^{I,W}\}$$

For other constructors, $\cdot^{I,W}$ can be defined analogously. It would also be possible to allow the operator **K** to occur in front of roles and to define the semantics of role expressions of the form **K**. analogously. However, since epistemic roles are not needed to explain the semantics of rules, we restrict ourselves to epistemic concepts.

3. Decompose $ALCNHR+K(D)$ knowledge base into partitions

After rules in product ontology are eliminated through operator K, the $ALCNHR^{+}K(D)$ knowledge base only includes concept definitions, which can be decomposed into three concepts:

Atomic concepts, which define the ground, constructs for ontology modeling. Objects responding to atomic concepts in information system are directly implemented by basic data structure. This in term connects the data level and semantic level in the hierarchy of knowledge representation. For example, in figure 1, i.e. part of a computer configuration model, the concept "HD1" own an attribute "storage_space", which is inherited form the further concept, whose value is an integer value. So "storage_space" is a concrete concept.

Abstract concepts, which are defined through relationships/attributes declarations with hybrid concepts and other abstract concepts, such as "HD".

Hybrid concepts, which are defined through relationships/attributes declarations with atomic concepts and other abstract concepts or hybrid concepts, such as "HD1". To avoid the undecidable inferential problems brought by concrete domain, hybrid concepts are decomposed into an abstract one, an image concrete concept which only contains the concrete concepts and their constrains projected from the source hybrid concept.The link relationship between image concrete concept and abstract concept is implied by the name of image concrete concept. So $ALCNHR^{+}K(D))$ knowledge base denoted as Π_{KB} can be divided into partitions as Π_{DL}, i.e. a set of DL-oriented statements which do not exceed the expressive power of the selected DL-based system, and Π_{CS} i.e. a set of non-DL statements which contains the concrete knowledge filtered out to from Π_{DL}. As a result, instead of reasoning with constrains directly, DL-based systems provide inferential services without being aware of the existence of constraint reasoning. All the information related to concrete domains is removed form concept definitions. Thus, only the proper DL-based constructors, which are admitted by the selected DL-based inferential engines, are left

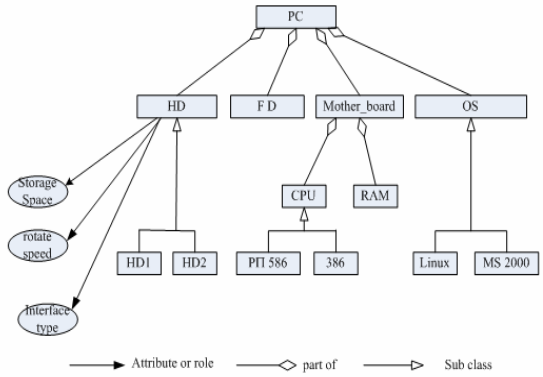

Figure 1. Part of product knowledge model for PC

For instance, let us suppose that the storage space of "HD1" type hard disk are to be required to be more than 4 GB, and the "MS 2000" need at least 2 GB storage space. In order to decompose the hybrid concept, we have:

$HD1 \subseteq Hard\_disk \cap$

$\forall storage\_space.storage\_space_{HD1}$

$MS\_2000 \subseteq Operation\_System$

$\cap storage\_space\_req.storage\_space\_req_{oper\_system}$

In the above expression, the "storage space" restriction is replaced by an atomic concept "storage_space" which has the same name with the attribute name, but with a subscript which denote where the atomic concept comes from. Meanwhile, the restrictions on the hybrid concept is given as

$$storage\_space_{HD1} \geq 4 \times 2^{30}$$

$$storage\_space\_req_{oper\_system} \geq 2 \times 2^{30}$$

Now, by normalizing the knowledge base we split the concepts definitions and restriction into two parts. First, we replace all the hybrid concepts with the wrapper concepts, which are rewrite only by relationship or attribute with abstract concepts, and add new atomic concepts, such as "storage_apaceHD1" into the DL parts. Second, all the image concrete concepts acting as constraints variables are stored in the non-DL part together with their default domain, such as

$storage\_space_{HD1}$ $storage\_space\_req_{oper\_system}$

$type : integer$ …… … $type : integer$

$domain : \geq 4 \times 2^{30}$ $domain : \geq 2 \times 2^{30}$

In default, domain field is the range allowed by data type. The above statements are translated into the underlying modeling languages of the cooperative inferential engines. Subsequently, translated statements are loaded into DL and CPL inferential engines. According to the results from both inferential engines, a reasoning coordinator creates hierarchical structures of hybrid concepts, which are introduced into DL definitions through the atomic axioms concepts. In our example, after loading the non DLs part into an external constraints solver, we obtain a new partial order:

$storage\_space\_req_{operation\_system}$

$\subseteq storage\_space_{HD1}$ Sending such information back to join the original DL part knowledge base, which can be used directly by DLs reasoner. We can conclude that, between satisfying other constraints, if a computer has a "HD1" type hard disk, operation system "linux" can be installed on it.

4. System architecture for product knowledge reasoning

The STEP standard, ISO 10303, is the predominant international standard for the definition, management, and interchange of product data, being used in a wide variety of industries from aerospace, shipbuilding, oil and gas to power generation [16]. Central to the standard is the product data model, which are specified in EXPRESS (ISO 10303-11), a modeling language combing ideas from the entity-attribute-relationship family of modeling languages with object modeling concepts [17]. To satisfy the large number of sophisticated and complex requirements put forwards by large-scale industry, the EXPRESS language has powerful expressing constructs to describe complicated product information, and had been used to build up a family of robust and time-tested standard application protocols. In term these have implemented in most Product data management (PDM) and Computer-Aided-X (CAX) systems. PDM systems manages "data about data" or metadata and provides data management and integration at the image, drawing and document levels of coarse-grain data. CAX systems have provided engineering applications with high-performance solutions. In our former works [18] [19], we had proposed a translation mechanism, which rewrites the EXPRESS, based product knowledge base into DL based. So the system architecture for product data reasoning is composed of three modules, as shows in figure2.

- The translator for EXPRESS schema to DLs;
- Parser for , which divides DLs with constraints and concrete domain to and sub knowledge base.
- Reasoning co-coordinator, which is the link between DLs reasoner and CS reasoner

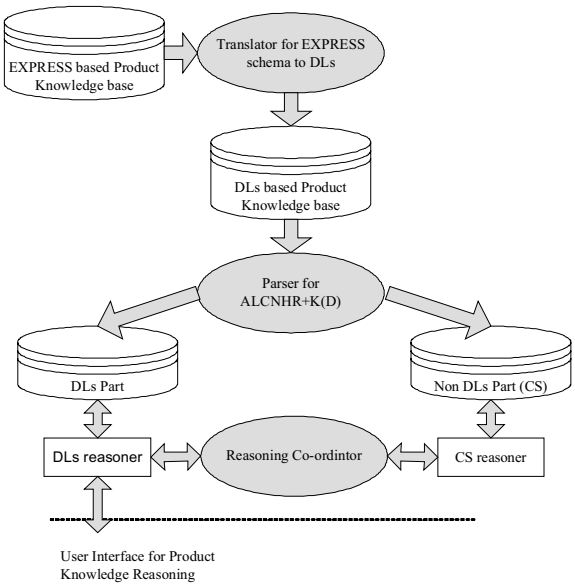

Figure1. Architecture for Product Knowledge Reasoning

The combined reasoning process is as follows:

1. Parse the input EXPRESS schema and translate it into the expressive DL language-ALCNHR+K(D).

2. Parse the DL based product knowledge baseextract the concrete image concepts form hybrid concepts and decompose it into homogeneous parts: DL, non-DL (the concrete value and constraints).
3. Check the consistency of constraints and propagate them in order to maintain a full path-consistency by reducing the set of possible values associated with each constrained variable.
4. Update DL-based representation with the quasi-ordering between the atomic concepts which are the corresponding image concept for each variable.
5. Update and classify the DL-based descriptions based on the new knowledge.

5. Conclusions and Future work

In previous sections we presented architecture for reasoning on product knowledge, which takes originally EXPRESS Schema as input. In order to capture the semantic of complicated product data model, the expressive language $ALCNHR^{+}K(D)$) is introduced. It not only can represent knowledge about concrete domain and constraints, but also can rule in some sense of closed world semantic model hypothesis. To avoid the undecidable inferential problems brought by the extension. A partition based reasoning approach is proposed. The usual reasoning problems, such as concept subsuming, can be resolved by the combined reasoning systems, which take the DL reason engine as the core part. Utilizing current Semantic Web technology, product knowledge can be embedded inside Web resources. One feature of this capability is the data sources, which are readily available for consumption by a wide variety of Semantic Web users. Our proposed product knowledge reasoning architecture can be used to Semantic Web based search engines and discovery services. For further work, we need to optimize the hybrid reasoning system to adapt the diverse application domain, which would gracefully support to automatically semantic web services composition and execution.

6. References

[1] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., and Rosati, R. 1998. Description logic framework for information integration. In Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98). 2–13.

[2] The Semantic Web lifts off 'by Tim Berners-Lee and Eric Miller, W3C. ERCIM News No. 51, October 2002

[3] Felix Metzger, "The challenge of capturing the semantics of STEP data models precisely", Workshop on Product Knowledge Sharing for Integrated Enterprises (ProKSI'96), 1996.

[4] F. Baader and U. Sattler, "Description Logics with Concrete Domains and Aggregation", In H. Prade, editor, Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98), pages 336-340. John Wiley & Sons Ltd, 1998.

[5] F. Baader and R. Küsters, "Unification in a Description Logic with Transitive Closure of Roles". In R. Nieuwenhuis and A. Voronkov, editors, Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001), volume 2250 of Lecture Notes in Computer Science, pages 217–232, Havana, Cuba, 2001. Springer-Verlag.

[6] V. Haarslev, C. Lutz, and R. Möller, "A Description Logic with Concrete Domains and Role-forming Predicates". Journal of Logic and Computation, 9(3):351–384, 1999.

[7] The Description Logic Handbook, edited by F. Baader, D. Calvanese, DL McGuinness, D. Nardi, PF Patel-Schneider, Cambridge University Press, 2002.

[8] Ian Horrocks, Ulrike Sattler, "Optimised Reasoning for SHIQ", ECAI 2002: 277-281.

[9] I. Horrocks, U. Sattler, and S. Tobies, "Practical Reasoning for Very Expressive Description Logics". Logic Journal of the IGPL, 8(3):239–264, May 2000.

[10] E. Compatangelo, H. Meisel, "K-Share: an architecture for sharing heterogeneous conceptualizations". In Intl. Workshop on Intelligent Knowledge Management Techniques (I-KOMAT'2002) - Proc. of the 6th Intl. Conf. on Knowledge-Based Intelligent Information & Engineering Systems (KES'2002), pages 1439–1443.

[11] Volker Haarslev, Ralf Möller, Michael Wessel, "The Description Logic ALCNHR+ Extended with Concrete Domains: A Practically Motivated Approach". IJCAR 2001: 29-44.

[12] Domazet D., "The automatic tool selection with the production rules matrix method". Annals of the CIRP, 1990, 39(1): 497□500.

[13] Volker Haarselev and Ralf Moller. RACER system Description. In proceedings of the International Joint Conference on Automated Reasoning(IJCAR 2001), Volume 2083, 2001.

[14] Dretske, Fred, "Epistemic Operators, The Journal of Philosophy", Vol. LXVII, No.24, Dec. 24, pp.1007-1023.

[15] Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W., and Schaerf, A., "Adding epistemic operators to concept languages". In Proceedings of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR'92). Morgan Kaufmann, Los Altos, 342–353.

[16] Mike Pratt, "Introduction to ISO 10303 - The STEP Standard for Product Data Exchange", ASME Journal of Computing and Information Science in Engineering, November, 2000

[17] Schenk, D.A., and Wilson, P.R., "Information Modeling: The EXPRESS Way", Oxford University Press, New York, NY, 1994 (ISBN 0-19-508714-3).

[18] Xiangjun Fu, Shanping Li, "Ontology Knowledge Representation for Product Data Model", Journal of Computer-Aided Design & Computer Graphics, to appear (in Chinese).

[19] Xiangjun Fu, Shanping Li, Ming Guo, Nizamuddin Channa "Methodology for Semantic Representing of Product Data in XML", In Proceedings of Advance Workshop on Content Computing, LNCS, 2004