A hybrid approach to extend DL-based reasoning with concrete domains

Bo Hu[1] Ernesto Compatangelo[2] Ines Arana[1]

[1] School of Computer and Mathematical Sciences,
The Robert Gordon University, Aberdeen, UK

[2] Department of Computing Science, University of Aberdeen, UK

29th August 2001

Abstract

We propose a new hybrid approach which extends the expressive power of DL languages by incorporating concrete domains. Without modifying the DL inference algorithms, our approach uses the results of other non-DL inferential engines to reason about terminological knowledge. Constraints involving concrete domains are reasoned and replaced with equivalent concept restrictions exclusively based on the expressive power of the DL languages selected as the DL-based inferential engines. Meanwhile, We outline a system architecture that can support such an approach, which involves a homogeneous knowledge representation and hybrid reasoning.

1 Extending existing DL-based system

Description Logics (DLs) are a well-known family of knowledge representation and reasoning formalisms [6]. They are featured by the ability of building up complex knowledge from basic notion of concepts (unary predicates) and roles (binary relations). Various DL-based systems are also available to provide inferences on such complex knowledge. Ever since they were introduced two decades ago, DLs have always been characterised by a reasonable trade-off between expressive power and computational complexity. However, this trade-off has not prevented the development of inferential engines based on expressive concept languages, such as $\mathcal{SHIQ}$ [10].

So far, the major effort in extending the expressive power of DL systems has been put in the enhancement or optimisation of the inference algorithms. Various extensions have been proposed or implemented in the last decade. For instance, concrete domains have been introduced into DLs which are normally

used to capture abstract domains [2]. A very expressive language which provides reasoning services on individuals and limited concrete knowledge (concrete data types) has also been introduced [5]. Despite their diversity, virtually all of the above approaches, which are based on the $\mathcal{ALC}$ language [13], extend the original $\mathcal{ALC}$ tableau inference algorithm in different ways [3].

Modifying the inference algorithm normally results in a well-performed new system with new or improved underlying inference capabilities. However, such approach is not very good at handling situations where knowledge engineers want to use the existing DL systems without touching anything "inside", enhancing them with proper extensions in a task-specific way. These situations provide reasonable motivations for the introduction of our approach.

In this paper, we neither introduce a revised DL-based modelling language nor a new inference algorithm, but a formal scheme which can be applied to existing DL systems. In another word, we focus on a top-up system which can be easily implemented and tailored to a particular application. Meanwhile, we intend to use such a system as the workbench to explore the role of hybrid approach in extending the expressive power of DLs.

Since we intend to keep our approach as portable and generic as possible, we deliberately do not use certain features provided by some DL systems. For instance, the CLASSIC Terminological Knowledge Representation and Reasoning System (TKRRS) [4] allows users to query the domain of a particular role while other TKRRS, such as FaCT [9] do not implement such mechanisms. Therefore, we have decided to avoid using them.

2 Representing concrete domains

The abstract characters of DLs make it difficult to naturally model the knowledge on concrete domains such as arithmetic ones while in certain applications, modelling on the concrete characters such as age, size, weight, etc is necessary.

Including concrete domains into DLs is always a very interesting issue. Previous efforts such as $\mathcal{ALC}$ (D) [2] and $\mathcal{ALCRP}$ (D) [8] allow concrete properties to be referred through functional role chains. For instance, if we want to constrain that the service time of an apprentice is no more than 5 years, we can defined "apprentice" as $\mathsf{Apprentice} \doteq \mathsf{Human} \sqcap \exists \mathsf{employer} \sqcap \mathsf{5year}(\mathsf{len\text{-}of\text{-}serv})$, where $\mathsf{len\text{-}of\text{-}sev}$ is the functional role. The predicate $\mathsf{5year}$ is defined using functions written in the host programming languages. One way to implement it can be: $\mathsf{5year}(x) = \ x \geq 5$. The approach of $\mathcal{ALC}$ (D) provides DLs a perfect means to describe the constraints on concrete domains, however, with high computational complexity [11]. Such situation inspired us to contrive a systems with limited increment on the complexity while obtain the similar expressive power, i.e. in our case, the expressive power of the concrete properties.

In this paper, we focus on concrete domains which are formally defined as followings:

A concrete domain $\mathcal{D}$ *is a pair* $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, *where*

1. $\Delta_{\mathcal{D}}$ *is a set called domain,*
2. $\Phi_{\mathcal{D}}$ *is a set of predicates;*
3. *Each predicate* $P \in \Phi_{\mathcal{D}}$ *is associated with an arity n and an n-ary predicate* $P^{\mathcal{D}} \subseteq \Delta^n_{\mathcal{D}}$.

For the sack of simplicity, we will restrict the predicates to those arithmetic and comparison ones and the domain $\Delta_{\mathcal{D}}$ to a finite one. Nevertheless, we are quite aware that Boolean operators can be introduced easily to form more complex expressions (e.g. meta-constraints). Furthermore, other concrete domains will be considered in successive research.

In this section, we will give the syntax and semantics of our conceptual modelling language. Since we are not restricted to a particular DL, here and in the following, we take $\mathcal{ALC}$ for the demonstration purpose. Nevertheless, such approach can be applied to virtually any DL systems.

Constructor	**Syntax**	**Semantics** (*Interpretation*)
Top (Universe)	$\top$	$\Delta^{\mathcal{I}}$
Bottom (Nothing)	$\bot$	$\emptyset$
Atomic Concept	A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic Role	R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Conjunction	$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
Disjunction	$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
Negation	$\neg\, \mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{D}^{\mathcal{I}}$
Universal quantification	$\forall\, \mathsf{R}.\, \mathsf{C}$	$\{\, c \in \Delta^{\mathcal{I}} \mid \forall\, d \in \Delta^{\mathcal{I}} : \langle\, c, d\, \rangle \in \mathsf{R}^{\mathcal{I}} \rightarrow d \in \mathsf{C}^{\mathcal{I}} \,\}$
Existential quantification	$\exists\, \mathsf{R}.\, \mathsf{C}$	$\{\, c \in \Delta^{\mathcal{I}} \mid \exists\, d \in \Delta^{\mathcal{I}} : \langle\, c, d\, \rangle \in \mathsf{R}^{\mathcal{I}} \wedge d \in \mathsf{C}^{\mathcal{I}} \,\}$

Table 1: $\mathcal{ALC}$ syntax and semantics

Let R *be a role,* H *the constrained element,* P *the defined predicate name. In addition to the syntax of* $\mathcal{ALC}$ *[13] (Table 1),*

1. $\forall \mathsf{R}.\mathsf{H}$ *in a concept, where* H *acts as the bridge between the abstract and the concrete domains;*
2. $P(\mathsf{H}_1, \ldots, \mathsf{H}_n)$ *specifies the constraint on* $\mathsf{H}_1, \ldots, \mathsf{H}_n$; *and* $CE(\mathsf{H}_1, \ldots, \mathsf{H}_n)$ *is the constraint expression which collects all the constraints on* $\mathsf{H}_1, \ldots, \mathsf{H}_n$ *or a subset of them;*

H_i ($i \leq n$) which are referred to as Hybrid Concepts are defined by giving each of them a unique name and mapping it to a subset of $\Delta_{\mathcal{D}}$.

In order to associate abstract and concrete domains, we introduce an assignment function $\lambda : \Delta^{\mathcal{I}} \rightarrow \Delta_{\mathcal{D}}$, i.e. $\lambda(\mathrm{H}^{\mathcal{I}}) \subseteq \Delta_{\mathcal{D}}$ which maps every *Hybrid Concept* to a subset of $\Delta_{\mathcal{D}}$. Thus, the semantics is given as

Let $\cdot^{\mathcal{I}}$ be the interpretation function, $\lambda(\cdot)$ the assignment, and $\Delta^{\mathcal{I}}$ a non-empty domain which is disjoint from $\Delta_{\mathcal{D}}$. In addition to the semantics of $\mathcal{ALC}$ [13] (Table 1),

1. $(\forall R.H)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y.\langle x,\ y\rangle \in R^{\mathcal{I}} \rightarrow y \in H^{\mathcal{I}}\}$;

2. *$P(H_1, \ldots, H_n)$ is satisfied iff*

$$\forall s_1 \in \lambda(H_1^{\mathcal{I}}), ..., \forall s_n \in \lambda(H_n^{\mathcal{I}}).P(s_1\ ...\ s_n);$$

3. *$CE(H_1, \ldots, H_n)$ is satisfied iff*

$$\begin{array}{l}\forall s_1 \in \lambda(H_1^{\mathcal{I}}), ..., \forall s_n \in \lambda(H_n^{\mathcal{I}}).\\ \qquad (\forall P \in CE(H_1, \ldots, H_n).P(s_1\ ...\ s_n)).\end{array}$$

By introducing the assignment function between abstract and concrete domains, we are able to embed concrete knowledge into so-called "*Hybrid Concepts*". Such embedding has non-trivial consequences, as referring to concrete knowledge in concept definitions is no longer required. More specifically, those inference algorithms which are designed to tackle with concrete domains become unnecessary, in that "wrapper concepts" (please refer to Section 3) which are the "incarnations" of *Hybrid Concepts* in DLs are introduced as atomic concepts and treated exactly the same way as normal concepts. In stead of reasoning with *Hybrid Concepts*, system replaces each of them with the wrapper concept. Meanwhile, the responsibility of reasoning with the concrete knowledge represented by *Hybrid Concepts* is delivered to the reasoning systems other than DL ones. For instance, in the above example the apprentice's length-of-service can be modelled as *Hybrid Concepts* represented using universal quantification in the hybrid concept definitions.

Thus, we have apprentices defined as

$$\mathsf{Apprentice} \doteq \mathsf{Human} \sqcap \exists \mathsf{employer} \sqcap \forall \mathsf{len\text{-}of\text{-}serv.Year}_{\mathsf{App}},$$

where $\mathsf{Year}_{\mathsf{App}}$ is *Hybrid Concept*. Moreover, we can specify slightly intricate facts by putting constraints on $\mathsf{Year}_{\mathsf{App}}$. For instance, a specialist is defined as

$$\mathsf{specialist} \doteq \mathsf{Human} \sqcap \exists \mathsf{employer} \sqcap \forall \mathsf{len\text{-}of\text{-}serv.Year}_{\mathsf{Spl}}.$$

Note that: at current stage, we do not restrict the *Hybrid Concepts* to be successors of functional roles (features), but we require that the role value restriction ($\forall$) is used instead of the role existence restriction ($\exists$). Ignoring the latter is motivated by pragmatic considerations. To simulate partial constrained role values, array-type variables may be required. However, partial orders among arrays are not straightforward.

The specifications and restrictions on the *Hybrid Concepts* are given separately as the conjunction of non-DL expressions:

$$\mathsf{Year}_{\mathsf{App}} \leq 5 \wedge \mathsf{Year}_{\mathsf{Spl}} \geq \mathsf{Year}_{\mathsf{App}} + 10.$$

In such an approach, more complex constraints than min and max can be specified. However, such constraints have to be global restrictions which hold universally. In another word, the constraints have to hold on a concept as a group rather than considering each of its instance individually. For instance, as long as one restricts the diameter of a certain type of cylinder, its cross-sectional area is constrained. Also, if we restrict the age of pupils and the average age difference between them and their parents, we constrain the age of the latter to a certain range. Of course, such difference can exist in a much more complex way than "20 years older". Specifying the universal constraints separately, we are granted the freedom to add, delete, change and satisfy them in a batch job.

Apparently, if we want to avoid the complex interaction between DL and non-DL expressions, no available DL system can properly handle such expressions. However, we can hide the actual inferences on the concrete domains from DL systems. More specifically, we allow conceptual hierarchies to be simulated by partial orders which are more general than subsumption relationships between concepts. In our case, with the help of formal numeric systems, we can create hierarchies of *Hybrid Concepts* (more detailed examples can be found in the next section). As a result, DL system is informed of such hierarchies through the subsumption relationship between the wrapper concepts. It then classifies the concepts (such as Apprentice, Specialist) which are built using the wrapper concepts.

Here and in the following, we will use Constraint Programming Languages (CPLs, also referred to as Constraint Solvers, CSs for short, in certain cases) for illustration purposes. Nevertheless, we notice that the actual applications are not restricted to such reasoning engines.

3 Tackling with the "wrapper concepts"

For the sake of simplicity and integrity, we use hybrid concept definitions to capture both abstract and concrete knowledge. However, as pointed in the previous

section, the latter has to be wrapped in order to be properly manipulated. Concepts which contains wrapped concrete knowledge are referred to as normalised concepts. They are defined in a broader sense as followings:

> *A normalised heterogeneous concept definition (**normalised concept**, for short) is the result of a normalisation process that for every occurrence of Hybrid Concepts an atomic concept will be introduced with which the former will be replaced. Such atomic concepts are referred to as "**w**rapper concepts".*

Hybrid concept definitions contained in a *Hybrid Knowledge Base* (HKB) are first analysed and processed by a *parser* which normalises the definitions and generates three sets of statements. These three sets are defined as followings:

- a set of DL-oriented statements which do not exceed the expressive power of the external DL system (warnings will be generated otherwise),
- a set of non-DL statements which express all the concrete knowledge,
- a set of *Hybrid Concepts* which connect DL and non-DL statements.

The hybrid characteristic of our approach is evident in the "polymorphism" of *Hybrid Concepts*. More specifically, *Hybrid Concepts* are represented by wrapper concepts in DL inferential engines while act as legal objects in non-DL reasoning systems, (e.g. constrained variables in CPLs).

For instance, let us suppose that in certain state X married persons are required to be 22-year-old or older. We define concept Married as those "legally married people", Golden-Couples as "people who have already celebrated their golden (50th) anniversary", and Senior-Citizen as "people who are 70 and older". Together with other necessary concepts, we have

$$\begin{aligned} \mathsf{Married} &\doteq \mathsf{Human} \sqcap (\exists \mathsf{has\text{-}spouse.Human}) \sqcap (\forall \mathsf{age.Age_{Mar}}) \\ \mathsf{Golden\text{-}Couples} &\doteq \mathsf{Human} \sqcap (\exists \mathsf{has\text{-}spouse.Human}) \sqcap (\forall \mathsf{married\text{-}year.Year50}) \\ &\quad \sqcap (\forall \mathsf{age.Age_{Gol}}) \\ \mathsf{Senior\text{-}Citizen} &\doteq \mathsf{Human} \sqcap (\forall \mathsf{age.Age_{Sen}}) \end{aligned} \tag{1}$$

Meanwhile, the restrictions on all the *Hybrid Concepts* are given as

$$\begin{aligned} \mathsf{Age_{Mar}} &\geq 22 \\ \mathsf{Year50} &= 50 \\ \mathsf{Age_{Sen}} &\geq 70 \end{aligned} \tag{2}$$

Because the actual constraint exists between two sets of values, rather than write an inequality, we use the relation between sets instead:

$$\mathsf{Age}_{\mathsf{GoI}} - \mathsf{Year50} \subseteq \mathsf{Age}_{\mathsf{Mar}} \tag{3}$$

Now, by normalising the knowledge base we split the above definitions and restrictions into three parts. First, we replace all the *Hybrid Concepts* with "wrapper concepts" and adding new atomic concepts, $\mathsf{Age}_{\mathsf{Mar}}$, $\mathsf{Age}_{\mathsf{GoI}}$, $\mathsf{Age}_{\mathsf{Sen}}$, and Year50 into the DL part. Second, all "ages" acting as constrained variables are stored in the non-DL part together with their default domain $[0..100]$[1] and the constraints defined in (2). Assuming that domain() is the assignment function, we can specify for each *Hybrid Concepts*:

$$\begin{aligned} &\text{non-DL part:} \\ &\qquad \mathsf{domain}(\mathsf{Age}_{\mathsf{Mar}}) = [0..100] \\ &\qquad \mathsf{domain}(\mathsf{Age}_{\mathsf{GoI}}) = [0..100] \\ &\qquad \mathsf{domain}(\mathsf{Age}_{\mathsf{Sen}}) = [0..100] \\ &\qquad \ldots \end{aligned} \tag{4}$$

Hybrid Concepts are also stored separately in the so-called "Link Pool", (see Figure 2 for the "Link Pool").

The above statements are translated into the underlying modelling languages of the external inferential engines. Such translations are carried out so as to keep our approach portable and implementation-independent. Subsequently, translated statements are loaded into external DL and CPL inferential engines. According to the results provided by these different engines, a reasoning coordinator (see Figure 3) creates hierarchical structures of *Hybrid Concepts*, which are then introduced into DL definitions through the "wrapper concepts".

In our example, after loading the non-DL part (2)+(3)+(4) into an external constraint solver, we obtain the reduced domains (e.g. $\mathsf{domain}(\mathsf{Age}_{\mathsf{GoI}}) = [72..100]$). Using the results from both DL and non-DL inferential engines, we create a new partial order among "ages" e.g. $\mathsf{domain}(\mathsf{Age}_{\mathsf{GoI}}) \subseteq \mathsf{domain}(\mathsf{Age}_{\mathsf{Sen}})$. Therefore, the corresponding subsumption relationships can be specified between wrapper concepts (e.g. $\mathsf{Age}_{\mathsf{GoI}} \sqsubseteq \mathsf{Age}_{\mathsf{Sen}}$). Sending such information back to join the original DL definitions contained in the DL part in (1), we can conclude that, among other conclusions, in the state X, people who have already celebrated their 50th anniversary are all senior citizens, i.e.

$$\mathsf{Golden\text{-}Couples} \sqsubseteq \mathsf{Senior\text{-}Citizen}.$$

[1] We assume that human can not live out 100.

Thinking globally, such conclusion is evident in the sense that although people get married and celebrate their Golden Anniversary (if there is one) in different ages, the ages of golden couples will always fall into the given range.

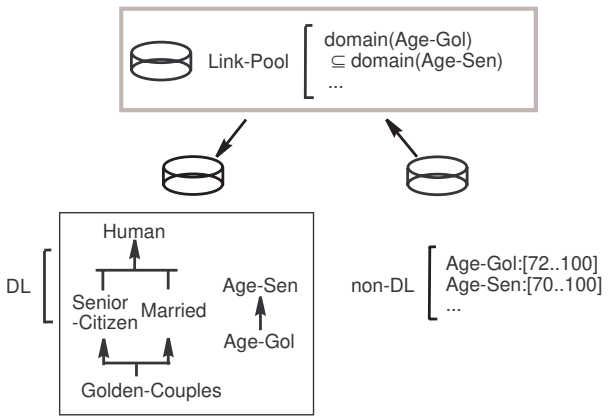

Figure 1: Reasoning with of "ages"

An obvious result of the above reasoning process is that, by satisfying the constraints, we explicitly express some knowledge which otherwise, remains implicit. In our case, by satisfying the constraints, our system can answer the query that "Is Golden-Couples a Senior-Citizen". Although the above "ages" example is a relatively simple one, more complex hybrid knowledge can be represented and processed using the same approach.

4 System Architecture

Since the heterogeneous knowledge used in our approach is evenly split into two separate "homogeneous" components, we use a hybrid architecture to provide the overall inference services. In our system (see Figure 2), the external DL inferential engine and the non-DL one (e.g. a constraint solver in our case) are used in a "peer to peer" way, i.e. neither of them acts as a client or a server with regard to the other one.

However, several points need to be remarked. First, to ensure the portability of our system, we introduce a decoupling between the DL and non-DL representations on one side and the actual inferential engines on the other. More specifically, the transformation process from a hybrid description to the one accepted by the selected DL or non-DL system is divided in two stages. During a first stage, the hybrid language is split into its two homogeneous DL and non-DL components. During a second stage, the two resulting components is actually expressed in an implementation-independent "intermediate" language.

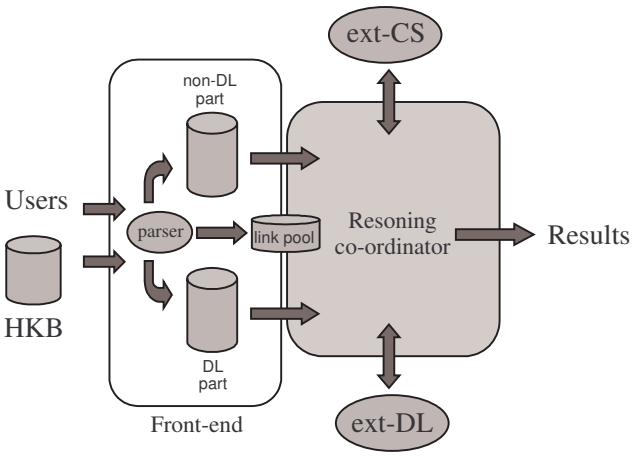

Figure 2: Hybrid architecture

Each of the two intermediate descriptions are subsequently transformed into the ones accepted by the adopted inferential engines. In this way, we decouple our system from the reasoners, thus allowing their modular replacement if necessary. Second, a *link pool* is created to store related data about each newly created "wrapper concept", such as name, position, and so on. This information can be used when "wrapper concepts" must be reclassified according to the results provided by the external inferential engines. Third, various systems can be used as external inferential engines. In this paper, we only analyse the situation where CPLs are selected to reason with numeric restrictions (such as $\mathsf{Age}_{\mathsf{GoI}} - \mathsf{Year50} \subseteq \mathsf{Age}_{\mathsf{Mar}}$ in our example) for the demonstration purposes.

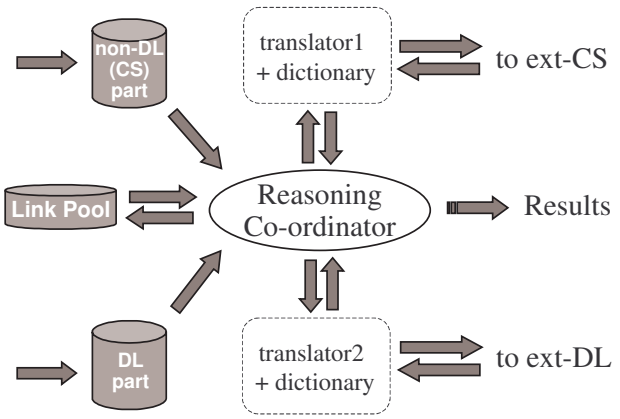

Figure 3: Reasoning Co-ordinator

5 Conclusions

We give an overview of a novel approach to extend the expressive power of existing DLs. This approach is based on a hybrid reasoning process. Thus, we also propose an architecture to support it.

The advantages of a hybrid system are system portability, extensibility, and robustness. Although similar approaches have been already proposed in the past (e.g. TexLog [1]), our approach benefits from the heterogeneous characteristics of *Hybrid Concepts*, which have a consistent semantics on both DL and non-DL (e.g. CPLs) sides. Such a bridging facility also differentiates our approach from other apparently similar ones.

At current stage, we haven't examined the complexity issue thoroughly. Nevertheless, there is great potential to optimized our system by tailoring the DL and CPLs inferential engines to particular applications. Such character is enhanced by selecting the external inferential engine with the right expressive power. In practice, the overall performance of our system can be considered separately. On the one hand, since we do not introduce any new constructors–only new concepts, we expect that the complexity of the DL inference will be in the same class as the original system. Moreover, we avoid the complex interaction between abstract and concrete domains by introducing the latter through "wrapper concepts". On the other hand, the Finite Constraint Satisfaction Problems (FCSPs) are NP-complete as a general class [12]. Pragmatic results shows that the performance varies from system to system [7]. However, selecting the suitable CPLs is not the concern of this paper.

Because of the adopted hybrid approach, the overall performance of our system could be influenced by the various translation and interfacing processes. This might be particularly evident if several different protocols must be used to link each of them to the other as well as to the controller. This drawback may not be a serious problem in all those cases which are not time-critical and in which reasoning with heterogeneous knowledge is necessary. However, further analysis on this issue is necessary.

References

[1] Andreas Abecker and Holger Wache. A layer architecture for the integration of rules, inheritance, and constraints. In *ICLP Workshop: Integration of Declarative Paradigms*, pages 12–22, 1994.

[2] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence IJCAI-91*, pages 452–457. Morgan Kaufmann Publ. Inc., San Mateo, CA, 1991.

[3] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 2001. To appear.

[4] Ronald J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, and A. Borgida. "Reducing" CLASSIC to practice: Knowledge representation theory meets reality. *Artificial Intelligence 114*, pages 203–237, 1999.

[5] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending RDF schema. In *Proceedings of the tenth World Wide Web conference WWWW'10*, 2001. To appear.

[6] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Foundation of Knowledge Representation*, pages 191–236. CSLI-Publications, 1996.

[7] A. Fernández and P. M. Hill. A comparative study of eight constraint programming languages over the Boolean and finite domains. *Journal of Constraints*, 5:275–301, 2000.

[8] Volker Haarslev, Carsten Lutz, and Ralf Möller. A description logic with concrete domains and role-forming predicates. *Journal of Logic and Computation*, 9(3), 1999.

[9] I. Horrocks. FaCT and iFaCT. In *Proc. of the Int. Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[11] C. Lutz. NExpTime-complete description logics with concrete domains. In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, number 2083 in Lecture Notes in Artifical Intelligence, pages 45–60, Siena, Italy, 2001. Springer Verlag.

[12] Alan K. Mackworth and Eugene C. Freuder. The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59(1–2):57–62, 1993.

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