Facilitating DL-based Hybrid Reasoning with Inference Fusion

Bo Hu[1], Inés[1] Arana and Ernesto Compatangelo[2]

[1] School of Computing, The Robert Gordon University,

Aberdeen AB25 1HG, Scotland

Tel: +44 1224 262702, Fax: +44 1224 262727

[2]Department of Computing Science, University of Aberdeen

Aberdeen AB24 3UE, Scotland

Tel: +44 1224 272397, Fax: +44 1224 273422

Abstract

We present an extension to DL-based taxonomic reasoning by means of the proposed *inference fusion*, i.e. the dynamic combination of inferences from distributed heterogeneous reasoners. Our approach integrates results from a DL-based system with results from a constraint solver under the direction of a global reasoning coordinator. *Inference fusion* is performed by (i) processing heterogeneous input knowledge, producing suitable homogeneous input knowledge for each specialised reasoner; (ii) activating each reasoner when necessary, collecting its results and passing them to the other reasoner if appropriate; (iii) combining the results of the

two reasoners. We discuss the benefits of our approach and demonstrate our ideas by proposing a language ($\mathcal{DL}$(D)/S) and a reasoning system (CONCOR) which uses knowledge bases written in $\mathcal{DL}$(D)/S and supports hybrid reasoning. We illustrate our ideas with an example.

Keywords: Description Logics, Hybrid Reasoning, Constraint Reasoning.

1 Motivation and background

The development of languages understandable by both human and machines is central to the *semantic web* [1]. Thus, research in this area has increased the interest of ontological modelling and reasoning. Current approaches to ontology reasoning during the knowledge lifecycle management are based on a wide variety of structured knowledge models, each enabling different automated capabilities. Some models, such as UML [2], represent knowledge at the conceptual level. Unfortunately, most of them have an ill-defined semantics and thus do not enable any semantic deduction. Frame-based models like Protégé [3] represent knowledge at the epistemological level (i.e. they use the two generic primitives class and role). Although these models enable automated inferences such as class membership, they do not support deductions such as subsumption. Finally, models based on Description Logics (DLs) like OIL/DAML+OIL [4] broaden the spectrum of frame-based inferences with a whole set of specialised deductions based on taxonomic reasoning.

Deductive services provided by DL-based reasoners include, among others, semantic consistency check subsumption, and concept classification [5]. DL-

based approaches are particularly appealing for applications such as ontology reasoning in the Semantic Web, where taxonomic reasoning has been recognised as one of the core inferences [4]. Moreover, DLs use the notions of concept (i.e. unary predicate) and role (i.e. binary relation) to model declarative knowledge in a structured way. 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, the expressive power needed to model complex real-world ontologies is quite high, so that ontology reasoning was initially ruled out of the list of services to be provided by ontology management tools [6]. Nevertheless, the OIL/DAML+OIL effort has re-introduced the issue of ontology reasoning as a first-class problem, providing a solution within the framework of a DL-based, frame-centred approach [4]. However, despite its expressivity, the OIL/DAML+OIL approach does not yet provide practical support to reasoning with concrete domains or local constraints (i.e. role-value maps). This is because the knowledge model of the iFaCT DL engine [7], which provides the deductive services for the ontology inference layer, does not currently include concrete domains or role-value maps.

During the last few years, much research has been devoted to the development of more powerful reasoning systems, Although single-purposed reasoning systems have improved substantially, their homogeneous approaches are limited in two ways: (i) the expressive power of their representation is restricted in

order to ensure computational tractability, completeness and decidability; (ii) the specialist nature of their reasoning means that they are only successful at carrying out particular inferential tasks. For instance, DL-based systems specialised in the construction of concept taxonomies from concept descriptions while constraint programming tools solve constraint problems. Although in the past there has been some research on the integration of hybrid homogeneous reasoning [8], little has been done on the integration of heterogeneous reasoning, e.g. the integration of DL-based and constraint-based reasoners.

Some approaches have been proposed to include *concrete domains*—and predicates on these domains—in DL-based concept definitions which are normally restricted to *abstract domains*. Despite the diversity of their representations, most of them are based on $\mathcal{ALC}$ [9] and its expressive successor $\mathcal{SHIQ}$ [10] and extend the original tableau-based algorithm [9] in different ways. It has been proved, however, that reasoning about extensions of $\mathcal{ALC}$ with concrete domains is generally intractable [11]. This problem can be mitigated only if suitable restrictions are introduced in the way of combining concept constructors [12].

Homogeneous reasoning systems (or systems with homogeneous inference algorithms) have encountered the difficulty of finding the right "trade-off" between expressiveness and computational complexity. We believe that if a knowledge model is too expressive to be analysed within the framework of DLs, then other representation and reasoning paradigms must be jointly used. Therefore, it's reasonable to consider a that a hybrid approach to heterogeneous knowledge management may provide, among other things, a wider and better support to

ontology reasoning. The benefits of such an approach in the context of ontology sharing through the articulation of ontology interdependencies is highlighted in [13].

In this paper, we thus present a generic schema to extend existing DL-based systems with the ability of representing and reasoning with numeric constraints. Our idea is materialised through a hybrid modelling language $\mathcal{DL}$(D)/S, and supported by an implemented hybrid reasoning system (HRS), CONCOR.

2 Practical approach for hybrid reasoning

Inference fusion is a generic schema for dynamically integrating heterogeneous inferential engines [14]. More specifically, we focus on a particular class of *inference fusion*-based HRSs, which fuse the T-Box deductions from a DL-based taxonomic reasoning system with constraint satisfaction inferences from a constraint solver under the direction of a global reasoning coordinator.

In order to ensure the autonomy of both inferential sub-systems (hereafter, referred to as engines), there should be a reliable mechanism responsible for the communicating between them. For this purpose, we introduce the bijection, *linkage*, which is responsible for mapping the intrinsic data structures in the DL-based system to the data structures in the constraint solver and *vice versa*. *Linkages* ensure that (i) the results from one system can be fed into the other system without increasing the original computational complexity of these two systems; and (ii) no changes are required on either reasoning system, i.e. the

underlying inference algorithms remain unchanged.

A Hybrid Knowledge Base (HKB), denoted as Π_{KB}, is first processed by a *parser* which fragments the descriptions and splits them into three sets, namely: (i) Π_{DL}, i.e. a set of DL-oriented statements which do not exceed the expressive power of the selected DL-based system, (ii) $\Pi_{\text{non-DL}}$, i.e. a set of non-DL statements which contains the concrete knowledge filtered out to form Π_{DL}, and (iii) $\Pi_{linkage}$, i.e. a set of *linkages* which are one-to-one relations connecting DL and non-DL statements.

Π = pi

As a result, instead of reasoning with constraints 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 from concept definitions. Thus, only the proper DL-based constructors which are admitted by the selected DL-based inferential engines are left.

The reasoning results from the non-DL system are reflected into the DL one using *linkages*. Therefore, the hybrid characteristics of our approach are evident in the "polymorphism" of *linkages* which are regarded as atomic concepts in the DL-based inferential engine while act as legal objects in the non-DL reasoning system (e.g. constrained variables in CSs).

For instance, let's assume that, in state X, all people participating in legal marriages should be at least 22 years old. In the meantime, only those who are older than 70 are counted as senior citizens. Amongst the married people, couples who have already celebrated their golden wedding anniversary should have been married for at least 50 years. The set of concepts and global con-

straints for this domain is as follows: Married-person who is between 22 and 100[1], Golden-couples who have been married for at least 50 years, and Senior-citizens who are between 70 and 100. Because of the difficulty of carrying out real calculations, e.g. addition of X+Y, DL-based systems may not be able to detect that a person who belongs to Golden-couple is also a Senior-citizen.

Our approach can facilitate such reasoning by splitting and redirecting knowledge to specialised reasoners. In the above example, a series of AGE_x will be defined as constrained variables with specified domains, e.g. 0..100. *Linkages* map a concrete variable AGE_x (used by the constraint solver) to an abstract atomic concept Age_x (referred in the DL concept definitions of Married-person, Golden-couple and Senior-citizen). Thus, the reasoning results w.r.t. AGE_x from the constraint solver are fed into the DL-based system. Subsequently, the subsumption relationship between Senior-citizen and Golden-couple can be detected by the DL-based (taxonomic) reasoning system.

3 Hybrid DL-based modelling with $\mathcal{DL}(\mathbf{D})/\mathbf{S}$

In this section, the hybrid modelling language $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ is proposed to illustrate the applicability of *inference fusion* in extending the DL-based systems. $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ extends $\mathcal{ALC}$ with various types of concrete constraints. Note that, because of the generic characteristics of *inference fusion* and the common availability of *linkages* in DLs, the use of $\mathcal{ALC}$ is not mandatory, i.e. other DLs

[1] We assume that the life span of human being does not exceed 100 years

could have been used for our purposes.

3.1 Syntax and Semantics of $\mathcal{DL}(\mathbf{D})/\mathbf{S}$

$\mathcal{ALC}$ concepts are built as follows. Let $\mathcal{A}$ be the set of concept names, $\mathcal{C}$ the set of arbitrary concept descriptions, $\mathcal{R}$ the set of role names and n an arbitrary non-negative integer. Starting with (i) $\mathsf{A} \in \mathcal{A}$, (ii) C, $\mathsf{D} \in \mathcal{C}$ and (iii) $\mathsf{R} \in \mathcal{R}$, concept terms can be defined inductively. A *concept definition* is either $\mathsf{A} \dot{\sqsubseteq} \mathsf{C}$ (partial definition) or $\mathsf{A} \doteq \mathsf{C}$ (full definition). An interpretation $\mathcal{I}$ for $\mathcal{ALC}$ is a couple $(\ \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$: the nonempty set $\Delta^{\mathcal{I}}$ is the domain of $\mathcal{I}$, while the $\cdot^{\mathcal{I}}$ function maps each concept to a subset of $\Delta^{\mathcal{I}}$ and each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation of $\mathcal{ALC}$ constructors is shown in Table 1.

Δ = delta

$\mathcal{ALC}$ has been extended with the ability to describe concrete knowledge. For instance, $\mathcal{ALC}$(D) [15] extends $\mathcal{ALC}$ with constructors allowing the definition of predicates over functional roles and role chains. Sound and complete algorithms for $\mathcal{ALC}$(D) exist for A-Box reasoning provided that $\mathbf{D}$ is an *admissible* concrete domain, e.g. $\mathcal{N}$ [15]. $\mathcal{SHOQ}$(D) extends $\mathcal{ALC}$ with constructors for *concrete datatypes* used to represent numbers and strings [16]. Sound and complete algorithms exist for reasoning in $\mathcal{SHOQ}$(D) provided that suitable restrictions are introduced [15]. Meanwhile, substantial efforts have been made on the implementations, e.g. $\mathcal{ALCRP}$(D) [17] and RACER [18].

Despite the difference in expressive and deductive powers, traditional approaches which extend DLs have concentrated on enhancing the algorithm originally devised for $\mathcal{ALC}$ [9], i.e. create a tableaux containing both concept con-

structors and constraint predicates, during which process, the complex intervention of abstract and concrete knowledge is inevitable. Thus, adding concrete domains (e.g. numeric constraints) directly to expressive DL-based systems may result in undecidable inferential problems [11].

We introduced the hybrid modelling language $\mathcal{DL}$(D)/S (Table 2) in order to extend DLs with concrete domains while avoiding a significant increase in the computational complexity of the DL-based systems [14]. The concrete domain is formally defined as a pair $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a finite set of numeric and symbolic constants and $\Phi_{\mathcal{D}}$ a set of algebraic and boolean operators. Φ = phi

Here, $rel \in \{=\}$, H is a hybrid concept, v an integer type variable and $\xi[v]$ the set of role cardinality constraints defined over v; λ' an assignment mapping v to a set of non-negative integers. The constraints are, therefore, specified through hybrid role successors (*hybrid concept*) H or role cardinality variables $v_1, \ldots, v_n$. The following concept contains a numeric constraint which restricts the number of airpads (α) to be twice the number of axis (β): α = alpha, β = beta

exists(α, β) (**and** Machine_Tool (**equal** α has-axis) (**equal** β has-airpad)\

(: **with** : **begin** $\alpha = 2\beta$: **end**))

3.2 Constraints in $\mathcal{DL}$(D)/S

Both global constraints over hybrid role successors and local constraints on role cardinalities are allowed in $\mathcal{DL}$(D)/S. A $\mathcal{DL}$(D)/S knowledge base ($\mathcal{DL}$(D)/S-KB) is represented as $\Omega = \mathcal{T} + \Psi$, where $\mathcal{T}$ is the set of concept definitions and multi-concept relationships (e.g. subsumption and disjointness among concepts) Ω = omega, Ψ = psi

and Ψ (short for $\Psi[\mathsf{H}_1,\ldots,\mathsf{H}_n]$) is the set of all global constraints $\psi[\mathsf{H}_1,\ldots,\mathsf{H}_n]$ defined over $\mathsf{H}_1,\ldots,\mathsf{H}_n$ or a subset of them. ψ = psi

Let $\lambda(\mathsf{H}^{\mathcal{I}}) \subseteq \Delta_{\mathcal{D}}$ be the assignment function which creates a concrete image (a concrete variable with associated domains) for an hybrid concept and assign a subset of $\Delta_{\mathcal{D}}$ to the concrete image, and $\lambda'(v_i) \rightarrow t_i \in \mathcal{N}$ mapping v_i a non-negative integer. We have: λ = lambda

$$
\begin{aligned}
&(\psi(\mathrm{H}_1,\ldots,\mathrm{H}_n))^{\mathcal{I}} \equiv \mathtt{sat}(\psi(\mathrm{H}_1,\ldots,\mathrm{H}_n)) \equiv \\
&\textstyle\bigwedge_{i=1}^{n} \forall x_i \in \lambda(\mathsf{H}_i^{\mathcal{I}}).(\exists y_1 \in \lambda(\mathsf{H}_1^{\mathcal{I}}),\ldots,\exists y_{i-1} \in \lambda(\mathsf{H}_{i-1}^{\mathcal{I}}), \\
&\quad \exists y_{i+1} \in \lambda(\mathsf{H}_{i+1}^{\mathcal{I}}),\ldots,\exists y_n \in \lambda(\mathsf{H}_n^{\mathcal{I}}).\psi(y_1,\ldots,y_{i-1},x_i,y_{i+1},y_n))
\end{aligned}
$$

i.e. for every value of the concrete image of H_i, there exist values in every H_j $(j = 1\ldots n, j \neq i)$ such that predicate ψ holds. Meanwhile, the collection $\Psi[\mathrm{H}_1,\ldots,\mathrm{H}_n]$ is satisfied $\mathtt{sat}(\Psi[\mathrm{H}_1,\ldots,\mathrm{H}_n])$ *iff*

$$\mathtt{sat}(\Psi[\mathrm{H}_1,\ldots,\mathrm{H}_n]) \equiv \forall\psi \in \Psi[\mathsf{H}_1,\ldots,\mathrm{H}_n].\mathtt{sat}(\psi)$$

A $\mathcal{DL}$(D)/S-concept C/ξ (where ξ is the set of concept-local role cardinality constraints that may be empty) is satisfiable w.r.t. $\xi[v_i]$ *iff* there is an assignment λ' such that $\mathsf{C}[\lambda'(v_i)] \neq \emptyset$ and $\xi[\lambda'(v_i)]$ hold for $i = 1\ldots n$ where $\lambda'(v_i) \rightarrow t_i \in \mathcal{N}$: ξ = xi

$$(\exists v_i.\mathsf{C}[v_i]/\xi[v_i])^{\mathcal{I}} = \exists t_i.(\mathsf{C}^{\mathcal{I}}[t_i] \neq \emptyset \wedge \xi[t_i]) \quad (i = 1\ldots n)$$

Our hybrid concepts capture both abstract knowledge and RC constraints. However, constraints need to be "wrapped" as they cannot be directly processed by a DL-based system. Concepts containing wrapped RC constraints are said

to be *normalised*.The normalisation of $\mathcal{DL}$(D)/S concepts:

Global constraints: (i) Generating a atomic concept for each hybrid concept H, (ii) creating a mapping between H and the corresponding constrained variable and (iii) removing all global constraints is as:

RC-constrained concept: (i) Replacing every sub-concept containing constraints on role cardinalities with an existential role restriction; (ii) Introducing an atomic concept for every set of constraints on role cardinalities; (iii) Removing the existential restrictions on RC variables and eliminating RC constraints by conjuncting atomic concepts at the same logical level;

Non-RC-constrained concept: If the concept is defined with the RC constrained roles acting as the subject of numeric role cardinality restrictions, (i) creating an existential role restriction to replace every sub-concept referring to RC-constrained roles; (ii) generating a set of numeric constraints to represent the numeric role cardinality restrictions; (iii) defining an atomic concept into the HKB and conjuncting it to the original concept at the same logical level. Concepts will not be changed otherwise.

For instance, the previous Machine_Tool example is transformed into

(**and** Machine_Tool (**some** has-axis) (**some** has-airpad) C1_axis-pad)

where the RC constraint (i.e. $\alpha = 2\beta$) is replaced by C1_axis-pad introduced as

an atomic concept. Meanwhile, if a concept in the same HKB is defined as

(**and** Machine_Tool (**atleast** 4 has-axis) (**atmost** 4 has-axis))

it will be normalised as:

(**and** Machine_Tool (**some** has-axis) C2_axis-pad)

where the RC constraints (e.g. $\{|\text{has-axis}| \leq 4\}$) is extracted and replaced by C2_axis-pad because that the same roles (i.e. has-axis and has-airpad) have been restricted by RC constraints in other concepts from the same HKB.

If we define that all concepts contains roles that restricted by RC constraints as RC related concept, then:

1. If two concepts C and D are RC related concepts (i.e. ξ may be empty but the concept contains roles restricted by RC constraints), the subsumption relationship between $\mathcal{DL}$(D)/S concepts is defined as follows:

   - *let C' and D' be the normalised concept definitions of C and D;*
   - *let* ξ'_C *(*ξ'_D*) be the union of original RC constraints* ξ_C *(*ξ_D*) and those generated from the normalisation of concept C (D).*

   Then, $D/\xi_D \sqsubseteq C/\xi_C$ *if concept D' is subsumed by C', i.e.* $D \sqsubseteq C$ *and constraint set* ξ'_D *entails constraint set* ξ'_C *in model* Σ*, namely* $\xi'_D \models_\Sigma \xi'_C$.

2. If otherwise the normal DL-based reasoning will be carried.

The *hybrid concept* is similar to the *concrete datatype* in $\mathcal{SHOQ}$(D) [16].

However, our approach differs from the latter in three aspects. Firstly, all the concept constructors are interpreted solely in abstract domains; associations between abstract and concrete domains are realised by an assignment function through hybrid concepts. Secondly, more complex global constraints can be modelled using role value constraints. Finally, the overall inferential process is distributed across different (specialised) engines and thus the overall complexity of the reasoning task may be reduced.

4 Hybrid reasoning with constraints

Our *linkages* are based on two observations. Firstly, DL-based systems can specify subsumption relationships between concepts (the "told" knowledge). For instance, in the iFaCT system [7], one can specify concept A to be subsumed by concept B as (implies A B). Most other DL systems such as LOOM [19] and RACER [18] have the same functionality.

Secondly, it is possible to obtain an ordering (e.g. *quasi-ordering* [20]) with the help of constraint solvers[2]. For instance, the entailment between two set of constraints can be seen as an ordering.

[2]Currently, Constraint Logic Programming (CLP) languages have been extend with the ability to tackle with different domains of computation, e.g. Boolean algebra, finite domains, *etc.* and, for part of these domains provide the decision about consistency and entailment of constraints (please refer to [21] for a detailed survey).

4.1 Ordering of constraints

When domain reduction can be carried out thoroughly and the constraint system can reach a stable status, the inclusion relationships between reduced domains are passed to the DL-based system. Such approach applies to cases when (i) variable domains exist independently; (ii) their images in DL-based systems can be extracted from the rest of a KB and (iii) the extracted knowledge can be referred to as an independent object in the KB. For instance, the life-span of human beings whose domain is $0 \ldots 150$ can be isolated from others easily and defined as a atomic concept in a DL-based knowledge base.

When constrained variables appear as the role number restrictions, the domain reduction technique is not applicable. Because constraints can be considered as the set of tuples of legal values that the constrained variables can take simultaneously [22], an inclusion between different sets of tuples can actually be established and manipulated.

The relationship obtained among concrete constraints is described by a *quasi-ordering*. A formal definition on the new concept, *quasi-ordering*, is introduced as follows:

Let α and β be the sets of compound labels (tuples). We say that α is prior to β in a *quasi-ordering* with regard to a model Σ, if every tuple in β also exists in α, i.e. $\beta \models_\Sigma \alpha$. In this case, we also say that β is tighter than α.

$\Sigma = \text{sigma}$

Note that the ordering among constraint sets is a partial ordering as it is reflexive, transitive and anti-symmetric. In cases when such ordering are mutual, α and β are equivalent.

Constraints in $\mathcal{DL}$(D)/S-KB are manipulated in two ways. Global role value constraints are removed in the sense that the same restrictions can be achieved by reducing the domains of constrained objects (i.e. maintaining a path consistency among the associated constrained domains of concrete images of the *hybrid concepts*). On the contrary, local RC constraints are enhanced by explicitly expressing the restrictions which are otherwise implicit (i.e. discover the entailments ordering and the disjointness).

4.2 Hybrid reasoning system, CONCOR

CONCOR is composed of four major parts: *engine interface*, *user(and KB) interface*, *internal storage* and *reasoning coordinator* which is at the heart of CONCOR. Hybrid knowledge is input into CONCOR through the *user(and KB) interface*. The *user(and KB) interface* contains a parser which checks the inputs for errors such as illegal syntax and invalid constructors, i.e. those constructors that are not admitted by the selected inferential engines. Well-formed concept descriptions are normalised and translated into intermediate forms and split into non-DL, DL and *linkage* pools which are referred to as *internal storage*.

Having parsed the input HKB, the *user(and KB) interface* passes control to the *reasoning coordinator* and the latter will decide which subsequent-inferential engines (SIEs) the contents of the *internal storage* should be sent to. Communications between the *reasoning coordinator* and the SIEs are carried out through the *engine interface*. An *engine interface* associated to an engine is responsible for transferring the data and control flows to this particular engine. The *en-*

gine interface helps to design CONCOR system in a modular manor: SIE can be replaced together with its interface, thus, theoretically the effect of exchanging SIE will not ripple off to other parts of the HRS.

The modular manor of CONCOR system is further guaranteed by introducing an intermediate language between user language and the underlying modelling languages of the selected SIEs. The intermediate language allows a standard translator to be designed for each inferential engine. It also reduces the programming tasks on any further extensions to the modelling language—only the parser residing in the User Interface need to be upgraded. Meanwhile, because the intermediate modelling language has a well-formed semantics, engine interfaces can actually be developed off-line with the help of certain tools.

CONCOR's reasoning process is as follows:

1. parse the input HKB and split it into small homogeneous parts: DL, non-DL (global and concept-local constraints), and *linkage*;
2. check the consistency of global constraints and propagate them in order to maintain a full path-consistency by reducing the set of possible values associated with each constrained variable;
3. update DL-based descriptions with the *quasi-ordering* (domain inclusion) between constrained variables;
4. check the consistency of concept-local RC constraints w.r.t. each individual RC constrained concept;
5. obtain *quasi-ordering* (entailment ordering) among all RC constraint sets;

6. update and classify the DL-based descriptions based on the new knowledge (*quasi-ordering*).

4.3 Hybrid reasoning with examples

We will use a toy example to demonstrate the merits of our hybrid approach. Assume that an estate agency X maintains a database of floor plans. Each design contains certain types of constraints on the number and style of rooms. The HKB is as follows:

(**def-primconcept** 'Floorplan 'top)

(**def-role** 'has_room) (**def-role** 'has_bathroom)

(**def-role** 'has_bedroom) (**def-role** 'has_internet_plug)

(**def-role** 'has_phone_plug)

(**decl-variable** 'Shape_SBaD [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

(**decl-variable** 'Shape_SBeD [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

(**decl-variable** 'Shape_SBaH [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

(**decl-variable** 'Shape_SBeH [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

(**decl-variable** 'Shape_SBaE [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

(**decl-variable** 'Shape_SBeE [`square`, `rect`, `oval`, `rhomb`, `cir`, `tri`])

```
(def-concept    'Ensuit_Design    '(exists (x y z)
                                     (and Floorplan
                                       (equal z has_rooms) (equal x has_bedrooms)
                                       (forall has_bedrooms Style_bed_Ensuit)
                                       (equal y has_bathrooms)
                                       (forall has_bathrooms Style_bath_Ensuit))
                                     (with :begin
                                       :body
                                         x = y, z ≥ y + x + 1
                                       :end) ))

(def-concept    'Residence_Design    '(exists (r be ba)
                                       (and Floorplan (equal r has_rooms)
                                         (equal be has_bedrooms)
                                         (equal ba has_bathrooms)
                                       (with :begin
                                         :body
                                           r > be + ba
                                         :end) ))
```

(**def-concept** 'Hitech_Design '(**exists** (x y z $n1$ $n2$)

(**and** Floorplan

(**equal** x has_rooms) (**equal** z has_phone_plug)

(**equal** y has_internet_plug)

(**equal** $n1$ has_bathrooms)

(**forall** has_bathrooms Style_bath_Hitech)

(**equal** $n2$ has_bedrooms)

(**forall** has_bedrooms Style_bed_Hitech))

(**with** :**begin**

:**body**

$x > n1 + n2,\ y = z,\ y = x$

:**end**)))

(**def-concept** 'Dorm_Design '(**exists** (x y z)

(**and** Floorplan

(**equal** x has_rooms) (**equal** y has_bedrooms)

(**forall** has_bedrooms Style_bed_Dorm)

(**equal** z has_bathrooms)

(**forall** has_bathrooms Style_bath_Dorm))

(**with** :**begin**

:**body**

$x > y + z,\ y = z$

:**end**)))

```
(def-concept    'Modern_Design    '(exists (r pl)
                                     (and Residence_Design
                                       (equal r has_rooms) (equal pl has_phone_plug)
                                     (with :begin
                                       :body
                                         r = pl
                                       :end) ))

(def-concept    'Style_bath_Dorm    '(and room (fallin shape Shape_SBaD) ))
(def-concept    'Style_bed_Dorm     '(and room (fallin shape Shape_SBeD) ))
(def-concept    'Style_bath_Hitech  '(and room (fallin shape Shape_SBaH) ))
(def-concept    'Style_bed_Hitech   '(and room (fallin shape Shape_SBeH) ))
(def-concept    'Style_bath_Ensuit  '(and room (fallin shape Shape_SBaE) ))
(def-concept    'Style_bed_Ensuit   '(and room (fallin shape Shape_SBeE) ))

(decl-constraint    'RoomShape    :with :BEGIN
                                  :BODY
                                    Shape_SBaD=[square, rect, rhomb ],
                                    Shape_SBeD=[square, rect, rhomb ],
                                    Shape_SBaE\=[cir, oval, tri, rhomb ],
                                    Shape_SBeH=[square, rect, oval ],
                                    Shape_SBaE=Shape_SBeE,
                                    disjoint(Shape_SBaH, Shape_SBeH)
                                  :END)
```

Reasoning about the above HKB with traditional DL-based systems may be either (i) possible but at the price of computational complexity, e.g. reasoning about the individual shapes; or (ii) not feasible, e.g. the reasoning with constraints on role cardinalities.

After the hybrid reasoning, a series of nontrivial conclusions can be drawn from the above example as:

$$\begin{aligned} \mathsf{Ensuit\_Design} &\sqsubseteq \mathsf{Dorm\_Design} \\ \mathsf{Dorm\_Design} &\sqsubseteq \mathsf{Residence\_Design} \\ \mathsf{Hitech\_Design} &\sqsubseteq \mathsf{Morden\_Design} \end{aligned}$$

5 Conclusions and future work

We have presented a new approach which extends taxonomic (DL-based) systems by combining the results of existing non DL-based reasoning systems. This approach aims at enabling *inference fusion* by dividing an input hybrid KB into smaller components, each containing the knowledge that can be processed by a different specialised reasoning system. Results of inferences are then fused.

Benefiting from the use of independent inferential engines and the polymorphous *linkages* which are required to have consistent semantics within different systems, our approach to *inference fusion* does not depend on a specific DL-based system or constraint solver.

In order to demonstrate the feasibility and applicability of our ideas, we have presented a hybrid modelling language, $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ which extends $\mathcal{ALC}$ and illustrated its usage in the context of *inference fusion* by means of an example.

The CONCOR architecture is proposed as the platform of carrying out *inference fusion* which has several advantages, such as system extensibility, simplicity and component isolation. Implementation of CONCOR system is completed which fuses inferences from the iFaCT DL-based system [7] and the Ecl^{i}ps^e

CS [23]. Small test cases have been reasoned about by CONCOR system giving promising results. Although no formal analysis of the CONCOR system has been made, its complexity can be estimated as follows:

- DL system: since we do not explicitly introduce any new type of reasoning or new concept constructors or operators, the complexity of the DL system remains unchanged. Meanwhile, by introducing a hybrid approach, we avoid the complex interventions between symbolic role number restrictions and other conceptual constructors by introducing the former through hybrid "wrapping" concepts. This removes one of the major sources of computational complexity [24] with regard to the extensions of DLs with concrete domains, if, again, only the DL-based inference is considered.

- Constraint reasoner: Finite Constraint Satisfaction Problems (FCSPs) are NP-complete as a general class [25]. Pragmatic results show that the performance varies from system to system. For a thorough analysis on different constraint systems, please refer to [26].

- Reasoning coordinator: thorough analysis on the algorithms working with the *reasoning coordinator* is forthcoming.

A formal evaluation of CONCOR and the theory of *inference fusion* using real-life examples is forthcoming.

Acknowledgements

This work is partially supported by an Overseas Research Scholarship from the British Council and by EPSRC under the AKT IRC grant GR/N15764.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, pages 28–37, May 2001.

[2] The Object Management Group OMG. OMG Unified Modeling Language Specification, March 2000. Available from http://www.omg.org/technology/documents/formal/uml.htm.

[3] N. F. Noy, R. W. Fergerson, and M. A. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *Proc. of the 12th Intl. Conf. on Knowledge Engineering and Knowledge Management (EKAW'2000)*, 2000.

[4] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. OIL: An Ontology Infrastructure for the Semantic Web. *Intelligent Systems*, 16(2):38–45, 2001.

[5] F. M. Donini et al. Reasoning in description logics. In *Foundations of Knowledge Representation*, pages 191–236. CSLI Publications, 1996.

[6] R. Fikes and A. Farquhar. Distributed Repositories of Highly Expressive Reusable Ontologies. *Intelligent Systems*, 14(2):73–79, 1999.

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

[8] B. Nebel. What is hybrid in hybrid representation and reasoning systems? In *Proc. of the 2nd Intl. Symp. on Computational Intelligence (CI'89)*, pages 217–228, 1989.

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

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th Intl. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number LNAI-1705, pages 161–180, 1999.

[11] C. Lutz. NExpTime-complete description logics with concrete domains. In *Proc. of the Intl. Joint Conf. on Automated Reasoning*, number LNAI-2083, pages 45–60, 2001.

[12] V. Haarslev, C. Lutz, and R. Möller. A description logic with concrete domains and role-forming predicates. *Jour. of Logic and Computation*, 9(3):351–384, 1999.

[13] E. Compatangelo and H. Meisel. $\mathcal{K}-$ShaRe: an architecture for sharing heterogeneous conceptualisations. In Proc. of I-KOMAT'2002-to appear, 2002.

[14] B. Hu, E. Compatangelo, and I. Arana. Coordinated reasoning with *inference fusion*. In E. Damiani *et. al.*, editor, *Proc. of the KES-2002, Sixth Intl. Conf. on Knowledge-Based Intelligent Information & Engineering Systems*, pages 156–160. IOS Press, 2002.

[15] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of the 12th Intl. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457. Morgan Kaufmann, 1991.

[16] I. Horrocks and U. Sattler. Ontology Reasoning in the $\mathcal{SHOQ}$(D) Description Logic. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 199–204. Morgan Kaufmann, 2001.

[17] V. Haarslev, R. Möller, and A. Turhan. ABox reasoner: Progress report. In *Proc. of the Intl. Workshop on Description Logics (DL'98)*, pages 82–86, 1998.

[18] V. Haarslev and R. Möller. High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 161–168, 2001.

[19] R. MacGregor, H. Chalupsky, and E. R. Melz. *PowerLoom Manual*. ISI, University of South California, 1997.

[20] B. Hu, E. Compatangelo, and I. Arana. A hybrid approach to extend DL-based reasoning with concrete domains. In *Proc. of the KI-2001 Workshop on Applications of Description Logics*, 2001.

[21] Joxan Jaffar and Michael J. Maher. Constraint Logic Programming: A Survey. *The Jour. of Logic Programming*, 19 & 20:503–582, 1994.

[22] E. P. K. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[23] P. Brisset et al. *ECL^{i}PS^e Constraint Library Manual, Rel. 5.2.* International Computers Ltd. and Imperial College London, 2001.

[24] F. Baader and U. Sattler. Description Logics with Symbolic Number Restrictions. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI'96)*, pages 283–287. John Wiley, 1996.

[25] A. K. Mackworth and E. C. Freuder. The Complexity of Constraint Satisfaction Revisited. *Artificial Intelligence*, 59(1–2):57–62, 1993.

[26] A. Fernández and P. M. Hill. A Comparative Study of Eight Constraint Programming Languages over the Boolean and Finite Domains. *Jour. of Constraints*, 5:275–301, 2000.

Table 1: Syntax and semantics of $\mathcal{ALC}$ constructors

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 2: Syntax and semantics of $\mathcal{DL}$(D)/S constructor (not in $\mathcal{ALC}$)

Constructor	**Syntax**	**Semantics** (*Interpretation*)
role value constraint(D)	$\forall \mathsf{R}_{\mathrm{H}}.\mathsf{H}$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y.\langle x,\ y\rangle \in \mathsf{R}^{\mathcal{I}} \ \rightarrow\ y \in \mathsf{H}^{\mathcal{I}}\}$
role cardinality constraint(S)	$(rel\ v\ \mathsf{R})$ $\exists v.\mathsf{C}[v]/\xi[v]$	$\{\ c \in \Delta^{\mathcal{I}} \mid \sharp\{\ d \in \Delta^{\mathcal{I}} : \langle\ c, d\ \rangle \in \mathsf{R}^{\mathcal{I}}\ \}\ rel\ \lambda'(\mathbf{v})\ \}$ $\mathsf{C}^{\mathcal{I}}[\lambda(v)]$ where $\xi[\lambda(v)]$ hold

Captions

Figure 1: System architecture of CONCOR.

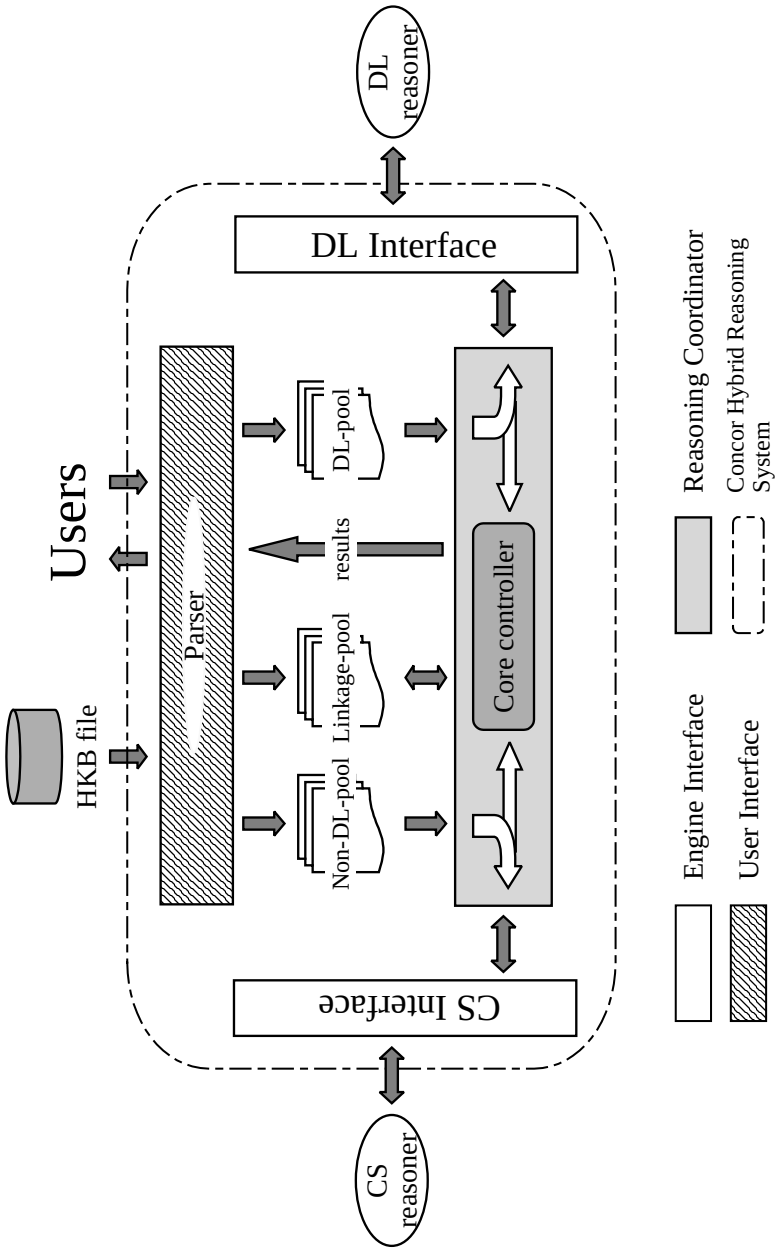

Figure 1: Knowledge-Based Systems, B. Hu, I. Arana and E. Compatangelo