# Infusion: a hybrid reasoning system with Description Logics

Bo Hu
The Robert Gordon University
Aberdeen, UK
bh@comp.rgu.ac.uk

Ernesto Compatangelo
University of Aberdeen
Aberdeen, UK
ecompata@csd.abdn.ac.uk

Inés Arana
The Robert Gordon University
Aberdeen, UK
ia@comp.rgu.ac.uk

**ABSTRACT**
We discussed a new approach using *inference fusion*, i.e. the cooperative reasoning from distributed heterogeneous inference systems, to extend the expressive and deductive powers of existing Description Logic (DL) based systems. More specifically, our approach integrates results from a DL reasoner with results from a constraint solver. *Inference fusion* (i) fragments heterogeneous input knowledge to generate suitable homogeneous inputs for the DL and constraint reasoners; (ii) passes control to each reasoner, retrieving the results and making them available to the other reasoner for further inferencing; and (iii) dynamically combines the results of the two reasoners to present the overall conclusion. We also outline the main features of *inference fusion* by way of a small example.

**KEY WORDS**
Knowledge Representation, Description Logics, Hybrid Reasoning System, Software tools

## 1 Introduction

The so-called "*Semantic Web*" has reignited the research into expressive knowledge representation and reasoning systems and, in particular, ontology modelling and reasoning [3]. Inferential engines based on Description Logics (DLs) are extremely powerful when reasoning about ontological knowledge, since they can automatically discover hierarchical structures of a set of vocabulary [5]. However, in general, their expressive power is restricted in order to reduce the computational complexity and to guarantee the decidability of their deductive algorithms. Consequently, this restriction prevents DL-based taxonomic reasoning from being widely applied to ontological services on heterogeneous domains (e.g. domains with integers, rational numbers, strings, *etc.*).

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 (e.g. FaCT [8] and RACER [7]), their homogeneous approaches are limited in three ways: (i) the inevitable limitation of available resources has emphasised the "trade-off", i.e. the expressive power of their representation is restricted in order to ensure the computational tractability, completeness and decidability; (ii) the specialist nature of their reasoning means that they are only successful at carrying out particular inferential tasks; and (iii) because of the diversity of real-life applications, new extensions to a reasoner always lead to new limitations to be discovered.

Because that developing a general and omnipotent reasoning system is impractical [4], *ad hoc* adoptions of an existing system to meet certain requirements in real life applications are inevitable. However, changing the underlying algorithm of certain reasoning systems might be very difficult for knowledge engineers to carry out at the application stage, as such process may involve in-depth knowledge of the reasoning systems and extensive programming skills. This is especially true when considering DLs, due to the insufficient functionality provided by the Knowledge Base Management System (KBMS) of DL reasoners.

In this paper, we present an *inference fusion* [12] (InFusion) approach as an effort to reduce the necessity of *ad hoc* extensions of DL-based systems in ontological reasoning. This approach extends DL-based systems with constraint solvers in a heterogeneous manor.

## 2 A hybrid representation for InFusion

DLs are based on the notion of concepts (i.e. unary predicates) and roles (i.e. binary relations). Using different conceptual constructs defined with uniform syntax and unambiguous semantics, complex concepts can be built up from simple components. For instance, if human and male are defined concepts with their intuitive meanings, the concept man can be defined as: (**and** human male). Meanwhile, if has-child is specified as a role, the concept Father, a man whose children are human, can be expressed as:

$$(\textbf{and}\ \mathsf{man}\ (\textbf{forall}\ \mathsf{has\text{-}child}\ \mathsf{human}))$$

In the past decades, different approaches have been proposed and implemented to extend the expressive and deductive powers of DL-based systems. The InFusion framework is proposed to provide hybrid reasoning based on the $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ language which extends the DLs, in general, with Hybrid Knowledge Base ($\mathcal{H}_{\mathrm{KB}}$)-global constraints on role successors [1] and concept-local constraints on role cardinalities [10, 11].

[1]The constrained role successors are referred to as ***hybrid concepts*** to be distinguished from normal DL concepts

We use the $\mathcal{ALCN}$, $\mathcal{ALC}$ [15] DL with role number restrictions ($\mathcal{N}$), DL language as the foundation of our hybrid representation since it provides most of the necessary constructs. Let $\mathsf{A}$ denote a concept name, $\mathsf{C}$ an arbitrary concept, $\mathsf{R}$ a role name and $n$ a non-negative integer. An interpretation $\mathcal{I}$ for $\mathcal{ALCN}$ is a tuple ( $\Delta^{\mathcal{I}}$, $\cdot^{\mathcal{I}}$) where the nonempty set $\Delta^{\mathcal{I}}$ is the domain of $\mathcal{I}$ and the $\cdot^{\mathcal{I}}$ function maps each concept to a subset of $\Delta^{\mathcal{I}}$ while each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The syntax and semantics of $\mathcal{ALCN}$ is presented in Table 1, where $rel \in \{\geq, \leq\}$.

Table 1. Syntax and semantics of $\mathcal{ALCN}$ constructs

| **Syntax** | **Semantics** (*Interpretation*) |
|---|---|
| $\top$ | $\Delta^{\mathcal{I}}$ |
| $\bot$ | $\emptyset$ |
| $\mathsf{A}$ | $\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| $\mathsf{R}$ | $\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| $\mathsf{C} \sqcap \mathsf{D}$ | $\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$ |
| $\mathsf{C} \sqcup \mathsf{D}$ | $\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$ |
| $\neg \mathsf{C}$ | $\Delta^{\mathcal{I}} \setminus \mathsf{D}^{\mathcal{I}}$ |
| $\forall \mathsf{R.C}$ | $\{ c \in \Delta^{\mathcal{I}} \| \forall d \in \Delta^{\mathcal{I}} : \langle c, d \rangle \in \mathsf{R}^{\mathcal{I}} \rightarrow d \in \mathsf{C}^{\mathcal{I}} \}$ |
| $\exists \mathsf{R.C}$ | $\{ c \in \Delta^{\mathcal{I}} \| \exists d \in \Delta^{\mathcal{I}} : \langle c, d \rangle \in \mathsf{R}^{\mathcal{I}} \wedge d \in \mathsf{C}^{\mathcal{I}} \}$ |
| $(rel\ n\ \mathsf{R})$ | $\{c \in \Delta^{\mathcal{I}} \mid \sharp\{d \in \Delta^{\mathcal{I}} : \langle c, d \rangle \in \mathsf{R}^{\mathcal{I}}\}\ rel\ n\}$ |

While DLs are normally used to model abstract domains, in domains such as engineering and architecture, modelling of numeric constraints is essential. New constructs are thus necessary to express the heterogeneous domains with both abstract and concrete knowledge. $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ is introduced to extend $\mathcal{ALCN}$ by incorporating the concrete knowledge shown in Table 2.

$\mathsf{H}$ is a *hybrid concept*, $v$ an integer type variable, $\Psi$ the set of role cardinality constraints and $\lambda$ an assignment mapping $v$ to a set of non-negative integers. $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ constraints are, therefore, specified through the *hybrid concepts*, i.e. $\forall x_1 \in \mathsf{H}_1, \ldots, \forall x_n \in \mathsf{H}_n.\mathrm{Q}(x_1, \ldots, x_n)$ or role cardinalities (RC) variables, i.e. $\exists v_1, \ldots, v_n.\mathrm{P}(v_1, \ldots, v_n) \in \Psi$ where $\mathsf{H}_1, \ldots, \mathsf{H}_n$ are *hybrid concepts*, $v_1, \ldots, v_n$ role cardinality variables and $\mathrm{P}$, $\mathsf{Q}$ constraint predicates.

Table 2. Syntax and semantics of $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ extensions

| **Syntax** | **Semantics** (*Interpretation*) |
|---|---|
| $\forall_h \mathsf{R.H}$ | $\{x \in \Delta^{\mathcal{I}} \mid \forall y.\langle x,\ y\rangle \in \mathsf{R}^{\mathcal{I}}\ \rightarrow\ y \in \mathsf{H}^{\mathcal{I}}\}$ |
| $(=\ v\ \mathsf{R})$ | $\{c \in \Delta^{\mathcal{I}} \mid \sharp\{d \in \Delta^{\mathcal{I}} : \langle c, d \rangle \in \mathsf{R}^{\mathcal{I}}\}\ = \lambda(v)\}$ |
| $\exists v.\mathsf{C}[v]/\Psi[v]$ | $\mathsf{C}^{\mathcal{I}}[\lambda(v)]$ where $\Psi[\lambda(v)]$ holds |

For example, the following LISP-style expression defines a father who "has twice as many sons as daughters":

(**exists** $(x\ y)$ (**and** father (**eq** $x$ has-son) (**eq** $y$ has-daughter) )
**:with :begin :body** $(x = 2 * y)$ **:end**)

Meanwhile, a $\mathcal{DL}(\mathrm{D})/\mathrm{S}$-$\mathcal{H}_{\mathrm{KB}}$ is a tuple $\mathcal{H}_{\mathrm{KB}} = (\mathcal{T}, \mathcal{C})$, where $\mathcal{T}$ is the set of concept definitions and multiple concept relations (e.g. father is subsumed by man and man is disjoint from woman) including the RC-constrained concepts, $\mathcal{C}$ is the set of global constraints (e.g. the restriction on human's lifespan $[0..100]$).

## 3 Fusion of DL-based and constraint-based inferences

In order to fully exploit the unparallel power of DL-based approaches in taxonomic reasoning, when reasoning about $\mathcal{DL}(\mathrm{D})/\mathrm{S}$-$\mathcal{H}_{\mathrm{KB}}$ with heterogeneous knowledge, we have developed a hybrid approach [12] that facilitates the collaboration of two disparate reasoners. The InFusion of disparate reasoners is defined as a three-stage process, namely, knowledge fragmenting, homogeneous reasoning and inference combination.

Two features are beneficial for reasoning with InFusion: (i) DL-based systems can specify (told) subsumption relationships between concepts, and (ii) an ordering (denoted as *quasi-ordering* [12]) can be introduced between different sets of constraints.

Our approach differentiates global and local numeric constraints and handles them in two different ways:

(i) Each *hybrid concept* is associated with a concrete image ("object identification"+"constrained domain"). The $\mathcal{H}_{\mathrm{KB}}$-global constraints are eliminated from the knowledge base as, after constraint propagation (i.e. maintaining a full path-consistency among the concrete images), the same restricting effects can be achieved by the *quasi-ordering* among the reduced domains of those constrained concrete images.

(ii) A new atomic concept is introduced to replace each set of local constraints. The restrictions of local constraints are enhanced by explicitly expressing the restrictions that are otherwise implicit, i.e. the entailments and disjointnesses among different RC constraint sets (*quasi-ordering*) are emulated by the subsumption relationships among the new concepts.

A heterogeneous $\mathcal{DL}(\mathrm{D})/\mathrm{S}$-$\mathcal{H}_{\mathrm{KB}}$ is reasoned by the InFusion reasoning system in the following steps:

1. $\mathcal{H}_{\mathrm{KB}}$ is normalised to generate three homogeneous sets of statements:

   (i) A set of DL statements (stored in *DL*-pool and donated as $\boldsymbol{S}_{\mathrm{DL}}$) which do not exceed the expressive power of the selected DL system;

   (ii) A set of non-DL statements (stored in *non-DL*-pool and donated as $\boldsymbol{S}_{\text{non-DL}}$) containing the concrete knowledge (both global and local constraints) that filtered out to make $\boldsymbol{S}_{\mathrm{DL}}$;

(iii) A set of *linkages* (stored in *linkage*-pool and donated as $\boldsymbol{S}_{linkages}$) which are bijections connecting DL and non-DL statements.

2. The set of $\mathcal{H}_{\text{KB}}$-global constraints is isolated and checked for satisfiability. In cases where an unsatisfiable global constraint exists, **false** is returned with a message to indicate the nature of the result.

3. All $\mathcal{H}_{\text{KB}}$-global constraints are propagated. The propagation of constraints applies path-consistency by removing the inconsistent values from the domains of concrete images (domain reduction). Using a constraint solver, (inclusion) *quasi-ordering* among the reduced domains, $O_{\subseteq}$, is retrieved and mapped to $\boldsymbol{S}_{\text{DL}}$ through *linkages*.

4. The $\mathcal{H}_{\text{KB}}$-global constraints are removed from the original $\mathcal{H}_{\text{KB}}$ since that the same restrictions can be emulated by synchronising the relationships among *hybrid concepts* with the inclusion relations among the domains of concrete images.

5. The concept-local role cardinality constraints are checked with regard to each concept. If an inconsistency is identified, the system returns the truth value **false** with a message.

6. The (entailment) *quasi-ordering*, $O_{\models}$, is generated with the help of a constraint solver and used to update the corresponding relationships among atomic concepts in $\boldsymbol{S}_{\text{DL}}$ through *linkages*.

7. The updated $\boldsymbol{S}_{\text{DL}}$ is classified to give the overall conceptual hierarchies of the original $\mathcal{H}_{\text{KB}}$.

As a result of the proposed reasoning method, all the numeric constraints are removed from the concept definitions, leaving only proper DL-based descriptions built up with the legal and admitted constructs of the selected DL-based inferential engine (e.g. $\mathcal{SHIQ}$ DL of the FaCT system). The reasoning process is summarised in Figure 1, where *Sat*$(\mathcal{H}_{\text{KB}})$ checks the satisfiability of the given knowledge base $\mathcal{H}_{\text{KB}}$.

## 4 The InFusion system

The InFusion theory is materialises by combining the TBox deductions with the constraint satisfaction inferences under the supervision of a reasoning agent. The system (see Figure 2) is composed of a *reasoning coordinator* (the interfacing agent), a DL reasoner and a constraint solver.

A combinational behaviour semantics is associated with the architecture as:

$$(\mathcal{DL}(\text{D})/\text{S-}\mathcal{H}_{\text{KB}})^{\mathcal{I}} = DL(\ \textit{linkage}(Con(\boldsymbol{S}_{\text{non-DL}}))\ \cup\ \boldsymbol{S}_{\text{DL}}\ )$$

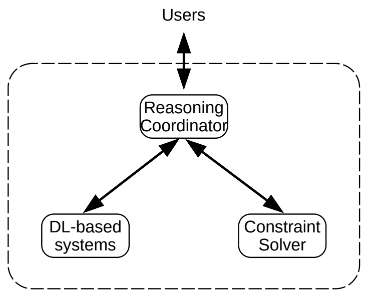
where $DL$ and $Con$ are DL-based systems and constraint solvers respectively and *linkage* the mapping mechanism between $\boldsymbol{S}_{\text{DL}}$ and $\boldsymbol{S}_{\text{non-DL}}$.

Figure 1. Reasoning algorithm for $\mathcal{DL}(\text{D})/\text{S-}\mathcal{H}_{\text{KB}}$

```
$sat :\ \mathcal{H}_{\text{KB}} \longrightarrow$ boolean

$sat(\mathcal{H}_{\text{KB}}) =$
  normalise $\mathcal{H}_{\text{KB}} \rightarrow\ \boldsymbol{S}_{\text{DL}},\ \boldsymbol{S}_{\text{non-DL}},\ \boldsymbol{S}_{linkage};$
  **if** $\forall C_{\text{global}} \in \boldsymbol{S}_{\text{non-DL}}.C_{\text{global}}$ satisfiable
  **then**
    propagate $C_{\text{global}};$
    obtain domain inclusion ordering, $O_{\subseteq};$
    $\boldsymbol{S}_{\text{DL}} = \boldsymbol{S}_{\text{DL}} \cup linkage(O_{\subseteq});$
    **if** $\forall\exists C_{\text{concept local}} \in \boldsymbol{S}_{\text{non-DL}}.C_{\text{concept local}}$ consistent
    **then**
      obtain entailment ordering, $O_{\models};$
      $\boldsymbol{S}_{\text{DL}} = \boldsymbol{S}_{\text{DL}} \cup linkage(O_{\models});$
      **if** satisfiable $\boldsymbol{S}_{\text{DL}}$
      **then** **true**: successful;
      **else** **false**: inconsistent $\mathcal{H}_{\text{KB}};$
    **else** **false**: inconsistent $C_{\text{concept local}};$
  **else** **false**: inconsistent $C_{\text{global}};$
```

Figure 2. InFusion's architecture



The *reasoning coordinator* is at the heart of InFusion and is responsible for redirecting knowledge to the specialised reasoning engines, analysing the results of their inferencing, deciding whether further processes should be carried out, updating the knowledge base files and returning results. It uses an internal language which acts as the mediator between the input and the underlying representation used by the selected DL or constraint system. This language (i) makes InFusion engine-independent, since whenever a reasoner is replaced, only the interface need to be switched, and (ii) reduces the programming effort on any further extensions to the modelling language, since only the parser residing between the intermediate language and user language need to be modified.

### 4.1 Complexity

Small test cases have been reasoned about by InFusion giving promising results. Although the overall analysis of the complexity of our system has not been completed, the complexity of each of its components is discussed below:

Because of the architecture of our system, the complexity can actually be considered in three relatively independent parts, i.e. the complexity of DL-based reasonings, the complexity of constraint solving, and the complexity of the global *reasoning coordinator*.

### 4.1.1 Complexity of DL-based reasoning

When considered from the DL's point of view, we do not explicitly introduce any new concept constructs—only new mechanisms to interpret the foreign constructs using normal DL-based ones. In another word, new constructs do not participate in either DL-based or constraint reasonings—they are removed after concept normalisation.

Moreover, we avoid the complex interventions between symbolic RC restrictions and other conceptual constructs by wrapping the former with atomic concepts. This removes one of the major sources of computational complexity [1] w.r.t. the extensions of DLs with concrete domains, if, again, only the DL-based inference is considered.

More specifically, we expect the complexity of the DL-based taxonomic inference to be in the same class as that of the original language without any new constructs or operators.

### 4.1.2 Complexity of constraint solving

When considering from the constraint's point of view, the Finite Constraint Satisfaction Problems (FCSPs) are NP-complete as a general class [14]. Pragmatic results shows that the performance varies from system to system. For a thorough analysis on different constraint programming systems, please refer to [6].

When considering the RC-constraints, FCSP is considered to have enough expressive power for modelling and solving the constraints on integer-type role cardinalities. On the contrary, the global constraints may be used to model problems with real numbers, e.g. mechanical domains, of which the constraint solving is prohibitively expensive. Nevertheless, DL with concrete domains is, in itself, complex—the satisfiability and subsumption of $\mathcal{ALC}(\mathrm{D})$-concepts are NEXPTIME-hard when (i) TBox is taken into account and (ii) satisfiability in concrete domain D is PTIME [13].

However, we can reduce the overall complexity of $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ reasoning by restricting the constraints that can be defined over the role successors to finite domains or other less complex domains.

### 4.1.3 Complexity of reasoning coordination

New sources of computational complexity of the InFusion approach mainly originate from the *reasoning coordinator*.

Reasoning processes of $\mathcal{DL}(\mathrm{D})/\mathrm{S}$ involve language analysing processes (i.e. syntax checking and language fragmenting) which are linear to the size of input $\mathcal{H}_{\mathrm{KB}}$.

Meanwhile, *quasi-ordering* is obtained by testing the relationships between each pair of different RC constraint sets or concrete images of *hybrid concepts*. Such tests are performed in a deterministic manor in both directions. Therefore, it is expected to be of $O(N^2)$ with regard to the size of input $\mathcal{H}_{\mathrm{KB}}$ (i.e. the number of RC-constrained concepts or *hybrid concepts*).

## 5 Hybrid reasoning with examples

In order to illustrate the reasoning process in detail, a toy example with numeric constraints is presented. Let us assume that company X plans to have a contest on property designs. To simplify the evaluation procedure, the organiser has defined what he means by a good design using the concept Good-Design. Among others, one of the submitted designs is Design-A. We have all the information stored in a $\mathcal{H}_{\mathrm{KB}}$ as shown in Figure 3.

Figure 3. hybrid knowledge base $\mathcal{H}_{\mathrm{KB}}$

```
(defprimconcept   Residence-Estate Top)
(defprimconcept   Room Top)

(defprimrole      has-bedroom)
(defprimrole      has-s-room)

(decl-variable 'Shape-et   [SQ, RECT, OVAL, CIR, POLY])
(decl-variable 'Shape-ba   [SQ, RECT, OVAL, CIR, POLY])
(decl-variable 'Area-ba    [0..100])
(decl-variable 'Area-ca    [0..100])

(defconcept   'Good-Design   '(and Residence-Estate
                                 (exists (x y)
                                   (and (eq x has-bedroom)
                                        (eq y has-s-room))
                                   (with :begin
                                      (x = 4y)
                                   :end))
                                 (forall has-bedroom Econ-Type)) )

(defconcept   'Econ-Type   '(and Room
                               (fallin has-area Area_et)
                               (fallin has-shape Shape_et)) )

(defconcept   'Type-A   '(and Residence-Estate
                            (exact 8 has-bedroom)
                            (exact 2 has-s-room)
                            (forall has-bedroom Bedroom-A)
                            (forall has-s-room Sittingroom-A)) )

(defconcept   'Bedroom-A   '(and Room
                               (fallin has-area Area-ba)
                               (fallin has-shape Shape-ba)) )

(defconcept   'Sittingroom-A   '(and Room
                                   (fallin has-area Area-sa)) )

(decl-constraint   'Rooms
                   :with :begin
                        Shape-et =[RECT, CIR, OVAL, SQ, POLY],
                        Shape-ba =[RECT],
                        20 ≤ Area-ba ≤ 40,
                        Area-ba ≤ Area-ca −10,
                        Area-ba ≥ Area-ca /2,
                        Area-ca = 50
                   :end)
```

After *normalisation*, the original hybrid $\mathcal{H}_{\text{KB}}$ is fragmented into homogeneous parts and stored into pools as explained in Section (3). The contents of different pools are illustrated in Figure 4, where $\boldsymbol{S}_{\text{non-DL}}$ is the union of $\boldsymbol{S}^{\text{Global}}_{\text{non-DL}}$ and $\boldsymbol{S}^{\text{Local}}_{\text{non-DL}}$, i.e.

$$\boldsymbol{S}_{\text{non-DL}} = \boldsymbol{S}^{\text{Global}}_{\text{non-DL}} \cup \boldsymbol{S}^{\text{Local}}_{\text{non-DL}}$$

Figure 4. contents of Pools

$\boldsymbol{S}_{\text{DL}}$

```
(defconcept Good-Design     (and Residence-Estate
                                 (some has-bedroom)
                                 (some has-s-room)
                                 C-gd
                                 (forall has-room1 Econ-Type)))
  (defconcept Econ-Type     (and Room (forall has-area Area-et)
                                 (forall has-shape Shape-et)))
   (defconcept Design-A     (and Residence-Estate
                                 (some has-bedroom)
                                 (some has-s-room)
                                 C-da
                                 (forall has-bedroom Bedroom-A)
                                 (forall has-s-room Sittingroom-A)))
  (defconcept Bedroom-A     (and Room (forall has-area Area-ba)
                                 (forall has-shape Shape-ba)))
(defconcept Sittingroom-A   (and Room
                                 (forall has-area Area-ca)))
```

$\boldsymbol{S}^{\text{Global}}_{\text{non-DL}}$

Shape-et $=$ [`RECT, CIR, OVAL, SQ, POLY`],
Shape-ba $=$ [`RECT`],
$20m^2 \leq$ Area-ba $\leq 40m^2$,
Area-ba $\leq$ Area-ca $- 10m^2$,
Area-ba $\geq$ Area-ca$/2$,
Area-ca $= 50m^2$

$\boldsymbol{S}^{\text{Local}}_{\text{non-DL}}$

RC-gd $=$ $\{|$has-bedroom$| = x$, $|$has-s-room$| = y$, $x = 10 * y\}$
RC-da $=$ $\{|$has-bedroom$| = 20$, $|$has-s-room$| = 2\}$

$\boldsymbol{S}_{linkage}$

| | | | |
|---|---|---|---|
| C-gd $\rightarrow$ | RC-gd | C-da $\rightarrow$ | RC-da |
| Shape-ba $\rightarrow$ | Shape-ba | Shape-et $\rightarrow$ | Shape-et |
| Area-ba $\rightarrow$ | Area-ba | Area-et $\rightarrow$ | Area-et |
| Area-ca $\rightarrow$ | Area-ca | | |

By consulting the constraint solver, *quasi-orderings* between the concrete images of *hybrid concepts* (i.e. Shape-et, *etc.*) and RC constraint sets (i.e. RC-gd, *etc.*) are established as followings:

$$\text{RC-da} \preceq_{\models} \text{RC-gd}$$
$$\text{Area-ba} \preceq_{\subseteq} \text{Area-et}$$
$$\text{Shape-ba} \preceq_{\subseteq} \text{Shape-et}$$

The orderings are mapped into $\boldsymbol{S}_{\text{DL}}$ as relationships between concepts introduced to replace concrete knowledge in the original $\mathcal{H}_{\text{KB}}$ as:

$$\text{C-da} \sqsubseteq \text{C-gd} \quad \text{Area-ba} \sqsubseteq \text{Area-et} \quad \text{Shape-ba} \sqsubseteq \text{Shape-et}$$

Thus, together with the original concept definitions in $\boldsymbol{S}_{\text{DL}}$, we can draw a conclusion that the design A is a sub-class of good designs for this particular contest, i.e.

$$\text{Design-A} \sqsubseteq \text{Good-Design}$$

Note that, for the $\mathcal{H}_{\text{KB}}$ presented in this paper, existing DL-based systems are either do not have the expressive and deductive powers to give the correct answers (e.g. the reasoning of RC constrained concepts) or the reasoning process is computational expensive because of the intervention between concrete and abstract knowledge (e.g. the interactions between concrete domains on one hand and individual fillers or inverse roles on the other [9]).

## 6 Implementation of InFusion

InFusion framework has been implemented which combines the inferential powers of the LISP-based DL system FaCT and the Prolog-based constraint solver ECL$^i$PS$^e$. Note that the selection of both FaCT and ECL$^i$PS$^e$ is not mandatory as InFusion is introduced as a generic framework which can be applied to other DL-based reasoners and constraint solvers as long as the requested functionalities (ordering and mapping) are provided.

### 6.1 Language parser and translator

Language parsing and translation are one of the core functionalities of InFusion systems (residing in the *reasoning coordinator*). Several programming languages (e.g. Prolog and Lisp) and tools are available for this purpose. In this investigation, a C/C++ based tool, Bison/Flex, is selected to assist in coding of parsers and translators.

### 6.2 Coordinator and User Graphic Interface

The *reasoning coordinator*, implemented with Tcl/Tk, benefits from a User Graphic Interface (UGI). End users give instructions to the system *reasoning coordinator* through buttons, menu items or dialog boxes.

As shown in Figure 5, end-users can obtain the system feedback both from the Output Window of the UGI and line-wisely from the console window. This is to provide users a flexible means to acquire more detailed explanations on errors and warnings which may not be provided by InFusion system. For instance, DL-based system FaCT provides verbose explanations on errors. For simplicity, such verbose information is not displayed in UGI Output Window. However, experienced users of FaCT may found it useful and thus can refer to the console window.

## 7 Conclusions

We have presented a hybrid reasoning system which fuses the inferences of a DL-based taxonomic reasoner with those of a constraint solver without increasing the computational complexity of the former system. We consider our approach as an exemplary solution to the general problem of reducing and distributing the reasoning with expressive

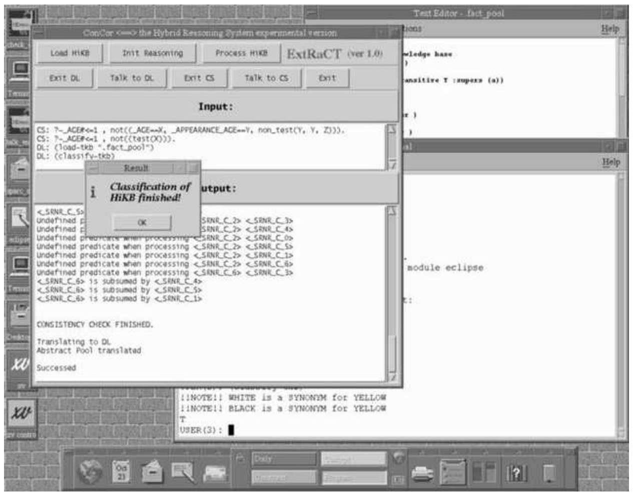


Figure 5. The implemented InFusion HRS

```
CS: ?-_AGE#<=1 , not((_AGE==X, _APPEARANCE_AGE==Y, non_test(Y, Y, Z))).
CS: ?-_AGE#<=1 , not((test(X))).
DL: (load-tkb ".fact_pool")
DL: (classify-tkb)
```

```
<_SRNR_C_5>
Undefined p                 <_SRNR_C_2> <_SRNR_C_3>
Undefined p                 <_SRNR_C_2> <_SRNR_C_4>
Undefined predicate when processing <_SRNR_C_2> <_SRNR_C_0>
Undefined predicate when processing <_SRNR_C_2> <_SRNR_C_5>
Undefined predicate when processing <_SRNR_C_2> <_SRNR_C_1>
Undefined predicate when processing <_SRNR_C_2> <_SRNR_C_6>
Undefined predicate when processing <_SRNR_C_6> <_SRNR_C_3>
<_SRNR_C_6> is subsumed by <_SRNR_C_4>
<_SRNR_C_6> is subsumed by <_SRNR_C_5>
<_SRNR_C_6> is subsumed by <_SRNR_C_1>


CONSISTENCY CHECK FINISHED.

Translating to DL
Abstract Pool translated

Successed
```

```
!!NOTE!! WHITE is a SYNONYM for YELLOW
!!NOTE!! BLACK is a SYNONYM for YELLOW
T
USER(3):
```

knowledge to the combination of inferences from heterogeneous reasoners. Our approach shows how to distribute and coordinate the reasoning task among different inferential engines, each dealing with a homogeneous part of the overall expressive power.

The proposed theory is demonstrated in a hybrid reasoner which combines inferences from both the FaCT DL-based reasoning system [8] and the ECL$^i$PS$^e$ constraint solver [2]. The hybrid characteristics of our approach are evident in the *linkages* mechanism which maps atomic concepts in the DL reasoner to the legal objects (e.g. concrete images and RC constraint sets) in the constraint solver, and *vice versa*. Our approach does not depend on the particular (DL and constraint) reasoners adopted, since the characteristics of the *linkages* makes the *reasoning coordinator* independent from any specific DL system or constraint problem solver [12].

## Acknowledgements

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

## References

[1] 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.

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

[3] J. Broekstra et al. Enabling knowledge representation on the web by extending rdf schema. In *Proc. of the 10th World Wide Web conference (WWWW'10)*, pages 467–478, 2001.

[4] J. Doyle and R. S. Patil. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.

[5] 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.

[6] 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.

[7] V. Haarslev and R. Möller. *RACER User's Guide and Reference Manual Version 1.6*. University of Hamburg, Computer Science Department, July 2001. Technical Report.

[8] I. Horrocks. *FaCT Reference manual Ver. 1.6*. Dept. of Computer Science, University of Manchester, UK, 1998.

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

[10] 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.

[11] 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.

[12] B. Hu, E. Compatangelo, and I. Arana. Facilitating dl-based hybrid reasoning with *Inference Fusion*. In *22nd SGAI Intl. Conf. on Knowledge-based Systesm and Applied Artificial Intelligence (ES'02)*. Springer-Verlag, 2002.

[13] 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.

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

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