

Inference Fusion: A Hybrid Approach to Taxonomic Reasoning

Bo Hu

The University of Southampton
Highfield
Southampton SO17 1BJ, UK
bh@ecs.soton.ac.uk

Inés Arana

The Robert Gordon University
St. Andrew Street
Aberdeen AB25 1HG, UK
ia@comp.rgu.ac.uk

Ernesto Compatangelo

University of Aberdeen
King's College
Aberdeen AB24 3UE, UK
ecompatata@csd.abdn.ac.uk

Abstract

We present a hybrid way to extend taxonomic reasoning using *inference fusion*, i.e. the dynamic combination of inferences from distributed heterogeneous reasoners. Our approach integrates results from a DL-based taxonomic reasoner with results from a constraint solver. *Inference fusion* is carried out by (i) parsing heterogeneous input knowledge, producing suitable homogeneous subset of the input knowledge for each specialised reasoner; (ii) processing the homogeneous knowledge, collecting the reasoning 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 to the ontological reasoning and demonstrate our ideas by proposing a hybrid modelling languages, $\mathcal{DL}(D)/S$, and illustrating its use by means of examples.

Motivation and background

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. Different from the Object-oriented and Frame-based approaches, models based on Description Logics (DLs) like OIL/DAML+OIL (Fensel *et al.* 2001) are equipped with a whole set of specialised deductions based on taxonomic reasoning. Such deductive services include, among others, semantic consistency check and contradiction detection, explication of hidden knowledge, subsumption, and concept classification (Donini *et al.* 1996). Therefore, 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 (Fensel *et al.* 2001). 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, because the expressive power needed to model complex real-world ontology is quite high, ontology reasoning was initially ruled out of the list of services to

be provided by ontology management tools (Fikes & Farquhar 1999). Nevertheless, it has been re-introduced by the OIL/DAML+OIL effort as a first-class issue, providing a solution within the framework of a DL-based, frame-centred approach (Fensel *et al.* 2001). 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 (Horrocks 1999), which provides the deductive services for the ontology inference layer, does not currently include concrete domains or role-value maps.

Some approaches have been proposed to include *concrete domains* in DL-based concept definitions which are normally restricted to *abstract domains*. Despite the diversity of their representations, most of them have based on *ALC* (Schmidt-Schauß & Smolka 1991) or its expressive successor *SHIQ* (Horrocks, Sattler, & Tobies 2000). They concentrated on extending the original tableau-based algorithm (Schmidt-Schauß & Smolka 1991), i.e. create a tableaux containing both concept constructors and constraint predicates, during which process, the complex intervention of abstract and concrete knowledge is inevitable. It has been proved that adding concrete domains (e.g. numeric constraints) directly to expressive DL-based systems may result in undecidable inferential problems (Lutz 2001).

The dilemma faced by DL-community brings up a new question: 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.

We believe that if a knowledge model is too expensive to be analysed by a reasoner (e.g. DLs) alone, other representation and reasoning paradigms must be jointly used. Therefore, it's reasonable to consider that a hybrid approach to heterogeneous knowledge management may provide, among other things, a wider and better support to ontology reasoning.

In this paper, we thus present a generic hybrid 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 $\mathcal{DL}(\mathcal{D})/S$, and explained with examples.

Inference fusion based on DLs

Inference fusion is critical in Hybrid Reasoning Systems (HRSs), which combine different kinds of specialised inferential sub-systems. In this paper, we focus on a particular class of HRSs, which fuse TBox deductions from a taxonomic reasoner with constraint satisfaction inferences from a constraint solver. In order to ensure the autonomy of both reasoners, we introduce the concept of *linkages* which are relations responsible for the inter-engine communication between DL systems and constraint solvers. They: (i) enable cooperative reasoning without changing the underlying inference algorithms of either reasoner; (ii) are expressive enough to describe all the reasoning results from one system to the other without dramatically increasing the original computational complexity.

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

As a result, all the information related to the numeric constraints is removed from the concept definitions. Thus, only the proper DL constructors which are admitted by the selected DL-based systems are left.

The reasoning results from non-DL systems are fed into the DL system by the ordering among *linkages*. The hybrid characteristics of our approach are evident in the “polymorphism” of *linkages* which are defined as atomic concepts in DL-based systems while act as legal objects in non-DL systems, (e.g. constrained variables in constraint solvers).

Hybrid modelling with inference fusion

We have introduced $\mathcal{DL}(\mathcal{D})/S$, a hybrid language which facilitates the integration of DL-based systems with *concrete domains*. The *concrete domains* are formally defined as a pair $\mathcal{D} = (\Delta_{\mathcal{D}}, \Phi_{\mathcal{D}})$, where $\Delta_{\mathcal{D}}$ is a set called the domain and $\Phi_{\mathcal{D}}$ is a set of predicates. Each predicate $\phi \in \Phi_{\mathcal{D}}$ is associated with an arity n and an n -ary predicate $\phi^{\mathcal{D}} \subseteq \Delta_{\mathcal{D}}^n$. In this paper, we restrict predicates to algebraic and boolean operators, while we restrict the domain $\Delta_{\mathcal{D}}$ to a finite one with numeric values and string constants.

In principle, because of the generic characteristics of *inference fusion* and the common availability of *linkages*, the use of a particular DL language is not mandatory, i.e. our approach to extending DLs with numeric constraints can potentially be applied to other existing DL-based systems.

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

We use the \mathcal{ALCN} , \mathcal{ALC} with role number restrictions, as the foundation of our hybrid representation since it provides most of the constructs necessary for our purposes. Let \mathcal{A}

denote a concept name, \mathcal{C} arbitrary concepts, \mathcal{R} a role name and n an non-negative integer. Concepts in \mathcal{ALCN} are:

$$\top | \perp | \mathcal{A} | \neg \mathcal{C} | \mathcal{C} \sqcap \mathcal{D} | \mathcal{C} \sqcup \mathcal{D} | \forall \mathcal{R}. \mathcal{C} | \exists \mathcal{R}. \mathcal{C} | (\leq n \mathcal{R}) | (\geq n \mathcal{R})$$

A *concept definition* is either $\mathcal{A} \sqsubseteq \mathcal{C}$ (partial definition) or $\mathcal{A} \doteq \mathcal{C}$ (full definition). An interpretation \mathcal{I} for \mathcal{ALCN} is a couple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ 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 interpretation of \mathcal{ALCN} constructors can be found in (Patel-Schneider & Swartout 1993).

Let all the aforementioned symbols be defined as previously, \mathcal{H} a hybrid concept and ξ a concrete predicate. In addition to the syntax of \mathcal{ALCN} , we introduce the following new constructors:

1. *hybrid role value restriction*: $\forall_h \mathcal{R}. \mathcal{H}$ is a concept specifying the value restrictions of role successors (\mathcal{H} is referred to as *hybrid concept* so as to be differentiated from the normal abstract role successors);
2. *role cardinality restriction*: $(= v \mathcal{R})$ is a concept where v is an \mathbb{Z}^* -type role cardinality (RC) variable;
3. *role cardinality constraint*: $\exists v_1, \dots, v_n. \mathcal{C} / \xi[v_1, \dots, v_n]$ is a concept specifying the set of all constraints on role cardinalities associated with concept \mathcal{C} .

With concept-local RC constraints, it is possible to restrict the numbers of roles without giving the exact values. For instance, using $\mathcal{DL}(\mathcal{D})/S$, a equipment with twice as many airpads as axes is described as:

$$\exists(\alpha \beta). (\text{Machine.Tool} \sqcap (= \alpha \text{ has-axis}) \sqcap (= \beta \text{ has-airpad})) / \{\alpha = 2\beta\}$$

where the RC constraints are specified as $\xi[\alpha, \beta] : \alpha = 2\beta$.

Meanwhile, global constraints are introduced through *hybrid concepts* \mathcal{H} as: $\psi(\mathcal{H}_1, \dots, \mathcal{H}_n)$ giving the restrictions on the role successors. Note that global constraint is not introduced by means of concept constructors and is not interpreted within abstract domains.

Let $\cdot^{\mathcal{I}}$ be the interpretation function, $\mathcal{C}[v_i/t_i]$ the concept obtained through bounding each variable in v_1, \dots, v_n to a number t_1, \dots, t_n in \mathbb{Z}^* , and $\lambda'(\cdot) : v_i \rightarrow t_i$ the assignment function. The meaning of the new constructors are interpreted as followings:

1. $(\forall_h \mathcal{R}. \mathcal{H})^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y. \langle x, y \rangle \in \mathcal{R}^{\mathcal{I}} \rightarrow y \in \mathcal{H}^{\mathcal{I}}\}$
2. $(= v \mathcal{R})^{\mathcal{I}} = \{c \in \Delta^{\mathcal{I}} \mid \#\{d \in \Delta^{\mathcal{I}} : \langle c, d \rangle \in \mathcal{R}^{\mathcal{I}}\} = \lambda'(v)\}$
3. $(\exists v. \mathcal{C}[v]/\xi[v])^{\mathcal{I}} = \mathcal{C}^{\mathcal{I}}[v/t] \wedge \xi[v/t]$.

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

$\mathcal{DL}(\mathcal{D})/S$ provides the capability to express both the HKB-global constraints and the concept-local constraints. A $\mathcal{DL}(\mathcal{D})/S$ knowledge base ($\mathcal{DL}(\mathcal{D})/S$ -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) and Ψ is the set of all global constraints $\psi[\mathcal{H}_1, \dots, \mathcal{H}_n]$ defined over $\mathcal{H}_1, \dots, \mathcal{H}_n$ or a subset of them.

To some extents, our *hybrid concept* is similar to the *concrete datatype* in $SHOQ(\mathcal{D})$ (Horrocks & Sattler 2001). However, they are different in two aspects. 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*. Moreover, the overall inferential process is distributed across different specialised engines and thus the intervention between different language components is avoided.

Let $\lambda(H^{\mathcal{I}}) \subseteq \Delta_{\mathcal{D}}$ be the assignment function which creates a concrete image for a *hybrid concept* and assign a subset of $\Delta_{\mathcal{D}}$ to the concrete image, $\lambda'(v_i) \rightarrow t_i \in \mathbb{Z}^*$ mapping v_i to a non-negative integer. We have:

$$(\psi(H_1, \dots, H_n))^{\mathcal{I}} \equiv \text{sat}(\psi(H_1, \dots, H_n)) \equiv \bigwedge_{i=1}^n \forall x_i \in \lambda(H_i^{\mathcal{I}}). (\exists y_1 \in \lambda(H_1^{\mathcal{I}}), \dots, \exists y_{i-1} \in \lambda(H_{i-1}^{\mathcal{I}}), \exists y_{i+1} \in \lambda(H_{i+1}^{\mathcal{I}}), \dots, \exists y_n \in \lambda(H_n^{\mathcal{I}}). \psi(y_1, \dots, y_{i-1}, x_i, y_{i+1}, y_n))$$

i.e. for every possible value of the concrete image of H_i , there exist values in every H_j ($j = 1 \dots n, j \neq i$) such that ψ holds. Meanwhile, the collection Ψ is satisfied $\text{sat}(\Psi[H_1, \dots, H_n])$ iff

$$\text{sat}(\Psi[H_1, \dots, H_n]) \equiv \forall \psi \in \Psi[H_1, \dots, H_n]. \text{sat}(\psi)$$

A $\mathcal{DL}(\mathcal{D})/S$ -concept C/ξ (ξ may be empty) is satisfiable w.r.t. $\xi[v_i]$ iff there is an assignment $\lambda'(v_i) \rightarrow t_i \in \mathbb{Z}^*$ such that $C[t_i] \neq \emptyset$ and $\xi[t_i]$ holds for $i = 1 \dots n$:

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

Both types of constraints need to be “wrapped” as they cannot be directly processed by DL-based systems. Concepts containing wrapped constraints are said to be *normalised*. The concept normalisation is specified as follows:

Global constraints: (i) generating an atomic concept for each *hybrid concept* H , (ii) creating a *linkage* between H and the corresponding constrained variable and (iii) removing all global constraints;

RC-constrained concept: (i) replacing every sub-concept containing RC constraints 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 knowledge base 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

$$\text{Machine_Tool} \sqcap \exists \text{has-axis} \sqcap \exists \text{has-airpad} \sqcap \text{C1_axis-pad}$$

where the RC constraint (i.e. “ $\alpha = 2\beta$ ”) is replaced by `C1_axis-pad` introduced as an atomic concept. Meanwhile, Concept (1) in the same HKB is *normalised* as (2)

$$\text{Machine_Tool} \sqcap (\geq 4 \text{ has-axis}) \sqcap (\leq 4 \text{ has-axis}) \quad (1)$$

$$\text{Machine_Tool} \sqcap \exists \text{has-axis} \sqcap \text{C2_axis-pad} \quad (2)$$

where the RC constraints (e.g. $\{|\text{has-axis}| \leq 4\}$) is extracted and replaced by `C2_axis-pad` because the role `has-axis` is a RC-constrained role.

If we define that all concepts containing roles which are restricted by RC constraints as RC-related concepts, then:

1. If two concepts C and D are RC-related concepts, the subsumption relationship is defined as:
 - (a) let C' and D' be the normalised concept definitions of C and D ;
 - (b) 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 , respectively).
 $D/\xi_D \sqsubseteq C/\xi_C$ 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 , $\xi'_D \models_{\Sigma} \xi'_C$.
2. If otherwise, the normal DL-based reasoning is carried.

Reasoning with concrete constraints

The *linkages* are based on two observations. Firstly, DL-based systems can specify subsumption relationships between concepts (the “told” knowledge), e.g. (implies $A \ B$) in iFaCT.

Secondly, it is possible to obtain an ordering (e.g. *quasi-ordering*) with the help of constraint solvers¹. For instance, the entailment relationship between two set of constraints is seen as an ordering.

Ordering of constraints

In our case, orderings w.r.t. constraints are obtained in different ways. 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 an approach applies to cases when (i) variable domains exist independently; (ii) their images in DL-based systems can be isolated from the rest of a HKB; and (iii) the isolated knowledge can be referred to as an independent object naturally. For instance, the life-span of human beings whose domain is $0 \dots 150$ can be isolated from others easily and defined and referred to as an atomic concept in a DL-based knowledge base.

When constrained variables appear as the RC restrictions, the domain reduction is not applicable. Since constraints can be considered as the set of tuples of legal values that the constrained variables can take simultaneously (Tsang 1993), an inclusion (entailment) between different sets of tuples can actually be established and manipulated.

The relationships among concrete constraints are 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. $\beta \models_{\Sigma} \alpha$.

¹Currently, Constraint Logic Programming (CLP) languages have been extended 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 (Jaffar & Maher 1994))

In this case, we also say that β is tighter than α . If such ordering are mutual, α and β are said to be equivalent.

Constraints in $\mathcal{DL}(\mathbf{D})/S$ -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 concrete images of the *hybrid concepts*). Contrarily, local RC constraints are enhanced by explicitly expressing the restrictions which are otherwise implicit (i.e. discover the entailments ordering and the disjointness).

Combinational behaviour semantics

The reasoning system of $\mathcal{DL}(\mathbf{D})/S$ -KB is build on the top of two subsequent engines, i.e. a DL system and a constraint solver. In our approach, no change to the underlying reasoning algorithms is necessary as a clear separation is kept between DL-based descriptions and constraint-based ones. Constraint solvers only reason with the constraint-related restrictions while DL-based systems focus on the taxonomic reasonings. The behaviour semantics of our knowledge base is defined with syntactic objects as:

$$(\mathcal{DL}(\mathbf{D})/S\text{-KB})^{\mathcal{I}} = DL(Con(\Pi_{\text{non-DL}}) \cup \Pi_{\text{DL}})$$

where DL is the DL-based system, Con is the constraint solver and $\Pi_{\text{non-DL}}$ and Π_{DL} are the sets of non-DL and DL-based descriptions respectively.

With the combinational behaviour semantics, a system is endowed with the ability to carry out real calculations (e.g. the sum of two numeric values) while preserves the *open world assumption* which is a distinctive characteristics of the DL-based approach to the ontological reasonings. Moreover, the system modularity and the simplicity of implementation are enhanced as both the DL and the Con reasoning engines can be off-the-shelf systems.

Modelling with $\mathcal{DL}(\mathbf{D})/S$

An example built up using the $\mathcal{DL}(\mathbf{D})/S$ language is presented in LISP-style in Figure (1), where terms in bold fonts are reserved words of the user language while constant strings are presented in typewriter font, e.g. `square`.

Starting with the topmost concept `Floorplan`, a series of roles, hybrid variables, concepts and constraints are defined. Note that all the concepts of room styles are only defined for demonstrating purposes. Hence, no further restrictions other than shapes are specified.

Reasoning about the HKB in Figure (1) using current DL-based systems may be: (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 numeric constraints on role cardinalities.

In a hybrid approach, the reasoning about concrete constraints and variables can be redirected to a constraint solver which is design particular for this type of inferences. Moreover, the interference between abstract and concrete knowledge, one of the major contributions to the high computational complexity, can be avoided, if the fragment of the HKB and the selection of the carrier (*linkages*) for inter-engine communications is carefully designed.

After the hybrid reasoning, a series of nontrivial conclusions can be drawn as follows:

Figure 1: $\mathcal{DL}(\mathbf{D})/S$ knowledge base example

```
(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, rhomb, cir, tri ])
(decl-variable 'Shape_SBeD [square, rect, rhomb, cir, tri ])
(decl-variable 'Shape_SBaH [square, rect, rhomb, cir, tri ])
(decl-variable 'Shape_SBeH [square, rect, rhomb, cir, tri ])
(decl-variable 'Shape_SBaE [square, rect, rhomb, cir, tri ])
(decl-variable 'Shape_SBeE [square, rect, rhomb, cir, tri ])

(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_Hi)
    (equal n2 has_bedrooms) (forall has_bedrooms Style_bed_Hi) )
  (with :begin :body
    x > n1 + n2, y = z, y = x
  :end) ))

(def-concept 'Ensuit_Design '(exists (x y z)
  (and Floorplan
    (equal x has_rooms)
    (equal x has_bedrooms) (forall has_bedrooms Style_bed_En)
    (equal y has_bathrooms) (forall has_bathrooms Style_bath_En) )
  (with :begin :body
    x = y, z ≥ y + x + 1
  :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 'Dorm_Design '(exists (x y z)
  (and Floorplan
    (equal x has_rooms)
    (equal y has_bedrooms) (forall has_bedrooms Style_bed_Do)
    (equal z has_bathrooms) (forall has_bathrooms Style_bath_Do) )
  (with :begin :body
    x > y + z, y = z
  :end) ))

(def-concept 'Style_bath_Do '(and room (fallin shape Shape_SBaD) ))
(def-concept 'Style_bed_Do '(and room (fallin shape Shape_SBeD) ))
(def-concept 'Style_bath_Hi '(and room (fallin shape Shape_SBaH) ))
(def-concept 'Style_bed_Hi '(and room (fallin shape Shape_SBeH) ))
(def-concept 'Style_bath_En '(and room (fallin shape Shape_SBaE) ))
(def-concept 'Style_bed_En '(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)
```

Conc. 1 `Modern_Design` is satisfiable:

because of the consistency of the overall RC constraints including the concept-local ones and those inherited from `Residence_Design` through “told” subsumption relationship;

Conc. 2 `Ensuit_Design` \sqsubseteq `Dorm_Design`:

(i) concepts `Style_bath_En` and `Style_bed_En` are subsumed by concepts `Style_bath_Do` and `Style_bed_Do` respectively, and (ii) RC constraints of `Ensuit_Design` is tighter than those of `Dorm_Design`;

Conc. 3 `Dorm_Design` \sqsubseteq `Residence_Design`, based on the entailment ordering between RC constraints;

Conc. 4 `Hitech_Design` \sqsubseteq `Morden_Design`:

the composite RC constraints of *Morden_Design* entails concept local RC constraints of *concept Hitech_Design*;

Conclusions

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 a HKB into smaller components, each containing the homogeneous 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 characteristics of *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, $\mathcal{DL}(D)/S$ which extends \mathcal{ALCN} and illustrated its usage in the context of *inference fusion* by means of an example.

An implementation of the *inference fusion* approach to extend DLs with the ability of modelling concrete knowledge is complete (Hu, Arana, & Compatangelo 2003). It fuses inferences from the FaCT DL-based taxonomic reasoning system (Horrocks 1999) and the $Ecl^i ps^e$ constraint reasoner (Brisset *et al.* 2001). Preliminary results are promising. Although no thorough analysis has been made, the computational complexity can be estimated as follows:

DL-based system: since we do not explicitly introduce any new types of reasoning or new constructors or operators, the complexity of the DL-based system is expected to remain unchanged. Meanwhile, by introducing a hybrid approach, we avoid the complex interventions between symbolic role number restrictions and other conceptual constructors by normalising the former with “wrapper” concepts. This removes one of the major sources of computational complexity (Baader & Sattler 1996) with regard to the extensions of DLs with concrete constraints, if, again, only the DL-based inference is considered.

Constraint reasoner: Finite Constraint Satisfaction Problems (FCSPs) are NP-complete as a general class (Mackworth & Freuder 1993). Pragmatic results show that the performance varies from system to system. For a thorough analysis on different constraint systems, please refer to (Fernández & Hill 2000).

Reasoning coordinator: on the general case, we expect the complexity of the overall coordinating algorithm to be $O(N^2)$ with regard to the size of the input HKB.

A formal evaluation of the implemented system 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

- Baader, F., and Sattler, U. 1996. Description Logics with Symbolic Number Restrictions. In *Proc. of the 12th European Conf. on Artificial Intelligence (ECAI'96)*, 283–287. John Wiley.
- Brisset, P.; Sakkout, H.; Frühwirth, T.; Gervet, C.; Harvey, W.; Meier, M.; Novello, S.; Le Provost, T.; Schimpf, J.; Shen, K.; and Wallace, M. 2001. *ECLⁱPS^e Constraint Library Manual, Rel. 5.2*. International Computers Ltd. and Imperial College London.
- Donini, F. M.; Lenzerini, M.; Nardi, D.; and Schaerf, A. 1996. Reasoning in description logics. In *Foundations of Knowledge Representation*. CSLI Publications. 191–236.
- Fensel, D.; van Harmelen, F.; Horrocks, I.; McGuinness, D.; and Patel-Schneider, P. 2001. OIL: An Ontology Infrastructure for the Semantic Web. *Intelligent Systems* 16(2):38–45.
- Fernández, A., and Hill, P. M. 2000. A Comparative Study of Eight Constraint Programming Languages over the Boolean and Finite Domains. *Jour. of Constraints* 5:275–301.
- Fikes, R., and Farquhar, A. 1999. Distributed Repositories of Highly Expressive Reusable Ontologies. *Intelligent Systems* 14(2):73–79.
- Horrocks, I., and Sattler, U. 2001. Ontology Reasoning in the $\mathcal{SHOQ}(D)$ Description Logic. In *Proc. of the 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI'01)*, 199–204. Morgan Kaufmann.
- Horrocks, I.; Sattler, U.; and Tobies, S. 2000. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL* 8(3):239–263.
- Horrocks, I. 1999. FaCT and iFaCT. In *Proc. of the Intl. Workshop on Description Logics (DL'99)*, 133–135.
- Hu, B.; Arana, I.; and Compatangelo, E. 2003. Facilitating dl-based hybrid reasoning with *Inference Fusion*. *Jour. of Knowledge-Based Systems*. (to appear).
- Jaffar, J., and Maher, M. J. 1994. Constraint Logic Programming: A Survey. *The Jour. of Logic Programming* 19 & 20:503–582.
- Lutz, C. 2001. NExpTime-complete description logics with concrete domains. In *Proc. of the Intl. Joint Conf. on Automated Reasoning*, number LNAI-2083 in LNAI, 45–60. Springer-Verlag.
- Mackworth, A. K., and Freuder, E. C. 1993. The Complexity of Constraint Satisfaction Revisited. *Artificial Intelligence* 59(1–2):57–62.
- Patel-Schneider, P. F., and Swartout, B. 1993. Description-logic knowledge representation system specification from the krss group of the arpa knowledge sharing effort.
- Schmidt-Schauß, M., and Smolka, G. 1991. Attributive concept descriptions with complements. *Artificial Intelligence* 48(1):1–26.
- Tsang, E. P. K. 1993. *Foundations of Constraint Satisfaction*. Academic Press.