The F-Logic Approach **for** Description Languages

Mira Balaban

Dept. of Mathematics and Computer Science

Ben-Gurion University of the Negev

P.O.B. 653, Beer-Sheva 84105, Israel

mira@bengus.bitnet

(972)-57-461622

Technical Report FC-93-02

Abstract

The Frame-logic (F-logic) approach of [20] is suggested as an underlying framework for description languages. F-logic is shown to provide a full account for description languages, without losing the direct semantics and the descriptive nature. It can support such desirable features as high order role fillers, collective entities, intensions, roles as first class objects and n-ary relationships. Yet, its semantics is first order. In an F-logic based description language, few description constructs are built in, and concepts, roles, **and** terminological operators are definable. Discussion of desirable features in descriptions is made possible within a single, uniform framework, that also coherently integrates with logic programming and deductive, object-oriented database technology. Typical descriptive operators can be defined in the language, thereby yielding a flexible description language, in which not all operators must be built in.

keywords: object-oriented representation, F-logic, description languages.

1 Introduction

Description languages (DLs) is a collective name for knowledge representation formalisms that concentrate on the management of essential descriptive vocabulary. It rests on the observation ([7]) that the natural ontology for providing information about a domain requires the ability to define, organize, and use intensional entities that stand for *concepts* and *roles* in a domain. DLs concentrate on providing means for:

- Definition of concepts and roles.
- Organizing concepts and roles in structures, called *taxonomies*, based on their analytical inter-relationships.
- Creating analytic descriptions using concepts and roles.
- Answering queries, by analyzing the analytic definitions of concepts/roles (and regardless of a temporary population of a knowledge base).

The main thrust of Description Languages/Logics (DLs) is given within their name: They concentrate on descriptional issues, i.e., on constructs that faithfully reflect their intended ontologies and processes. Hence, DLs support built-in *terminological* operators for building complex descriptions, where the meanings of the operators are directly defined in terms of the intended entities. The various DLs vary in their terminological operators' vocabulary, in the amount of separation between their terminological and contingent components, and the strict commitment of their inference algorithms to the well defined direct semantics. However, all DLs insist on terminological operators and direct semantics.

The status of descriptions, and the emphasis on direct semantics is the major distinction between DLs to other logics in AI. DLs are unique in their provisions for complex descriptions. Claimed possible embeddings of DLs in classical logic (e.g., [16]) lose the direct semantics of complex descriptions, which is the bare bone of DLs. That is, DLs can be embedded in classical logic, but the resulting classical language **is not a DL**, as descriptions are translated into collections of independent assertions.

The two main trends in DLs today are:

1. Small languages, supported by sound, complete, and tractable inference algorithms ([21, 36, 5, 31]).
2. Rich languages, supported by possibly incomplete inferences ([25, 27, 51, 11]).

In general it seems that there is a growing agreement that DLs should be strengthened to meet more faithfully users' requirements. [51] seems to suggest complete retrospection into the basic assumptions. Some requirements and expectations (see [28]) are:

- Wider representational and inferential mechanisms: Non-taxonomic relations like connectivity, n-ary relations ([49, 51, 3]), part-whole relations ([14]), higher order constructs ([42]), collective entities ([12]), default assumptions and subsumption ([43, 1, 35, 38, 11]), intensions and views ([51]), cyclic terminological definitions ([25, 27, 34]), and self references ([48]).
- More flexible definitions ([11, 51]).
- Uniform proof theory ([45]).
- Integration with logic programming, object-oriented systems, functional programming, and database technology ([15, 41, 30]).
- Incremental development of implementations ([4]).

F-logic ([20, 19]) is a logic designed for reasoning about object-oriented domains. It assumes an ontology of objects that form a hierarchy, and are also inter-related by a membership relation. Information about the objects is given in methods or attributes that are attached to objects, and in types of these methods. F-logic provides means for typing and type checking, and supports reasoning about inheritance of types and of methods. Its syntax allows for high order constructs, like quantification over methods, but its semantics is carefully kept first order, since quantification is restricted to namable methods alone. A sound and complete proof theory is given in [20].

In this paper we suggest to use the F-logic approach as a basis for a rich description language that can be termed DFL (for Description F-logic Language). The advantage would be that of a full fledged logic that integrates notions of description, reasoning, object-orientation and logic programming; can serve as a unifying formalism for current description languages; and can extend their expressivity in the listed above directions. In particular, DFL will have few term constructors directly built into its descriptive intensional semantics, will have a uniform proof theory, and will be able to coherently integrate with logic programming and object-oriented deductive databases. Term forming operators of DLs that are not built into the semantics, can always be **axiomatized** in the language, thereby yielding a flexible DL. Specialized DLs' inference algorithms, like algorithms for normalization of descriptions, subsumption and classification, can be integrated into the general deductive machinery for processing DFL knowledge bases. The DLs' algorithms can be viewed as a special purpose inference component, built into the general one.

Axiomatization of term forming operators in a Description F-logic Language is **markedly** different from embedding in predicate calculus, since DFL has a typical DL's ontology. Hence, all operators and other components of the language

can be given their direct, natural intended meaning, and apart from possible minor syntactical differences, there is no translation algorithm. Instead, this view amounts to building high level conceptualizations on top of a lower level one, that provides certain primitives as building blocks. The main point is that the low and high level representations have the same semantics. The theme of this paper is that F-logic, although not in itself a description language (the management of analytic definitions of concepts/roles is not its major focus), has the potential for becoming a general description language, since the core of its ontology is the *objects' hierarchy* and the associated *methods*, and its descriptive capabilities include *object constructors*, that can support complex descriptions.

In the rest of this paper we, first, shortly introduce DLs (Section 2) and F-logic (Section 3). In Section 4 we discuss the potential of F-logic as a basis for a Description F-Logic (DFL), and argue that standard DLs form a subset of F-logic. Section 5 discusses the way F-logic can account for desirable extensions for DLs, and Section 6 concentrates on issues in the integration of a terminology into an F-logic knowledge base. Section 7 is the conclusion, and proofs are postponed to the Appendix.

2 Description Languages

Description languages form a spinoff of the KL-ONE school ([9, 52]), that concentrates on languages that provide constructs for management of analytic domain terminology. DLs emphasize the importance of direct, well defined semantics, and of a limited, but tractable, inferential service. DLs assume that a typical domain's terminology is built around *concepts* and *roles*, which stand for subsets and binary relations over a domain of individuals, respectively. The concepts are assumed to form a *taxonomy*, i.e., a partially ordered structure that is derived from the lattice of subsets. Accordingly, the alphabet of a DL includes an unlimited supply of symbols for concepts, roles and individuals, and a small set of term forming operator symbols. Descriptive terms are formed by applying term forming operators to concept/role symbols. For example, the concept of a *frozen-dinner*, i.e., a ready-made evening meal, whose courses are all frozen, and at least one is spicy, can be described by the concept term:

$$
\begin{array}{ll}
\textbf{and}(& \mathit{ready\text{-}made\text{-}meal}, \\
 & \textbf{some}(\mathit{eating\text{-}time},\ \mathit{evening}), \\
 & \textbf{all}(\mathit{course},\ \textbf{and}(\mathit{food},\ \textbf{some}(\mathit{form},\ \mathit{frozen}),\ \textbf{all}(\mathit{form},\ \mathit{frozen}))), \\
 & \textbf{some}(\textbf{compose}(\mathit{course},\ \mathit{taste}),\ \mathit{spicy}))
\end{array}
\tag{1}
$$

In this term *ready-made-meal* is a *concept symbol*, intended to represent the concept of a ready made meal; *eating-time* is a *role symbol*, intended to represent the role (binary relation) of eating time; **some**(*eating-time*, *evening*), is a concept term that represents the concept of all elements that are related via *eating-time* to some element from the *evening* concept; **all**($form$, $frozen$) is a concept term that represents the concept of all elements that are related via the *form* role only to elements of the *frozen* concept; **compose**(*course*, *taste*) is a role term that represents the composition of the *course* and *taste* roles.

A *terminology* is formed by associating concept/role symbols with definitions, which are descriptive terms. For example, the above concept term can be used to define the *defined-concept* symbol *frozen-dinner*. The *defined-role* symbol *course.taste* can be defined as the composition of the *course* and *taste* roles:

$$\begin{aligned} frozen\text{-}dinner &\doteq \langle term\ 1 \rangle \\ course.taste &\doteq \textbf{compose}(course,\ taste) \end{aligned}$$

Most DLs concentrate on defined concepts, and include only limited capabilities for forming non-atomic role terms (if at all).

The currently standard semantics of DLs is set theoretic: Concept terms are interpreted as denoting sets of domain elements, role terms denote binary relations over the interpretation domain, and definitions are interpreted by set equality.

A typical DL knowledge base has, in addition to its terminological component, a component that describes knowledge about individual objects. For example:

$$dinner.9.5 \ : \ frozen\text{-}dinner$$

asserts an individual that belongs to the *frozen-dinner* concept. The individuals' component is usually referred to as the *Assertional KB*, or *A-box*: It manages knowledge about individual concrete objects. The terminological component, i.e., concept/role definitions is usually referred to as the *Terminological KB*, or *T-box*. In addition, various systems allow formulae like restricted rules. For example:

$$frozen\text{-}dinner \longrightarrow lousy\text{-}meal$$

can be a rule that implies that all individuals that belong to frozen-dinner concept also belong to the lousy-meal concept.

The central management service provided by a description KR system is the automatic classification of concepts (roles) into their proper position in the taxonomy, based on their descriptions. The taxonomy is used for answering queries about the inter-relationships between concepts/descriptions, about the properties of concepts and individuals, and about membership of individuals in concepts. The conventional method is to classify, i.e., find the proper position in

the taxonomy, for every description that appears in a query. The answer to the query, then, results from observing possible subsumption relationships between concepts/descriptions that appear in the query, based on their relative position in the taxonomy.

Inferencing in DLs is essentially different from standard inference in AI, or in logic databases. The main point of diversion is that DL systems consider definitions in the vocabulary, not a temporary population in a knowldege base. For example, the query:

Are all children of c doctors?

is understood as an essential query about c. That is, this query is understood as the query:

Is c subsumed by **all**(child, doctor) ?

and not as the query:

Are all known children of c members of the doctor concept?

The latter query, demonstrates the typical database and AI approaches, that operate under the Closed World Assumption (CWA). In contrast, DLs can be understood as operating under the Open World Assumption (OWA), where a momentary population in a knowledge base does not provide sufficient evidence on essential definitions in the terminology. DLs are unique in developing knowledge bases for essential domain vocabularies.

Determination of subsumption between concepts' (roles') descriptions is the main inferential task, since all other inferential operations are reduced to subsumption. Hence, major efforts were devoted to the study of the subsumption problem in the context of different terminologies ([22, 37, 46, 33]), to the study of subsumption/classification algorithms in existing systems ([2, 31]), and to the development of subsumption algorithms ([44, 17, 47].

The conclusion of this intensive study of subsumption is that keeping subsumption sound, complete, and tractable is not feasible, and some criteria must be relaxed. That is, in order to keep the subsumption problem tractable, the set of term forming operators has to be severely limited ([46, 36, 22]). [32] lists known complexity results for subsumption, and for subsumption algorithms in different systems. CLASSIC ([5]) is a formal, well defined system with limited expressivity, but tractable and complete inference machinery. Nevertheless, growing experience with existing systems shows that the idea of a small, tractable and complete DL component that will be integrated within a larger knowledge base system (as suggested in [36]) is not realistic, mainly since users of a restricted DL knowledge base use ad-hoc methods that jeopardize the soundness of the whole system ([11]). The LOOM system ([26]), takes a contrary approach to that of CLASSIC: It

emphasizes the strong expressivity of the language, at the expense of completeness. The BACK system ([23, 24]) is a formal, well defined system, which is more expressive than CLASSIC, with an incomplete subsumption algorithm. [2] reports on experiments in approximating completeness in an expressive DL.

Most presentations in [28] emphasize the need for extending the expressive power and inferential capabilities of DL systems. Another important point seems to be the integration of DL systems with existing systems, like conventional data bases. The main point here is that experience shows that both, the conventional approach, that operates under the CWA, and the definitional approach, that operates under the OWA are needed, and may, ideally, complement each other. A recent ongoing effort to standardize description languages is reported in [40].

2.1 Formal Definition of Description Languages

Recall that a terminology is a set of definitions of the form

$$\textit{defined-concept/role-symbol} \doteq \textit{concept/role-description}$$

The formal definition of DLs distinguishes the concept/role symbols that are defined (appear at the left side of a definition in the terminology), from those that are not. The defined symbols are called *concept/role name symbols*, and the rest are *primitive concept/role symbols*. The concept/role symbols consist of all primitive and name concept/role symbols. The formulae of DLs are of three kinds:

1. Definitions, e.g., a parent is a person with at least one child:

$$\textit{parent} \doteq \textbf{and}(\textit{person}, \textbf{at-least}(1, \textit{child}))$$

2. Rules, e.g., if an individual is a student then s/he eats junk-food:

$$\textit{student} \Rightarrow \textbf{all}(\textit{eat}, \textit{junk-food}))$$

3. Isa assertions, e.g., John is Mary's child:

$$(\textit{Mary}, \textit{John}) : \textit{child}$$

As an example for a typical DL we adopt the DL presented in[45], but we let the set of concept/role forming operators vary.

Syntax:

Symbols of the language: Primitive concept symbols (c_p), concept name symbols (c_n), primitive role symbols (r_p), role name symbols (r_n), object symbols (o), two special concept symbols, *top* and *bottom*, and a finite set P of concept and role forming operators.

Terms: Terms are either *concept terms*, or *role terms*. Concept/role terms are all concept/role symbols, and all syntactically legal applications of a concept/role forming operator to terms, respectively. A concept term of any kind is denoted c, and a role term of any kind is denoted r.

Formulae[1]: $c_n \doteq c$; $r_n \doteq r$; $c_1 \Rightarrow c_2$; $r_1 \Rightarrow r_2$; $o : c$; $(o_1, o_2) : r$.

Description languages with this set of formulae differ from each other just in the sets of allowed operators. Hence, we denote a typical DL with operators set P by L_P.

Semantics:

Meaning of formulae is given by a set theoretic semantics. An *interpretation* **I** is a pair (D, Υ), of a domain D and an interpretation function Υ, such that concept symbols are assigned subsets of D, role symbols are assigned binary relations over D^2, object symbols are assigned elements of D, $\Upsilon(top)$ is D, and $\Upsilon(bottom)$ is $\emptyset$ (the empty set). The interpretation function Υ is augmented to all concept/role terms by building into it a fixed meaning for each operator in P. For example, for $P = \{$ **and, all, some, and-role, inverse** $\}$, the meaning of concept terms is defined as follows:[3]

$$
\begin{aligned}
\Upsilon(\ \textbf{and}(\ c_1,\ c_2\ )\ ) &= \Upsilon(c_1) \cap \Upsilon(c_2) \\
\Upsilon(\ \textbf{all}(\ r,\ c\ )\ ) &= \{\ d \in D / \Upsilon(r)(d) \subseteq \Upsilon(c)\ \} \\
\Upsilon(\ \textbf{some}(\ r,\ c\ )\ ) &= \{\ d \in D / \Upsilon(r)(d) \cap \Upsilon(c) \neq \emptyset\ \} \\
\Upsilon(\ \textbf{and-role}(\ r_1,\ r_2\ )\ ) &= \Upsilon(r_1) \cap \Upsilon(r_2) \\
\Upsilon(\ \textbf{inverse}(\ r\ )\ ) &= \{\ (d, e) \in D \times D / (e, d) \in \Upsilon(r)\ \}
\end{aligned}
$$

Satisfaction of formulae in an interpretation is defined by interpreting $\doteq$ as set equality, $\Rightarrow$ as set inclusion, and $:$ as membership over D. An interpretation is a *model* for a set of formulae if it satisfies all formulae. A formula γ is logically implied from a terminology T, i.e., T $\models \gamma$, if it is satisfied in every model of the terminology.

[1] Different notations have been used in the DLs' literature. Primitive definitions have been denoted also using $\sqsubseteq$ or :< ; membership have been denoted also using $\in$ or :: ; $\Rightarrow$ was also used for denoting subsumption. The above formulation follows the notation in [45].

[2] In [8, 34], role symbols are assigned total functions from D to $P(D)$. This interpretation of role symbols is somewhat closer to F-logic's view of methods.

[3] We view relations as set-valued functions, whenever it simplifies the presentation. In the definitions of the meaning of the operators, we let, for a role symbol r and a domain element d, $\Upsilon(r)(d)$ denote the set of all domain elements related to d via $\Upsilon(r)$, i.e., the set $\{\ e / (d, e) \in \Upsilon(r)\ \}$.

A concept term t is *coherent* with respect to a terminology T, if there exists a model (D, Υ) of T, such that $\Upsilon(t) \neq \Phi$. The main relationship between terms is the *subsumption* relation: Term t_1 is subsumed by term t_2 ($t_1 \sqsubseteq t_2$) in a terminology T, if and only if in every model (D, Υ) of T, $\Upsilon(t_1) \subseteq \Upsilon(t_2)$. *Equivalence* of terms is defined as two way subsumption. The central role that the subsumption relationship between concept terms plays in the management of DL knwoledge bases was already discussed above.

Proposition 2.1 *$t_1 \sqsubseteq t_2$ in T, iff $T \models \quad t_1 \Rightarrow t_2$.*

Proof: Immediate from definition of subsumption, and semantics of $\Rightarrow$. $\square$

3 F-Logic

An F-logic domain U consists of *objects* and *methods*. In addition, there are *object constructors*, which are functions defined on objects, a *partial ordering* $\preceq_U$ on objects, that stands for the subset relationship, and a binary relation $\in_U$ on objects, that stands for the membership relationship. A condition set on $\preceq_U$ and $\in_U$ guarantees that membership in an object is extended to a super-class object. The underlying intensional approach assumes that the "essence" of an object lies in its behavior. Hence, objects can be just anything that we wish to talk about, like *Mary*, *Mary's car*, *the cars of Ben-Gurion University employees*, and the role of being a *mother*. In particular, there is no a priori distinction between individual to class objects, i.e., an object can be both, depending on its behavior/relationship to other objects. If $o_1 \preceq o_2$, then o_1 is understood as a subset of o_2; if $o_1 \in o_2$, then o_1 is understood as a member of o_2 (the subscript U is omitted, for simplicity). This approach is particularly powerful, since a collective entity can be viewed, both, as a class of objects, and as an individual object, that can be a member of another (class viewed) object. Another advantage is that a singleton is identified with its member. For example, the following "chains" of $\in$, $\preceq$ relations can be present in U:

$$maryAsChild \preceq mary \in studentCommittee \begin{cases} \preceq & student \\ \in & univCommittee \end{cases}$$

Of course, a distinction between individuals to classes (concepts), and other distinctions can be enforced by imposing sorts.

Methods are partial functions of objects. There are single-valued (scalar) and set-valued methods. Methods describe the *behavior* of objects, and provide *information* about objects. For example, *spouse-of*, *children-of*, and information on

a bank-account, can be captured by methods. The *children-of* method is an example of a method that takes additional arguments: "children-of an object o_1 with another object o_2", is an application of the method on o_1, with o_2 as an extra argument (or in the *context* of o_2). A method like *spouse-of*, that does not take additional arguments (i.e., a function of one argument) is called an *attribute*. *Name-of*, *age-of*, *address-of*, are all attributes. Methods that describe actions, like *buy*, *meet*, *treat*, etc., can typically take additional arguments, for all the parameters of the actions. These are n-ary functions, (n $>$ 1).

Methods are also classified into *inheritable* and *non-inheritable*. For example, *color* is an inheritable attribute of *bear*, *averageSalary* is a non-inheritable attribute of $faculty$, and *children-of* is a non-inheritable set-valued method of $Mary$ (since various specializations of $Mary$, e.g., $MaryAsChild$ need not inherit the value of *children-of* at $Mary$). The inference machinery uses the distinction between inheritable to non-inheritable methods for propagating values of inheritable methods, down the $\preceq$ hierarchy, as long as no overwriting is caused. Inheritance extends also into the $\in$ relationship, but is blocked after one step. This way, $Mary$, being a *student*, inherits the *registered-in-college* attribute-value pair, while its $MaryAsChild$ specialization does not inherit this property; $studentCommittee$ can inherit the *committee-size* property of $univCommittee$, but inheritance does not extend to Mary.

The "secret" behind F-logic is the *high-logization* of methods: Methods and types are **reified** by their object-names, and quantification over them is carried just over their object-names. Section 3.2 shortly summarizes the main ideas in the semantics of F-logic.

Observation: An F-logic ontology is already a description language ontology.

Argumentation: The ontology of a description language consists of *concepts, roles* and *individuals*. In an F-logic's ontology we have objects and methods. The objects capture both, concepts and individuals. The difference appears to be that F-logic's descriptions are formed with methods, while traditional DLs' descriptions are formed with roles. The following table summarizes the correspondence between DLs' roles to F-logic's methods:

DLs	F-logic
n-ary feature	single-valued method
n-ary role	set-valued method
(binary) feature	single-valued attribute
(binary) role	set-valued attribute

We argue that replacing methods for roles provides further flexibility and expressivity, while not losing the descriptive character. With methods, finer distinctions among the different kinds — attribute-features, non-attribute-features,

attribute-roles, non-attribute-roles — are clearly made in the ontology, and are easy to capture in the syntactical representation (see below). Moreover, the methods ontology can be naturally integrated with conventional representational constructs, like "plain" functions and relations. In the rest of this paper we use the DL terminology: An *attribute-feature* is a unary single-valued method, an *attribute-role* is a unary set-valued method, a *non-attribute-feature* is an n-ary single-valued method (n > 1), a *non-attribute-role* is an n-ary set-valued method (n > 1), a *feature* is a single-valued method of any arity, a *role* is a set-valued method of any arity. The term *method* refers to features and roles, indifferently.

3.1 F-logic Syntax

The terms of F-logic are expressions that denote objects in the domain. For example, *mary*, 3, **and**(*polygon, 3Sides*), **and**(*polygon, 3Angles*), *cars-of(employees(bgu))*, are *id-terms*, denoting objects. The id-terms **and**(*polygon, 3Sides*) and **and**(*polygon, 3Angles*) may denote two distinct intensional objects with the same extension, i.e., the set of triangles. In the above id-terms, **and**, *employees*, and *cars-of* are *object constructors*: They denote total functions on the domain U, that map objects to objects. Variables can also appear in id-terms, as in classical first order logic (we use capital first letters to denote variables).

The atomic formulae of F-logic, called F-molecules, are of three kinds: *is-a* F-molecules, *data* F-molecules, and *signature* F-molecules.

1. Is-a F-molecules map to the partial ordering, and to the membership relation on U. For example,

Is-a F-molecules	Meaning
mary : woman	The object denoted by *mary* is $\in_U$ related to the one denoted by *woman*.
woman :: person	The object denoted by *woman* is $\preceq_U$ related to the one denoted by *person*.
and(*polygon, 3Sides*) :: *polygon*	Subset relationship ($\preceq_U$) between the denotations of the two id-terms.

2. Data F-molecules are assertions about the values that methods (features, roles) get on objects. For example:

Data F-molecules	Meaning	Explanation
mary[*husband-of* $\rightarrow$ *fred*]	Fred is Mary's husband.	*husband-of* is an *attribute-feature*.
bear[*color* $\bullet\!\!\rightarrow$ *grey*]	Bears are grey.	*color* is an *inheritable attribute-feature*.
mary[*teach* $\twoheadrightarrow$ {*aut.*, *graph.*}]	Mary teaches aut. & graph.	*teach* is an *attribute-role*.
bear[*color* @ north $\bullet\!\!\rightarrow$ *white*]	Northern bears are white.	*color* is an *inheritable non-attribute-feature*.
son(*m*)[*children*@*j* $\twoheadrightarrow$ {*pat*}]	Pat is a child of m's son with j.	*children* is a *non-attribute-role*.

3. Signature F-molecules are assertions about the types of features and roles. For example,

$$\text{X}[\ \textit{husband-of} \Rightarrow (\textit{male})\ ]$$

$$\text{X}[\ \textit{children} \ @\ \text{Y} \Rightarrow\!\!> (\textit{person})\ ]$$

where X and Y are assumed to be universally quantified, assert that values of the *husband-of* feature must be males, and the values of the *children* role must be of type *person*. The rule:

$$\text{X} : \textbf{and}(\ \textit{female, married}\ ) \longleftarrow X[\ \textit{husband-of} \Rightarrow ()\ ]$$

where X is assumed to be universally quantified, asserts that if the feature *husband-of* applies to an object **o** denoted by X, then **o** must be a member of the **and**(*female, married*) class, i.e., a married female [4].

F-logic includes also regular atomic formulae of a first order language. Its formulae are constructed using connectives and quantifiers in the usual first order manner. Here are some examples, taken from [20] (the syntax is somewhat relaxed and simplified):

1. $faculty[\ boss \Rightarrow (faculty,\ manager)$;

   $age \Rightarrow midaged$;

   $highestDegree \Rightarrow degree$;

   $papers \Rightarrow\!\!> article$;

   $highestDegree \bullet\!\!\rightarrow phd$;

   $avgSalary \rightarrow 50000]$

   This statement declares typing information (i.e., signatures) for the attribute-features *boss*, *age*, and *highestDegree* of $faculty$, and for the attribute-role *papers* of $faculty$. Also, the value of the inheritable attribute-feature *highestDegree* at $faculty$ is specified as *phd*, and the value of the non-inheritable attribute-feature *avgSalary* at $faculty$ is specified as 50000.

2. $Car : dieselCars(Year) \longleftarrow Car : car[\ engineType \rightarrow \text{“}diesel\text{”};$

   $makeYear \rightarrow Year\ ]$

   This statement defines a family of classes, parameterized by $Year$. Each class, e.g., $dieselCars(1900)$ contains all diesel cars made in 1900.

[4] A rule is an implication whose conclusion is an F-molecule.

3. Polymorphic typing:

$list(T)[\ first \Rightarrow \mathrm{T};$

$rest \Rightarrow list(T);$

$length \Rightarrow int;$

$append\ @\ list(T) \Rightarrow list(T)\ ]$

4. Knowledge base browsing:

$interestingAttributes(X)[\ attributes\ \twoheadrightarrow\ L]\ \longleftarrow\ X : faculty[\ L \rightarrow Z : person\ ]$

$interestingAttributes(X)[\ attributes\ \twoheadrightarrow\ L]\ \longleftarrow\ X : faculty[\ L \twoheadrightarrow Z : person\ ]$

These rules define, for every member, **o**, of the *faculty* object, a new object, $interestingAttributes(\mathbf{o})$, with a set-valued attribute, *attributes*, whose value at that object is the set of all attributes of o that have a *person* value.

5. Concepts' similarity via analogy, as in "A Pig-Like-Person is similar to a Pig by his/her nose, legs, and smell" (an example borrowed from [16]:

$mary[\ smell \rightarrow P\ ]\ \longleftarrow\ like(mary,\ pig,\ [nose,\ legsform,\ smell]) \wedge pig[smell \rightarrow P].$

This rule states that if *mary* is similar to a pig by her nose, legsform, and smell, and if *pig*'s smell is some value P, then *mary*'s smell is also P (we use PROLOG's lists' notation). This rule can be generalized into any *like* similarity, as follows:

$X[\ Attr \rightarrow P\ ]\ \longleftarrow\ like(X,\ Y,\ PropList) \wedge Y[Attr \rightarrow P] \wedge member(Attr,\ PropList).$

6. Methods' dependency:

$X[heightClass\ \rightarrow\ tall]\ \longleftarrow\ X : person[height\ \rightarrow\ H]\ \wedge\ H \geq 1.8m$

The following subsection summarizes the semantics of F-logic. It is not essential for the rest of the paper.

3.2 F-logic Semantics

Methods are reified by objects of the domain in a semantic structure. The reification is accomplished by associating, with each object, a feature (or actually, infinity of features, one for each arity), a role (again infinity), an inheritable feature (infinity), an inheritable role (infinity), a type for the feature (infinity), and a type for the role (infinity). We can think about an object **d** of U as an association:

(**d**,

array-of-features, array-of-roles, array-of-inheritable-features, array-of-inheritable-roles, array-of-feature-types, array-of-role-types)

Features, roles, and their types are always referenced indirectly, via their object-names in U, and the specified arity. This is the "secret" behind the high-logization of F-logic: Features, roles and types are **reified** by their object-names, and quantification over them is carried just over their object-names. The association is given by six functions, $I_{\rightarrow}$, $I_{\rightarrow\!\!\!\!\rightarrow}$, $I_{\bullet\rightarrow}$, $I_{\bullet\rightarrow\!\!\!\!\rightarrow}$, $I_{\Rightarrow}$, and $I_{\Rightarrow\!\!\!\!\Rightarrow}$, that assign to an object **d** the six mentioned above arrays of features, roles, inheritable-features, inheritable-roles, types-of-features, and types-of-roles, respectively. ($I_{\rightarrow}^{(k)}$ denotes the k+1 element of $I_{\rightarrow}$, for k≥0; the superscript corresponds to the number of additional arguments that the feature/role takes, besides **d**.)

A semantic structure for an F-logic language is a tuple $\mathbf{I} = \langle U, \preceq_U, \in_U, I_F, I_{\rightarrow}, I_{\rightarrow\!\!\!\!\rightarrow}, I_{\bullet\rightarrow}, I_{\bullet\rightarrow\!\!\!\!\rightarrow}, I_{\Rightarrow}, I_{\Rightarrow\!\!\!\!\Rightarrow}\rangle$, where $(U, \preceq_U)$, and $\in_U$ are the partially ordered domain, and the membership relation, as described in the previous section. I_F is the interpretation mapping for object constructors, a standard function mapping. The other six mappings are the associations of domain objects with features, roles, and types, as explained above.

Is-a F-molecules are assertions about subset relationships ($\preceq_U$) and membership relationships ($\in_U$) between objects denoted by id-terms in the molecule.

Data F-molecules are assertions about the value of a feature or a role at a given object. In a data F-molecule with a non-inheritable-feature:

$$o[f@arg_1, \ldots, arg_n \rightarrow val],$$

$o, f, arg_1, \ldots, arg_n, val$, are id-terms. In a given semantic structure and variable assignment $\mathbf{I}$, the id-terms $o, arg_1, \ldots, arg_n$, val, are mapped to objects of the partially ordered domain (U, $\preceq_U$). The id-term f is mapped to the non-inheritable feature with n arguments, associated with the object to which the symbol f is mapped, i.e., to $I_{\rightarrow}^{(n)}(\ f^{\mathbf{I}}\ )$. The term is true in $\mathbf{I}$ if $I_{\rightarrow}^{(n)}(f^{\mathbf{I}})$ is defined in ($o^{\mathbf{I}}, arg_1^{\mathbf{I}}, \ldots, arg_n^{\mathbf{I}}$) and equals $val^{\mathbf{I}}$. For example, $\mathbf{I} \models mary\ [\ husband\text{-}of \rightarrow\ fred\ ]$, means that the non-inheritable feature attribute (1-ary method) associated with $husband\text{-}of^{\mathbf{I}}$ has the value $fred^{\mathbf{I}}$ at $mary^{\mathbf{I}}$.

The meaning of data F-molecules with non-inheritable-roles, that make assertions about role values, is similar:

$$\mathbf{I} \models o[r@arg_1, \ldots, arg_n \twoheadrightarrow \{val_1, \ldots, val_n\}]$$

holds in $\mathbf{I}$ if $I_{\twoheadrightarrow}^{(n)}(r^{\mathbf{I}})$ is defined at ($o^{\mathbf{I}}$, $arg_1^{\mathbf{I}}, \ldots, arg_n^{\mathbf{I}}$), and its set value includes the set { $val_1^{\mathbf{I}}, \ldots, val_n^{\mathbf{I}}$ }.

The meaning of inheritable data F-molecules is defined similarly, using the $\bullet\!\rightarrow$ and the $\bullet\!\twoheadrightarrow$ mappings. Inheritable data F-molecules are used to select a preferred (canonical) model for an F-logic program. The rational behind the preference

criterion is that inheritable features/roles should, preferably, propagate down the $\preceq_U$ hierarchy in an interpretation, to objects where their values are undefined. The propagation is blocked by the $\in_U$ relation, where a single step inheritance can still apply.

Signature F-molecules assign types to features and roles. A signature F-molecule with a feature:

(2) $$o[f@arg_1, \ldots, arg_n \Rightarrow (val_1, \ldots, val_m)],$$

serves as a typing expression for two kinds of applications of $f^{\mathbf{I}}$: Application of $f^{\mathbf{I}}$, as a non-inheritable feature, to objects $o'^{\mathbf{I}}$ that are members of $o^{\mathbf{I}}$, and applications of $f^{\mathbf{I}}$, as an inheritable feature, to objects $o'^{\mathbf{I}}$ that are subclasses of $o^{\mathbf{I}}$. That is, molecule 2 provides typing to the (feature) data F-molecules:

1.

(3) $$o'[f@arg'_1, \ldots, arg'_n \rightarrow val],$$

where o' : o, and $arg'_i : arg_i$, for $1 \leq i \leq n$, hold in $\mathbf{I}$.

2.

(4) $$o'[f@arg'_1, \ldots, arg'_n \bullet\!\!\rightarrow val].$$

where o' :: o, and $arg'_i : arg_i$, for $1 \leq i \leq n$, hold in $\mathbf{I}$.

Molecules 3 and 4 are correctly typed by signature 2 if $val^J \in val_i^J$, for every $1 \leq i \leq m$. Signature F-molecules with roles account for the typing of non-inheritable and inheritable data F-molecules with roles, in a similar fasion. The type correctness conditions, enforced on F-logic programs, requires that all feature/role data F-molecule are correctly typed by all signature F-molecules that can serve as their typing expressions. For example,

(5)

son-only[*children* @ *Y* $\Rightarrow\!\!>$ (*person, male*);
avgChildNo $\rightarrow$ *4*; /* Note that this is not a typing expression; it has to be typed by yet another typing expression (see (2) below) */
wishedChildGender $\Rightarrow$ (*gender*)]

enforces the following typing:

1. The first signature expression restricts the values that the non-inheritable role associated with $children^{\mathbf{I}}$ can take. It says that at objects o that are members of $son\text{-}only^{\mathbf{I}}$, these values must be members ($\in$ related) of $person^{\mathbf{I}}$

and $male^{\mathbf{I}}$. The formal expression of this restriction is: $I^{(1)}_{\rightarrow\rightarrow}(children^{\mathbf{I}})\ (o^{\mathbf{I}}, Y^{\mathbf{I}}) \in person^{\mathbf{I}}$, and $I^{(1)}_{\rightarrow\rightarrow}(children^{\mathbf{I}})\ (o^{\mathbf{I}}, Y^{\mathbf{I}}) \in male^{\mathbf{I}}$, for every semantic structure and variable assignment $\mathbf{I}$ of 5.

2. The non-inheritable feature associated with $avgChildNo^{\mathbf{I}}$ must be correctly typed by another signature F-molecule like

   $$people\text{-}group[\ avgChildNo \Rightarrow integer],$$

   where $son\text{-}only^{\mathbf{I}} \in people\text{-}group^{\mathbf{I}}$, and $4^{\mathbf{I}} \in integer^{\mathbf{I}}$, for every semantic structure and variable assignment $\mathbf{I}$ of 5.

3. The value of the inheritable feature associated with $wishedChildGender^{\mathbf{I}}$ at objects o that are subclasses of $son\text{-}only^{\mathbf{I}}$ must be members of $gender^{\mathbf{I}}$.

4 The Potential of F-logic as a General Description Language (DFL)

In section 3 we observed that F-logic's ontology is a typical DL's ontology. In this section we observe F-logic's languages, and show that standard DL's are natural subsets of such languages. This observation implies that a potential DFL can provide a uniform roof for specialized DLs. First we consider standard constructs of description languages, and observe which constructs have analogous constructs in F-logic, which constructs are missing in F-logic, and which constructs have different analogous constructs in F-logic. Then, in 4.2, we explore how F-logic can account for terminological operators. In particular, we show that expected subsumption relationships, that result from the intended meaning of the operators, are preserved. In 4.3 we formally compare the standard set-theoretic semantics of DLs, with the object-oriented semantics of F-logic. We show that no DL property is lost if we consider DL expressions and formulae as F-logic id-terms and F-molecules. The conclusion of this section is that F-logic has a natural DL ontology, all DL constructs either already exist or can be added to F-logic, all DL constructs can be directly built into F-logic's semantics, or alternatively, be axiomatized, and nothing is lost along the switch from the DL's set-theoretic semantics to the more conventional object-oriented semantics. Hence, F-logic seems to provide a good basis for the development of a general DL, termed DFL.

4.1 Description-oriented view of F-logic

The objects in an F-logic's ontology are the counterpart of concepts in a DL's ontology. The partial ordering $\preceq_U$ stands for concept subsumption, and the binary relation $\in_U$ stands for membership of an object in a concept. Objects are not split into disjoint sets of concepts (classes) and objects (individuals), although such a separation can be enforced in a

sorted version of an F-logic language. The advantage in such **overloading** of the Object notion is that an object can serve, depending on context, both as an individual and as a concept. When two objects are $\preceq_U$ related, they are both interpreted as concepts; when two objects are $\in_U$ related, the first is interpreted as an object element in the second, which is taken as a concept. This uniformity enables us to account for collective entities and higher order role-fillers.

The methods in an F-logic's ontology are the counterpart of roles in a DL's ontology. We have already argued that replacing methods (functions) for roles (binary relations) is advantageous, as it leads to finer characterization of different kinds of methods, can naturally account for n-ary roles (e.g., action-roles that take additional arguments), and supports the distinction between roles to plain relations.

Turning now to consider expressions of F-logic vs. expressions of DLs, we see that there are three major differences:

1. The meaning of object constructors is not built into the semantics of F-logic. Hence, in order to use standard terminological operators such as **and** or **some**, they should be either axiomatized or built into F-logic's semantics.

2. Since methods are functions and not binary relations as roles are, their specifications take the form of asserting a value for an object (possibly with additional arguments), rather than specifying the membership of a pair in a role.

3. Relationships between methods, such as equality and implication are not built into F-logic's semantics. Hence, the specification of such relationships should be unfolded into relationships between the values that methods take on objects.

The last difference points into an essential DL construct that is, indeed, missing in F-logic. In a Description F-logic (DFL), equality and implication of roles and features should be built into the semantics (as objects' equality is). The second point of difference is just cosmetic, and the advantage of using methods instead of roles has already been discussed above.

The semantics of standard terminological operators must be taken care of in a DFL. If the decision is in favor of implementing terminological operators in F-logic's semantics, this can be done, as each such operator has a direct meaning in F-logic's ontology. For example,

- The **and** terminological operator is the **greatest-lower-bound (glb)** operator on $(U, \preceq_U)$.

- The **all** operator is a type restriction on a method.

- The **at-least**, **at-most**, and **some** operators are cardinality restrictions on methods, etc.

Yet, the decision about direct embedding of terminological operators in F-logic's semantics should be carefully considered, since the available pool of terminological operators in the current DLs standard [40] is quite large. It seems preferable to implement just a small number of primary operators, and to take all others as definable. This way, the logic can be kept stable, and would not need to be changed with every little change in the set of terminological operators. This approach is significantly different from the current DL's convention, where in each DL, **all** terminological operators are built into the semantics. In particular, our approach can lead to a uniform representation for all DLs, and can provide a basis for comparing different DLs by redefining all DLs within a prospect DFL.

In the following subsection we show how F-logic can account for some typical terminological operators, and observe that once these operators are axiomatized (alternatively implemented), a DL's description is already an F-logic's object specification. Note that using F-logic as a general framework for DLs is markedly different from embedding DLs in First Order Logic (FOL), since no DL construct has a direct meaning in FOL. In F-logic we deliberately decide which constructs are directly embedded in the semantics, and which are specified via axiomatization.

4.2 Terminological Operators in F-logic

We consider several most standard terminological operators (whose set-theoretic meaning was given in 2.1). Each operator is first described in terms of F-logic's semantics, and then axiomatized. We note that all operators can be built directly into the semantics, if desirable, as they have direct meaning in F-logic's ontology. Variables are capitalized, and unless otherwise specified, are assumed universally quantified.

[**and**] The object constructor **and** denotes the **glb** operator on $(U, \preceq_U)$. It can be axiomatized as follows:

$$X \ :: \ \mathbf{and}(\ C_1,\ \ldots,\ C_n\ ) \quad \equiv \quad \bigwedge_{i=1}^{n} (\ X \ :: \ C_i\ ) \tag{6}$$

Note that **and** is an object constructor, and that it can have changing arity[5], while $\bigwedge$ is the regular conjunction connective. Alternative axiomatization using rules:

$$X \ :: \ \mathbf{and}(\ C_1,\ \ldots,\ C_n\ ) \quad \longleftarrow \quad \bigwedge_{i=1}^{n} (\ X \ :: \ C_i\ ) \tag{7}$$

$$\mathbf{and}(\ C_1,\ \ldots,\ C_n\ ) \ :: \ C_i \qquad 1 \leq i \leq n \tag{8}$$

[5]The changing arity is supported by the Hilog enhancement to F-logic [10].

The *extension* of **and**($C_1, \ldots, C_n$), i.e., the objects related to it via $\in$, can be characterized as the intersection of the extensions of the C_i-s, as follows:

(9) $$X \ : \ \mathbf{and}(\ C_1, \ \ldots, \ C_n\ ) \quad \equiv \quad \bigwedge_{i=1}^{n} (\ X \ : \ C_i\ )$$

[**exists**] The concept forming operator **exists**, selects all objects on which a (set-valued) role is defined. That is, the built-in meaning of **exists** in a DL semantic structure (D, Υ) is:

$$\Upsilon(\ \mathbf{exists}(\ r\ )\ ) \quad = \quad \{\ d \in D / \Upsilon(r)(d) \neq \emptyset\ \}$$

F-logic enables fine distinctions in possible meanings of **exists**: It can select objects on which a role is *defined*; alternatively, it can select objects on which a role is *applicable*. Axiomatization for the *defined* meaning is:

(10) $$C \ :: \ \mathbf{exists}(\ R\ ) \quad \equiv \quad C[R \bullet\!\!\rightarrow\!\!\!\rightarrow \{\}]$$

or, using rules only:

(11) $$C \ :: \ \mathbf{exists}(\ R\ ) \quad \longleftarrow \quad C[R \bullet\!\!\rightarrow\!\!\!\rightarrow \{\}]$$

(12) $$\mathbf{exists}(\ R\ )[R \bullet\!\!\rightarrow\!\!\!\rightarrow \{\}]$$

Note that this axiomatization of **exists**(R) does not explicitly require the existence of a value object for R, since R maybe defined but its value maybe the empty set. A definition that exactly accounts for the conventional meaning of **exists** should require the existence of a value object for R, as follows:

(13) $$C \ :: \ \mathbf{exists}(\ R\ ) \quad \equiv \quad \exists D, C[R \bullet\!\!\rightarrow\!\!\!\rightarrow \{D\}]$$

We prefer the former definition, as in 10, since the later is the definition of **at-least**(1, R) (see 21). Axiomatization for the *applicable* meaning is:

(14) $$C \ :: \ \mathbf{exists}(\ R\ ) \quad \equiv \quad C[R \Rightarrow\!\!\!\!\Rightarrow ()]$$

or, using rules only:

(15) $$C \ :: \ \mathbf{exists}(\ R\ ) \quad \longleftarrow \quad C[R \Rightarrow\!\!\!\!\Rightarrow ()]$$

(16) $$\mathbf{exists}(\ R\ )[R \Rightarrow\!\!\!\!\Rightarrow ()]$$

[**all**] The **all** operator selects all objects at which the values of a (set-valued) role are restricted by a given class. In F-logic's terms, **all** selects all objects that satisfy a typing restriction:

(17) $$C_1 \ :: \ \mathbf{all}(\ R, \ C\ ) \quad \equiv \quad C_1[R \Rightarrow\!\!\!\!\Rightarrow (C)]$$

Note that this definition of **all** is finer than the set-theoretic meaning, since in the latter, $\mathbf{d} \in \mathbf{all}(R, C)^{\mathbf{I}}$ holds if $\mathrm{R}(\mathbf{d}\ ) = \emptyset$. For example, if *table*, *child*, and *doctor*, get their intended meaning in a terminology T, then *table* $\sqsubseteq$ **all**(*child, doctor*) holds in T. The F-logic definition, on the contrary, requires that the role be applicable to every subclass of **all**(*R, C*), and restricts its values to be members of C. Hence, since it is unreasonable to have the typing $table[child \Rightarrow\!\!> (doctor)]$, $\neg(table :: \mathbf{all}(child, doctor)\ )$ holds in every model of 17.

Axiomatization using rules alone:

(18) $$C_1\ ::\ \mathbf{all}(\ R,\ C\ )\quad \longleftarrow \quad C_1[R \Rightarrow\!\!> (C)]$$

(19) $$\mathbf{all}(\ R,\ C\ )[R \Rightarrow\!\!> (C)]$$

Equivalence with 17 results from the mandatory inheritance of types in F-logic.

The **all** operator can be axiomatized without signature F-molecules, as follows:

(20) $$C_1\ ::\ \mathbf{all}(\ R,\ C\ )\ \equiv\ \forall C_2,\ (\ C_1[R\bullet\!\!\rightarrow\!\!> \{C_2\}]\ \longrightarrow\ C_2 :: C\ )$$

Subsumption relationships between terminological terms, that result from the fixed meaning of the operators, and not from a temporary population of a knowledge base, extend to the F-logic formulation. For example, in every DL

$$\mathbf{at\text{-}least}(1, r) \doteq \mathbf{exists}(r)$$

$$\mathbf{at\text{-}least}(2, r) \sqsubseteq \mathbf{exists}(r)$$

hold, for every role term r, and every terminology. In F-logic, **at-least**(*n,r*) is a cardinality restriction on the values of the role r, i.e.,

(21) $$C_1\ ::\ \mathbf{at\text{-}least}(\ 1,\ R\ )\quad \equiv \quad \exists C_2,\ C_1[R\bullet\!\!\rightarrow\!\!> \{C_2\}]$$

(22) $$C_1\ ::\ \mathbf{at\text{-}least}(\ 2,\ R\ )\quad \equiv \quad \exists C_2,\ C_3,\ \ C_2 \neq C_3\ \wedge\ C_1[R\bullet\!\!\rightarrow\!\!> \{C_2, C_3\}]$$

Hence, assuming the *defined* meaning for **exists** (or alternatively, the *applicable* meaning and well typing),

(23) $$\mathbf{at\text{-}least}(\ 1,\ R\ )\ ::\ \mathbf{exists}(R)$$

(24) $$\mathbf{at\text{-}least}(\ 2,\ R\ )\ ::\ \mathbf{exists}(R)$$

hold in every model of 10, 21, 22. Of course, would the meaning of these operators be built into the semantics of a DFL, formulae 23, 24 would have been tautologies of that DFL.

Assuming the *applicable* meaning of **exists**:

$$\mathbf{all}(\ R,\ C\ )\ \ ::\ \ \mathbf{exists}(R)$$

holds in every model of 14, 17, for every R and C. Also, since inheritance of signatures is mandatory, we have:

$$\text{if}\ \ C_1\ ::\ C_2\ \ \text{then}\ \ \mathbf{all}(\ R,\ C_1\ )\ \ ::\ \ \mathbf{all}(R,\ C_2)$$

holds in every model of 17, for every R and $C_1,\ C_2$. More about subsumption in DFL, in Section 6.1.

Inheritance of poperties in DLs, that result from the fixed meaning of the operators, also extend to the F-logic formulation. For example, in DLs, if $o\ :\ \mathbf{at\text{-}least}(1,\ r)$ holds in a semantic structure $(D,\ \Upsilon)$, then there exists an element d in D such that $(\Upsilon(o),\ d) \in \Upsilon(r)$. That is, individuals inherit their properties from concepts they belong to. An analogous situation in the F-logic translation arises when $o\ :\ \mathbf{at\text{-}least}(1,\ r)$ holds in a semantic structure $\mathbf{I}$, that satisfies axiom 21. By 21, $(\ \exists C,\ \mathbf{at\text{-}least}(1,\ r)[r\bullet\!\!\rightarrow\!\!\!\!\rightarrow \{C\}]\ )$ holds in $\mathbf{I}$, and by the semantics of non-monotonic inheritance suggested in [20], also $(\ \exists C,\ o[r \rightarrow\!\!\!\!\rightarrow \{C\}]\ )$ holds in $\mathbf{I}$. Moreover, for all c such that $c\ ::\ \mathbf{at\text{-}least}(1,\ r)$ holds in $\mathbf{I}$, $\exists C,\ c[r\bullet\!\!\rightarrow\!\!\!\!\rightarrow \{C\}]$ also holds.

Role forming operators like **and-role**, **inverse**, **identity**, and **compose** can also be bult-in as object constructors. For example:

[**and-role**] In the method-oriented approach of F-logic, $\mathbf{and}(\ R_1, \ldots, R_n\ )$ is a method defined/applicable only where all R_i-s are, and it gets only values common to all R_i-s. Its axiomatization, using the *defined* view:

(25) $$C_1[\mathbf{and}(\ R_1, \ldots, R_n\ ) \rightarrow\!\!\!\!\rightarrow \{C_2\}] \quad \equiv \quad \bigwedge_{i=1}^{n}\ (\ C_1[R_i \rightarrow\!\!\!\!\rightarrow \{C_2\}]\ )$$

(26) $$C_1[\mathbf{and}(\ R_1, \ldots, R_n\ )\bullet\!\!\rightarrow\!\!\!\!\rightarrow \{C_2\}] \quad \equiv \quad \bigwedge_{i=1}^{n}\ (\ C_1[R_i\bullet\!\!\rightarrow\!\!\!\!\rightarrow \{C_2\}]\ )$$

Note that the object constructor **and** can have changing arity, and is used, both as a concept constructor (6, 7, 8) and as a role constructor (25, 26). This is in line with F-logic's view of roles (methods) as objects.

If methods' equality and implication are built into a DFL, then methods' implication forms a partial ordering, and the language includes some "methods-analogue" for "::". In that case, the operator **and**, when used as a role constructor, denotes the **glb** operation on that ordering, and the axiomatization is similar to 7, 8, with the "::"operator replaced by a methods' ordering operator.

[**inverse**]

(27) $$C_1[\mathbf{inverse}(\ R\ ) \rightarrow\!\!\!\!\rightarrow \{C_2\}] \quad \equiv \quad C_2[R \rightarrow\!\!\!\!\rightarrow \{C_1\}]$$

(28) $$C_1[\mathbf{inverse}(\ R\ )\bullet\!\!\rightarrow\!\!\!\rightarrow \{C_2\}] \quad \equiv \quad C_2[R\bullet\!\!\rightarrow\!\!\!\rightarrow \{C_1\}]$$

Having embedded concept and role forming operators in F-logic, a role/concept description is written as in any DL. For example, the CLASSIC definition for a student that drives only sports cars of Italian makers, drives at least one and at most two such cars, can be introduced in the following terminology:

$driver \doteq \mathbf{exists}(thingDriven)$

$showOffPerson \doteq \mathbf{and}(\ person,$

$\mathbf{all}(\ thingDriven, \mathbf{and}(\ sportsCar, \mathbf{all}(\ maker, italian\ )))\ )$

$1\text{-}2CarOwner \doteq \mathbf{and}(\ \mathbf{at\text{-}least}(1, thingDriven), \mathbf{at\text{-}most}(2, thingDriven)\ )$

$showOffStudent \doteq \mathbf{and}(\ student, showOffPerson, 1\text{-}2CarOwner\ )$

Assume that the hierarchy component includes:

$sportsCar :: car$

$car :: vehicle$

and the typing component includes:

$X[thingDriven \Rightarrow\!\!\!> \{vehicle\}]$

Then, under the *applicable* meaning of **exists**, the following subsumption relationships (captured by the "::" operator) hold in this terminology:

$showOffPerson :: driver$

$1\text{-}2CarOwner :: driver$

$showOffStudent :: showOffPerson$

$showOffPerson :: person$

4.3 Replacing the Standard Set-theoretic Semantics by the OO Semantics of F-logic

In this section we show that when the set-theoretic semantics of DLs is replaced by the OO semantics of F-logic, nothing is lost, i.e., logical implication is preserved. We do it by considering the terminological operators one by one. First, we define DLs as (notational variants of) sublanguages of F-logic, and show that if no concept/role forming operators are allowed into a DL, then the F-logic view preserves logical implication (including subsumption). Then we characterize a

correspondence relationship between F-logic theories and DLs. We demonstrate that the axiomatizations provided in the last section for some operators indeed correspond to DLs with these operators.

For simplicity we ignore the type component of F-logic. This is achieved by assuming that the following formulae hold in all semantic structures that we consider:

$$\forall_{ind} X,\ X\ :\ top$$

$$\forall R,\ top[R \Rightarrow\!\!\!> top]$$

This assumption guarantees that all methods' applications that arise in the F-logic variant of a DL, are well typed.

As an example for a typical DL we adopt the DL presented in [45], where we let the set of concept/role forming operators vary. The syntax and semantics of this DL were already presented in 2.1. We recall that a typical DL with operators set P is denoted L_P.

The F-logic language that corresponds to the DL L_P is denoted L_P^{FL}. This is a sorted F-logic language, with sorts for primitive concepts, concept names, concepts, primitive roles, role names, roles, and individuals. The sort of *top* and *bottom* is Primitive concept. The set of id symbols includes all concept/role/object symbols of L_P, appropriately sorted. The set of object constructors is P, also appropriately sorted. Variables are sorted as well. The formulae of L_P^{FL} are notational variants of L_P's formulae (denoted by an FL superscript). The syntactical translation is needed since methods' equality is not built into the semantics of F-logic, and membership in a role is replaced by an assertion about a method's value. Once these differences are taken care of, L_P is already a subset of F-logic. The formulae of L_P^{FL}, and their correspondence to formulae of L_P, are defined in the table below.

$\gamma \in L_P$	$\gamma^{FL} \in L_P^{FL}$
$c_n \doteq c$	$c_n \doteq c$
$c_1 \Rightarrow c_2$	$c_1 :: c_2$
$o\ :\ c$	$o\ :\ c$[6]
$r_n \doteq r$	$\forall_{ind} X, Y, (X[r_n \twoheadrightarrow \{Y\}] \equiv X[r \twoheadrightarrow \{Y\}])$
$r_1 \Rightarrow r_2$	$\forall_{ind} X, Y, (\ X[r_1 \twoheadrightarrow \{Y\}] \ \longrightarrow\ X[r_2 \twoheadrightarrow \{Y\}])$
$(o_1, o_2) : r$	$o_1[r \twoheadrightarrow \{o_2\}]$

The subscribed quantifier "$\forall_{ind}$" quantifies over the sort of individuals.

We start by considering the languages $L_\emptyset$, i.e., a DL with no concept/role forming operators, and $L_\emptyset^{FL}$.

[6] A simpler, altough less intuitive, translation is obtained if an L_P formula $o : c$ is replaced by the L_P^{FL} formula $o :: c$.

Definition 4.1 *Let*

$$T_0 = \{ \quad (\forall X, (X :: top)), (\forall X, (bottom :: X)),$$
$$(\forall X, Y, (X : Y \longrightarrow (sort(X) = individual \wedge sort(Y) = concept))),$$
$$(\forall_{ind} X, \forall_{concept} Y, (\neg(X :: Y))) \quad \},$$

and let I be an $L_\emptyset$ interpretation, and J be an $L_\emptyset^{FL}$ interpretation such that $J \models T_0$. We say that I and J correspond *if for every $L_\emptyset$'s formula γ: $I \models \gamma \quad iff \quad J \models \gamma^{FL}$.*

If I and J are corresponding interpretations we denote J as I^{FL}, or denote I as J^{DL}. Since interpretations correspondence is a symmetric relation, I can be denoted $(\mathrm{I}^{FL})^{DL}$.

Lemma 4.2 *For every I, an $L_\emptyset$ interpretation, there exists a corresponding I^{FL}, which is an $L_\emptyset^{FL}$ interpretation, and vice versa.*

Proof: This result is somewhat unexpected since the semantical assumptions in DLs are stronger than in F-logic. The result relies on the limited assertional power of DLs. It is obtained by proving an auxiliary lemma that provides sufficient conditions for interpretations correspondence, and showing that interpretations that satisfy these conditions do exist. The auxiliary lemma and the proof appear in the Appendix. □

Theorem 4.3 *$L_\emptyset$ and $L_\emptyset^{FL}$ preserve logical implication. That is, for every set of formulae Γ, and a formula γ in $L_\emptyset$:* $\Gamma \models \gamma \quad iff \quad T_0, \Gamma^{FL} \models \gamma^{FL}$.

Proof: $\Leftarrow$

Assume $\mathrm{I} \models \Gamma$. Then by 4.2, $\mathrm{I}^{FL} \models \Gamma^{FL}, T_0,$ which implies $\mathrm{I}^{FL} \models \gamma^{FL}$, which implies $\mathrm{I} \models \gamma$ by 4.2, since I also corresponds to I^{FL}.

The other direction is proved similarly. □

Definition 4.4 *A theory T of L_P^{FL}* corresponds *to L_P if for every set of formulae Γ, and a formula γ in L_P:* $\Gamma \models \gamma \quad iff$ $T, \Gamma^{FL} \models \gamma^{FL}$.

Corollary 4.5 *T_0 corresponds to $L_\emptyset$.*

Corollary 4.6 *Let c_1, c_2 be concept terms of L_P, and T be an L_P^{FL} theory that corresponds to L_P. Then, $c_1 \sqsubseteq c_2$ holds in a terminology Γ iff $T, \Gamma^{FL} \models c_1^{FL} :: c_2^{FL}$.*

Theorem 4.7 *Let* $P = \{$**and, all, at-least1, and-role** $\}$, *and*

$$
\begin{aligned}
T_P = T0 \cup \{ \quad & C_1 :: C_2 \quad \equiv \quad (\forall O,\ O : C_1 \longrightarrow O : C_2) \\
& C \ :: \ \mathbf{and}(\ C_1,\ \ldots,\ C_n\ ) \quad \equiv \quad \textstyle\bigwedge_{i=1}^{n}(\ C \ :: \ C_i\ ) \\
& O \ : \ \mathbf{and}(\ C_1,\ \ldots,\ C_n\ ) \quad \equiv \quad \textstyle\bigwedge_{i=1}^{n}(\ O \ : \ C_i\ ) \\
& O_1 \ : \ \mathbf{all}(\ R,\ C\ ) \quad \equiv \quad \forall O_2,\ (\ O_1[R \twoheadrightarrow \{O_2\}] \ \longrightarrow \ O_2 : C\ ) \\
& O_1 \ : \ \mathbf{at\text{-}least1}(\ R\ ) \quad \equiv \quad \exists O_2,\ O_1[R \twoheadrightarrow \{O_2\}] \\
& O_1[\mathbf{and}(\ R_1, \ldots, R_n\ ) \twoheadrightarrow \{O_2\}] \quad \equiv \quad \textstyle\bigwedge_{i=1}^{n}\ (\ O_1[R_i \twoheadrightarrow \{O_2\}]\ )
\end{aligned}
$$

where, unless otherwise specified, all variables (start with upper case letter) are universally quantified. Then, T_P *corresponds to* L_P.

Proof: The proof is similar to the proof of Theorem 4.3. It is based on the following Lemma, which is analogous to Lemma 4.2:

Lemma 4.8 *For every* I, *an LP interpretation, there exists a* $J(= I^{FL})$, *an* L_P^{FL} *interpretation, such that for every* L_P*'s formula* γ,

$$I \models \gamma \quad iff \quad J \models \gamma^{FL}, T_P.$$

The proof of this Lemma is based on an auxiliary Lemma, analogous to Lemma 8.1, which is proved by induction on the number of operators in terms, starting with $P_1 = \{$**and**$\}$, and increasing it in three steps with the other operators (i.e., $P_4 = P$). The proof is tedious since for each P_i it has to be repeated. □

5 F-logic as a Vehicle for Extending the Expressivity of Description Languages

In this section we investigate how and when the object-oriented, intensional, higher ordered nature of F-logic can be used to extend the expressive power of standard description languages, and to provide a common ground for the integration of description languages with general knowledge representation systems. In particular, we show how an F-logic based description language can account for desired features of DLs, that are problematic in the standard account of DLs.

5.1 High Order Roles and Operator Forming Operators

High order roles hold between concepts. In the set-theoretic semantics it means binary relations on the set of subsets of the domain. Standard DLs do not allow high order roles. Yet, the need for such roles is frequently pointed out. For example, [51] suggests high order roles, parameterized by regular roles. Two such roles are **AE**(R), and **SR**(R) (AE is an abbreviation for All-Exists, and SR is an abbreviation for Subject-Restriction), that denote binary relations between concepts. They can be used to capture the following relationships:

Every person lives in some place:	(person, place) $\in$ **AE**(lives)
If X lives in an apartment then X is a person:	(person, apartment) $\in$ **SR**(lives)

Note that **AE** and **SR** act like role forming operators, for high order roles.

In F-logic, high order roles form no extension or exception to the rest of the language, and require no special treatment. **AE**(R) and **SR**(R) can be defined as non-inheritable roles, as follows:[7]

(29) $$C_1[\mathbf{AE}(R) \twoheadrightarrow \{C_2\}] \quad \equiv \quad \forall\, X : C_1,\ \exists\, Y : C_1,\ X[R \twoheadrightarrow \{Y\}]$$

(30) $$C_1[\mathbf{SR}(R) \twoheadrightarrow \{C_2\}] \quad \equiv \quad \forall\, X,\ (\ if\ X[R \twoheadrightarrow \{Y\}] \ \wedge\ Y : C_2\ \ then\ X\ :\ C_1\ )$$

The above examples can be asserted as follows:

$$person[\ \mathbf{AE}(\ live\ ) \twoheadrightarrow \{place\}\ ]$$

$$person[\ \mathbf{SR}(\ live\ ) \twoheadrightarrow \{apartment\}\ ]$$

The roles discussed in the subsection about collective entities, below, are also high order.

Operator forming operators is another trend in parameterization of constructs by other constructs. For example, Woods suggests to capture concepts such as "A student who takes every Math course" by associating the role *take-course* with a tag **MA**, such that

$$[student]\ /\ (\ \mathbf{MA}[take\text{-}course] : [math\text{-}course]\ )$$

denotes the intended subconcept of *student*. Essentially, this amounts to a concept forming operator **MA**(*take-course*), such that **MA**(*take-course*)(*student*,*math-course*) has the set-theoretic meaning:

$$\{\ \mathbf{d}\ /\ \mathbf{d} \in \Upsilon(student), \Upsilon(math\text{-}course) \subseteq \Upsilon(take\text{-}course)(\mathbf{d})\ \}$$

[7] It makes no sense to build operators like **AE** and **SR** into the semantics, as there are many other similar operators that have the same status.

MA itself is an *operator forming operator*, and **MA**(R) is a family of concept forming operators, each parameterized by a role.

In the standard set-theoretic approach to DLs, where the set of operators is small, fixed, and each operator's meaning is built into the semantics, a parameterized family of operators is not allowed. In F-logic, with the high-logization provided by the HiLog enhancement ([10]), **MA**(R) can be captured as a family of object constructors, parameterized by R. **MA**(R)(C_1, C_2) selects the greatest subconcept of C_1, that is inheritably related to every member of C_2, via R. **MA** can be defined as follows:

(31) $$X :: \mathbf{MA}(R)(C_1, C_2) \quad \equiv \quad (X :: C_1 \wedge \ \forall\, Y, (\ if\ Y : C_2\ then\ X[R\bullet\!\!\rightarrow\!\!\!\rightarrow \{Y\}]))$$

With this definition, the object denoted by the id-term **MA**(*take-course*)(*student*,*math-course*) stands for the intended concept.

Similarly, cardinality operators like **at-least**(n, R), can be "upgraded", or crystallized, by using **at-least**(n) as a family of concept forming constructors, with the definition:

$$\mathbf{at\text{-}least}(n)(R) \doteq \mathbf{at\text{-}least}(n, R).$$

5.2 Collective Entities

Collective entities are sets of other entities. The need for such entities arise when sets have properties, usually based on properties of their members ([12]). Properties such as $average(salary)$ or $min(bookPrice)$ are typical, certainly non-inheritable, properties of collective entities. The semantical approach of F-logic bears several desirable characteristics for handling collective entities, as listed below:

- Individuals can be identified with their singletons[8].
- Collective entities are identified neither with their members, nor with their sub-concepts. This property is captured by the following "proper subsets" relations, denoted $\subset_\prec$ and $\subset_\in$:

  [$\subset_\prec$] Let $\prec_U$ be the quasi ordering derived from $\preceq_U$. Define:

  $$a \subset_\prec b \ \ iff \ \ \{\mathbf{d} \in U \ / \ \mathbf{d} \prec_U a\} \ \subseteq \ \{\mathbf{d} \in U \ / \ \mathbf{d} \prec_U b\}$$

  [$\subset_\in$] Define:

  $$a \subset_\in b \ \ iff \ \ \{\mathbf{d} \in U \ / \ \mathbf{d} \in_U a\} \ \subseteq \ \{\mathbf{d} \in U \ / \ \mathbf{d} \in_U b\}$$

[8]The superfluous status of this distinction for knowledge representation is noted in [50].

Then we can have $a \subset_{\prec} b$ and $b \subset_{\prec} a$, but not $a = b$, and similarly for $\subset_{\in}$. (Note that $\prec_U$ implies $\subset_{\prec}$, but not vice versa.) That is, F-logic can account for different objects with the same members, or with the same subconcepts. For example, we can account for the two different objects appleCharterMembers and beatles, that stand in $\subset_{\prec}$ relation with each other (an example taken from [12]).

- Relationships that involve collective entities are easy to express. Examples of such relationships (following [12]), and their expression in F-logic follows:

  1. "John leads the Beatles": $John[lead \twoheadrightarrow \{beatles\}]$.

  2. "Every member of the Beatles sang yellowSubmarine":

     That is, the collective entity *beatles*, collectively sang *yellowSubmarine*, which implies that every member of the beatles sang *yellowSubmarine*. This relationship can be captured by a role forming operator **distributive**, parameterized by *sing*:

     $$beatles[\mathbf{distributive}(sing) \twoheadrightarrow \{yellowSubmarine\}],$$

     with axiomatization:

     $$C_1[\mathbf{distributive}(R) \twoheadrightarrow \{C_2\}] \quad \equiv \quad \forall X, (\ if\ X : C_1\ then\ X[R \twoheadrightarrow \{C_2\}]\ )$$

  3. "Every member of the Beatles has met Brian within some sub-collection of the Beatles":

     That is, for every member of the Beatles, there is a sub-colloection of the Beatles within which that member has met Brian. This relationship can be expressed with a role forming operator **cumulative**, parameterized by *meet*:

     $$beatles[\mathbf{cumulative}(meet) \twoheadrightarrow \{brian\}],$$

     with axiomnatization:

     $$C_1[\mathbf{cumulative}(R) \twoheadrightarrow \{C_2\}] \quad \equiv \quad \forall X, (\ X : C_1 \ \longrightarrow\ \exists S, X : S :: C_1 \wedge S[R\bullet\!\!\twoheadrightarrow \{C_2\}]\ )$$

  4. "Strikers throw tomatoes":

     That is, every striker throws tomatoes as part of some group of strikers:

     $$strikers[\mathbf{cumulative}(throw) \twoheadrightarrow \{tomatoes\}].$$

Note that all relationships are expressed with non-inheritable roles. Consequently, they can not be inherited by sub-concepts, or members of the collective entities.

5.3 N-ary Relationships

In description languages, where concepts are defined via the properties of their binary roles, n-ary relations, usually, have a secondary status ([49, 3]). The conventional handling is to turn (reify) an n-ary relationship into an object with n roles, relating it to its arguments. In F-logic, n-ary relationships can be captured in three ways: As n-ary methods, as entities related via n roles to their arguments, and as plain relations. The three ways highlight different aspects of relationships. As an n-ary method, the n-ary relationship describes the relationship between two objects, with dependency on other objects, as in: $patient[treater@place, time \Rightarrow doctor]$ (The treater of a patient at a certain place and time is a doctor.). In this formulation, n-ary relationships are not reified into entities, and are treated as n-ary roles. As entities, n-ary relations are captured as in: $medical\text{-}treatment[patient \Rightarrow person, treater \Rightarrow doctor, location \Rightarrow hospital, time \Rightarrow time]$. As plain relations, we have for example, $medical\text{-}treatment(john, Dr.Fu, hospital1, 9.9.92)$.

5.4 Roles as First Class Objects

The importance of roles as a special kind of objects is emphasized in [25, 27, 51]. The idea is that roles can be described, structured, classified, and in particular, distinguished from other relations. In F-logic, since roles are namable objects handling roles as objects is a natural feature.

- **Role typing**: $apple[color \Rightarrow\!\!\!> \{color\}]$
- **Cardinality restrictions**: The brute force way to impose cardinality restrictions on roles is to explicitly assert the existence of the required number of role values, as in:

$$\forall X,\ \exists Y, Z, (Y \neq Z \wedge X[R \twoheadrightarrow \{Y, Z\}]),$$

A more concise way:

$$\mathbf{exists}(R)\ ::\ \mathbf{at\text{-}least}(2)(R)$$

We can also change our view of **at-least**(n) from a family of operators into a class of roles, defined by:

$$R\ ::\ \mathbf{at\text{-}least}(n) \quad \equiv \quad \mathbf{exists}(R)\ ::\ \mathbf{at\text{-}least}(n)(R)$$

The cardinality of *parent* can be imposed as:

$parent$ :: **at-least**(2)

$parent$ :: **at-most**(2)

- **Excluded role fillers**:

$$X :: \mathbf{exc}(C)(R) \quad \equiv \quad (\ (\ if \ X[R \rightarrow Y] \ then \ \neg(Y :: C) \ ) \ \wedge \ (\ if \ X[R \twoheadrightarrow Y] \ then \ \neg(Y :: C) \ ) \ )$$
$$(\ (\ if \ X[R\bullet\!\!\rightarrow Y] \ then \ \neg(Y :: C) \ ) \ \wedge \ (\ if \ X[R\bullet\!\!\twoheadrightarrow Y] \ then \ \neg(Y :: C) \ ) \ )$$

  Turning the view of $\mathbf{exc}(C)$ from a family of operators into a class of roles, we can define:

$$R \ :: \ \mathbf{exc}(C) \quad \equiv \quad \mathbf{some}(R) \ :: \ \mathbf{exc}(C)(R)$$

- **Kinds of roles**: Roles can be classified by their nature, as in **intrinsic**(color, wine), **extrinsic**(price, wine), **part-of**(courses, meal), **essential**(color, white-wine) and **derived**(color, chardonay-wine) (examples are taken from [6]). The classification of roles by kinds can be particularly helpful in the knowledge engineering stage.

- **Justify the existence of a concept**: A role can be used to justify the existence of a concept (like the notion of *qua-concept* in [13]), as in:

  $X : father \quad \equiv \quad X[father \twoheadrightarrow \{\}]$,

  which allows into the $father$ concept only objects for which the $father$ role is defined.

5.5 Cycles and self-Reference

Terminological cycles, i.e., definitions of concepts that "use" the defined concept, either directly or not, pose a problem in DLs, since their semantics and computation are controversial. In particular, the standard normalize-compare algorithms for computing subsumption do not apply. Nebel, in[34], discusses several candidates for semantics of cycles, and suggests alternative approaches for computing subsumption. A DL based on F-logic can reason with cyclic terminological definitions, via its smooth integration with the rest of the F-logic knowledge base. For example, a definition like

human $\doteq$ **and**(mammal, **all**(parent, human), **exactly**(2)(parent)),

together with:

1. The hierarchy information:

   mammal :: animate

   teacher :: person

   person $\doteq$ *human*,

2. The contingent information:

   fred : mammal

$fred$: **all**$(parent, teacher)$

$fred[parent \twoheadrightarrow \{bob, mary\}]$,

3. Axiom 9 and appropriate axiomatization for members of **exactly**$(2)(R)$ objects,

can support positive answers to queries like:

? $human :: animate$	(Is *human* subsumed by *animate*?)
? $fred$: **all**$(parent, animate)$	(Is $fred$ a member of **all**$(parent, animate)$?)
? $fred$: $human.$	(Is $fred$ a member of *human*?)

The last query is considered "incorrect" by terminological reasoners, as it is true only under the closed world assumption.

This approach is in line with the direction of **several layered** systems, as described in [25], where not all concepts have the same status, and are not handled by a unique reasoning tool.

Self-reference is another problematic issue in DLs. It requires conjunction and composition of roles, which are a source for computational and descriptive problems. Consider Schmiedel's example from [48]:

> A person who has a son and a daughter who visit the same school, and where the daughter is older than the son.

The terminological representation of the first part is something like:

and(person, **some**(**rvm**(son.school, daughter.school))),

and combining the second qualification is hard, if not impossible. In F-logic, this concept can be defined using the logic mechanism in the standard way. A **description** for that concept requires a **special form** of object constructors that can impose conditions on a descriptive term, in the style of Woods' suggestion for a *General Structural Link* (**gsl**) of the form:

C [*roles-specifications-part; conditions-part*].

The required **gsl** constructor can have the syntax:

$$\textbf{gsl}(< role_1 > \twoheadrightarrow < var_1 >, \ldots, < role_n > \twoheadrightarrow < var_n >; < \textbf{conditions}(< var_1 >, \ldots, < var_n >) >)$$

The specification of Schmidel's example, using this operator:

childSameSchool $\doteq$ **and**(*person,*

$$\textbf{gsl}(\ son \twoheadrightarrow Son,\ daughter \twoheadrightarrow Dtr;$$

$$Son\ [\ school \rightarrow Schl,\ age \rightarrow AgeSon\ ],$$

$$Dtr\ [\ school \rightarrow Schl,\ age \rightarrow AgeDtr\ ],$$

$$AgeSon \leq AgeDtr\ )\ )$$

Using so called **dot expressions** ([18]) to specify role chains can greatly simplify the use of this constructor:

$childSameSchool \doteq \textbf{and}(\ person,$

$$\textbf{gsl}(\ son \twoheadrightarrow Son,\ daughter \twoheadrightarrow Dtr;$$

$$Son.school = Dtr.school,\ Son.age \leq Dtr.age\ )\ )$$

The definition of **gsl** in F-logic is:

$$X :: \textbf{gsl}(\ R_1 \twoheadrightarrow Y_1, \ldots,\ R_n \twoheadrightarrow Y_n;\ \textbf{conditions}(Y_1, \ldots,\ Y_n)\ ) \quad \equiv$$

$$(\ X[R_1 \twoheadrightarrow \{Y_1\}, \ldots, R_n \twoheadrightarrow \{Y_n\}] \wedge \textbf{conditions}(Y_1, \ldots, Y_n)\ )$$

This special form can be made part of F-logic, using the HiLog enhancement. Of course, more general forms are possible, such as a form that allows for arguments to roles, as in:

> A person who has, with the same spouse, a son and a daughter who visit the same school, and where the daughter is older than the son.

This example can be formulated as follows:

$childSameSchool \doteq \textbf{and}(\ man,$

$$\textbf{gsl}(\ son@Woman \twoheadrightarrow Son,\ daughter@Woman \twoheadrightarrow Dtr;$$

$$Son.school = Dtr.school,\ Son.age \leq Dtr.age\ )\ )$$

5.6 Intensions

Representing intensions in F-logic is straightforward, since its semantical objects do not stand just for themselves, as in classical logic, but have other objects associated with them. We have already dealt with intensional objects in subsection 5.2. The *member extension* of a semantical object is all objects $\in_U$ related to it. The *concept extension* of a semantical object is all objects $\preceq_U$ related to it. Intensional imaginary objects, such as "a round square "will have no objects in

their member extensions[9]. Intensional descriptions of the same object, such as "morning star" and "evening star" or "CFG language" and "NPDA language"[10], will have the same member extension. Views of individuals, such as "mary as a teacher" $\preceq_U$ "mary as a woman", are understood as further specifications.

6 Integrating Terminologies with F-logic Knowledge Bases

Most DL researchers believe that terminological reasoners should be used in complex applications. To this end there are two main approaches: Either to strengthen the terminological reasoner itself, or to integrate terminological reasoners within general KR systems. The LOOM approach adopts the first option, while the CLASSIC approach follows the second one. Yet, it seems that even augmented terminological reasoners will probably need to be combined with other forms of reasoning. So, although there is no agreement as to the desired expressivity of DLs, it seems that integration with other reasoners is unavoidable ([39, 41, 27, 48, 15, 51, 11, 29]).

A major issue in designing integration schemes is to avoid usual problems of **mismatch**. The F-logic approach seems promising as an underlying integration framework, for the following reasons:

- True terminological definitions are possible.
- Standard DL algorithms are correct in F-logic (see 6.1).
- F-logic can support all typical deductive and object-oriented databases reasoning.
- F-logic is very expressive but computable.
- A Description F-logic (DFL) reasoner can accommodate a separate terminological component, with independent processing methods[11]. Clarifying the semantics of a combined DL – F-logic system is a subject for further investigation (see 6.2).

[9] A *round square* can be captured by the concept term **and**(*round*, *square*), whose denotation has no members. Note that unless $round \doteq$ **not**(*square*) holds in a semantic structure, the denotation of **and**(*round*, *square*) need not, necessarily, be *bottom*.

[10] This example is due to Woods.

[11] Interestingly, this framework seems to be a dual to the CLASSIC viewpoint, in which everything beyond CLASSIC's capabilities becomes part of a black box test. In the approach outlined here, the powerful integrative scheme of the DFL is the dominant one, and within that scheme the limited terminological reasoner is incorporated, possibly as a black box.

6.1 DL subsumption algorithms as Correct F-logic's Inference Algorithms

We shortly consider two approaches for checking subsumption and coherence: The constraints based approach of [47], and the normalize-compare algorithms of CLASSIC ([44]) and of [32]. We show that these approaches can be simulated by similar F-logic inference algorithms, that apply to descriptions written in a DFL, and all properties of the algorithms are preserved. The purpose of this section is to show that, computationally, nothing is lost by considering a description language as a subset of F-logic, and standard computation methods apply without any meaningful changes. Moreover, the above methods constitute a natural part of F-logic, thereby saving the need to resort to a different, specialized formalism, developed particularly for solving subsumption - coherence problems in concept descriptions.

6.1.1 The Constraints-Based Approach of Schmidt-Schuaß and Smolka

The language considered in that work concentrates on concept descriptions. The alphabet includes symbols for concept (denoted A, B) and roles (denoted R), a special *top* concept symbol $\top$, and five concept forming operators: $\forall$, $\exists$, $\sqcap$, $\sqcup$, $\neg$. Concept terms (denoted C, D) have the syntax:

$$C \leftarrow A \mid \forall R : C \mid \exists R : C \mid C \sqcap D \mid C \sqcup D \mid \neg C.$$

The meaning of terms is defined in the standard way, using set-theoretic semantics, interpreting $\sqcap$ as intersection, $\sqcup$ as union, and $\neg$ as complement. This language, called ALb, is peculiar in including the three main boolean operators. The complement operator is hardly used (in an explicit way) in other DLs. The authors show that deciding coherence in a language that includes the three boolean operators is NP-hard; hence, deciding subsumption in that language is co-NP-hard. The main thrust of the paper is the linear time translation of concept descriptions into *constraints systems*, in a way that preserves coherence. That is, a concept description is coherent if and only if the constraints system it is translated into is satisfiable. The authors then characterize *complete constraints systems*, where satisfiability can be checked in linear time, and show that the constraints systems obtained from a concept description can be complemented. The rest of that work includes coherence checking algorithms based on various methods for complementing constraints systems, and investigation of the complexity of coherence for the above language, and some of its sublanguages.

Let ALb^{FL} be the F-logic language that corresponds to ALb. We define an F-logic theory T_{ALb}, and with respect to which we define *coherence* in ALb^{FL}, and show that concept descriptions in ALb^{FL} preserve their properties from ALb. We show that given a description in ALb^{FL}, we can mimic the transformations defined in the work of [47], so that we get similar algorithms. In particular, we get the following transformations:

1. A concept description c gets transformed to an ALb^{FL} formula c^{FL}, such that c is coherent if and only if c^{FL} is satisfiable (linear time).

2. The formula c^{FL} is *simplified* into a formula c_s^{FL}, with satisfiability being preserved (linear time).

3. The formula c_s^{FL} is *complemented*, into a formula c_{sc}^{FL}, with satisfiability being preserved. The satisfiability of c_{sc}^{FL} can be checked in linear time. This step is non-deterministic.

The rest of [47] algorithms can be simulated in the same way. Hence, the whole work described in [47] applies, as a natural part of F-logic, to the terminological component of a DFL knowledge base.

The language ALb^{FL} has the concept forming operators **all, exists, and, or, not**, whose meaning is defined by the theory T_{ALb}, as follows:

Definition 6.1

1)	$C_1 :: \mathbf{all}(R, C)$	$\equiv$	$(\forall C_2, C_1[R\bullet\!\!\rightarrow\!\!\!\rightarrow \{C_2\}] \longrightarrow C_2 :: C)$
2)	$C_1 :: \mathbf{exists}(R, C)$	$\equiv$	$(\exists C_2, C_1[R\bullet\!\!\rightarrow\!\!\!\rightarrow \{C_2\}] \wedge C_2 :: C) \wedge \neg(C_2 \doteq \mathbf{not}(top))$
3)	$C_1 :: \mathbf{and}(C, D)$	$\equiv$	$(C_1 :: C) \wedge (C_1 :: C)$
4)	$\mathbf{or}(C, D) :: C_1$	$\equiv$	$(C :: C_1) \wedge (D :: C_1)$
5)	$X : \mathbf{or}(C, D)$	$\equiv$	$(X : C) \vee (X : D)$
6)	$C :: top$	7)	$\mathbf{not}(top) :: C$
8)	$\mathbf{not}(\mathbf{not}(top)) \doteq top$	9)	$\mathbf{and}(C, \mathbf{not}(C)) \doteq \mathbf{not}(top)$
10)	$\mathbf{or}(C, \mathbf{not}(C)) \doteq (top)$		
11)	$X : C \longrightarrow \neg(X : \mathbf{not}(C))$	12)	$X : \mathbf{not}(C) \longrightarrow \neg(X : C)$

A ground id-term that consists of these operators is called a *concept description.*

Definition 6.2 *A concept description c is incoherent if $T_{ALb} \models c \doteq \mathbf{not}(top)$. A concept description c is coherent if it is not incoherent. A concept description c is subsumed by a concept description d, denoted $c \sqsubseteq d$, if $T_{ALb} \models c :: d$.*

Proposition 6.3 *A concept description c is coherent if and only if $\exists X, X : c$ is satisfiable in T_{ALb}.*

Let *simple-Alb*^{FL} be ALb^{FL} with the **not** operator being applied only to variables or to constants. Then we have:

Proposition 6.4 *Deciding coherence in simple-$ALb^F L$ is NP-hard.*

Proof: Reduction from the satisfiability problem of propositional formulas in CNF, following the lines of the proof in [47]. □

If the operators **and** and **or** are assumed to distribute over each other, then a model of T_{ALb} is a boolean lattice, and DeMorgan laws hold with respect to **not, and, or**. In that case, a concept description in ALb^{FL} can be simplified in linear time, and Proposition 6.4 holds for ALb^{FL} as well. For distributive **and/or** we also have:

Proposition 6.5 *Subsumption is equivalent to incoherence:*

$$c \sqsubseteq d \; if \; and \; only \; if \; \mathbf{and}(c, \mathbf{not}(d)) \; is \; incoherent.$$

Hence, deciding subsumption in ALb^{FL} with distributive **and/or** is co-NP-hard.

We now show the sequence of transformations. A conjunctive formula in ALb^{FL} is written as a set of its conjuncts.

1. A concept description c is transformed into a formula c^{FL} of the following form: $\{ \; a \; : \; X, \; X \; :: \; c \; \}$, where a does not occur in c.

   Proposition 6.6 *c is coherent if and only if c^{FL} is satisfiable.*

2. c^{FL} is transformed into c_s^{FL} by repeatedly applying the following transformations:

Replace $X \; :: \; \mathbf{all}(r, \; c)$	*by* $(\; \forall Z, \; X[r \bullet\!\!\rightarrow \{Z\}] \longrightarrow Z :: Y \;)$ *and* $Y :: c \; , \quad Y$ *is new in* c_s^{FL}.
Replace $X \; :: \; \mathbf{exists}(r, \; c)$	*by* $(\; X[r \bullet\!\!\rightarrow \{Y\}] \;), \; b : Y,$ *and* $Y :: c \; , \quad b, \; Y$ *are new in* c_s^{FL}.
Replace $X \; :: \; \mathbf{and}(c, \; d)$	*by* $X :: c$ *and* $X :: d$.
Replace $X \; :: \; \mathbf{or}(c, \; d)$	*by* $X :: \mathbf{or}(Y, \; Z) \; , \; Y :: c$ *and* $Z :: d \; , \quad Y, Z$ *are new in* c_s^{FL}.
Replace $X \; :: \; top$	*by* $True$

   Proposition 6.7 *c^{FL} is satisfiable if and only if c_s^{FL} is.*

3. c_s^{FL} is transformed into a *complemented* c_{sc}^{FL}, by repeatedly applying the following transformations:

   Initially, c_{sc}^{FL} is c_s^{FL}.

   - *If* $(\; \forall Z, \; X[r \bullet\!\!\rightarrow \{Z\}] \longrightarrow Z :: Y \;), \quad X[r \bullet\!\!\rightarrow \{W\}]$ and $b : W$ are in c_{sc}^{FL},

     add $b : Y$.

   - *If* $a : X$ and $X :: \mathbf{or}(Y, \; Z)$ are in c_{sc}^{FL}, and neither $a : Y$ nor $a : Z$ is in c_{sc}^{FL},

     add $a : W$, where W can be either Y or Z.

   Proposition 6.8 *c_s^{FL} is satisfiable if and only if c_{sc}^{FL} is.*

Definition 6.9 *A clash in* c_{sc}^{FL} *is a subset of formulas of the form:* $\{\ a : X,\ X :: c,\ a : Y,\ Y :: \mathbf{not}(c)\ \}$, *or* $\{\ a : X,\ X :: \mathbf{not}(top)\ \}$.

Proposition 6.10 c_{sc}^{FL} *is satisfiable if and only if it includes no clash.*

Theorem 6.11 *A concept description* c *in simple-*ALb^{FL} *is coherent if and only if* c_{sc}^{FL} *includes no clash.*

6.1.2 The Normalize – Compare Approach for Deciding Subsumption

The *normalize-compare* approach is the most popular/standard method for checking subsumption between concept/role descriptions. This is a two step method, where in the *normalize* step concept descriptions are transformed into some normal form, and in the *compare* step, subsumption is decided by comparing normalized descriptions. In a DFL, the *normalize* step can be viewed as a preprocessing step for optimizing or simplifying concept descriptions. The *compare* step is a specialized inference algorithm for checking the hierarchical relationship between normalized descriptions. To support this claim, we consider several example rules for normalization and comparison. The normalization rules are borrowed from [44], and the comparison rules are taken from [32].

Normalization rules:

- $c \longrightarrow \mathbf{and}(c)$.
- $\mathbf{and}(\mathbf{and}(c),\ d) \longrightarrow \mathbf{and}(c,\ d)$.
- $\mathbf{and}(\mathbf{all}(r, c),\ \mathbf{all}(r, d)\ ) \longrightarrow \mathbf{all}(r, \mathbf{and}(c, d)\ )$.
- $\mathbf{all}(r, top) \longrightarrow top$.
- $\mathbf{and}(\mathbf{at\text{-}least}(n)(r),\ \mathbf{at\text{-}least}(m)(r) \longrightarrow \mathbf{at\text{-}least}(n)(r), \qquad if\ n \geq m$.

Clearly, in a DFL with the meaning of the operators appropriately defined or built-in, these transformations replace concept/role descriptions by equal concept/role descriptions. Hence, normalization can be perceived as a correct optimization step.

Comparison rules:

The COMPARE algorithm of [32] takes two normalized concept/role descriptions as inputs, and COMPARE(c, d) is true only if $d \sqsubseteq c$, with respect to an empty terminology (the assumption is that all defined concepts/roles where already

replaced by their definitions, and terminological definitions do not contain cycles). Some comparison rules follow:

- If d is **not**(top) return true.
- If c is **and**$(e_1,\ \ldots,\ e_n)$, test COMPARE(e_i, d), for all $1 \leq i \leq n$.
- If c is **at-least**$(n)(r)$ and d is **at-least**$(m)(p)$, test COMPARE(r, p) and $n \leq m$.
- If c is **at-least**$(n)(r)$ and d is **and**$(d_1,\ \ldots,\ d_m)$, test COMPARE(c, d_i) for some i such that $1 \leq i \leq m$.

In a DFL with role comparison and implication built-in, all these rules are correct, in the sense that a positive answer indeed implies subsumption between the given descriptions.

6.2 Towards a Description F-Logic

An F-logic knowledge base consists of several components including a hierarchy component, an equalities component, a type declaration component, and a rules component. A DFL knowledge base should have an additional *terminological* component. The main issue is to coherently integrate the terminological component with the rest of the knowledge base. This is not straightforward since the standard approach to the semantics of knowledge bases in artificial intelligence and databases is justified by the closed world assumption, while description languages operate under the open world assumption. For example, if we consider the definition of the **all** operator:

$$C_1 \ :: \ \mathbf{all}(\ R,\ C\ ) \quad \equiv \quad (\ \forall C_2,\ C_1[R\bullet\!\!\rightarrow\!\!\!\rightarrow \{C_2\}] \longrightarrow C_2 :: C\ ),$$

then, under the closed world assumption, if all ground instances of the right hand side hold in a canonical model of a knowledge base, then also C_1 :: **all**$(\ R,\ C\ )$ hold. This conclusion is in contradiction to the DLs approach, since it derives from instantaneous assertions about objects, and not from their essential terminological properties.

It seems that the semantics of a DFL knowledge base cannot be just a special case of F-logic's one, but need to be defined in a way that will combine the two different assumptions. Tentatively, this can be done by associating, with each terminological operator a *terminological predicate*, which in turn, is associated with the operator's definition. For example, the definition of **and** can be:

$$X :: \mathbf{and}(C_1, C_2) \ \longleftarrow \ \mathbf{and\text{-}pred}(X, C_1, C_2)$$

$$\mathbf{and\text{-}pred}(\mathbf{and}(C_1, C_2), C_1, C_2)$$

$$\mathbf{and\text{-}pred}(X, C_1, C_2) \ \longleftarrow \ X :: C_1, X :: C_2$$

$$\mathbf{and\text{-}pred}(X, C_1, C_2) \ \longleftarrow \ X :: Y, \mathbf{and\text{-}pred}(Y, C_1, C_2)$$

In the definition of the canonical model of a DFL knowledge base, terminological predicates should enjoy a special treatment, based on their meanings. The **and-pred**, for example, does not need any special consideration, but the **all-pred** does need. A formal definition of the combined semantics is a subject for further research.

7 Conclusion

The main message of this paper is that description languages can form a natural subset of an F-logic language, which provides a common ground for the development of description languages, for the integration of description languages with other systems, and for extensions of description languages. The main distinction between the standard description languages school to F-logic involves the lack of built in terminological operators, the replacement of roles by methods , and the lack of built in methods' relationships.

We have shown that terminological operators have direct meaning in F-logic's ontology, and hence can be built into the semantics. Such an approach will be in harmony with the standard DL approach, where **all** descriptive operators are built into the semantics. Yet, in a logic, it seems unreasonable and undesirable to have a semantics that is sensitive to every little change in the language. Moreover, we have seen that the ability to define new descriptive operators in terms of existing ones is a powerful machinery, that provides more flexibility and durability. A reasonable way out of these two extremes seems to agree on a small stable set of constructors to be built into the semantics of a DFL. Another DL oriented necessity pointed in the paper, is to build roles/methods' relationships into the semantics of a DFL. This modification would turn DLs, even syntactically, into subsets of F-logic.

8 Appendix

Proof of 4.2 is easily obtained from the following auxiliary Lemma:

Lemma 8.1 *Let* $I = (D,\ \Upsilon_I)$ *be an* $L_\emptyset$ *interpretation, and* $J = (U,\ \preceq_U,\ \in_U,\ J_F,\ J_\rightarrow,\ J_{\rightarrow\rightarrow},\ J_{\bullet\rightarrow},\ J_{\bullet\rightarrow\rightarrow},\ J_\Rightarrow,\ J_{\Rightarrow\Rightarrow})$ *be an* $L^{FL}_\emptyset$ *interpretation. Then the following conditions guarantee correspondence relation between I and J:*

1. *For every object symbol o in* $L_\emptyset$, $\Upsilon_I(o) = J_F(o)$. *This condition implies:* $\{\ d\ /\ d \in U,\ d = J_F(o)$, *for an individual sort symbol o in* $L^{FL}_\emptyset\ \} \subseteq D$.

2. *For every object symbol o, and every concept symbol c in* $L_\emptyset$, $\Upsilon_I(o) \in \Upsilon_I(c)$ *iff* $J_F(o) \in_U J_F(c)$.

3. *For any two object symbols* $o_1,\ o_2$ *in* $L_\emptyset$, $J_F(o_1) \notin J_F(o_2)$;

   For any two concept symbols $c_1,\ c_2$ *in* $L_\emptyset$, $J_F(c_1) \notin J_F(c_2)$;

   For every object symbol o, *and every concept symbol* c *in* $L_\emptyset$, $J_F(c) \notin J_F(o)$;

   For every object symbol o, *and every concept symbol* c *in* $L_\emptyset$, $J_F(o) \not\preceq J_F(c)$. *This condition guarantees that* J *satisfies the third and forth axioms in* T_0 *(see Definition 4.1).*

4. *For every* c_1, c_2, *concept symbols in* $L_\emptyset$, $\Upsilon_I(c_1) \subseteq \Upsilon_I(c_2)$ *iff* $J_F(c_1) \preceq_U J_F(c_2)$. *That is, the elements of* $(U, \preceq_U)$ *that interpret symbols of sort concept behave like elements of the power set of* D. *In particular, this condition guarantees that* J *satisfies the first two axioms in* T_0 *(see Definition 4.1).*

5. *For every role symbol* r *in* $L_\emptyset$, *and elements* $d_1,\ d_2$ *from* D, $(\ d_1,\ d_2\ ) \in \Upsilon_I(r)$ *iff* $J^{(0)}_{\twoheadrightarrow}(J_F(r))(\ d_1\ )$ *is defined and includes* d_2.

Proof: We show that if I and J satisfy conditions (1) through (5) then for every γ, a formula of $L_\emptyset$, $I \models \gamma$ iff $J \models \gamma^{FL}$.

For γ being one of $c_n \doteq c$, $c_1 \Rightarrow c_2$, and $o : c$, the results for γ^{FL} follow immediately from conditions (2) and (4).

For γ being $r_n \doteq r$, γ^{FL} is

$$\forall_{ind} X, Y,\ (X[r_n \twoheadrightarrow \{Y\}] \quad \equiv \quad X[r \twoheadrightarrow \{Y\}]).$$

$I \models r_n \doteq r$ iff

$\Upsilon_I(r_n) = \Upsilon_I(r)$ iff for every $d_1,\ d_2 \in D$, $(d_1, d_2) \in \Upsilon_I(r_n)$ iff $(d_1, d_2) \in \Upsilon_I(r)$.

Assume now that $I \models r_n \doteq r$ holds.

$J \models \gamma^{FL}$ iff

For every variable assignment V to $L^{FL}_\emptyset$,

$J \models (X^V[r_n \twoheadrightarrow \{Y^V\}] \equiv X^V[r \twoheadrightarrow \{Y^V\}])$

Let V be an arbitrary variable assignment to $L^{FL}_\emptyset$. Extend $L_\emptyset$ with the individual variable symbols, taken as object symbols, and extend Υ to be defined on the new symbols, in a way that satisfies conditions (1), (2), and (3) (where V is substituted for J_F whenever the symbol is a variable). Then,

$J \models (X^V[r_n \twoheadrightarrow \{Y^V\}]$ iff

$J^{(0)}_{\twoheadrightarrow}(J_F(r_n))(\ X^V\ )$ is defined and includes Y^V iff

$(X^V,\ Y^V) \in \Upsilon_I(r_n)$, by conditions (1) and (5), iff

$(X^V,\ Y^V) \in \Upsilon_I(r)$ by the above assumption, iff

$J^{(0)}_{\twoheadrightarrow}(J_F(r))(\ X^V\ )$ is defined and includes Y^V, by condition (5), iff

$J \models (X^V[r \twoheadrightarrow \{Y^V\}]$.

The other direction is proved similarly.

For γ being $r_1 \Rightarrow r_2$ the proof is similar.

We now show for γ being $(o_1,\ o_2) : r$. In this case γ^{FL} is $o_1[r \twoheadrightarrow \{o_2\}]$.

$I \models (o_1,\ o_2) : r$ iff

$(\Upsilon_I(o_1),\ \Upsilon_I(o_2)) \in \Upsilon_I(r)$ iff

$(J_F(o_1),\ J_F(o_2)) \in \Upsilon_I(r)$, by condition (2), iff

$J^{(0)}_{\twoheadrightarrow}(J_F(r))(\ J_F(o_1)\ )$ is defined and includes $J_F(o_2)$, by condition (5), iff

$J \models o_1[r \twoheadrightarrow \{o_2\}]$. □

Lemma 4.2 *For every I, an $L_\emptyset$ interpretation, there exists a corresponding I^{FL}, which is an $L^{FL}_\emptyset$ interpretation, and vice versa.*

Proof: Given I, an $L_\emptyset$ interpretation with domain D, a corresponding I^{FL} interpretation for $L^{FL}_\emptyset$ is one that has a domain $(U, \preceq_U, \in_U)$, such that U includes D, and in addition has an element for each subset of D; $\preceq_U$ and $\in_U$ mimic the subset and membership relation over D, respectively; assignments to $L^{FL}_\emptyset$'s symbols follow conditions (1) through (5) in 8.1.

Let J be an $L^{FL}_\emptyset$ interpretation that satisfies T_0, and has domain $(U,\ \preceq_U,\ \in_U)$. The specification of I $(= J^{DL})$, a corresponding $L_\emptyset$'s interpretation, is somewhat more tricky since J's domain need not be a lattice at all, and need not be isomorphic to a power set structure. We suggest to take U for the domain of I, with the following assignments for $L_\emptyset$'s symbols:

For object symbols o, $\Upsilon_I(o) = J_F(o)$.

For concept symbols c, $\Upsilon_I(c) = \{d \mid d \preceq_U J_F(c)\} \cup \{d \mid d \in_U J_F(c)\}$

For role symbols r,

$$\Upsilon_I(r) = \{(d_1, d_2) \mid J^{(0)}_{\twoheadrightarrow}(J_F(r))(\ d_1\ )\ \textit{is defined and includes}\ d_2\}$$

It is not hard to see that I and J satisfy the five conditions in 8.1, and hence they are corresponding with respect to $L_\emptyset$. □

Acknowledgements

Special thanks to Michael Kifer, for encouragement, valuable discussions, and detailed comments on an earlier draft. Thanks also to Robert Mac Gregor, Peter Patel-Schneider and Joachim Quantz for their helpful comments.

References

[1] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *KR-92*, pages 306–317, 1992.

[2] F. Baader, B. Hollunder, B. Nebel, H.J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *KR-92*, pages 270–281, 1992.

[3] S. Bergamaschi, S. Lodi, and C. Sartori. Representational extensions of dls. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 11–13, 1992.

[4] A. Borgida. Towards the systematic development of description logic reasoners: Clasp reconstructed. In *KR-92*, pages 259–269, 1992.

[5] A. Borgida, R.J. Brachman, D.L. McGuinness, and L.A. Resnick. Classic: A structural data model for objects. In *ACM-SIGMOD-89*, Portland, OR, 1989.

[6] R.J. Brachman, , D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with classic: When and how to use a kl-one like language. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, 1991.

[7] R.J. Brachman and H.J. Levesque. Competence in knowledge representation. In *AAAI-82*, pages 189–192, Pittsburgh, PA, 1982.

[8] R.J. Brachman and H.J. Levesque. The tractability of subsumption in frame-based description languages. In *AAAI-84*, pages 34–37, Austin, Texas, 1984.

[9] R.J. Brachman and J.G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[10] W. Chen, M. Kifer, and D.S. Warren. Hilog: A foundation for higher-order logic programming. *J. of Logic Programming*, 15(3):187–230, February 1993.

[11] J. Doyle and R.S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *J. of Artificial Intelligence*, 48(3):261–297, 1991.

[12] E. Franconi. Collective entities and relations in concept languages. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 31–35, 1992.

[13] M.W. Freeman. The qua link. In J.G. Schmolze and R.J. Brachman, editors, *Proceedings of the 1981 KL-One Workshop*, pages 54–64, Jackson, NH, 1981. Bolt Beranek and Newman Inc.

[14] M. Gehrke. Particles of the part whole relation. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 36–38, 1992.

[15] P. Hanschke. How to benefit from terminological logics. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 45–48, 1992.

[16] P.J. Hayes. The logic of frames. In D. Metzing, editor, *Frame Conceptions and Text Understanding*, pages 46–61, Berlin, 1979. Walter de Gruyter and Co.

[17] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *ECAI-90*, pages 348–353, 1990.

[18] M. Kifer, W. Kim, and Y. Sagiv. Querying object oriented databases. In *ACM SIGMOD*, pages 393–402, 1992.

[19] M. Kifer and G. Lausen. F-logic: A higher-order language for reasoning about objects, inheritance, and scheme. In *SIGMOD-89*, 1989.

[20] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. Technical Report #93/06, Dept. of Computer Sciencee, SUNY at Stony Brook, April 1993. to appear in JACM.

[21] H.J. Levesque and R.J. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In R.J. Brachman and H.J. Levesque, editors, *Readings in Knowledge Representation*, pages 41–70, Calif., 1985. Morgan Kaufman Publishers Inc.

[22] H.J. Levesque and R.J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[23] K.V. Luck, B. Nebel, C. Peltason, and A. Schmiesel. The back system. Technical Report KIT Report 29, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1985.

[24] K.V. Luck, B. Nebel, C. Peltason, and A. Schmiesel. The anatomy of the back system. Technical Report KIT Report 41, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1987.

[25] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 385–400. Morgan Kaufmann, 1991.

[26] R. MacGregor. Inside the loom description classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

[27] R. MacGregor. What's needed to make a description logic a good kr citizen. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 53–55, 1992.

[28] R. MacGregor, D. McGuinness, E. Mays, and T. Russ, editors. *Working notes, AAAI Fall Symposium on Issues in Description Logics*. AAAI, 1992.

[29] B. Mark. 10 years don't mean nothing. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 59–60, 1992.

[30] A. Napoli. Subsumption and classification-based reasoning in object-based representations. In *ECAI-92S*, pages 425–429, 1992.

[31] B. Nebel. Computational complexity of terminological reasoning in back. *J. of Artificial Intelligence*, 34:371–383, 1988.

[32] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Dissertation, University of Saarlands, Saarbrücken, 1989.

[33] B. Nebel. Terminological reasoning is inherently intractable. *J. of Artificial Intelligence*, 43:235–249, 1990.

[34] B. Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 331–361. Morgan Kaufmann, 1991.

[35] L. Padgham and B. Nebel. Combining classification and nonmonotonic inheritance reasoning: A forst step. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 64–71, 1992.

[36] P.F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the 1983 KL-ONE Workshop*, Denver, Colorado, 1984.

[37] P.F. Patel-Schneider. Undecidability of subsumption in nikl. *J. of Artificial Intelligence*, 39:263–272, 1989.

[38] P.F. Patel-Schneider. Defaults and descriptions. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 74–75, 1992.

[39] P.F. Patel-Schneider. Partial reasoning in knowledge representation systems based on description logics. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 74–75, 1992.

[40] P.F. Patel-Schneider and B. Swartout. Description logic specification – from the krss effort. Technical report, AT&T Bell Labs, June 1993.

[41] A B. Pfahringer. The logical way to build a dl-bsed kr system. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 76–77, 1992.

[42] J. Quantz. A step towards second order. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 78–82, 1992.

[43] J. Quantz and V. Royer. A preference semanics for defaults in terminological logics. In *KR-92*, pages 294–305, 1992.

[44] L.A. Resnick, A. Borgida, R.J. Brachman, D.L. McGuinness, and P.F. Patel-Schneider. Classic description and reference manual for common lisp implementation. Technical Report Version 1.02, AT&T Bell Labs, 1990.

[45] V. Royer and J. Quantz. Deriving inference rules for terminological logics. In D. Pearce and G. Wagner, editors, *Logics in AI, JELIA'92*, pages 84–105, Berlin: Springer, LNAI 633, 1992.

[46] M. Schmidt-Schauß. Subsumption in kl-one is undecidable. In *Proceedings, Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431, Toronto, Ontario, Canada, 1989.

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

[48] A. Schmiedel. For a more expressive query language. In *Working notes, AAAI Fall Symposium on Issues in Description Logics*, pages 98–102, 1992.

[49] J.G. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *Conference on Principles of Knowledge Representation and Reasoning*, pages 432–443, Toronto, Ontario, Canada, 1989.

[50] J.F. Sowa. Towards the expressive power of natural language. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 157–190. Morgan Kaufmann, 1991.

[51] W.A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 45–94. Morgan Kaufmann, 1991.

[52] W.A. Woods and J.G. Schmolze. The kl-one family. *Computers and Mathematics with Applications*, Special Issue on Semantic Networks in Artificial Intelligence, 1992.