# DFL – A Hybrid Integration of Descriptions and Rules, using F-Logic as an Underlying Semantics

**Mira Balaban** and **Adi Eyal**
Ben-Gurion University of the Negev
ISRAEL

**Abstract.** In this paper we use F-Logic ([8]) as an underlying framework for a hybrid construction of descriptions and rules. The hybrid framework, termed $\mathcal{DFL}$, is *modular*, and enjoys a *compositional semantics*. In $\mathcal{DFL}$, the knowledge base manages a database of explicit descriptions, by consulting two separate reasoners: *DL* – The *Description Languages* reasoner, and *R* – The *Rules* reasoner. The reasoners can operate under different semantical policies. Four different compositional semantics possible for the hybrid $\mathcal{DFL}$ framework are discussed and compared.

## 1 INTRODUCTION

Descriptions and Rules are different, complementary, essential forms of knowledge. Descriptions are analytic and closed; rules are contingent and open. Historically, descriptions and rules were developed along separate lines, by different communities. The two forms can be integrated either by compiling one form within the other, or by constructing a hybrid framework. The first solution yields a coherent framework, but requires reconstruction of one approach within the other, and makes one approach sub-ordinate to the other. The hybrid solution keeps the modular independent status of each approach, but needs an underlying integration framework, in which a coherent compositional semantics can be defined.

We use F-Logic as an underlying framework for a hybrid construction of descriptions and rules since both are *natural subsets of F-logic*. The hybrid framework follows four desirable principles:

- Modularity of the source forms of knowledge.
- Compositional semantics, i.e., the semantics is composed from the separate semantics of the source forms.
- Query sensitivity, i.e., the semantics should reflect expected query-answering behavior.
- Preserve intrinsic properties of the source forms, e.g., openness for rules.

Rules are traditionally used in deductive database and in expert systems, as a means for expressing implicit knowledge, that extends the explicitly given facts. In DLs, although there is a growing agreement that rules are essential, there is no agreement on an integration framework, and the standard formal treatment is restricted to descriptions ([9; 7; 4; 5]). The different reasoning policies and semantical approaches of descriptions and rules pose a major obstacle. The conventional semantics of descriptions is set-theoretic, while the semantics of rules is model-theoretic; description systems usually reason under the so called *Open World Assumption (OWA)*, while rule based systems usually adopt the more conventional *Closed World Assumption (CWA)*.

In ([1]) we argue that F-Logic ([8]) can serve as a unifying formalism for current description languages. In particular it is shown that it provides a faithful account, in terms of direct syntax, semantics and inference algorithms, to current Dls. A DL knowledge base, under this account, reasons about a data base of descriptions, by consulting an "oracle" of axiomatizations of DL operators. The semantics of the knowledge base depends on the descriptions and the oracle. Inference can be accomplished using any specialized DL inference algorithm, that is faithful to the semantics. It is shown that proper axiomatizations of DL operators exist for meaningful DLs, and that standard inference algorithms are indeed applicable, under these conditions.

In ([2]), a hybrid model called $\mathcal{DFL}$, that integrates descriptions and rules, is introduced. In $\mathcal{DFL}$, the knowledge base manages a database of explicit descriptions in an arbitrary, but decidable description language. The database consists of the following descriptions (object, concept and role terms are denoted o, c, and r, respectively):

1. Extensional assertions of the form $o \in c$, and $(o_1, o_2) \in r$.
2. Intensional descriptions of the form $c_1 \leq c_2$, $r_1 \leq r_2$, $c_n \doteq c$, and $r_n \doteq r$, where $c_n$ and $r_n$ are concept and role symbols, respectively.

The knowledge base reasons about the given descriptions by consulting two separate reasoners:

1. *DL* – The *Description Languages* reasoner: A decidable reasoner, that reasons on the basis of the intended meaning of the description operators that form the descriptions.

2. $R$ – The *Rules* reasoner, that reasons on the basis of given rules and some agreed upon semantical policy (e.g., perfect model). The rules are logic rules whose atoms are descriptions.

This architecture is described in Figure 1.

While in query mode the $\mathcal{DFL}$ manager dispatches queries to the two reasoners. The reasoners make efforts to answer. If they succeed, they return an answer(s) to the manager. Intermediate results, obtained by one reasoner, can be useful to the other. Note that the two reasoners can operate under different semantical policies. Problems of mismatch among the different components are avoided due to the common underlying semantics of F-Logic.

## 2 Example

We use the industrial plants example from the BACK manual ([7]), to demonstrate the operation of the $\mathcal{DFL}$ architecture. Descriptions are built with the operators **and, all, some, domain, range, inv**. The *Descriptions Database*, *D*, is given in Table 1. The *DL* reasoner reasons on the basis of the built-in meaning of the description operators. The *R* reasoner consults the set of rules *RULES*, given in Table 2.

**Reasoning**: Consider the query "Find a mechanical product that is produced by a dangerous plant":

$$\begin{aligned} ?X \in \textbf{and}( \ & mechanical\_product, \\ & \textbf{some}(\textbf{inv}(produces), dangerous\_plant)) \end{aligned}$$

The answer $product_2$ can be obtained only with collaboration of the two reasoners. *DL*, using ($t12$), ($a1$), ($a3$), infers that $product_2$ is a mechanical product. *R*, using ($r1$) and ($a2$), infers that $waste_2$ is a toxic waste. *DL* infers, from ($a4$), that $(dump, waste_2\ ) \in \textbf{inv}(buried\_at)$, and from ($t13$) and ($t10$), that *dump* is a *risky_place*. *DL* infers from ($t12$) and ($a1$), that $plant_1$ is a plant, which enables *R* to infer, using ($a5$) and ($r2$), that $plant_1$ is also a dangerous plant. Eventually, *DL* infers, using ($a3$), that $product_2$ is in $\textbf{some}(\textbf{inv}(produces), dangerous\_plant)$.

## 3 DLs – as a sorted F-Logic Language

Let $P$ be a (finite) set of description operators. We define $L_P$, a description language with description operators in $P$, as a sorted, rather restricted, F-Logic language. The definition is not dependent on $P$.

### Syntax:

1. **Symbols**:
   $P$ – a finite set of *description operators.*
   $C$ – a set of *concept symbols.* $C = C_p \cup C_d \cup \{top,\ bottom\}$, where $C_p$ is the set of *primitive concept symboles*, and $C_d$ is the set of *defined concept symbols.*
   $R$ – a set of *role symbols.* $R = R_p \cup R_d$, where $R_p$ is the set of *primitive role symboles*, and $R_d$ is the set of *defined role symbols.*
   $O$ – a set of *individual symbols* (also called *object symbols*).
   $S$ – a set of three *sort symbols*: $\{concept,\ role,\ individual\}$.
   $\Psi$ – a *sort assignment.* $\Psi : C \rightarrow \{concept\}$, $\Psi : R \rightarrow \{role\}$, $\Psi : O \rightarrow \{individual\}$; an n-ary operator in $P$ is assignd a sort in the form of an $n+1$ tuple over S. For example, $\Psi(\textbf{all}) = (role,\ concept,\ concept)$.

2. **Well-Sorted Terms**: All concept, role, and object symbols are terms; their sorts are given by $\Psi$. Complex terms are formed by well sorted applications of description operators to terms. A complex term $\textbf{op}(t_1,\ \ldots,\ t_n)$, where $\Psi(\textbf{op}) = (s_1,\ \ldots,\ s_{n+1})$, is well sorted if the sort of $t_i$ $(1 \le i \le n)$ is $s_i$. The sort of the term is $s_{n+1}$. Below we use $c$, $r$, $o$ to denote concept, role, and individual (object) terms, respectivly. Defined concept or role symbols are denoted $c_d$ and $r_d$, respectively.
3. **Formulae**: $c_d \doteq c$; $c_1 \le c_2$; $o \in c$;
   $\forall_{ind} X, Y, (X[r_d \twoheadrightarrow \{Y\}] \equiv X[r \twoheadrightarrow \{Y\}])$;[1]
   Syntactic shortcut: $r_d \doteq r$.
   $\forall_{ind} X, Y, (\ X[r_1 \twoheadrightarrow \{Y\}] \longrightarrow X[r_2 \twoheadrightarrow \{Y\}])$;
   Syntactic shortcut: $r_1 \le r_2$.
   $o_1[r \twoheadrightarrow \{o_2\}]$; Syntactic shortcut: $(o_1, o_2) \in r$.

### Semantics:

The semantics of $L_P$, as an F-Logic language, is defined over a partially ordered domain $U$, with a greatest and a least elements, where terms are mapped to elements of $U$, and the symbols *top* and *bottom* are assigned the greatest and least elements, respectively. Formulae are interpreted by interpreting $\doteq$ and $\le$ between concept terms as equality and the partial ordering, respectively; $\in$ between an object term (usually a symbol) and a concept term is interpreted as the membership binary relation over $U$; $\doteq$ and $\le$ between role terms are interpreted as methods' equality and implication, respectively; $\in$ between a pair of object terms (symbols) and a role term is interpreted as a method's value assertion.

The intended meaning of the description operators is given by an F-Logic theory $FL_P$, called *the corresponding theory* for $L_P$. $FL_P$ is not part of $L_P$. Its main property is that it provides equivalence with the standard set-theoretic semantics of description languages. That is, for every set of formulae $\Gamma$, and a formula $\gamma$ in $L_P$:

$$\Gamma \overset{DL}{\models} \gamma \quad iff \quad FL_P,\ \Gamma \overset{FL}{\models} \gamma \quad ,$$

where $\overset{DL}{\models}$ denotes logical implication under the set-theoretic semantics of description languages, and $\overset{FL}{\models}$ denotes logical implication in F-Logic. For further details consult [1], where a corresponding theory for $P = \{$**and, all, at-least1, and-role** $\}$ is given.

---

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

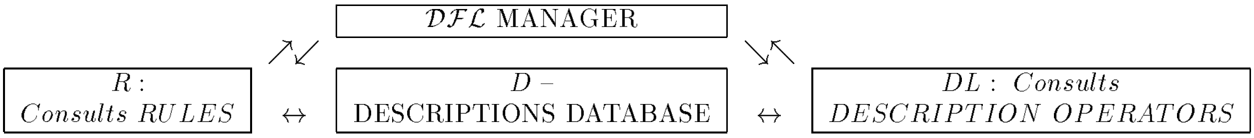
$\mathcal{DFL}$ MANAGER
$R:$ Consults RULES
$D$ – DESCRIPTIONS DATABASE
$DL:$ Consults DESCRIPTION OPERATORS

Figure 1: Architecture of a $\mathcal{DFL}$ KB

| Kind | No. | Description in words | description |
|---|---|---|---|
| INTENSIONAL | | | |
| Primitive-concept | t1) | *product is_a top* | $product \leq top$ |
| | t2) | *place is_a top* | $place \leq top$ |
| | t3) | *mechanical_product is_a product* | $mechanical\_product \leq product$ |
| | t4) | *waste is_a product* | $waste \leq product$ |
| | t5) | *radioactive_material is_a product* | $radioactive\_material \leq product$ |
| | t6) | *toxic_waste is_a product* | $toxic\_waste \leq product$ |
| | t7) | *plant is_a top* that is *located_at place* | $plant \leq \mathbf{and}(top, \mathbf{all}(located\_at, place))$ |
| | t8) | *dangerous_plant is_a plant* | $dangerous\_plant \leq plant$ |
| Primitive-role | t9) | A *plant produces products*, or, *produces* is a relation between *plants* and *products* | $produces \leq \mathbf{and}(\mathbf{domain}(plant), \mathbf{range}(product)\ )$ |
| | t10) | A *product* may be *buried_at* a *place* | $buried\_at \leq \mathbf{and}(\mathbf{domain}(product), \mathbf{range}(place)\ )$ |
| | t11) | *located_at* is a relation between objects and *places* | $located\_at \leq \mathbf{range}(place)$ |
| Defined-concept | t12) | A *mechanical_plant* is a *plant* that *produces* only *mechanical_products* | $mechanical\_plant \doteq \mathbf{and}(plant, \mathbf{all}(produces, mechanical\_product))$ |
| | t13) | A *risky_place* is a *place* where a *toxic_waste* is *buried_at* | $risky\_place \doteq \mathbf{and}(place, \mathbf{some}(\mathbf{inv}(buried\_at), toxic\_waste))$ |
| EXTENSIONAL | | | |
| Concept-member | a1) | $plant_1$ is a *mechanical plant* | $plant_1 \in mechanica\_plant$ |
| | a2) | $waste_2$ is a *radioactive waste* | $waste_2 \in \mathbf{and}(waste, radioactive\_material)$ |
| Role-member | a3) | $plant_1$ produces a product $product_2$ | $(plant_1\ , product_2) \in produces$ |
| | a4) | $waste_2$ is *buried_at* place *dump* | $(waste_2\ , dump) \in buried\_at$ |
| | a5) | $plant_1$ is *located_at dump* | $(plant_1\ , dump) \in located\_at$ |

Table 1: $D$ – The Descriptions Data Base; intensional descriptins and extensional assertions

| Kind | No. | Description in words | rule |
|---|---|---|---|
| *RULES* | r1 | A *radioactive_material* that is also a *waste*, is a *toxic_waste*. | $X \in toxic\_waste \longleftarrow$<br>$X \in \mathbf{and}(radioactive\_material, waste).$ |
| | r2 | If a *plant* is *located_at* a *risky_place*, it is a *dangerous_plant*. | $Y \in dangerous\_plant \longleftarrow$<br>$Y \in plant,\ (Y, X) \in located\_at,$<br>$X \in risky\_place.$ |

Table 2: The set of rules *RULES*

A major result of [1] is that given a corresponding theory $FL_P$ to $L_P$, the semantics of F-Logic provides a full account to $L_P$, i.e., it correctly simulates logical implication and subsumption relations, while preserving the direct semantics. This is summarized in the following corrolary:

**Corollary 3.1** *Let $t_1, t_2$ be terms of $L_P$, and $FL_P$ an F-Logic's theory that corresponds to $L_P$. Then, $t_1$ is subsumbed by $t_2$ in a terminology $\Gamma$ iff $FL_P,\ \Gamma \overset{FL}{\models} t_1 \leq t_2$.*

## 4 Compositional Semantics

The compositional semantics preserves the **modularity** of the *DL* and the *R* reasoners. It is composed from the separate semantics of *DL* and *R*, which may operate along different reasoning policies. The semantics consists of a set of syntactic objects, either in DL terms, or in terms of the underlying F-Logic formalism. It is constructed by *iteration* of the *separate* semantics $\mathcal{DL}$ and $\mathcal{R}$ of the *DL* and the *R* reasoners, respectively. $\mathcal{DL}$ and $\mathcal{R}$ are also sets of syntactic objects. This way the principles of *modularity* and *Compositionality* are kept. The general structure of the compositional semantics is visualized in Figure 2. $\mathcal{DFL}$ is formally defined as follows:

Define: $T(KB) \overset{def}{=} \mathcal{DL} \cup \mathcal{R}$
and $T^0(KB) = S^0$ – semantics dependent initial version.
$T^{k+1}(KB) = T(T^k(KB)) \quad k \geq 0$
$T^\omega(KB) = \bigcup_{k=0}^{\infty} T^k(KB)$
Then: $\mathcal{DFL}(KB) \overset{def}{=} T^\omega(KB)$

The compositional semantics does not specify the $\mathcal{DL}$ and the $\mathcal{R}$ semantics. We investigate four alternative compositional semantics, called *H*, *F*, *singleF*, and *OF*, that differ in the separate $\mathcal{DL}$ and $\mathcal{R}$ being used, and in the sort of syntactic objects being processed. In the *H* semantics the syntactic objects are ground atoms of the underlying F-Logic formalism; in the *F* and the *singleF* semantics the syntactic objects are ground descriptions; in the *OF* semantics the syntactic objects are descriptions (not necessarily ground), and rules of the *R* reasoner ([3]). Hence, *H* is neither query sensitive nor open, *F* and *singleF* are query sensitive but not open, and *OF* is both query sensitive and open. The *H* semantics is not natural since it follows the standard Herbrand style semantics, rather than reflecting the subject matter. The expressivity relations between the four semantics are:

$$H \leq F = singleF \leq OF$$

with the reservations: 1)$F = singleF$ holds only when the *R* reasoner consults a set of definite positive rules, without negation, and 2) The *OF* semantics is defined only for an *R* reasoner that consults a set of definite positive rules. For detailed explanations and proofs consult [2].

In the following example, a $\mathcal{DFL}$ framework over a DL language with a single description operator, **atleast1**, is presented. In the *OF* semantics, role relations that are obtained by the *R* reasoner, allow the *DL* reasoner to obtain new descriptions, that could not be obtained otherwise.

**Example 1**

$P:$ $\{\mathbf{atleast1}\}$
$D:$ 1)$c \leq \mathbf{atleast1}(r_1, d)$
$RULES:$ 2)$(X,Y) \in r_2 \longleftarrow p.$
3)$p \longleftarrow (X,Y) \in r_1.$

*The OF semantics:*
$\mathcal{DFL}(D \cup RULES) = OF(D) =$
$\{$ (1), (2), (3), 4)$r_1 \leq r_2$
5)$c \leq \mathbf{atleast1}(r_2, d)$,
6)$\mathbf{atleast1}(r_1, c) \leq \mathbf{atleast1}(r_2, c)$,
7)$\mathbf{atleast1}(r_1, d) \leq \mathbf{atleast1}(r_2, d)\}$

The *F* semantics:
$\mathcal{DFL}(D) = F(D) = \{\ (1)\ \}$

## 5 Current Research

We are now interested in specific evaluation methods for query-answering in the $\mathcal{DFL}$ framework. We look into top-down and bottom-up approaches.

In the *DL* reasoner, we can use an off-the-shelf DL system, like BACK ([7]) or CLASSIC ([9]). The disadvantage of this solution is that no partial results are collected during evaluation, and practically, it requires duplication of the descriptions database. A bottom-up DL reasoning method is preferable, and the work of [10] can lead to one.

In the *R* reasoner, we intend to use either a top-down + tabulation method ([6; 11]), or a bottom-up + magic set method. We note that description operators play the role of object constructors. Hence, unless the DL language includes no description operators, the semantics may be infinite.

## References

[1] M. Balaban. The f-logic approach for description languages. *Annals of Mathematics and Artificial Intelligence*, 15,19-60,1995.

[2] M. Balaban. The f(rames)-logic approach for description languages ii: A hybrid integrating of rules and descriptions. Technical Report FC 94-10 (submitted). ftp lace.bgu.ac.il, cd ftp/pub/mira.

[3] A. Bossi, M. Gabbrielli, G. Levi, and M. Martelli. The s-semantics approach: Theory and applications. *J. of Logic Programming*, 12, 1993.

[4] F. Donini, M. Lenzerini, D. Nardi, A. Schaerf, and W. Nutt. Adding epistemic operators to concept languages. In *KR-92*, pages 342–353, 1992.

[5] P. Hanschke and . Hinkelmann. Combining terminological and rule-based reasoning for abstraction processes. In *German Conference on AI-92, Springer LNCS 671*, 1992.

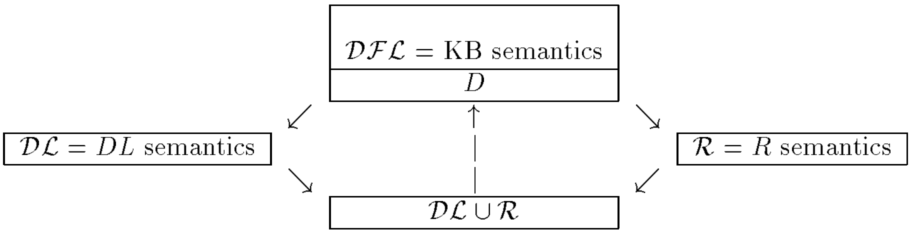
$\mathcal{DFL}$ = KB semantics
$D$
$\mathcal{DL} = DL$ semantics
$\mathcal{R} = R$ semantics
$\mathcal{DL} \cup \mathcal{R}$

Figure 2: Structure of the compositional semantics

[6] T. Hisao and S. Taisuke. OLD Resolution with Tabulation. 3rd Int conference on Logic Programing. 1986.

[7] T. Hoppe, C. Kindermann, J. Quantz, A. Schmiedel, and M. Fischer. Back v5: Tutotial and manual. Technical Report KIT – report 100, Technische Universitat Berlin, March 1993.

[8] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. In *JACM*, 1995.

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

[10] V. Royer and J. Quantz. On intuitionistic query answering in description bases. In *CADE*, 1994.

[11] Wunderwald E.Jens. Memoing Evaluation by Source-to-Source Transformation. In LOPSTR'95.