# Description Logic *vs.* Order-Sorted Feature Logic

Hassan Aït-Kaci

ILOG, Inc.
hak@ilog.com

**Abstract.** *We compare and contrast Description Logic ($\mathcal{DL}$) and Order-Sorted Feature ($\mathcal{OSF}$) Logic from the perspective of using them for expressing and reasoning with knowledge structures of the kind used for the Semantic Web.*

## Introduction

The advent of the Semantic Web has spurred a great deal of interest in various Knowldege Representation formalisms for expressing, and reasoning with, so-called formal *ontologies*. Such ontologies are essentially sets of expressions describing data and properties thereof. Data is generally organized into ordered hierarchies of set-denoting concepts where the order denotes set inclusion. Properties are binary relations involving these concepts. This set-theoretic semantics is amenable to a crisp formal rendering based on first-order logic, and therefore to proof-theoretic operational semantics.

Description Logic ($\mathcal{DL}$) and Order-Sorted Feature ($\mathcal{OSF}$) logic are two mathematical formalisms that possess such proof-theories. Both are direct descendants of Ron Brachman's original ideas [1]. This inheritance goes through my own early work formalizing Brachman's ideas [2], which in turn inspired the work of Gert Smolka, who pioneered the use of constraints *both* for the $\mathcal{DL}$ [3] *and* $\mathcal{OSF}$ [4] formalisms. While the $\mathcal{DL}$ approach has become the mainstream of research on the Semantic Web, the lesser known $\mathcal{OSF}$ formalisms have evolved out of Unification Theory [5], and been used in Constraint-Logic Programming and Computational Linguistics [6–19].

In this short communication (extracted from [20]), we compare and contrast $\mathcal{DL}$ and $\mathcal{OSF}$ logics with the purpose of using them effectively for ontological representation and reasoning.

## Relation between $\mathcal{DL}$ and $\mathcal{OSF}$ Formalisms

The two formalisms for describing attributed typed objects of interest—*viz.*, $\mathcal{DL}$ and $\mathcal{OSF}$—have several common, as well as distinguishing, aspects. Thanks to both formalisms using the common language of $\mathcal{FOL}$ for expressing semantics, they may thus be easily compared—see, for example, [21, 22]. We here brush on some essential points of comparison and contrast.[1]

**Common Aspects** $\mathcal{DL}$ reasoning is generally carried out using (variations on) Deductive Tableau methods [23].[2] This is also the case of the constraint propagation rules of Fig. 1, which simply mimick a Deductive Tableau decision procedure [24].[3] $\mathcal{OSF}$ reasoning is performed by the $\mathcal{OSF}$-constraint normalization rules of Figs. 2 and 3, which implement a logic of sorted-feature equality.

[1] Due to severe, and strictly enforced, space limitation in these proceedings, most of the points we make here are further elaborated for the interested reader in [20].

[2] Although one can find some publications on Description Logics that do not (fully) use Tableaux reasoning for their operational semantics and mix it with resolution (*i.e.*, Prolog technology), the overwhelming majority follow the official W3C recommendations based on Tableaux methods for TBox reasoning.

[3] The constraint-rule notation we use is Plotkin's SOS style [30]. The constraint system $\mathcal{ALCNR}$ is given here as an exemplar of a DL Tableaux-based reasoning system. It is neither the most expressive nor the most efficient. However, it uses the same style of formula-expansion rules used by all Tableaux-based DL systems such as, in particular, the ever-growing family of esoterically-named Description Logics $\mathcal{SHIQ}$, $\mathcal{SHOIN}$, $\mathcal{SHOIQ}$, $\mathcal{SHOQ}$(D), $\mathcal{SRIQ}$, and other $\mathcal{SROIQ}$, which underlie all the official nocturnal bird languages promoted by the W3C to enable the Semantic Web—see for example the "official" DL site (`http://dl.kr.org/`) as well as the output of one of its most prolific spokesperson (`http//www.cs.man.ac.uk/~horrocks/Publications/`).

($\mathcal{C}_{\sqcap}$) **CONJUNCTIVE CONCEPT**:

$$\left[\begin{array}{l}\textbf{if } x : (C_1 \sqcap C_2) \in S \\ \textbf{and } \{x : C_1, x : C_2\} \not\subseteq S\end{array}\right] \qquad \frac{S}{S \cup \{x : C_1, x : C_2\}}$$

($\mathcal{C}_{\sqcup}$) **DISJUNCTIVE CONCEPT**:

$$\left[\begin{array}{l}\textbf{if } x : (C_1 \sqcup C_2) \in S \\ \textbf{and } x : C_i \notin S \ (i = 1, 2)\end{array}\right] \qquad \frac{S}{S \cup \{x : C_1\}}$$

($\mathcal{C}_{\forall}$) **UNIVERSAL ROLE**:

$$\left[\begin{array}{l}\textbf{if } x : (\forall R.C) \in S \\ \textbf{and } y \in R_S[x] \\ \textbf{and } y : C \notin S\end{array}\right] \qquad \frac{S}{S \cup \{y : C\}}$$

($\mathcal{C}_{\exists}$) **EXISTENTIAL ROLE**:

$$\left[\begin{array}{l}\textbf{if } x : (\exists R.C) \in S \textbf{ s.t. } R \stackrel{\text{DEF}}{=} (\bigsqcap_{i=1}^{m} R_i) \\ \textbf{and } z : C \in S \implies z \notin R_S[x] \\ \textbf{and } y \text{ is new}\end{array}\right] \qquad \frac{S}{S \cup \{xR_iy\}_{i=1}^{m} \cup \{y : C\}}$$

($\mathcal{C}_{\geq}$) **MIN CARDINALITY**:

$$\left[\begin{array}{l}\textbf{if } x : (\geq n.R) \in S \textbf{ s.t. } R \stackrel{\text{DEF}}{=} (\bigsqcap_{i=1}^{m} R_i) \\ \textbf{and } |R_S[x]| \neq n \\ \textbf{and } y_i \text{ is new } (0 \leq i \leq n)\end{array}\right] \qquad \frac{S}{\begin{array}{l}S \cup \{xR_iy_j\}_{i,j=1,1}^{m,n} \\ \quad \cup \{y_i \not\equiv y_j\}_{1 \leq i < j \leq n}\end{array}}$$

($\mathcal{C}_{\leq}$) **MAX CARDINALITY**:

$$\left[\begin{array}{l}\textbf{if } x : (\leq n.R) \in S \\ \textbf{and } |R_S[x]| > n \\ \textbf{and } y, z \in R_S[x] \\ \textbf{and } y \not\equiv z \notin S\end{array}\right] \qquad \frac{S}{S \cup S[y/z]}$$

**Fig. 1.** Some $\mathcal{DL}$-constraint propagation rules ($\mathcal{ALCNR}$)

§**Object Descriptions—**Both the $\mathcal{DL}$ and $\mathcal{OSF}$ formalisms describe typed attributed objects. In each, objects are data structures described by combining set-denoting concepts and relation-denoting roles.

§**Logic-Based Semantics—**Both $\mathcal{DL}$ and $\mathcal{OSF}$ logic are syntatic formalisms expressing meaning using conventional logic styles. In other words, both formalisms take their meaning in a common universal language—*viz.*, (elementary) Set Theory. This is good since it eases understanding each formalism in relation to the other thanks to their denotations in the common language.

§**Proof-Theoretic Semantics—**Both $\mathcal{DL}$ and $\mathcal{OSF}$ logics have their corresponding proof theory. Indeed, since both formalisms are syntactic variants of fragments of $\mathcal{FOL}$, proving theorems in each can always rely on $\mathcal{FOL}$ mechanized theorem proving.

§**Constraint-Based Formalisms—**Even further, both $\mathcal{DL}$ and $\mathcal{OSF}$ logic are operationalized using a constraint-based decision procedure. As we have expounded, this makes both paradigms amenable to being manipulated by rule-based systems such as based on $\mathcal{CLP}$, rewrite rules, or production rules.

§**Concept Definitions—**Both $\mathcal{DL}$ and $\mathcal{OSF}$ provide a means for defining concepts in terms of other concepts. This enables a rich framework for expressing recursive data structures.

**Distinguishing Aspects** There are also aspects in each that distinguish the $\mathcal{DL}$ and $\mathcal{OSF}$ formalisms apart. However, several of these distinguishing features are in fact cosmetic—*i.e.*, are simply equivalent notation for the same meaning. Remaining non-cosmetic differences are related to the nature of the deductive processes enabled out by each formalism.

§**Functional Features *vs.* Relational Roles—**The $\mathcal{OSF}$ formalism uses *functions* to denote attributes while the $\mathcal{DL}$ formalism uses *binary relations* for the same purpose. Many have argued that this difference is fundamental and restricts the expressivity of $\mathcal{OSF}$ *vs.* $\mathcal{DL}$. This, however, is only a cosmetic difference as we have already explained. First of all, a function $f : A \mapsto B$ *is* a binary relation since $f \in A \times B$. It a *functional* relation because it obeys the axiom of functionality; namely, $\langle a, b\rangle \in f \;\&\; \langle a, b'\rangle \in f \;\Rightarrow\; b = b'$. In other words, a function is a binary relation that associates at most one range element to any domain element. This axiom is fundamental as it is is used in basic $\mathcal{OSF}$ unification *"Feature Functionality"* shown in Fig. 2. Indeed, the correctness of this rule relies on the semantics of features as functions, not as relations.

---

$(\mathcal{O}_1)$ **SORT INTERSECTION**: $\dfrac{\phi \;\&\; X : s \;\&\; X : s'}{\phi \;\&\; X : s \wedge s'}$

$(\mathcal{O}_2)$ **INCONSISTENT SORT**: $\dfrac{\phi \;\&\; X : \bot}{X : \bot}$

$(\mathcal{O}_3)$ **FEATURE FUNCTIONALITY**: $\dfrac{\phi \;\&\; X.f \doteq X' \;\&\; X.f \doteq X''}{\phi \;\&\; X.f \doteq X' \;\&\; X' \doteq X''}$

$(\mathcal{O}_4)$ **VARIABLE ELIMINATION**: $[\,\textbf{if } X \neq X' \textbf{ and } X \in \textsc{var}(\phi)\,]$ $\dfrac{\phi \;\&\; X \doteq X'}{\phi[X/X'] \;\&\; X \doteq X'}$

$(\mathcal{O}_5)$ **VARIABLE CLEANUP**: $\dfrac{\phi \;\&\; X \doteq X}{\phi}$

---

**Fig. 2.** Basic $\mathcal{OSF}$-constraint normalization rules

However, a relation $R \in A \times B$ is equivalent to either of a pair of set-denoting functions—*viz.*., either the function $R[\_] : A \mapsto \mathbf{2}^B$, returning the *R-object* (or *R-image*) set $R[x] \subseteq B$ of an element $x \in A$; or, dually, the function $R^{-1}[\_] : B \mapsto \mathbf{2}^A$, returning the *R-subject* (or *R-antecedent*) set $R^{-1}[y] \subseteq A$ of an element $y \in B$. Indeed, the following statements ($s_1$)–($s_3$) are equivalent:

$$\begin{array}{lrl} \forall \langle x, y\rangle \in A \times B, & \langle x, y\rangle \in R & (s_1) \\ & y \in R[x] & (s_2) \\ & x \in R^{-1}[y] & (s_3) \end{array}$$

Therefore, it is a simple matter for the $\mathcal{OSF}$ formalism to express relational attributes (or roles) with features taking values as sets. This is trivially done as a special case of the *"Value Aggregation"* $\mathcal{OSF}$ unification rule shown in Fig. 3, using a set data constructor—*i.e.*, a commutative idempotent monoid.

($\mathcal{O}_6$) **PARTIAL FEATURE**:

$$\left[\text{ if } s \in \textsc{dom}(f) \text{ and } \textsc{ran}_s(f) = s' \right] \quad \frac{\phi \;\&\; X.f \doteq X'}{\phi \;\&\; X.f \doteq X' \;\&\; X : s \;\&\; X' : s'}$$

($\mathcal{O}_7$) **WEAK EXTENSIONALITY**:

$$\left[\begin{array}{l}\text{if } s \in \mathcal{E} \text{ and } \forall f \in \textsc{arity}(s) : \\ \{X.f \doteq Y, X'.f \doteq Y\} \subseteq \phi\end{array}\right] \quad \frac{\phi \;\&\; X : s \;\&\; X' : s}{\phi \;\&\; X : s \;\&\; X \doteq X'}$$

($\mathcal{O}_8$) **VALUE AGGREGATION**:

$$\left[\begin{array}{l}\text{if } s \text{ and } s' \text{ are both subsorts of} \\ \text{commutative monoid } \langle \star, \mathbf{1}_\star \rangle\end{array}\right] \quad \frac{\phi \;\&\; X = e : s \;\&\; X = e' : s'}{\phi \;\&\; X = e \star e' : s \wedge s'}$$

**Fig. 3.** Additional $\mathcal{OSF}$-constraint normalization rules

§**Sets *vs.* Individuals—**Because the $\mathcal{OSF}$ formalism has only set-denoting sorts, it is often misconstrued as unable to deal with individual elements of these sets. However, as explained in [20], this is again an innocuous cosmetic difference since elements are simply assimilated to singleton-denoting sorts.

§**No Number Restrictions *vs.* Number Restrictions—**Strictly speaking, the $\mathcal{OSF}$ formalism has no special constructs for number restrictions as they exist in $\mathcal{DL}$. Now, this does not mean that it lacks the power to enforce such constraints. Before we show how this may be done, however, it important to realize that it may not always be a good idea to use the $\mathcal{DL}$ approach to do so.

Indeed, as can be seen in Fig. 1, the *"Min Cardinality"* rule ($\mathcal{C}_{\leq}$) will introduce $n(n-1)/2$ new disequality constraints for each such constraint of cardinality $n$. Clearly, this is a source of gross inefficiency a $n$ increases. Similarly, the *"Existential Role"* rule ($\mathcal{C}_{\exists}$) will systematically introduce a new variable for a role, *even when this role is never accessed!* It does so because, it materializes the full extent of role value sets. In other words, $\mathcal{C}$ constraint-propagation rules flesh out complete skeletons for attributed data structures whether or not the actual attribute values are needed.

By contrast, it is simple and efficient to accommodate cardinality constraints in the $\mathcal{OSF}$ calculus with value aggregation using a set constructor (*i.e.*, an idempotent commutative monoid $M = \langle \star, \mathbf{1}_\star \rangle$), and a function $\textsc{card} : M \mapsto \mathbb{N}$ that returns the number of elements in a set. Then, imposing a role cardinality constraint for a role $r$ in a feature term $t = X : s(r \Rightarrow S = \{e_1, \ldots, e_n\} : m)$, where sort $m$ denotes $M$'s domain, is achieved by the constraint $\varphi(t)$ & $\textsc{card}(S) \leq n$—or $\varphi(t)$ & $\textsc{card}(S) \geq n$. If the set contains variables, these constraints will residuate as needed pending the complete evaluation of the function **CARD**. However, as soon as enough non-variable elements have materialized in the set that enable the decision, the constraint will be duly enforced. Clearly, this "lazy" approach saves the time and space wasted by $\mathcal{DL}$-propagation rules, while fully enforcing the needed cardinalities.

Incidentally, note also that this principle allows not only min and max cardinality, but any constraints on a set, whether cardinality or otherwise. Importantly, this foregoing method works not only for sets, but can be used with arbitrary aggregations using other monoids.

§**Greatest Fix-Point *vs.* Least Fix-Point—**It is well known that unfolding recursive definitions of all kinds (be it function, relation, or sort) is precisely formalized as computing a fix-point in some information-theoretic lattice. Indeed, given a complete lattice $\mathfrak{L} \stackrel{\text{DEF}}{=} \langle D^{\mathfrak{L}}, \sqsubseteq^{\mathfrak{L}}, \sqcap^{\mathfrak{L}}, \sqcup^{\mathfrak{L}}, \top^{\mathfrak{L}}, \bot^{\mathfrak{L}} \rangle$ and a monotone function[4] $\mathcal{F} : D^{\mathfrak{L}} \mapsto D^{\mathfrak{L}}$, Tarski's fix-point theorem[5] states that the set $\textsc{fp}(\mathcal{F}) \stackrel{\text{DEF}}{=} \{x \in D^{\mathfrak{L}} \mid \mathcal{F}(x) = x\}$ of

[4] That is, such that: $\forall x, y \in D^{\mathfrak{L}},\ x \sqsubseteq^{\mathfrak{L}} y \implies \mathcal{F}(x) \sqsubseteq^{\mathfrak{L}} \mathcal{F}(y)$.

[5] See, *e.g.*, [25].

fix-points of $\mathcal{F}$ is itself a complete sublattice of $\mathfrak{L}$. Moreover, its bottom element is called $\mathcal{F}$'s *least fix-point* (LFP), written $\mathcal{F}^{\uparrow}$, defined by Equation (1):

$$\mathcal{F}^{\uparrow} \stackrel{\text{DEF}}{=\!=} \bigsqcup_{n\in\mathbb{N}}^{\mathfrak{L}} \mathcal{F}^n\big(\bot^{\mathfrak{L}}\big) \tag{1}$$

and its top element is called $\mathcal{F}$'s *greatest fix-point* (GFP), written $\mathcal{F}^{\downarrow}$, defined by Equation (2):

$$\mathcal{F}^{\downarrow} \stackrel{\text{DEF}}{=\!=} \bigsqcap_{n\in\mathbb{N}}^{\mathfrak{L}} \mathcal{F}^n\big(\top^{\mathfrak{L}}\big) \tag{2}$$

where:

$$\mathcal{F}^n(x) = \begin{cases} x & \textbf{if } n = 0, \\ \mathcal{F}(\mathcal{F}^{n-1}(x)) & \textbf{otherwise}. \end{cases}$$

Informally, $\mathcal{F}^{\uparrow}$ is the *upward* iterative limit of $\mathcal{F}$ starting from the least element in $D^{\mathfrak{L}}$, while $\mathcal{F}^{\downarrow}$ is its *downward* iterative limit starting from the greatest element in $D^{\mathfrak{L}}$. One can easily show that $\mathcal{F}(\mathcal{F}^{\uparrow}) = \mathcal{F}^{\uparrow}$ [resp., $\mathcal{F}(\mathcal{F}^{\downarrow}) = \mathcal{F}^{\downarrow}$], and that no element of $D^{\mathfrak{L}}$ lesser than $\mathcal{F}^{\uparrow}$ [resp., greater than $\mathcal{F}^{\downarrow}$ ] is a fix-point of $\mathcal{F}$.

One may wonder when one, or the other, kind of fix-point captures the semantics intended for a set of recursive definitions. Intuitively, LFP semantics is appropriate when inference proceeds by deriving *necessary consequences* from facts that hold true, and GFP semantics is appropriate when inference proceeds by deriving *sufficient conditions* for facts to hold true.[6] Therefore, LFP computation can model only well-founded (*i.e.*, terminating) recursion, while GFP computation can also model non well-founded (*i.e.*, not necessarily terminating) recursion. Hence, typically, LFP computation is naturally described as a *bottom-up* process, while GFP computation is naturally described as a *top-down* process.

An example of GFP semantics is given by the Herbrand-term unification. Indeed, this process transforms a set of equations into an equivalent one using sufficient conditions by processing the terms top-down from roots to leaves. The problem posed is to find sufficient conditions for a term equation to hold on the constituents (*i.e.*, the subterms) of both sides of the equation. For first-order terms, this process converges to either failure or producing a most general sufficient condition in the form of a variable substitution, or equation set in solved form (the MGU). Similarly, the $\mathcal{OSF}$-constraint normalization rules of Figs. 2, 4, 5, and 3 also form an example of converging GFP computation for the same reasons. Yet another example of GFP computation where the process may diverge is the lazy recursive sort definition unfolding described in [26].

On the other hand, constraint-propagation rules based on Deductive Tableau methods such as used in [3] or shown in Fig. 1 are LFP computations. Indeed, they proceed bottom-up by building larger and larger constraint sets by completing them with additional (and often redundant) constraints. In short, $\mathcal{OSF}$-constraint normalization follows a reductive semantics (it eliminates constraints) while $\mathcal{DL}$-constraint propagation follows an inflationary semantics (it introduces constraints). As a result, $\mathcal{DL}$'s tableau-style reasoning method is *expansive*—therefore, *expensive* in time and space. One can easily see this simply by realizing that each rule in Fig. 1 builds a larger set $S$ as it keeps adding more constraints and more variables to $S$. Only the *"Max Cardinality"* rule ($\mathcal{C}_{\leq}$) may reduce the size of $S$ to enforce upper limits on a concept's extent's size by merging two variables. Finally, it requires that the constraint-solving process be decidable.

By contrast, the $\mathcal{OSF}$ labelled-graph unification-style reasoning method is more efficient both in time and space. Moreover, it can accommodate semi-decidable—*i.e.*, undecidable, though recursively enumerable—constraint-solving. Indeed, no rule in Figs. 2, 4, 5, and 3 ever introduces a new variable. Moreover, all the rules in Fig. 2 as well as the rule 3, except for the *"Partial Feature"* rule, all eliminate constraints. Even this latter rule introduces no more constraints than the number of features in the whole constraint. The rules in Figs. 4 and 5 may replace some constraints with more constraints, but the introduced constraints are all more restrictive than those eliminated.

[§]**Coinduction *vs.* Induction—**Remarkably, the interesting duality between least and greatest fix-point computations is in fact equivalent to another fundamental one; namely, *induction vs. coinduction* in computation

[6] One might also say that LFP is *deductive* since it moves from premiss to consequent, and that GFP is *abductive* since it moves from consequent to premiss.

$(\mathcal{O}_9)$ **NON-UNIQUE GLB**:

$$\left[\begin{array}{l}\textbf{if } \{s_1\}_{i=0}^{n} = \max_{\leq}\{t \in \mathcal{S} \mid t \leq s\} \\ \textbf{and } t \leq s'\}\end{array}\right] \frac{\phi \ \& \ X : s \ \& \ X : s'}{\phi \ \& \ \left(X : s_1 \ \| \ \ldots \ \| \ X : s_n\right)}$$

$(\mathcal{O}_{10})$ **DISTRIBUTIVITY**:

$$\frac{\phi \ \& \ \left(\phi' \ \| \ \phi''\right)}{\left(\phi \ \& \ \phi'\right) \ \| \ \left(\phi \ \& \ \phi''\right)}$$

$(\mathcal{O}_{11})$ **DISJUNCTION**:

$$\frac{\phi \ \| \ \phi'}{\phi}$$

**Fig. 4.** Disjunctive $\mathcal{OSF}$-constraint normalization

$(\mathcal{O}_{12})$ **DISEQUALITY**:

$$\frac{\phi \ \& \ X \neq X}{\bot}$$

$(\mathcal{O}_{13})$ **COMPLEMENT**:

$$\left[\begin{array}{l}\textbf{if } s' \in \max_{\leq}\{t \in \mathcal{S} \mid t \not\leq s \\ \textbf{and } t \not\leq s\}\end{array}\right] \frac{\phi \ \& \ X : \overline{s}}{\phi \ \& \ X : s'}$$

**Fig. 5.** Negative $\mathcal{OSF}$-constraint normalization

and logic, as nicely explained in [27]. Indeed, while induction allows to derive a whole entity from its constituents, coinduction allows to derive the constituents from the whole. Thus, least fix-point computation is induction, while greatest fix-point computation is coinduction. Indeed, coinduction is invaluable for reasoning about non well-founded computations such as those carried out on potentially infinite data structures [28], or (possibly infinite) process bisimulation [29].

This is a fundamental difference between $\mathcal{DL}$ and $\mathcal{OSF}$ formalisms: $\mathcal{DL}$ reasoning proceeds by actually building a model's domain verifying a TBox, while $\mathcal{OSF}$ reasoning proceeds by eliminating impossible values from the domains. Interestingly, this was already surmised in [3] where the authors state:

> *"[...] approaches using feature terms as constraints [...] use a lazy classification and can thus tolerate undecidable subproblems by postponing the decision until further information is available. [...these] approaches are restricted to feature terms; however, an extension to KL-ONE-like concept terms appears possible."*

Indeed, the extended $\mathcal{OSF}$ formalism overviewed in [20] is a means to achieve precisely this.

## Conclusion

We have briefly reviewed two well-known data description formalisms based on constraints, Description Logic and Order-Sorted Feature Logic, explicating how they work and how they are formally related. We

have identified similarities and differences by studying their set-theoretic semantics and first-order logic proof-theory based on constraint-solving. In so doing, we identified that the two formalisms differ essentially as they follow dual constraint-based reasoning strategies, $\mathcal{DL}$ constraint-solving being inductive (or eager), and $\mathcal{OSF}$ constraint-solving being coinductive (or lazy). This has as consequence that $\mathcal{OSF}$ logic is more effective at dealing with infinite data structures and semi-decidable inference.

It seems therefore evident that, since the $\mathcal{DL}$ and $\mathcal{OSF}$ formalisms are one another's formal *duals*, both semantically and pragmatically, we should be well-advised to know precisely *when* one or the other technology is more appropriate for *what* Semantic Web reasoning task.

## References


1. Brachman, R.: A Structural Paradigm for Representing Knowledge. PhD thesis, Harvard University, Cambridge, MA, USA (1977)
2. Aït-Kaci, H.: A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Types. PhD thesis, University of Pennsylvania, Philadelphia, PA (1984)
3. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence **48** (1991) 1–26
4. Smolka, G.: A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland, Stuttgart, Germany (1988)
5. Schmidt-Schauß, M., Siekmann, J.: Unification algebras: An axiomatic approach to unification, equation solving and constraint solving. Technical Report SEKI-report SR-88-09, FB Informatik, Universität Kaiserslautern (1988) [Available online [7]].
6. Aït-Kaci, H.: An introduction to LIFE—Programming with Logic, Inheritance, Functions, and Equations. In Miller, D., ed.: Proceedings of the International Symposium on Logic Programming, MIT Press (1993) 52–68 [Available online [8]].
7. Aït-Kaci, H., Dumant, B., Meyer, R., Podelski, A., Van Roy, P.: The Wild LIFE handbook. [Available online [9]] (1994)
8. Aït-Kaci, H., Podelski, A.: Towards a meaning of LIFE. Journal of Logic Programming **16**(3-4) (1993) 195–234 [Available online [10]].
9. Carpenter, B.: The Logic of Typed Feature Structures. Volume 32 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK (1992)
10. Dörre, J., Rounds, W.C.: On subsumption and semiunification in feature algebras. In: Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science (Philadelphia, PA), Washington, DC, IEEE, Computer Society Press (1990) 301–310
11. Dörre, J., Seiffert, R.: Sorted feature terms and relational dependencies. In Nebel, B., von Luck, K., Peltason, C., eds.: Proceedings of the International Workshop on Terminological Logics, DFKI (1991) 109–116 [Available online [11]].
12. Emele, M.C.: Unification with lazy non-redundant copying. In: Proceedings of the 29th annual meeting of the ACL, Berkeley, California, Association for Computational Linguistics (June 1991)
13. Emele, M.C., Zajac, R.: A fixed point semantics for feature type systems. In: Proceedings of the 2nd International CTRS Workshop, Montreal (June 1990). Number 516 in Lecture Notes in Computer Science, Springer-Verlag (1992) 383–388
14. Fischer, B.: Resolution for feature logics. In: *GI-Fachgruppe über Alternative Konzepte für Sprachen und Rechner*, GI Softwaretechnik Trends (1993) 23–34 [Available online [12]].
15. Smolka, G.: Feature constraint logic for unification grammars. Journal of Logic Programming **12** (1992) 51–87


[7] `http://www.ki.informatik.uni-frankfurt.de/papers/schauss/unif-algebr.pdf`
[8] `http://koala.ilog.fr/wiki/pub/Main/HassanAitKaci/ilps93.ps.gz`
[9] `http://citeseer.ist.psu.edu/134450.html`
[10] `http://www.hpl.hp.com/techreports/Compaq-DEC/PRL-RR-11.pdf`
[11] `http://elib.uni-stuttgart.de/opus/volltexte/1999/93/`
[12] `http://www.infosun.fmi.uni-passau.de/st/papers/resolution/`

16. Zajac, R.: Towards object-oriented constraint logic programming. In: Proceedings of the ICLP'91 Workshop on Advanced Logic Programming Tools and Formalisms for Natural Language Processing, Paris (1991)
17. Treinen, R.: Feature trees over arbitrary structures. In Blackburn, P., de Rijke, M., eds.: Specifying Syntactic Structures. CSLI Publications and FoLLI (1997) 185–211 [Available online [13]].
18. Müller, M., Niehren, J., Treinen, R.: The first-order theory of ordering constraints over feature trees. Discrete Mathematics & Theoretical Computer Science **4**(2) (2001) 193–234 [Available online [14]].
19. Müller, M., Niehren, J., Podelski, A.: Ordering constraints over feature trees. Constraints **5**(1–2) (2000) 7–42 Special issue on CP'97, Linz, Austria. [Available online [15]].
20. Aït-Kaci, H.: Data models as constraint systems: A key to the semantic web. Research paper, submitted for publication, ILOG, Inc. (2007) [Available online [16]].
21. Nebel, B., Smolka, G.: Representation and reasoning with attributive descriptions. In Blasius, K., Hedtstuck, U., Rollinger, C.R., eds.: Sorts and Types in Artificial Intelligence. Volume 418 of Lecture Notes in Artificial Intelligence., Springer-Verlag (1990) 112–139
22. Nebel, B., Smolka, G.: Attributive description formalisms and the rest of the world. In Herzog, O., Rollinger, C.R., eds.: Text Understanding in LILOG: Integrating Computational Linguistics and Artificial Intelligence. Volume 546 of Lecture Notes in Artificial Intelligence., Springer-Verlag (1991) 439–452
23. Manna, Z., Waldinger, R.: Fundamentals of deductive program synthesis. In Apostolico, A., Galil, Z., eds.: Combinatorial Algorithms on Words. NATO ISI Series. Springer-Verlag (1991) [Available online [17]].
24. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A.: Reasoning in description logics. In Brewka, G., ed.: Principles of Knowledge Representation. CSLI Publications, Stanford, CA (1996) 191–236 [Available online [18]].
25. Birkhoff, G.: Lattice Theory. 3rd edn. Volume 25 of Colloquium Publications. American Mathematical Society, Providence, RI, USA (1979)
26. Aït-Kaci, H., Podelski, A., Goldstein, S.C.: Order-sorted feature theory unification. Journal of Logic Programming **30**(2) (1997) 99–124 [Available online [19]].
27. Sangiorgi, D.: Coinduction in programming languages. Invited Lecture ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages—POPL'04 (2004) [Available online [20]].
28. Aczel, P.: Non Well-Founded Sets. Center for the Study of Language and Information, Stanford, CA, USA (1988) [Available online [21]].
29. Baeten, J.C.M., Weijland, W.P.: Process Algebra. Volume 18 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK (1990)
30. Plotkin, G.D.: A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Århus, Århus, Denmark (1981) [Available online [22]].

[13] http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/FeatArbStruct.ps
[14] http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/dm040211.ps
[15] http://www.ps.uni-sb.de/Papers/abstracts/ftsub-constraints-99.html
[16] http://koala.ilog.fr/wiki/bin/view/Main/HassanAitKaci/semwebclp.pdf
[17] http://citeseer.ist.psu.edu/manna92fundamentals.html
[18] http://citeseer.ist.psu.edu/article/donini97reasoning.html
[19] http://www.hpl.hp.com/techreports/Compaq-DEC/PRL-RR-32.pdf
[20] http://www.cs.unibo.it/~sangio/DOC_public/TalkPOPL.ps.gz
[21] http://standish.stanford.edu/pdf/00000056.pdf
[22] http://citeseer.ist.psu.edu/plotkin81structural.html