The Essence of $\mathcal{LIFE}$ for the Semantic Web

Hassan Aït-Kaci*

hak@acm.org

Version of August 29, 2014

Abstract

This is a short informal paper overviewing the objectives of the Semantic Web and potential approaches to meet these objectives in terms of formal automated reasoning. It offers to use a reasoning method that would be an alternative to the predominant OWL-based systems—all of whose formal reasoning is derived from Description Logic. The proposed alternative approach—Order-Sorted Feature (OSF) graph constraint solving—is derived from extensions of unification seen as a constraint system. $\mathcal{LIFE}$ is a declarative Constraint-Logic Programming (CLP) system where computation exploits the OSF formalism for the efficient solving of constraints expressed on data organized as sorted graphs. Its name is derived acronymically from $\mathcal{L}$ogic, $\mathcal{I}$nheritance, $\mathcal{F}$unctions, and $\mathcal{E}$quations—its four basic formal paradigms. Drawing from familiar experience with database systems, we first quickly recall what exists today and what "alternative" approaches may be suitable for addressing the challenges faced by the Semantic Web. In essence, this essay exposes in simple terms *why*, *how*, and *what* $\mathcal{LIFE}$'s computational reasoning may be used for the Semantic Web.

Why the Semantic Web?

The advent of the Semantic Web offers an opportunity to explore effective and scalable reasoning with massive actual, albeit raw, data. The formal motivation for the system we are to propose hinges on a key fact: namely, there seems to be a timely pervasive adoption of an emerging standard to represent all Semantic Web (SW) data and knowledge as labeled graphs—most notably, the W3C's Resource Description Framework (RDF)[1] and its Linked Data extension.[2]

There is a huge market in the Semantic Web. Since RDF is becoming a universally accepted Semantic Web standard, then any commercial system that can reason with RDF-based data/knowledge, not only efficiently, but also in a scalable manner, will result in guaranteed profits. The best-known tools for the required technology work only on toy-size examples. The approach proposed in this paper has the capacity to overcome such limitations. The rest of this document explains in simple terms what must be done and how it may be done.

What can be done today?

Since the 1970's, technology has evolved to find ways to share, retrieve, and process information in the form of *data*. Today, the world is trying to network *knowledge*, not just data. Just as it was necessary to standardize data to be shared (*e.g.*, Relational Model), it is now necessary to standardize knowledge to be shared.

*http://www.hassan-ait-kaci.net/

[1] http://en.wikipedia.org/wiki/Resource_Description_Framework

[2] http://www.w3.org/standards/semanticweb/data

The challenge, however, is to agree first on a standard knowledge representation format just as this was done for data; namely, relational tables and relational calculus acting on them. The difficulty is that, in the case of knowledge, there are drastically different levels of computational power depending on the language at hand: from finite decidable models, to simple propositional, to first-order or even second-order logics, all variations of which involve a careful specification of *what* can be expressed and *how* it can be verified.

So, today, this difficulty has yet to be understood depending on which specific level of knowledge representation sophistication is required for which task. Then, ultimately, the goal is to create a system with the intelligence, not just to transmit and process data, but to be able to infer knowledge from data, from which then it can learn and evolve.

How is data/knowledge represented today?

Data representation

Relational databases The enormous majority of data existing today is represented in some form of relational database format—namely, tables of named tuples. Standards for this exist that have been used for decades.

Knowledge representation

> ***"Knowledge representation (KR)** and reasoning is an area of Artificial Intelligence whose fundamental goal is to represent knowledge in a manner that facilitates inferencing [sic] (i.e., drawing conclusions) from knowledge."*[3]

- **RDF**—The Resource Description Framework—is a set of universal XML-based formats defining a standard for representing all linked data for the Semantic Web proposed by the W3C (hence, the name "*Linked Data*" to designate the standard). Informally, one could say that RDF is to linked data what the Relational Model is to relation tables. RDF's triple-based graph model is bringing such expressivity to networked data—a triple is simply an arc in a graph of the form: ⟨*Subject*, *Predicate*, *Object*⟩,[4] which graphically can be depicted as in Figure 1.

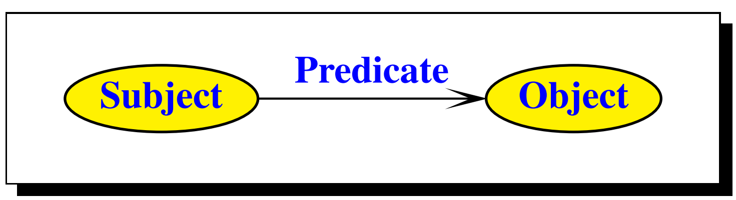

Figure 1: An RDF triple denotes an arrow between two sorted nodes

- **Linked Data**—This is a distributed data representation paradigm based on RDF, the standard that sees everything as a set of triples.[5] The idea behind the Linked Data proposal is quite simple—*do to data what HTML has done to text: interconnect it through Internet.*
- **OWL**—The Ontology Web Language is actually a family of logical languages based on one specific logic called Description Logic (DL) [HS02, BN03].

[3] http://en.wikipedia.org/wiki/Knowledge_representation_and_reasoning

[4] Also: ⟨*Agent*, *Verb*, *Complement*⟩.

[5] See http://blogs.ecs.soton.ac.uk/webteam/2011/07/17/linked-data-vs-open-data-vs-rdf-data/ for a short description of how these notions relate.

- **OSF**—Order-sorted Feature logic is also a family of constraint languages based on representing everything as typed featured graphs [AK11].

The reader interested in understanding how OWL-type and OSF-type formalities differ and relate is referred to [AK07b].

What sort of knowledge representation systems exist today?

There are two "official" Semantic Web families of formalisms that the W3C is officially "supporting:"[6]

- RDF-based: RDF, RDF Schema, RDFa, Linked Data, SKOS;[7]
- OWL-based: SHIF, SHIN, SHOIN, CIQ, SHIQ, SHOQ, SHOIQ, SRIQ, SROIQ, *etc.*, ...

With this proliferation of knowlege representation systems, understanding what can be done by what system is therefore very important.

What are the challenges that we are facing today?

Current systems can only process and manipulate data as long as it is finite. Even then, a bottom-up approach building the models explicitly with all its numerous elements, most of which are not needed, is a crude waste of resources. *Yet, it is current standard!*

Today, there tends to be confusion between the Semantic Web's goal and the means to achieve it. Indeed, DL-based systems, such as the OWL family of logics, are just tools among many, not the SW's goal. The SW's only sensible objective is: *effective, correct and controllable reasoning based on formal tools*—*any tool* not just DL-based [Sta09].

Even if one chooses DL-based logics, the promoters of these systems have yet to develop a clear meta-ontology of their various co-related ontological systems. In other words, one is at a loss having to choose one or the other of these existing tools that best fit the job one wants to do. Indeed, very often, the chosen system is "*too powerful*"—metaphorically, it is like *"using a bazooka to kill a fly!"*[8]

The Web of $\mathcal{LIFE}$: Reconciling *data* bases with *knowledge* bases

> My dear Watson, how often have I said to you that when you have eliminated the impossible, whatever remains, however improbable, must be the truth?
>
> SIR ARTHUR CONAN DOYLE—*The Sign of Four*

There are two kinds of information in any data and knowledge base: *extensional* and *intensional*:

- *extensional* information designates the actual contents of a data/knowledge base (in other words, all the actual elements themselves);
- *intensional* information denotes data through descriptions (for example, types, logical statements, natural language description, *etc.*, ...).

Databases have existed for decades. Knowledge bases are just nascent. Both technologies deal with both intensional and extensional information, however in different ways. Indeed, databases have honed and optimized the relational technology to deal with large finite extensional sets; therefore, in databases: *extension = data* and *intension = schema*. On the other hand, knowledge bases focus on efficient reasoning

[6] By "supporting" we mean for which there are W3C Working Groups.

[7] See also the recent proposals to use easier-to-read formats than provided by the XML-based RDF (namely, JSON and JSON-LD).

[8] It may *occasionally* work but *always* with guaranteed considerable collateral damage! ... ☺

over generally relatively small extensional sets; therefore, in knowledge bases: *intension = formula* and *extension = model*. Interfacing both worlds has thus become a necessity and a challenge. An important difference to keep in mind is that databases deal with *finite models*, whereas knowledge bases also deal with *infinite models* (which is why database systems are *bottom-up systems* and knowledge base systems are *top-down systems*).

In essence, $\mathcal{LIFE}$ is a simple albeit powerful system that can do just that—reconcile data and knowledge base management since:

- it treats all information and knowledge as ***constraints***,
- all information and data is represented as ***graph***, and
- it is ***not limited to finite*** knowledge.

It does differ in a fundamental way from other similar approaches—such as the official OWL family of languages (based on DL)—in that $\mathcal{LIFE}$ solves constraints by top-down graph-unification while DL solves constraints by explicit bottom-up model construction (*i.e.*, the very same flaw we are decrying regarding the database approach to knowledge-base management!). The picture in Figure 2 summarizes quite accurately this author's stance on the Semantic Web issue . . . ☺[9]

Figure 2: Why do simple when we can do complicated?

An alternative approach

Again, the fundamental difference between data-based and knowledge-based information processing is that databases are finite by necessity while knowledge bases may denote all sorts of models, whether finite or infinite.

[9]This picture (without the added annotation regarding the Semantic Web) and the Henry Ford quote are borrowed from IBM Canada Ltd.'s Martin Wildberger, VP, WW Information Management Development, in his keynote address at IBM's CASCON 2009 [Wil09].

The same essential concept—namely, that of *relation*—takes on important different subtle interpretations depending on whether it is seen as a Relational Database (RDB) relation, or a First-Order Logic (FOL) relation, or a Constraint-Logic Processing (CLP) relation. To add to confusion, the word "relation" is used in all three contexts as synonymous of "predicate."[10]

So what do the three have in common and how do they differ?

Common features:

- a relation is a set of records;
- all three have intensional and extensional parts;
- their Model Theory and Proof Theory coincide.[11]

Differences:

- RDB: all relation extensions are finite; intensional information amounts to type schemata;
- FOL: relation extensions may be finite or infinite; there are two views of FOL—Model Theory, which relies on extensional information (*what is denoted* regardless of how it is computed) and Proof Theory, which uses intensional information (*how to prove* what is denoted);
- CLP: a relation is a constraint where there exists an efficient proof algorithm for the specific class of predicates it denotes. Thus, CLP is necessarily intensional if there is an algorithm for it; it is extensional only when is finite and given as a table.

... and—hey, by the way! ☺—OSF logic can efficiently and scalably solve constraints that are labeled graphs; a.k.a.: Linked Data graphs!

Recapitulation

An immense opportunity comes up with Linked Data as it is universally RDF-based and the sort of graphs specifiable in these formalisms is expressible as a particularly interesting class of constraints: namely, OSF constraints. ***This means that all data and knowledge can be treated as a constraint. In other words, there is a special efficient method on how to process a type of data when that data is treated as a constraint.*** Table 1 on Page 7 summarizes the above.

Therefore, the essence of the approach proposing to use OSF constraint solving for processing Semantic Web knowledge whenever it is appropriate, takes advantage of the following facts:

- **Linked data**—this is a system in which information is represented as sorted labeled graph—this is today a universal data model that everybody agrees on.
- **Differences between DL and OSF can come handy**—especially since:
  - DL is expansive—hence, expensive—and can only describe finitely computable sets; whereas,
  - OSF is contractive—hence, efficient—and can also describe recursively-enumerable sets.
- **Constraint Solving is appropriate for Knowledge Representation**—there are two kinds of constraints when dealing with knowledge representation:
  1. *structural type constraints:* objects, classes, and inheritance.
  2. *non-structural type constraints:* path equations, relational constraints, and type definitions.

[10] See Table 1 on Page 7 for a summary.

[11] As illustrated in Figure 3 on Page 8.

Conclusion

The W3C's Linked Data format is nothing but a data type for graphs: everything is a set of interconnected nodes (a graph is a set of RDF triples). $\mathcal{LIFE}$ enables effective computing of relations in the forms of predicates (*à la* Prolog) and functions (*à la* Rewrite Rules) over the universal data type called *OSF structures that is essentially a formalization of RDF-type graphs as constraints*. **Therefore, this is a system capable of reasoning efficiently over Linked Data.**

The Web of $\mathcal{LIFE}$ is a declarative paradigm using OSF-constraint solving for an operational semantics on RDF-like Semantic Web graphs—*viz.*, Linked Data. What is offers differs in essential ways from existing OWL-based technology using DL tableau-based model-building proofs that cannot scale up [Sri09]. By contrast, reasoning with RDF-graphs based on OSF constraint-solving is efficient and can scale up.[12] Other built-in practical advantages of OSF constraint-based reasoning is its automatic "*memoizing*" capacities whereby all proofs are remembered for free and never duplicated, all this with virtually no overhead penalty in either memory or time [AK07a].

It is not sufficient for an idea—however clever and practical—to work. It is only a requirement. The real judge is whether the sponsors of such an idea can make profit to a commensurate amount. It is this author's conviction that the line of ideas that have been discussed in this document offers such potential.[13]

References

[AK07a] Hassan Aït-Kaci. Data models as constraint systems—a key to the Semantic Web. *Constraint Processsing Letters*, 1(1):33–88, November 2007. [Available online[14]].

[AK07b] Hassan Aït-Kaci. Description Logic *vs.* Order-Sorted Feature Logic. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Proceedings of the 2007 International Workshop on Description Logics (DL2007)*, Brixen-Bressanone, Italy, June 8–10 2007. [Available online[15]].

[AK11] Hassan Aït-Kaci. A sorted-graph unification approach to the Semantic Web. Invited keynote lecture,[16] *IEEE/WIC/ACM International Conference on Web Intelligence*, Lyon, France, August 22–27, 2011. [Available online[17]].

[BN03] Franz Baader and Werner Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 2, pages 47–100. Cambridge University Press, 2003. [Available online[18]].

[HS02] Ian Horrocks and Ulrike Sattler. Description logics. Tutorial given at ECAI-2002, Lyon, France, July 23 2002. [Available online[19]].

[Sri09] Kavitha Srinivas. OWL reasoning in the real world: Searching for Godot. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Proceedings of the 22nd International Workshop on Description Logics (DL 2009)*, Oxford, United Kingdom, July 27–30 2009. CEUR Workshop Proceedings. Invited lecture [Available online[20]].

[Sta09] CACM Staff. Children's Magic Won't Deliver the Semantic Web. *Communications of the ACM*, 52(3):8–9, March 2009. [Available online[21]].

[Wil09] Martin Wildberger. Computing for a smarter planet. Keynote Speaker, Plenary Presentation, CASCON 2009, Markham, ON, Canada, November 3 2009. [Available online[22]].

[12] See http://www.youtube.com/watch?v=8uOgG6CJ8iY to understand why.
[13] http://www.hassan-ait-kaci.net/my-scientific-outlook.html
[14] http://www.cs.brown.edu/people/pvh/CPL/Papers/v1/hak.pdf
[15] http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-250/paper_2.pdf
[16] http://wi-iat-2011.org/WI_2011/keynote-speaker-hassan-ait-kaci/
[17] http://www.wi-iat-2011.org/ressources/speakers/slides/AitKaci-wiiat11-talk.pdf
[18] http://citeseer.ist.psu.edu/baader03basic.html
[19] http://www.cs.man.ac.uk/~horrocks/Slides/ecai-handout.pdf
[20] http://www.cs.ox.ac.uk/DL2009/proceedings/invited/Srinivas.pdf
[21] http://doi.acm.org/10.1145/1467247.1467250
[22] http://dl.acm.org/citation.cfm?id=1723028&CFID=71209407&CFTOKEN=34400094

<table>
<tr><th>RDB</th><th>FOL</th><th>CLP</th></tr>
<tr>
<td>Predicate as finite relation (i.e., table)

Bottom-up computation

$r: \begin{array}{|c|c|c|}\hline X_1 & \cdots & X_n \\ \hline a_{11} & \cdots & a_{1n} \\ \hline a_{21} & \cdots & a_{2n} \\ \hline \vdots & \vdots & \vdots \\ \hline \end{array}$

+ Select, Project, Join, Transitive Closure</td>
<td>Predicate as first-order formula

Top-down computation

The relation r is a set of n-tuples verifying Rule (1), which is read: "the n-tuple of terms $\langle t_1, \ldots, t_n \rangle$ belongs to the relation r if (and only if) some formula F holds."

$$r(t_1, \ldots, t_n) \ \ \mathbf{if(f)} \ \ F. \qquad (1)$$
where the t_i's denote a first-order term (e.g., Prolog) and F is a formula.</td>
<td>Predicate as constraint formula

Computation = solving algorithm

The relation r is a set of n-tuples verifying Rule (2) using a specific algorithm for φ, where φ denotes a constraint (not necessarily first-order as long as we have an efficient procedure for it). Rule (2) is read: "the n-tuple of variables $\langle X_1, \ldots, X_n \rangle$ belongs to the relation r if (and only if) some formula F holds and some constraint φ is verified."

$$\begin{array}{l} r(X_1, \ldots, X_n) \ \ \mathbf{if(f)} \ \ F \\ \mid \varphi(X_1, \ldots, X_n). \end{array} \qquad (2)$$
CLP recognizes sub-formulas that can be processed with specific efficient algorithms. For example if φ is limited to first-order term equations, Prolog's computation amounts to CLP where the constraint-solving algorithm is first-order term unification. Namely, Formula (1) for Prolog (e.g., Horn Logic), when seen as (2), becomes:

$$\begin{array}{l} r(X_1, \ldots, X_n) \ \ \mathbf{if(f)} \ \ F \\ \mid X_1 = t_1, \ldots, X_n = t_n. \end{array}$$</td>
</tr>
</table>

Table 1: How do RDB, FOL, and CLP compare?

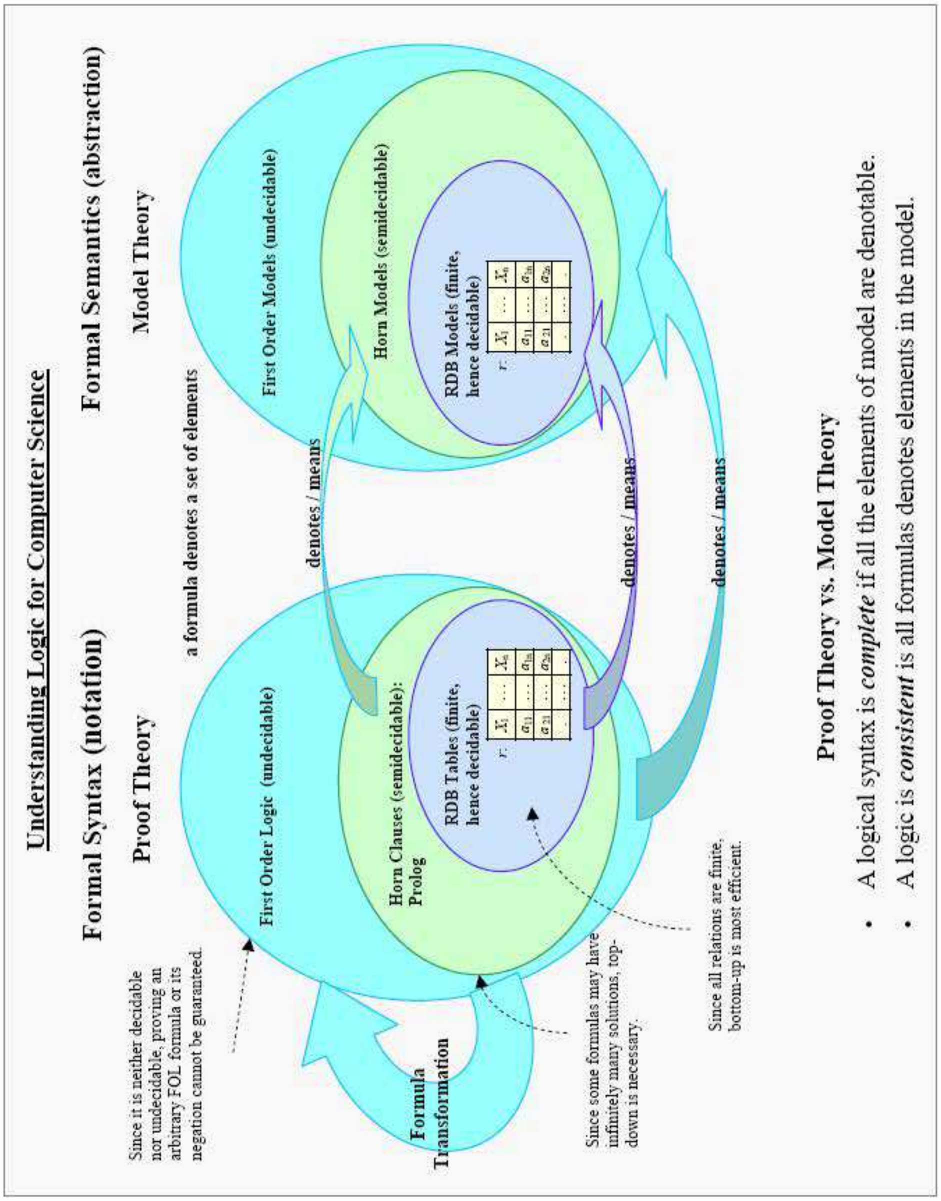

Figure 3: Understanding Logic