Litteratus Calculus: a manifesto for a demographic way to build a sustainable Semantic Web

Jean Rohmer
Centre des Nouvelles Technologies
d’Analyse d’Information (CENTAI)
Thales Communications France
jean.rohmer@fr.thalesgroup.com

ABSTRACT
Litteratus Calculus is a proposal to help Semantic Web enjoy the good properties which made the success of the original Web, i.e. natural language, navigation, simple search, freedom of formats, without sacrificing the structural and normative qualities of semantics. This poster outlines the underlying formalism of Litteratus Calculus, explains how it relates to Semantic Web standards, and describes some promising experiments. It is an invitation to a shared reflection on these points.

KEYWORDS: semantic and natural language, semantic writing, semantic web evolution

1. IN SEARCH OF ANOTHER BOOSTER FOR THE SEMANTIC WEB
In parallel with a growing interest from the scientific IT community, eagerness to see the Semantic Web (SW in short) more widely used is a constant since its inception. See for instance the search of its “killer app” [1]. This continuous expectation contrasts with the speed at which the original Web spread out. In some sense, with the Web, Sir Tim Berners-Lee gave us the solution, when for the Semantic Web he gave us the challenge. Which may reveal equally fruitful if we accept to lucidly address it.

We feel that two dangers threat the SW. The first one is to be seen as pure Software Engineering: “another way to develop applications with data models, programming languages, programmers”. The second danger is to consider that formal logic should be the preferred representation and computation mechanism for the SW. Another idea to boost the SW is to see it as a natural choice when the idea of the SW arose. However, they seem too close to machines and too far from people, since in one hand Software Engineering and Formal Logic so easily put a grasp at them, and since in another hand it is so difficult to automatically translate text to them.

Note 1. The slow growth of the SW may originate from its aim for a better understanding between machines, which got the main visibility at the detriment of more people-oriented features [2].

Note 2. As soon as initiatives like Semantic Desktops, Semantic Wikis and hopefully Litteratus Calculus will yield a new semantic compost, the above-mentioned disciplines might indeed become more instrumental

2. LITTERATUS CALCULUS
In Litteratus Calculus (LC in short) litteratus stands for people and calculus for machines. LC can be seen as a generalization of Semantic Networks (SN in short), which, as shown by [7] are one of the preferred formalisms to represent formal knowledge since the Ancient Greeks. Ubiquitous in Artificial Intelligence, they were a natural choice when the idea of the SW arose. However, they seem too close to machines and too far from people, since in one hand Software Engineering and Formal Logic so easily put a grasp at them, and since in another hand it is so difficult to automatically translate text to them.

Litteratus Calculus first step is to replace the SN notion of triple (e.g. Subject / Verb / Object) by the notion of minimal autonomous sentence.

A minimal autonomous sentence is a sentence in any human readable language which is atomic, i.e. cannot be replaced by one or more shorter sentences, and autonomous in the sense it is self-sufficient to be understood alone by some community of people.

We call such a sentence an inferon.

RDF statements are inferons, Logic Clauses are inferons, Natural Language sentences may be inferons. Examples: (the first sentence of this paragraph)

“In the expression Litteratus Calculus (LC in short) litteratus stands for people and calculus for machines.” is not an inferon; not minimal

“(LC in short)” is not an inferon; not autonomous

“In the expression Litteratus Calculus litteratus stands for people” is an inferon: minimal and autonomous

Litteratus Calculus second step is to consider, given two inferons, the set of their common words, called interlogos. Example:

Inferon 1: “In the expression Litteratus Calculus litteratus stands for people”

and Inferon 2: “Litteratus Calculus is designed for people”

have in common the interlogos “Litteratus Calculus for people”

Litteratus Calculus uses as few software engineering, formal logic and computational linguistics as possible so as to dramatically increase the number of contributors to the SW.

Finally, we call argos a set of inferons with their interlogos. An argos is a bipartite sub-graph of inferons and interlogos.

3. LITTERATUS CALCULUS: MORE CONTRIBUTORS, AND MORE RESPONSIBLE ONES
The Litteratus Calculus project can now be made more explicit:

- Let people write inferons —and only inferons—
- Build automatically the resulting argos and interlogos
- Provide tools to make navigation, computations, inferences on the argos network
- Provide tools based on analogy and emergence to assist people writing new inferons

Key points which will be illustrated in the Poster:

Everything is represented with inferons: facts, ontologies, rules, queries … there is an unique “semantic soup of inferons”

The main logic / computational operation is analogy. For instance, using a rule or query is making an analogy between the argos representing the rule or query and argos of basic facts (everybody understands analogy, which is not the case for formal logic)

Two basic operations in LC permit to compute resemblance and differences between inferons, interlogos, argos, based upon graph topology. For instance the resemblance between two interlogos is a set of argos. (Consider: “Jack and John work in companies which both have customers which receive grants from projects sponsored by European Union”)

Inferons writing should be an altruist activity: when typing a new inferon, the user can see how it relates to existing ones -through argos. He can see for instance how it closes circuits, helping to fire rules or answer queries. If the writer considers that his new inferon is not well connected, he may decide to add supplementary inferons to fill the gap (exactly as we proceed during a face-to-face conversation). This way, useful ontologies will be built in a need-driven process. LC encourages alterity (alter IT!). The more responsive is the system, the more responsible becomes the user. LC aims at replacing Software Engineering by a conscious discipline and altruism from users.

4. PRACTICAL STEPS TOWARDS LITTERATUS CALCULUS
All the expected LC properties remain valid if we restrict ourselves to simple inferons (RDF, SVO, triples …). And in fact we experienced that it is extremely fruitful to explore two tracks in parallel: one “low” track with simple Semantic Networks, one “high” track with true natural language inferons.

As reported in [6], we develop and use since 1993 a simple Semantic Networks Editor, IDELIANCE, which has been intensively used by individuals and groups to write and share knowledge (users include French Military Intelligence, L’Oreal, Air Liquide, Merck Pharmaceutical Labs, Thales). With IDELIANCE, users of various professional profiles directly create shared semantic networks, after some hours or days –age depending- of training. We call this low track Litteratus Calculus “A”, and the high track Litteratus Calculus “B”.

LC “A” is of course easier to manipulate inside a machine. This generates many ideas of algorithms like: computing all argos between two subjects, all circular argos visiting a given set of subjects, and filter argos according to the nature of subjects and verbs they are made of. We also developed mechanisms of suggestions when writing new inferons: given the current environment of a subject in the graph, users are suggested new statements by analogy with similar graph configurations.

Once proved in LC “A”, these features can then be taken as objectives to be transposed in LC “B”, no longer in terms of subjects and relations, but in the more complex, less formal lattice of inferons, interlogos and argos. These transpositions from LC “A” to LC “B” often invite us to state the problem in more general terms, leading to more general features, which, in return, give new specifications for LC “A”.

One can ask the question: why not concentrate on LC “A” and improve it? The answer refers to our demographic point: we noticed that at most 1 (one) per cent of people—among a normal business population—spontaneously adhere to LC “A”—rather than remaining in the traditional textual / document mode.

Our bet is, with LC “B”, to raise this percentage to about 10%. (it is important to realize that writing inferons is not writing plain text as usual. Even with much less constraints than with LC “A”, it harnesses people’s reflection—not a bad point in other respects)

One of the initial possible outcomes of Litteratus Calculus is to promote new ways of scientific publishing, as anticipated and proposed in [5].

5. REFERENCES
http://www.semanticdesktop.org/xwiki/bin/SemanticDesktopWS2005/
http://wiki.ontoworld.org/index.php/ABCDEF
[7] Sowa John F. Semantic Networks
http://www.jfsowa.com/pubs/semnet.htm