Towards a People's Web: Metalog

Massimo Marchiori
MIT CSAIL, The World Wide Web Consortium (W3C)
Cambridge, MA (USA)
and
Dept. of Computer Science, University of Venice (Italy)
massimo@w3.org

In memoriam: To my father Orfeo, a marvellous person, with a tragic end.

Abstract

This paper introduces Metalog, a query/logical system designed to allow reasoning on the Web. Metalog tries to start filling in the so-called people axis, *where the Web is tailored for the people, and not just for the machine. Besides allowing the formulation of declarative logical rules, Metalog's distinctive feature is to lower the entry access level, by employing a Pseudo Natural Language (PNL) interface, which is particularly easy to understand. This allows almost everybody to use Metalog, even without any particular expertize in the field. This capability, together with other advanced solutions that enhance customization and user friendliness, are key components for a wide adoption of intelligent semantic web technologies.*

1. Introduction

The need for an "intelligent Web" has been growing fast in the last years, thanks to the widespread adoption and growth of the Web: conjoint factors like more and more users, more and more demanding applications, a huge enlargement in scope, have all contributed both to the success of the current Web, and also raised the expectations from the audience, that is currently experiencing all the limitations that the original Web Architecture presents. To recover from this situation, the W3C has proposed a new Web structure where more qualified information can be put on the Web, and sophisticated reasoning can be performed: the so-called Semantic Web (cf. http://www.w3.org/2001/sw). The main idea of the Semantic Web is to provide a flexible "basic semantic language", RDF (cf. [3] and the latest W3C specifications), which makes possible to codify the basic bricks of reasoning: then, on top of this "universal semantic alphabet", more and more sophisticated technologies can be employed, so to bring the expressive power to higher levels. The overall architecture is often simplified by the well-known "Semantic Web Tower", where various technologies are stacked on top of RDF, including the upper layers that deal with reasoning and intelligent web applications.

However, what has been somehow missing so far has been technologies that follow not just the technological axis, but the *people axis*, i.e., technologies that empower the people and try to make the semantic web closest to the widest possible audience, possibly sacrificing some of its power. The people axis is of equal importance, as the ultimate goals of an Intelligent Web is both machine and users, and while the first are tackled with the technological axis (the ones present in the Semantic Web Tower, and other related tools), there is still the big need to start filling the other orthogonal axis, the people one.

This paper describes the current status of the Metalog project ([5]), the first query/logical system developed for the Semantic Web, whose main design choice is to start filling the people axis.

Metalog tries to blend two common necessities in the Semantic Web: on the one hand, the ability of *reasoning* on the Web. On the other hand, a key aspect which is often underestimated: the ability to bring these advanced technologies to the widest possible audience. This aspect is particularly crucial in this early adoption phase, where the real usefulness, and the same concepts, of Semantic Web, still have to reach sufficient critical mass among the public.

To this extent, Metalog uses a so-called Pseudo Natural Language (PNL) interface, which is much similar to natural language, and therefore allows an easy interfacing to the more complex underlying technologies of the Semantic Web. Metalog interfaces the PNL with an underlying logical extension of the RDF semantics, the "MLL" (Metalog Logical Level), which is in turn based on an extension of the RDF model, the Metalog Model Level (the "MML"). In the following, we will describe first the foundational layers on which such logical extensions are given, then describe the PNL, and finally introduce the latest developments within the Metalog project.

2. The Metalog Model Level

The Metalog Model Level (MML for short) is a natural extension of RDF with logical operators. The Metalog operators are identified with a URI reference ([1]), and lie in the Metalog namespaces, which all start with the URI “http://www.w3.org/RDF/Metalog” . For brevity, we will in the following associate the prefix ”ml” with the above URI (so, writing for example ml:foo instead of the longer http://www.w3.org/RDF/Metalog#foo). Similarly, we will indicate with ”rdf” the standard RDF namespace URI, i.e. “http://www.w3.org/1999/02/22-rdf-syntax-ns” (cf. [3,2]).

All of the new operators that the MML provides are represented in the RDF model (graph level) in an uniform way: an operator denoted by the URI α, having as operands $\beta_1, \ldots, \beta_k$, is encoded (using a “subject predicate object” notation a la n-triples) via

```
_:an <ml:operator> <α>.
_:an <rdf_1> β1.
...
_:an <rdf_k> βk.
```

The MML provides the basic logical operators, that is to say the logical “and”, “or” and “implication”. These are denoted with ml:and, ml:or and ml:imply respectively. It provides a negation (not) operator, and the classic comparison and math operators for equality, inequality, $>=$, $<=$, $+$, $-$, $*$ and $/$. It also provides an ml:name extension, which is used to name variables. Finally, it also provides two annotation extensions, ml:annotation and ml:ns, which will be explained later in the paper.

2.1. The Metalog Logic Level

The Metalog Logic Level, or MLL for short, is the logic that expresses the semantics of the MML. It is essentially a subset of (infinitary) equational First Order Logic. A full formal mapping from the MML to the MLL would be too lengthy, and unnecessary complicated (given the minor role that the MLL plays, as we will see): therefore, we just give the basic intuition of the recursive interpretation mapping, whose details should be anyway straightforward.

• The ml:and, ml:or and ml:imply and “not” operators are translated into logical conjunction, logical disjunction, logical implication and logical negation respectively.
• Ground mathematics, and equality/inequality, can be naturally mapped into this logic (this also means that the corresponding infinitary axioms for the comparison and equality predicates has to be present in the MLL).
• Each literal/URI-reference is mapped into some constant via an injective mapping.
• ml:name, ml:annotation and ml:ns are mapped into distinguished (i.e., distinct and outside of the image of the literals/URI) constants.
• Each triple (S,P,O) that neither has Metalog extensions, nor is part of an RDF container description is mapped into the predicate P(S,O).
• RDF container descriptions of type Seq (Bag, Alt resp.) containing k objects are mapped correspondingly into a distinguished k-ary operator SEQ$_k$ (BAG$_k$, ALT$_k$ resp.) applied to (the mapping of) its k operands.

Moreover, the BAG$_k$ and ALT$_k$ operators are given the following axiomatization:

B1 $\mathrm{BAG}_k(t1, \ldots, t_k) \rightarrow \mathrm{BAG}_k(t_{\pi(1)}, \ldots, t_{\pi(k)})$ for every permutation π of $[1, k]$
A1 $\mathrm{ALT}_k(t_1, \ldots, t_k) \rightarrow \mathrm{ALT}_k(t_1, t_{\pi(2)}, \ldots, t_{\pi(k)})$ for every permutation π of $[2, k]$
A2 $\mathrm{ALT}_k(t_1, \ldots, t_k) \rightarrow \mathrm{ALT}_i(t_1, \ldots, t_i)$ for every i in $[1, k]$
A3 $\mathrm{ALT}_k(t_1, \ldots, t_k) \rightarrow t_i$ for every i in $[1, k]$

So, this mapping just defines the natural interpretation of the MML (i.e., that ml:and works in fact like a conjunction, ml:or like an or, and so on). Axiom B1 formally represents the intuition that in a Bag order does not matter, while axioms A1, A2 and A3 represent the intuition that an Alt is a series of alternatives (“one of these”), and that the first alternative is distinguished.

The MLL is in a sense the “ideal” interpretation for the Metalog extensions (in fact, for any logical extension that wants to provide FOL-like capabilities, RDF containers, math). However, of course, it is of limited practical use: in order to use such a logic (for instance, to prove theorems, i.e. to answer to queries) we would have to employ powerful machinery, and worst of all, with no guarantee of good computability.

Therefore, while the MLL nicely represents the semantics of the Metalog extension, in the abstract, we will later need to downscale it, and try to find a more restricted setting with more performing computational properties.

3. The PNL

As said, Metalog’s distinguished feature is to provide a top-level interface that tries to climb up, as far as possible, the “people’s axis”, so to be intuitive enough for normal people to understand. To this extent, Metalog employs so-called “discourses”, which are Pseudo Natural Language (PNL) sentences that resemble everyday’s common writing. The best way to understand discourses is to start with a simple example, so to have an idea of how things work. One of the simplest discourses from the online Metalog sample distribution is the following:

```
comment: one of the simplest Metalog sessions.

comment: we start defining what things are.
JOHN represents the person "John_Smith" from
the company "http://www.example.com/staff".
IS represents the verb "is" from the collection
"http://www.relationships.example.org/verbs".

comment: now we say something.
JOHN IS "tall like a tower".
comment: and now we ask something.
do you know whether JOHN IS SOMETHING?
```

When this program is loaded in Metalog, it is actually colored in different ways, so to better outline the various components of the discourse. The first, second, fifth and sixth sentences would output as red, indicating that these are comments: as it can be guessed, these parts are superfluous in a Metalog discourse, they are just there for better description.

The third and fourth sentences would appear as green lines: green indicates that these are so-called "representation" parts. Representations are useful to denote shorthands: in these sentences, we associate some entity (like `JOHN`) to its corresponding concept (`the person "John_Smith" from the company "http://www.example.com/staff".`). This means in a certain sense that, anywhere, writing the whole `the person "John_Smith" from the company "http://www.example.com/staff".` is pretty much equivalent to writing its representation (the shorter `JOHN`).

The last two sentences (the sixth and the eighth) would appear on the Metalog screen in blu, signaling that they can be either "assertions" or "queries". An assertion is a sentence where we state something. For instance, the first of such blue lines in the Metalog discourse (`JOHN IS "tall like a tower".`) states precisely what it says. A query, instead, is a sentence where we are asking Metalog for answers. In fact, the second blue line is a query (`do you know whether JOHN IS SOMETHING?`). Queries can be easily distinguished because they end with a question mark ("`?`"). All the other sentences, instead, must end with a dot ("`.`"), as it can be seen from this example.

Representations, assertions, and queries are the three types of sentences in any Metalog discourse. We will explain the first two now, and the query part subsequently.

3.1. Representations

As we have seen in the first example, representations are a useful mean to associate some word (like JOHN) to its corresponding representation (the person "John_Smith" from the company "http://www.example.com/staff"). This means that whenever, in the following, the word JOHN is used, it is precisely like if we had written its whole representation instead. So, representations are a helpful way to make a Metalog discourses much more readable. Their usage is somehow necessary when working in a Semantic Web environment, because of the fact that objects are usually represented using URIs (or, more generally, URI references, cf. [1]). This means that sentences can rapidly become unreadable to people, without appropriate countermeasures.

The keyword "represents" is used to denote a representation. Before it, there is the word whose meaning is stated, and after that follows the associated meaning. Like every sentence, also a representation sentence is terminated with a dot at the end.

Not every word can be associated with a representation. Only words that are in *upper case* can. This is the reason why our first example had JOHN and not instead John or john. Words that are all written in upper case are also called *variables*, which means that they do not have a meaning per se, but instead their meaning have to be explicitly specified. In the above case, the meaning of JOHN has been assigned to the person "John_Smith" from the company "http://www.example.com/staff".

3.2. Names

Names (literals/strings) can be written in Metalog by simply enclosing them within double quotes, which is the rather common and expected choice. However, it is a common prerogative of the Semantic Web to use URI references (in particular, in the RDF Data Model); and, the large collections of URI references that are used mostly have the same few URI components, just to indicate that they belong to some common category. For instance, in XML Schema datatypes all what changes is the fragment identifier part, while the basic URI stays the same. This implies that it might be useful to separate the URI information from the fragment identifier, to get a better readability. The Metalog PNL allows such a facility, via the "from" keyword. URI references can be built by first writing the fragment identifier part, then the "from" keyword, and then the URI component; this way, we state more clearly the intuitive notion that URI references are a kind of "qualified names", names with a context. And, we pave the way for easier reusability/grouping of multiple names within the same context.

Examples of such names in context were present in the above Metalog discourse, where certainly writing `JOHN represents the person "John_Smith" from the company "http://www.example.com/staff".` sounds a bit more natural and less "computer-like" than writing the equivalent `JOHN represents "http://www.example.com/staff#John_Smith".`

3.3. Assertions

Assertions are the main sentences in a Metalog discourse, the ones where we state something.

The basic brick of these sentences follows the very simple pattern "subject predicate object". In the previous example, the assertion we had was: `JOHN IS "tall like a tower".` This indicates that JOHN is the subject of the sentence, IS is the verb, and "tall like a tower" is the object of the sentence. Note that, like said earlier, a dot ends the sentence.

Assertions can be much more sophisticated than that. For example, "and" and "or" can be used with the usual meaning of conjunction and disjunction. For example, we could write a sentence like

```
JOHN and MARY ARE "tall".
```

Metalog assertions can also express deduction rules. For example, we could write

```
if SOMEONE is "tall" then that SOMEONE
could TRY VOLLEYBALL.
```

3.4. How things work

As said, the PNL layer sits above the Metalog logic layer. The way to pass from one to another is via a careful parsing that goes thru the Metalog dialogue and extract the corresponding logical meaning.

Metalog uses a set of *reserved keyword*, whose position in the sentence determines their meaning (so, the Metalog grammar is not context-free). Each Metalog sentence is tokenized by using the reserved keyword, the names therein, and the variables (everything else is discarded). Then, the parsing proceeds to the translation phase.

The way this is done is for the Metalog assertions is, technically speaking, thru a 3-states left to right parsing (with forward lookahead). The three levels of parsing roughly correspond to the subject (level 0), predicate (level 1) and object (level 2) states. If a name or a variable is found at the current parsing point of a sentence, then the current state is filled in with that value, the parsing point advances within the sentence to the next token, and the state is increased by one modulo 3 (so, it cycles within the three states).

The three main classes of reserved keywords are those corresponding to the Metalog logic conjunction, disjunction and implication. These are, respectively, "`and`" and "`or`" for the first two, and "`then`", "`imply`" and "`implies`" for the third. Then, each time one of these keyword is found, Metalog tries to interpret them following the basic English grammar rules.

In some cases, this is easy: for instance, the interpretation of the "then" (and its other two synonymous) is always the same: the translation of everything to its left implies (at the Metalog logic level) the translation of everything to its right. So, in this case, the state is only needed to determine whether we have an error or not (for instance, it is obvious that a "then" cannot occur in state 1, where a predicate is expected).

However, in other cases things are not so straightforward, like in the case of the "and" keyword. Indeed, such keyword does not have a unique meaning, as in English it could either stand for a logical conjunction, but also as the connector for, say, a sequence of objects. For example, consider the sentence
`RALPH and JOHN LIKE MARY and LUCY LIKE TOM.`
It is rather clear that the first "and" has to be interpreted as a connector (as it occurs at state 2, it cannot be otherwise). But, the second "and" can't be disambiguated just looking at its state (0), as two interpretations are in principle equally possible: either "and" as a logical conjunction (i.e., a new sentence is starting afterwards), or "and" as a sequence connector (i.e., a sequence where MARY is the first element, and the second is coming). In order to disambiguate, the parsing needs to lookahead: doing so, we discover that the latter option (((RALPH and JOHN) LIKE (MARY and LUCY)) LIKE TOM.) does not make sense. Therefore, the parser assigns to this "and" the logical conjunction meaning.

Note that, within a discourse, a sequence with the "and" connector is naturally interpreted in Metalog logic level as an RDF Bag container (i.e., order doesn't matter), while a sequence with the "or" connector is interpreted as an RDF Alt container. Metalog also has the expressive power to encode the other RDF container, the "Seq" (ordered sequence), making it a a subcase of the Bag one: if the sequence is followed by the keyword "order", then instead of a Bag we obtain a Seq (again, note that this requires lookahead). So for instance, one could write `RALPH and JOHN in this order` to indicate that order is significant.

A full description of all the Metalog keywords is not within the scope of this paper, but en passant we can mention that Metalog has also support for basic math (via the keywords "add", "plus", "sub", "minus", "times", "mult", "div", "divided", with the obvious meaning), and for comparisons like strict comparison ("greater", "less") and equality ("different", "equal", "equals"). Also, strict comparison and equality can be composed to form things like "greater or equal", that have the obvious meaning (note that, also in this case, we need lookahead).

Finally, we have said above that Metalog supports variables (which are either substituted with the corresponding representation or, if no representation is present, are mapped into logical variables in the Metalog logical level). Because of the philosophy of the PNL, this should have been done in such a way to preserve, as far as possible, the natural (English-like) appearance of the sentence. For this reason, the design choice, as partly hinted at when talking about Representations, has been to represent all variables *in upper case*. Therefore, no "special symbols" are introduced in Metalog discourses. All the keywords, as the reader might have noted, are in lower case; moreover, no mixed case words are allowed (so, writing for example "Somebody" is not allowed): this minimizes the risk of confusion between a variable and something else.

3.5. Ambiguities

The field of natural language processing (NLP) is well known for the extreme challenges that it poses: in particular, the biggest problem is the ambiguity one: in other words, many possible meanings could be possible, and to find out which one to choose might be very hard, or even impossible. The philosophy of the PNL approach is somehow middle-way between the benefits of full natural language, and the benefits of precise computation. So, the PNL is not a normal "Natural Language" interface. Pseudo here means that the natural language that is used here is not the whole natural language that we all use, but a subset of it (roughly, it corresponds to those very basic grammar rules we were taught in our first years at school).

More precisely, the current PNL has been designed to be an "*r*-language", which means, it is very easy to *r*ead (and hence, to understand). In fact, if used properly, the PNL makes trivial for people to understand concepts and sentences in the Semantic Web.

On the other hand, if you are a developer or want to write information using the PNL, you can't just write everything: as said, the current PNL limits the language to a very simple form; this implies some amount of adaptation is still needed in the writing phase, although it is not that much (essentially, just forget all the involved grammar you learnt in high school and think you are a kid...).

This little price to pay has an important consequence: the PNL is an *unambiguous* language. In other words, there are no ambiguities in what you can write, every sentence has a precise and determined meaning, not subject to interpretation and doubts. So, the current PNL sacrifices the total freedom of natural language, where ambiguities and interpretation problems exist, to give a more restricted, but totally safe environment: *a discourse written in the PNL has the same meaning for every Metalog processor.*

It should be noted that the fact the PNL is unambiguous does not mean that the subset of natural language it deals with is completely unambiguous too: this would have limited too much the expressive power of the PNL. It simply means that, in the (relatively few) cases of ambiguity, the PNL assigns a well-determined meaning. In particular, as the PNL parser works left to right, every ambiguity is resolved with the simple rule to always keep the left sentence as big as possible. For instance, consider the following sentence, where we have augmented the previous example:
```
RALPH and JOHN LIKE MARY and LUCY and
ANDREA LIKE TOM.
```
This is ambiguous in natural language because it could well either mean (((RALPH and JOHN) LIKE (MARY and LUCY)) and

(ANDREA LIKE TOM).), or also (((RALPH and JOHN) LIKE MARY) and ((LUCY and ANDREA) LIKE TOM).) . But as said, as parsing occurs left to right this is always interpreted by Metalog with the first option: (((RALPH and JOHN) LIKE (MARY and LUCY)) and (ANDREA LIKE TOM).) .

In natural language, the easiest way to disambiguate a sentence like the one above would have been for instance to use a comma. And indeed, Metalog uses the same way: the comma (`,`) can be used, whenever needed, to disambiguate. So for instance, writing `RALPH and JOHN LIKE MARY, and LUCY and ANDREA LIKE TOM.` would have made Metalog interpret the sentence with the second possibility: (((RALPH and JOHN) LIKE MARY) and ((LUCY and ANDREA) LIKE TOM).) .

4. Computable Metalog

The PNL interpretation within the MML should be just plain obvious, as described in the parsing process. From there, we know the meaning, passing on to the MLL. However, as said when introducing the MLL, such a logic is of little practical use, and we need to find a subset with nicer computational properties that can be successfully employed. One such subset of choice is Logic Programming (Horn logic with the negation as failure), that can be run using Prolog (even if with some minor differences like literal and clause ordering). Therefore, the current implementation of Metalog interfaces to a Prolog system (to be precise, with SWI-Prolog, [4]), using it as inferential engine. The mapping rules for the logical part are rather straightforward, and are expressed in Table 1. Note that rules 1) and 2) are different because rule 1) only applies when we are translating a sentence, while rule 2 applies when we are already within it; the other rules apply with no restrictions. Containers are more of a problem, as already introducing rule B1 for the Bag would break the termination property in Prolog (i.e., we will risk to get in a lot of cases nonterminating programs). A good compromise solution is to simulate them using the explicit replacements sketched in Table 2. Note the choice regarding Alt, whose axiomatization has been reduced in order to be more effective (loss of power would occur in extremely limited cases here).

$\|\Theta_1 \wedge \Theta_2 \wedge ... \wedge \Theta_n\|$	$\Longrightarrow$	$\|\Theta_1\|$ $\|\Theta_2\|$ ... $\|\Theta_n\|$	1
$\|\Delta_1 \wedge \Delta_2 \wedge ... \wedge \Delta_n\|$	$\Longrightarrow$	$\|\Delta_1\|\ ,\ \|\Delta_2\|,\ ...\ ,\ \|\Delta_n\|$	2
$\|\Theta_1 \vee \Theta_2 \vee ... \vee \Theta_n\|$	$\Longrightarrow$	$\|\Theta_1\|\ ;\ \|\Theta_2\|\ ;\ ...\ ;\ \|\Theta_n\|$	3
$\|\Theta_1 \rightarrow \Theta_2\|$	$\Longrightarrow$	$\|\Theta_2\|\ :-\ \|\Theta_1\|$	4
$\|(\Theta_1 \vee \Theta_2) \rightarrow \Theta_3)\|$	$\Longrightarrow$	$\|\Theta_3\|\ :-\ \|\Theta_1\|$ $\|\Theta_3\|\ :-\ \|\Theta_2\|$	5
$\|\Theta_1 \rightarrow (\Theta_2 \wedge \Theta_3)\|$	$\Longrightarrow$	$\|\Theta_2\|\ :-\ \|\Theta_1\|$ $\|\Theta_3\|\ :-\ \|\Theta_1\|$	6

Table 1. (Logical) mapping of MLL to Prolog.

$\|Seq_n(\Gamma_1,\ \Gamma_2,\ ...\ ,\ \Gamma_n)\|$ Metalog Sequence Container	$\Longrightarrow$	$[\ \|\Gamma_1\|,\ \|\Gamma_2\|,\ ...\ ,\ \|\Gamma_n\|\ ]$ Prolog List
$\|Bag_n(\Gamma_1,\ \Gamma_2,\ ...\ ,\ \Gamma_n)\|$ Metalog Bag Container	$\Longrightarrow$ $\Longrightarrow$ $\dots$ $\Longrightarrow$	$[\ \|\Gamma_1\|,\ \|\Gamma_2\|,\ ...\ ,\ \|\Gamma_n\|\ ]$ $[\ \|\Gamma_2\|,\ \|\Gamma_1\|,\ ...\ ,\ \|\Gamma_n\|\ ]$ $\dots$ $[\ \|\Gamma_{\pi 1}\|,\ \|\Gamma_{\pi 2}\|,\ ...\ ,\ \|\Gamma_{\pi n}\|\ ]$ Explosion in n! distinct assertions (permuting elements of the Bag)
$\|Alt_n(\Gamma_1,\ \Gamma_2,\ ...\ ,\ \Gamma_n)\|$ Metalog Alt Container	$\Longrightarrow$ $\Longrightarrow$ $\dots$ $\Longrightarrow$	$\|\Gamma_1\|$ $\|\Gamma_2\|$ $\dots$ $\|\Gamma_n\|$ Explosion in n distinct assertions (selecting singletons of the Alt)

Table 2. Mapping MLL containers into Prolog

5. Where's the Query?

We have seen how discourses are written in Metalog, but we haven't so far talked about a corresponding "query language" for it. The point is, when talking about Semantic Web and intelligent web applications, it can be confusing to write about the "query layer" alone, because it is so tightly linked with the reasoning/logic layer that they should in fact be part of the same realm.

What is the point? In the upper layers of the Semantic Web, there is simply more and more semantics that enters the game. A meaningful query language, in general, is just supposed to ask a specific query with respect to some knowledge (produced by the semantics). Because, the meaning is given by the semantics, and we want to query the meaning. If we don't, then this "query language" is not a Semantic Web one, it's just a query language that is querying something different (like, say, low-level syntax structures). And as such, it ought to be properly classified as part of the lower infrastructure that makes the system work, not of the Semantic Web upper layers.

So, simple enough, if the query language is part of the Semantic Web, its questions should be semantical.

Now, suppose a system has a semantics defined by an entailment relationship $\vdash$: that is to say, "A $\vdash$ B" means that from A, it follows B. Then, a (semantical) query language should be able to ask questions about the truth of entailments, like "is it true that A $\vdash$ B ?". However, such kind of queries would be a problem for Metalog, because one should have a way to express the $\vdash$ relationship in a natural English way.

There is however a way out, that Metalog employs. A simpler class of possible queries is to ask whether something is true or not. This can be written as "$\vdash$ A", which is an abbreviation for "True $\vdash$ A": that is to say, roughly speaking, from the Truth it follows A. This is a very handy and effective way to ask questions. In natural language, it corresponds to the "Is it true that...?".

We can then employ the following classic relationship, which holds true in our case: $A \vdash B \Leftrightarrow \vdash if\ A$ then B. This means that we can get rid of the entailment relationship "$\vdash$": any question of the form "Is it true that A $\vdash$ B ?" is equivalent by the above to the "Is it true that $\vdash$ if A then B ?" So, in general, we don't need to carry around the $\vdash$ to express a query: we can just ask "if A then B ?". Metalog uses this strategy to express queries: a query is just a sentence, which ends with a question mark instead than a dot. So,

for instance, the query that we have seen within the first Metalog discourse at the beginning of the paper was:
`do you know whether JOHN IS SOMETHING?`

which asks the system about the truth of the assertion. So, Metalog does not clutter the interface with an explicit reference to the particular inference mechanism ($\vdash$), and keeps the English-like flavor of the interface at a natural level.

Moreover, *there is no need to learn a separate, query language*, different from the PNL: the principle is *one language, one query*.

5.1. Feedback

If they were just to return truth, Metalog queries would be of poor help. In fact, one of the biggest challenges faced in the project has been to go beyond, and return much more meaningful answers. This is obtained by a process of instantiations of the answers (where free variables are bound to results, much like Prolog), and, above all, via a return process called *feedback*. Feedback allows to re-compute back an answer, and try to model it in the form of a natural language reply. To this extent, feedback utilizes special annotations, which codify at the RDF level the information expressed by the representations, and use them back after a reply is given by the inference engine, so to build up a meaningful answer. For example, the above query would return that
`JOHN IS "tall like tower".`

together with the opportune (*minimal*) set of representations needed to understand the answer. This method elegantly ensures the crucial property, in a Web environment, that the representation information is not lost during the contact with an inference engine (whatever it is), and that it can so be faithfully transported within the Semantic Web, so any other application can recreate the user-friendly information needed to help the user.

6. More towards the People

While, until now, we have described Metalog up to the v2.0, which is publicly available (cf. [5]) , in the following we describe some of the features that are under investigation for its future versions. All the features are aimed to strengthen even more the crucial point behind Metalog: to provide an even more satisfactorily experience for *the people* using the system.

6.1. Smart Queries

The fact Metalog has a query language which is the same as its regular expression language is extremely helpful for users (as seen, the "one language , one query"). However, one might need to extend the query language, so to make a richer variety of queries possible.

For example, consider the query that we have seen in the first Metalog discourse of this paper: `do you know whether JOHN IS SOMETHING?`. Another reasonable way to express this query, using "natural language", would be for example `IS JOHN SOMETHING?`. So, in some cases it would make sense to accept more queries, in addition to the ones obtained by the normal sentences of Metalog. We call this ability *Smart Querying*: a Smart Query is a query, possibly not in of the form of a normal assertion, that is mapped into an assertion via an opportune normalization. Smart Querying can be implemented in the following way: usually, interpretation of a sentence is *local* w.r.t. assertions, i.e., only the representations are taken into account to build up the context, and not the other assertions. With Smart Querying activated, the assertions in the current discourse are instead considered, forming up (with assertions) the current *context* of the query. Within this context, URI-references are then categorized as subject, verbs or objects, according to their use in the discourse. This then allows for possible tranformations/interpretations of a Smart Query, based on the additional information given by the context. For instance, when a Smart Query occurs with a "verb" (according to the context) appearing at state 0, followed by a "subject" at state 1, they are reversed. Therefore, a query like the aforementioned one, `IS JOHN SOMETHING?`, would have IS interpreted as a verb by the context, and JOHN interpreted as a subject: just like in natural language, this is interpreted as a sign that an interrogative verbal form is likely in use, and as such this Smart Query is normalized into the query `JOHN IS SOMETHING?`. Other possible Smart Queries are for instance queries that have some components missing: for instance, a smart query like `what about JOHN?` is translated into the query `JOHN VERB OBJECT?`, that asks for what things John can do.

This way, the system is getting closer and closer to the final user who is querying the system, by providing a much bigger flexibility, and trying to mimic even more natural language behavior. All this, staying within the deterministic nature of the interface (so the query has always a well-defined, unambiguous meaning).

6.2. Meta-Metalog

The current PNL interface is very easy to use, and serves rather well its needs: to provide an easy introductory access to the Semantic Web ideas and technologies to the people. However, this same simplicity has its own limitations. For instance, nowadays, the OWL standard for Web Ontologies has reached its final stage. This means that it would be very nice to extend the PNL with more keywords so to be able to grasp some of the more fundamental OWL constructs. Another example is the recent restructuring of the RDF data model (cf. [2]), which also approached completion: data types have been introduced, and it would be nice to have a simplified access to them using the PNL. Yet another example is the problem of *multi-linguism*, i.e. the possible use of another spoken language for the PNL, like for instance French or Spanish or Chinese: in this case, the simplified grammar rules might be (even substantially) different, and in any case the keywords should be changed.

Now, of course, extensions to RDF can, in principle, be expressed by using the current PNL, e.g. by enumerating the triples. However, usage of a PNL is supposed to make easy, and *natural*, the access to this information structuring. The moment we are forced to go down-level, the utility of using a PNL, rather than another technical language (like N-triples, or RDF/XML) becomes weaker and weaker. Instead, a good PNL should be able to take this more advanced concepts, and provide a natural formulation for people to use.

This problem, that can be in general seen as the problem of *evolution* of a PNL.

Also, when we are dealing with multi-linguism, for example, we are in presence not just of evolutions (in the sense of additional features) of the system, but of possible *alterations* of the grammar/keywords: this would seem to imply we have to rebuild a PNL from scratch, according to its language locale.

The above issues, which pertain in the general realm of PNL handling, can be tackled in the following way: rather than having each time to rebuild a different/more evolved PNL, one could extend Metalog with a new level, Meta-Metalog. Meta-Metalog is a "meta" level, that can in a sense reason on Metalog (in our case, on Metalog's PNL). In order to do so, there is the need for an appropriate conceptualization of an "abstract PNL" in the Semantic Web (so, at RDF/MML level). This can be introduced under a new namespace (http://www.w3.org/2003/m2), "m2" for short. Among the "meta" concepts introduced, are the concept of keyword (via m2:keyword), of arguments to a predicate (m2:arguments), and a high-level representation of the logic operators in the MLL (m2:and, m2:or, m2:imply). This allows to write Meta-Metalog discourses where the abstract "meta" representation of a discourse is interpreted according to some rules defining a PNL.

So, for instance, the PNL itself could be described entirely within Meta-Metalog. For example, the definition of the "imply" keyword and its rules could be given by the following snippet:

```
METAMETALOG represents "http://www.w3.org/2003/m2".
KEYWORD represents "keyword" from METAMETALOG.
ARGUMENTS represents "arguments" from METAMETALOG.
   ...
LOGICAL-IMPLY represents "imply" from METAMETALOG.
   ...
IMPLY represents "http://www.w3.org/RDF/Metalog#Implies".
IMPLY is a KEYWORD with arity 2.
   ...
if IMPLY has as ARGUMENTS X and Y in this order,
then X LOGICAL-IMPLY Y.
```

Note that obviously a change in the processing model is needed for Meta-Metalog discourses (and so, meta-PNL execution). Parsing is much more sophisticated and happens dynamically, as the PNL is generated on the fly. In the first parsing phase, keywords are extracted by the m2:keyword's. This generates a tree-like RDF structure, that is subsequently interpreted by the Meta-Metalog discourse, and put in logical relationship to its MLL structure.

The big advantages of a "meta" solution like Meta-Metalog is that we don't need any more to build ad-hoc PNLs for every different need of evolution/alteration, as we are using the same basic PNL as a unified meta-language to reason about the interpretation of the data. This meta-modelling allows "programming" of a PNL, always staying within the Semantic Web world. This approach is so powerful that it can also deal with Smart Queries. For instance, remember the example we have seen previously about the smart query `IS JOHN SOMETHING?`. Meta-Metalog could also provide the conceptualization of a syntactic query operator (m2:qop) and of a logical query (m2:query), together with the concepts of subject, predicate and verb within a sentence. Then, for example, the corresponding smart-query rule (inversion of subject and predicate) could be expressed in Meta-Metalog with the following additional snippet:

```
VERB represents "http://www.w3.org/2003/m2#verb".
QUERYOP represents "http://www.w3.org/2003/m2#qop".
QUERY represents "http://www.w3.org/2003/m2#query".
if a QUERYOP has ARGUMENTS X and Y and Z in this
order, and Y is a VERB, then there is a QUERY
with ARGUMENTS Y and X and Z in this order.
```

So, with the appropriate modelization (i.e., bringing the topic of interest within the Semantic Web world, like in this case for queries), we manage to reuse Meta-Metalog to provide Smart Queries capabilities; again, without having to build an ad-hoc parser for each type of Smart Query we might want to incorporate into the system.

7. Conclusions

The Semantic Web has a tremendous potential to enhance the future Web, and to bring classification and reasoning as first class objects in the Web. However, it is still lacking in the *people axis*, where intelligent technologies are specifically tailored for the people, rather than for the machines. Metalog can be seen as a first attempt to showcase such promises, in a way that is natural and easy-friendly to as many people as possible, via the introduction of the PNL interface, on top of a clean logical extension of RDF. The big promise is to bring this capabilities even further, as the Semantic Web tower grows and grows (datatypes in RDF, OWL, various needs like smart queries and multilinguism). To this extent, the introduction of meta-features might prove to be the right way to fulfill this promise, and to pave the way for a variety of even more user-friendly entry level tools (e.g. using voice interfaces, or fuzzy natural language processing). Ultimately, the aim is to get into the user's needs in a seamless way: much like, currently, the "e-" prefix is synonymous with electronic interfaces that empower the generic user (e-mail, e-commerce, e-learning, etc.), we envision a future where semantic web technologies will play a similar seamless role, and maybe the introduction of a *new "s-" prefix* will be in order, together with the birth of new distinguished paradigms (like s-search, s-query and so on). Meanwhile, in this initial period, user-friendly tools that start to hide the complexity, like ones using PNL solutions, are especially needed to start filling in the people axis, in this crucial phase where the Semantic Web needs to gain critical mass to the wider audience of the big WWW community. Because, after all, *it's the people who matter*.

8. Acknowledgments

Thanks to Antonio Epifani, Samuele Trevisan, and Janne Saarela for their contribution to the Metalog project, and to Tim Berners-Lee, Maria Claudia Buzzi, Elena Marchiori, Jan Rutten and Franca Tressoldi for their continuous support. And, thanks to you, dad, you are always with me, deep inside.

Bibliography

[1] T.Berners-Lee, R.Fielding and L.Masinter, Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396, 1998.
[2] G.Klyne and J.J.Carroll (Eds.), Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C. 2004.
[3] O.Lassila and R.Swick (Eds.), Resource Description Framework (RDF) Model and Syntax Specification. W3C, 1999.
[4] J.Wielemaker, SWI-Prolog 5.2.9 Ref. Manual. SWI, 2003.
[5] The Metalog Project, http://www.w3.org/RDF/Metalog/. W3C, 1998-2004.