UNIVERSITE DE TECHNOLOGIE DE COMPIÈGNE
Département de Génie Informatique

# MOSS 7 - Paths to Queries

Jean-Paul Barthès

BP 349 COMPIÈGNE
Tel +33 3 44 23 44 66
Fax +33 3 44 23 44 77

Email: barthes@utc.fr

N228
July 2008

## Warning

This document describes how to build a formal MOSS query from a list of words.

The current research version of MOSS 7 runs in a MacIntosh Common Lisp environment (MCL 5.2 for OSX). It has been ported to Allegro Common Lisp (ACL 6.1 and 8.1 running under Windows XP).

## Keywords

Knowledge representation, MOSS, query, natural language.

## Revisions

| **Version** | **Date** | **Author** | **Remarks** |
|---|---|---|---|
| 1.0 | Jul 08 | Barthès | First Issue for MOSS v7 |

## MOSS documents

Related documents

- UTC/GI/DI/N196L - PDM4
- UTC/GI/DI/N206L - MOSS 7 : User Interface (Macintosh)
- UTC/GI/DI/N218L - MOSS 7 : User Interface (Windows)
- UTC/GI/DI/N219L - MOSS 7 : Primer
- UTC/GI/DI/N220L - MOSS 7 : Syntax
- UTC/GI/DI/N221L - MOSS 7 : Advanced Programming
- UTC/GI/DI/N222L - MOSS 7 : Query System
- UTC/GI/DI/N223L - MOSS 7 : Kernel Methods
- UTC/GI/DI/N224L - MOSS 7 : Low Level Functions
- UTC/GI/DI/N225L - MOSS 7 : Dialogs
- UTC/GI/DI/N228L - MOSS 7 : Paths to Queries

Readers unfamiliar with MOSS should read first N196 and N219 (Primer), then N220 (Syntax), N218 (User Interface). N223 (Kernel) gives a list of available methods of general use when programming. N222 (Query) presents the query system and gives its syntax. For advanced programming N224 (Low level Functions) describes some useful MOSS functions. N209 (Dialogs) describes the natural language dialog mechanism.

## Contents

# 1 Introduction

When using an ontology and/or a knowledge base, most of the reasoning is done through querying. When the user is interacting by means of natural language, one has to build the queries from the user's input. The latter may be noisy when coming from a speech analysis device. The approach developed in this document is to use the structure of the knowledge base for building a formal query from the user's input, i.e. from a list of words.

To do so we use the specific structure of the MOSS representation, and in particular the entry points.

### Examples

- From the input:

  ```
  "personne" "président" "utc"
  ```

  We produce:

  ```
  (($E-PERSON
    ($S-TEACHING-ORGANIZATION-PRESIDENT.OF (> 0)
     ($E-TEACHING-ORGANIZATION ($T-ORGANIZATION-ACRONYM :IS "Utc")))))
  ```

- From the input:

  ```
  "utc"
  ```

  We produce:

  ```
  (($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc")))
  ```

### Hypotheses

Very little is assumed, however

- we assume that the target concept is the first one in the sentence;
- the series of words in input are extracted from the original input and correspond to terms designing concepts, properties or entry points. Empty words or words not understood are discarded.

# 2 Algorithm

## 2.1 Global Approach

The main idea of the algorithm is to mark the concepts, properties and entry points in the graph representing the knowledge base and to produce a series of paths that includes the marked items in the order they have been given. The approach is overly simple but nevertheless efficient in many cases.

### 2.2 Algorithm

We proceed in several steps:

- First the words are transformed into class-ids, relation-ids, or entry-points. Thus, we obtain an ordered list of places (nodes, relations) that the path must traverse to obtain the formal query.

  Entry points have a different role: they contain a terminal property an operator, and the class to which it may apply. If the class is missing from the original marked items, then it can be recovered from the entry point frame.

- Second, we solve special cases directly. E.g. when there is a single item, or a single entry point frame...

- The main algorithm is to build paths starting from a specific class, the target, that will go through all the items of the ordered list. At each step, we check if we came to the next item of the ordered list in one of the paths. If so, we throw away all the other possibilities, betting on the shortest path heuristic. We then proceed from there. If we did not come to the next item, then a number of path are generated using the relations and inverse relations attached to the class.

To avoid fanning, we limit the possible length of a path, quitting when it is reached.

### 2.3 Processing the Initial Input

The input sentence (list of words) is processed in several steps.

#### 2.3.1 Screening Relevant Terms

The initial input consists of a list of words (strings) from which punctuation marks have been removed, e.g.

```
("quel" "est" "le" "numéro" "de" "téléphone" "du"  "président" "de" "l" "utc")
```

A first function, find-best-entries, takes the list as an input and outputs two lists;

- a list of entry points corresponding to ontology or knowledge base items
- a list of unrecognized words

On the previous example;

```
(HAS-NUMERO TELEPHONE HAS-PRESIDENT UTC)
("quel" "est" "le" "de" "du" "de" "l")
```

#### 2.3.2 Building an Initial List

From the previous list of relevant entry points we replace with internal ids for MOSS objects and class frames for knowledge base entry points. This is done by the function path-process-word-list. The function returns two lists:

- the translation;
- a list of unused words or words not understood.

A class frame is a list containing a concept id and an attribute clause, e.g.

```
($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc"))
```

The expression is built from the entry point object. In the example: ”Utc”

**Examples**

```
? (moss::path-process-word-list
   '("quelle" "est" "l""adresse" "privée" "du" "président" "de" "l" "utc"))
($E-HOME-ADDRESS $S-PRESIDENT ($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc")))
("quelle" "est" "l" "du" "de" "l")

? (moss::path-process-word-list
   '("quel" "est" "le" "numéro" "de" "téléphone" "du" "président" "de" "l" "utc"))
($E-PHONE $S-PRESIDENT ($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc")))
("quel" "est" "le" "de" "du" "de" "l")

? (moss::path-process-word-list
   '("quel" "est" "le" "numéro" "de" "portable" "du" "président" "de" "l" "utc"))
($E-CELL-PHONE $S-PRESIDENT ($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc")))
("quel" "est" "le" "de" "du" "de" "l")
```

## 2.4 Special Cases

Next step is to process special cases. We have several special cases:

- input is a single term corresponding to a class, we return it as a list;
- input is a single entry point, we return it as a list;
- input is a single object relation, we do not process it yet.

## 2.5 Building the Path

In the general case we do the following:

1. We first remove from the list all leading ids that do not correspond to a concept in order to determine a target concept (it may be a class frame). If the remaining list contains a single element, we are done.
2. We fuse adjacent class frame corresponding to the same class. If the remaining list contains a single element, we are done.
3. We navigate from the first concept until we reach the next or until the path becomes too long. To do so, we fan out from the end of the current paths through the neighbors of the current class, actually building a spanning tree. We repeat the process until we hit a relation with the same name as the target relation, or a class that contains the target class. In that case we only keep the branches that end up on the target item, the rationale being that we favor shortest paths.
4. Then we build a set of formal queries. For simple question, the resulting list contains usually a single query.

## 2.6 Results

Some results are shown in the following examples.

**Examples**

```
? (moss::path-build-query-list '( "utc"))
(($E-UNIVERSITY ($T-ORGANIZATION-ACRONYM :IS "Utc")))

? (moss::path-build-query-list '( "personne"))
(($E-PERSON))

? (moss::path-build-query-list '( #+MCL "téléphone" #-MCL "tÈlÈphone" "utc"))
(($E-PHONE
  ($S-ORGANIZATION-PHONE.OF (> 0)
       ($E-ORGANIZATION ($T-ORGANIZATION-ACRONYM :IS "Utc")))))

? (moss::path-build-query-list '("personne" "président" "utc"))
(($E-PERSON
  ($S-TEACHING-ORGANIZATION-PRESIDENT.OF (> 0)
   ($E-TEACHING-ORGANIZATION ($T-ORGANIZATION-ACRONYM :IS "Utc")))))

? (moss::path-build-query-list '( "téléphone"  "président"  "utc"))
(($E-PHONE
  ($S-PERSON-PHONE.OF (> 0)
   ($E-PERSON
    ($S-TEACHING-ORGANIZATION-PRESIDENT.OF (> 0)
      ($E-TEACHING-ORGANIZATION ($T-ORGANIZATION-ACRONYM :IS "Utc"))))))
```

The resulting list of queries can then be used by the calling program to access entities corresponding to the queries.

## 3 Conclusion

This simple set of functions is useful in the case of an agent receiving a natural language input. It can use the them to build formal queries to access its own knowledge base, trying to make sense of the input.

Of course there are limitations to the approach, namely:

- when the input is so noisy that terms are missing or not understood (which happens with vocal inputs);
- when the path contains the same class to be traversed several times;
- when some of the crucial data does not have corresponding entry points (e.g. an unknown family name)
- when the path is too long or too complex.

However, in general as a first attempt, the functions presented here are quite useful.