Deliverable D6.1

Development of the Search, Inference and Ranking Modules

Project:	Mineral Intelligence Capacity Analysis
Acronym:	MICA
Grant Agreement:	689468
Funding Scheme:	Horizon 2020
Webpage:	www.mica-project.eu
Work Package:	Work Package 6
Work Package Leader:	Daniel Cassard (BRGM)
Deliverable Title:	Release of the Search and Ranking Modules Development
Deliverable Number:	D6.1
Deliverable Leader:	LIG
Involved beneficiaries:	BRGM, GeoZS, GEUS, GTK, JRC, NERC (BGS)
Dissemination level:	PU Public
Version:	Final
Status:	Submitted
Year:	2017
Author:	Danielle Ziébelin
Reviewed by:	Daniel Cassard
Approved by:	Erika Machacek, Kisser Thorsøe

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 689648.

Notice

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

PURPOSE

The MICA Expert System is currently being developed in the frame of the H2020 MICA project and more precisely within its work package 6 (WP6) entitled 'The European Raw Materials Intelligence Capacity Platform (EU-RMICP)' whose partners are BRGM, GeoZS, GEUS, GTK, JRC, NERC (BGS) and LIG (Laboratoire d'Informatique de Grenoble).

The description of the project can be found on the project website (www.mica-project.eu/). To briefly summarize: MICA (Mineral Intelligence Capacity Analysis – 2015-2017) has as objective to develop a platform of knowledge, the EU-Raw Materials Intelligence Capacity Platform (or EU-RMICP), integrating metadata on data sources related to primary and secondary mineral resources and providing the end users with an expertise on the methods and tools used in mineral intelligence. In practice, the system is to be capable of bringing relevant 'answers' of the type 'how to proceed for …' on almost any questions related to mineral resources, covering significant parts of mineral supply chains, from prospecting to recycling, taking into account the environmental, technical, political and social dimensions.

To meet this challenge, the EU-RMICP is based on an ontology of the domain of mineral resources (coupled with more generic cross-functional ontologies, relative to commodities, time and space), which represents the domain of the questions of the users (experts and non-experts). The user navigates this ontology by using a Dynamic Decision Graph (DDG) which allows him/her to discover the solutions which he/she is looking for without having to formulate any question. The system is coupled with a 'RDF TripleStore', a database storing the ontologies, factSheets, docSheets and flowSheets (i.e., specific formatted forms) respectively related to methods, documentation and scenarios and metadata (Figure 1).

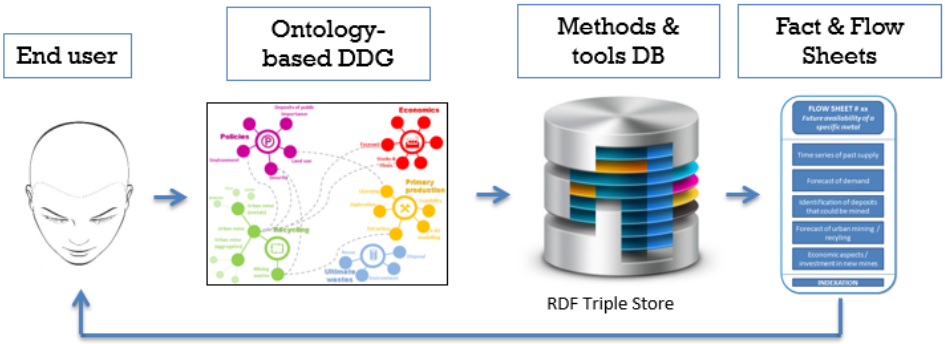

Figure 1 WP6 objective, and the Dynamic Decision Graph.

In order to make the system more powerful and most convenient for the end user, it was in WP6 decided that the results will be presented by pertinence or relevance. In other words, the answers brought by the system will take into account all the information related to the navigation on the ontology-based DDG. This will allow ranking the fact/doc/flowSheets presented to the end user.

The purpose of D6.1 is to present the methodology and the queries and algorithms implemented in order to get this result.

EXECUTIVE SUMMARY

The present report deals with the ranking by pertinence of the different answers brought by the MICA Expert System to a question or a query of an end user.

The system has been conceived in a way so the end user doesn't have to formulate the question in mind. The navigation on the ontology-based Dynamic Decision Graph (DDG) allows the user to select the concepts and sub-concepts which are closest to what he/she has in mind. This is possible because the Main Ontology has been designed by the MICA Experts to cover most of the topics related to the Raw Materials domain from prospecting to recycling, taking into account the environmental, technical, political and social dimensions. To each of the concepts and sub-concepts are attached to one or several 'Sheets' describing methods and tools, giving recipes on 'how to proceed to get such or such result', and providing also the end user with the best documents and data sources related to the question in mind.

These different sheets (docSheets for documentation, factSheets for methods and tools, flowSheets for complex scenarios, linkSheets for 'external high-quality' resources …) are attached to one or several concepts, but they can also be linked together: e.g., a factSheet can be linked to another related factSheet, to one or several docSheets which describe some aspects, to some piece of EU legislation, to some types of data … In other words, a sheet is not an isolated element; when navigating on the DDG and choosing one or several concepts, the end user will get very closely related sheets, but also some other sheets – less closely related – but which make sense.

This report explains how to manage this CONTEXTUAL approach. The data/information/ knowledge brought by the DDG have to be ordered, hierarchised by pertinence. This is done using the information that will be given by the end user during its navigation.

This report doesn't (re)describe all the developments made in MICA to arrive to this task entitled 'Search and Ranking Modules Development'. This has been done in a first deliverable (D6.0) released in December 2016 (Cassard *et al.* 2016) to accompany the delivery of the first operational prototype of the EU-RMICP. A complete overview of the whole system will be provided at the end of the project in the form of an exhaustive note accompanying the release of the final version of the EU-RMICP (D6.2).

The present report describes the 3 modules/algorithms which have been developed to obtain the expected result:

- **Search and Select**: find all the resources annotated by the concepts, these concepts having been selected by the user when 'making the query', i.e., during the navigation over the Main Ontology in the DDG.
- **Inference and Saturation**: inference means creating new facts (new assertions and new relations) in the TripleStore to facilitate and optimize searches. Because the strategy to create these new facts consists in deducting all possible assertions, this action is called 'saturation'. The TripleStore is therefore saturated by these deductions. This module is

based on the semantics of the SKOS[1] knowledge representation language which describes the ontology.

- **Ranking**: the search results, found by the search module, are ranked according to their relevance. The relevance is based on the proximity between the semantic annotation of the resources and the search concepts expressed by the user in the query.

[1] SKOS – Simple Knowledge Organization System – see: www.w3.org/2009/08/skos-reference/skos.html

DELIVERABLE REPORT

1. Introduction

Search, inferencing and ranking functionalities involve exchanges between the DDG and associated visualization applications, SPARQL Endpoint (Fuseki), TripleStore (TDB), MICASheetEditor and SheetRepository (Figure 2). The exchanged information flows between these modules are queries written in SPARQL, Fuseki answers (URI of resources), and annotated sheets.

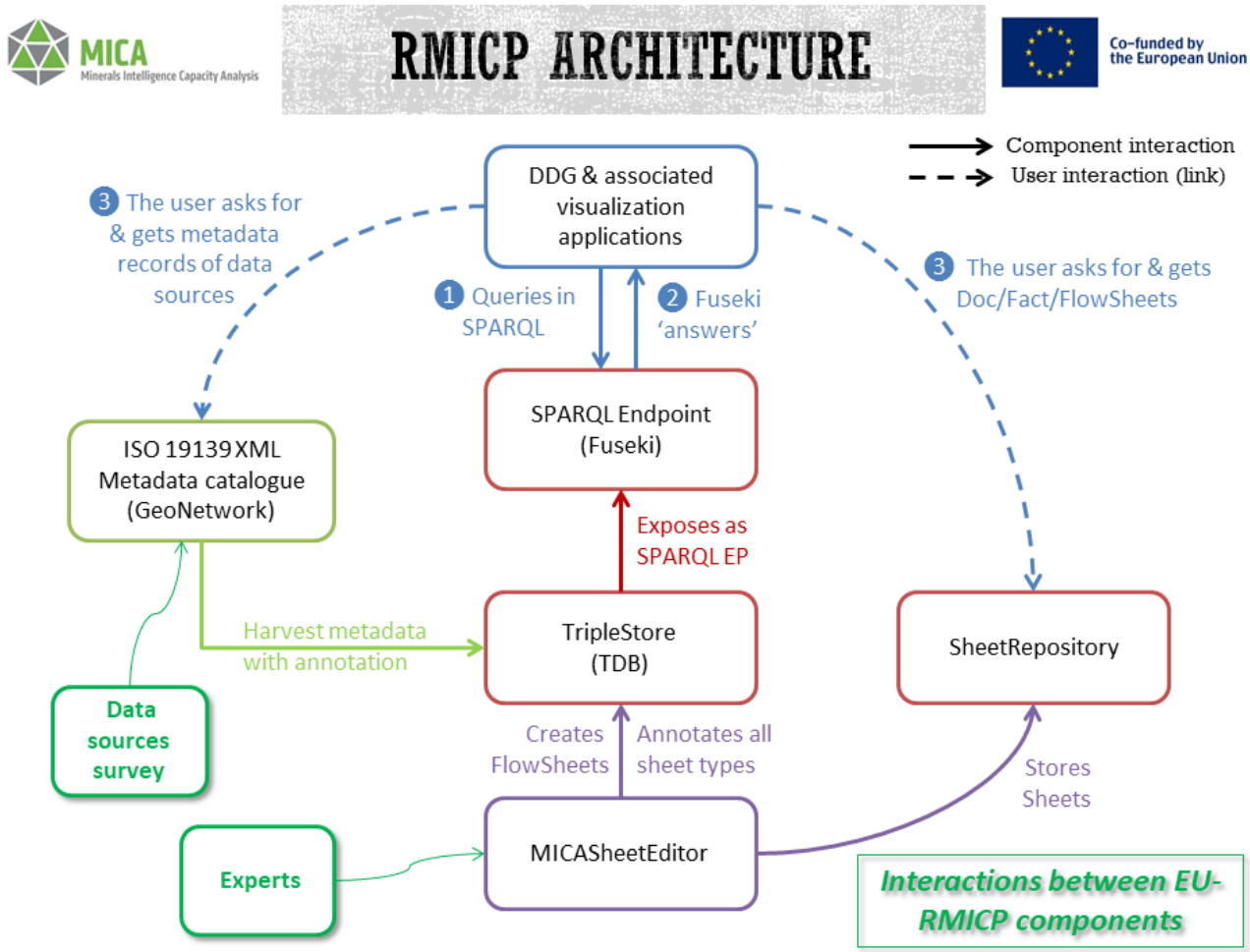

Figure 2 Interactions between the EU-RMICP components.

The three functions presented in this report are used (i) to construct queries from the DDG, (ii) to search for the requested annotated items (i.e., sheets) and (iii) to provide an ordered selection of sheets.

The chosen option will do all these searches through queries and rules instead of using algorithms. This technical choice is justified by the flexibility of queries and rules and the agility of their development (ease of modification and extension) as opposed to the creation of a more complex

search and inference algorithm which would be more difficult to modify. The sheets are searched and selected through queries whose details are given in section 2.

To facilitate this search, it is necessary to saturate the inference graph by making all possible deductions. This allows accessing all the sheets linked to the selected concepts, so these concepts are part of the direct indexing of the sheets or inherited concepts. The inference functions are used to complete the graph. These inferences make it possible to find all the sheets associated with a search. This module is presented in section 3.

Finally, once all the sheets are identified on every possible level and with every possible link, a ranking function is used in order to sort the results. Several strategies are possible in order to carry out this ranking. In this report we present the two solutions that have been used, one for single concept approach, and one for multiple-concept approach. In this module, we also favored a flexible solution and the agility of the developments. This module is presented in section 4.

2. Search and Select

The search is done through queries, which are divided into four types: a search for concepts and sheets, and on questions that relate to information and documents associated with sheets. The set of implemented features is of course extensible.

All queries are documented in detail and are accessible on the project server. The documentation is located at https://gloucklegnou.github.io/MICA/MICAQueriesDDG.html. It provides documentation on all queries codes.

Sample Query can be seen in Figure 3.

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX micavocab: <https://w3id.org/mica/ontology/MicaOntology/>

SELECT ?conceptURI ?label
WHERE {
    ?conceptURI a skos:Concept;
                skos:prefLabel ?label;
                skos:broader micavocab:MICA.
}
order by ?label
```

Figure 3 Search code example.

In the following paragraphs the query titles by category (for more details see the documentation) which have been implemented are listed.

Concept centric queries

Get top level domain concepts (top level concepts in MICA main ontology)
Select all concepts which have a given concept as parent
Get direct sub-concepts of a given concept
Get direct parent concept of a given concept
Get all concepts of a given concept scheme
Get statistics by concept
Get all sheets of a given type directly related to a given concept
Get all sheets of a given type related to a given concept
Get all flowsheets related to a given concept
Get all questions related to a given concept
Get all concepts of a given concept scheme related to a given concept
Get metadata (label definition, description, links…) for a given concept
Get all concepts containing a text in their label or definition or other available metadata

Sheet centric queries

Sheets can be any type of MICAContentType, MethodsAndTools, Documentation and Legislation...
Get all sheets of a given type of content
Get all sheets related to a given sheet
Get all questions related to a given sheet
Get all the flowSheets related to a given sheet
Get all concepts related to a given sheet
Get metadata (label definition, description, links…) for a given sheet
Get statistics by sheet
Get all sheets of a given type containing a text in their title or summary or other available metadata

FlowSheet centric queries

Get all FlowSheets
Get all MICAKnowledgeElements belonging in flowSheet
Get all sheets related to a given flowSheet
Get all questions related to a given flowSheet
Get all the flowSheets related to a given sheet
Get all concepts related to a given sheet
Get metadata (label definition, description, links…) for a given flowSheet
Get statistics by flowSheet
Get all sheets of a given type containing a text in their title or summary or other available metadata

Questions

Get all questions
Get all sheets related to a given question
Get all questions related to a given question
Get statistics by question
Get all metadata (question, links…) about a given question
Get all questions containing a given text in their description

In order to clarify how these queries are exploited by the Endpoint module, we give the example that was presented in Report D6.0 (Cassard *et al.* 2016). The principles are the same for all of the previously mentioned requests.

For example, to '*retrieve all factSheets about "Mining Wastes" with factSheets about related methods'*, the following SPARQL query are used.

```
SELECT DISTINCT ?fsd ?d ?fsm ?meth
WHERE {
 {  ?fsd model:hasDomainConcept ?d.
    FILTER( ?d = micavocab: Mining_Wastes ||
        EXISTS {?d skos:broaderTransitive micavocab:Mining_Wastes})
```

```
  }
  OPTIONAL {
   { ?m  skos:inScheme micavocab:MethodsScheme;
        skos:related ?d.
     {  ?fsm model:hasMethodConcept ?m.
        BIND (?m as ?meth)
     }
     UNION {
       ?m1 skos:broaderTransitive ?m.
       ?fsm model:hasMethodConcept ?m1.
       BIND (?m1 as ?meth)
     }
   }
   UNION {
     ?d1 skos:broaderTransitive ?d.
     ?m  skos:inScheme micavocab:MethodsScheme;
        skos:related ?d1.
     {  ?fsm model:hasMethodConcept ?m.
        BIND (?m as ?meth)
     }
     UNION {
        ?m1 skos:broaderTransitive ?m.
        ?fsm model:hasMethodConcept ?m1.
        BIND (?m1 as ?meth)
     }
   }
  }
} ORDER by ?fsd ?fsm
```

The results of this query are shown in Figure 4.

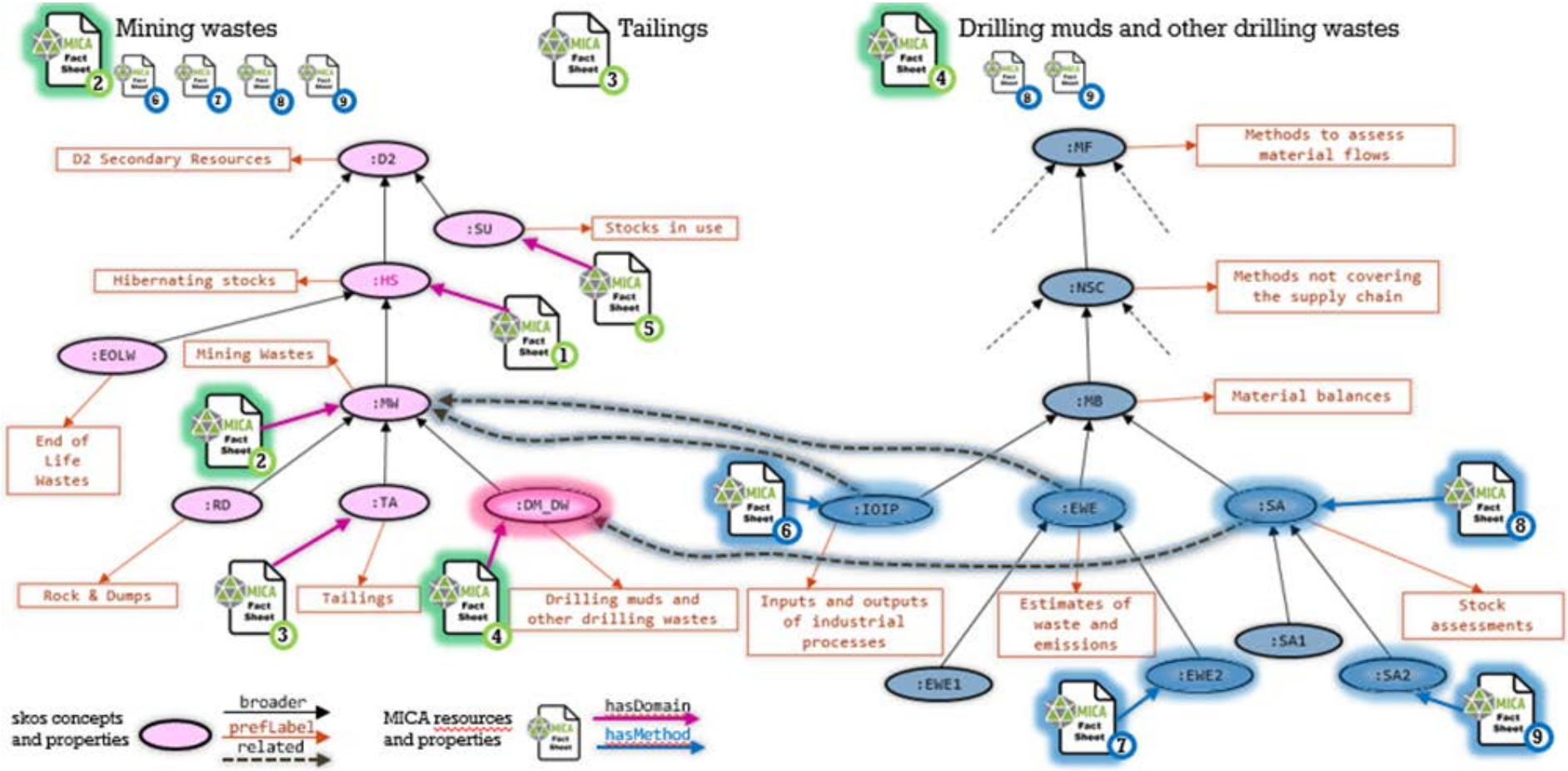

Figure 4 Using the concepts hierarchy to retrieve factSheets and relevant methods.

3. Inference and Saturation

Inference means creating new facts (new assertions and new relations) in the TripleStore to facilitate and optimize searches. Because the strategy to create these new facts consists in deducting all possible assertions, this action is called 'saturation'. The TripleStore is therefore saturated by these deductions. This module is based on the semantics of the SKOS[2] knowledge representation language which describes the ontology.

The inference module can deduce all the possible relations between all the elements present in the TripleStore. The module is divided into two functionalities: Configure Fuseki server to support inferences and make SKOS inferences. These inferences are made by using 18 rules which can be expanded as needed. We present in the rest of this report the list of rules that have been implemented. The full documentation is available at: https://glouck legnou.github.io/MICA/TestInferences.html.

R1: create the narrower property
R2: create the directly broaderTransitive property
R3: create the hierarchical broaderTransitive property
R4: create the directly narrowerTransitive property
R5: create the hierarchical narrowerTransitive property
R6: create the reflexive broaderTransitive and narrowerTransitive
R7: create the symetric property for related concept
R8: create Heritage on data
R9: create Heritage on class
R10: create Heritage on property
R11: create the symetric property for relatedTo MICAResource
R12: create the DomainConcept
R13: create the MethodConcept
R14: create the DataConcept
R15: create the TemporalConcept
R16: create the SpatialConcept
R17: create the CommodityConcept
R18: create the ValueSupplyChainConcept

Figure 5 illustrates the use of rule R1.

[2] SKOS – Simple Knowledge Organization System – see: www.w3.org/2009/08/skos-reference/skos.html

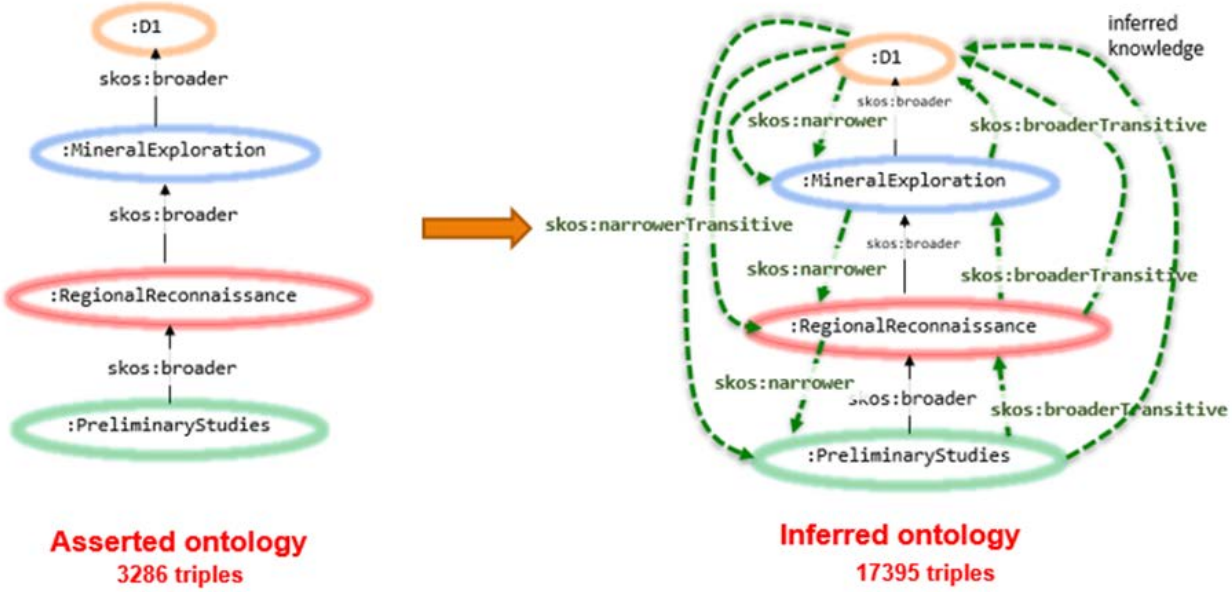

Figure 5 Saturation of inference graph using the broaderTransitive rule.

4. Ranking

Once the results of the queries are obtained by applying the inference module and then computing the query, the results are ordered according to their relevance. In this module, the techniques are also flexible, enabling an easy and agile extension and allowing their development as necessary. The complete documentation and the code associated with the algorithm are available at https://glouck legnou.github.io/MICA/MICARankingQueries.html.

Before presenting the ranking algorithm, we give an example of five resources of type MICASheet as defined and annotated by the concepts C_X. Figure 6 shows the graph which corresponds to this dataset. Note that only hierarchical concept relations and MICA resources annotations are given.

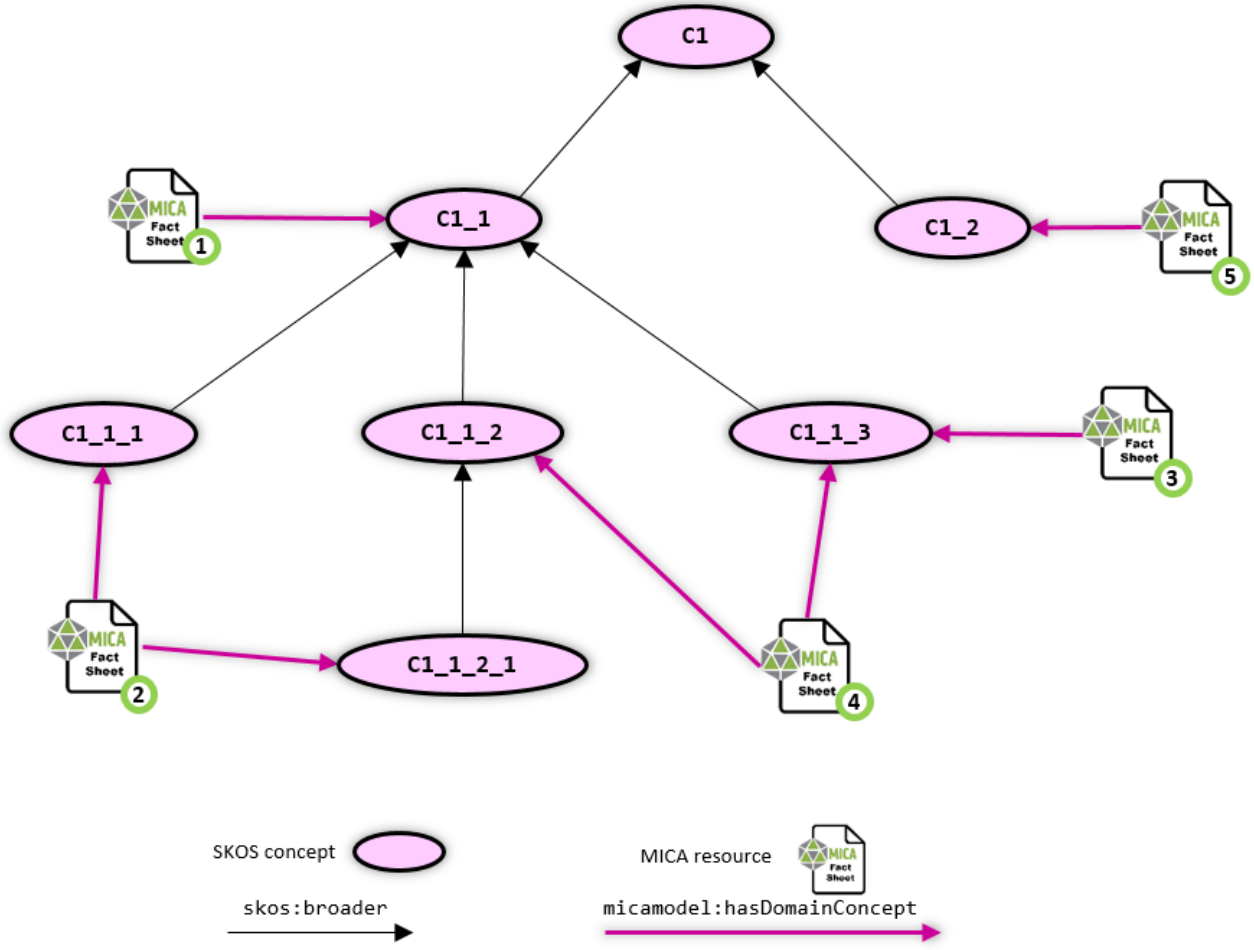

Figure 6 Ranking example.

From the example given in Figure 6, the MICA resources annotation is:

micaresource: R1 annotated by micavocab:C1_1
micaresource: R2 annotated by micavocab:C1_1_1 & micavocab:C1_1_2_1
micaresource: R3 annotated by micavocab:C1_1_3
micaresource: R4 annotated by micavocab:C1_1_2 & micavocab:C1_1_3
micaresource: R5 annotated by micavocab:C1_2

If a user selects C1_1 in the query, the results of the ranking can be seen in Table 1.

Table 1 Ranking of the results of the 'C1_1' concept selection.

sheetURI	conceptURIs	rank	nbmax	nbConceptURIs
micaresource:R1	"micavocab:C1_1"	"0"	"0"	"1"
micaresource:R4	"micavocab:C1_1_3; micavocab:C1_1_2"	"1"	"1"	"2"
micaresource:R3	"micavocab:C1_1_3"	"1"	"1"	"1"
micaresource:R2	"micavocab:C1_1_1; micavocab:C1_1_2_1"	"1"	"2"	"2"

If the rank is equal we take into account only nbmax value and if nbmax value is equal we take into account only nbConceptURIs value. And finally, if nbConceptURIs is equal we display the results randomly.

Algorithm for ranking query results with a one-concept search:

- C is a searched concept
- Ci is a subconcept of C which annotated the resources Ri such as Ci≤C
- di is a distance between Ci and C such as Ci-C

For every resources Ri, we compute distance di for every annotated concept. For example:

- R1((C1, d1) ... (Ci,di)...(Ck,dk))
- R2((C1, d1) ... (Ci,di)...(Ck,dk))
- …
- Rn((C1, d1) ... (Ci,di)...(Ck,dk))

Then, for every resources we search for the following variables:

- rank= min(di)
- nbMax=max(di)
- nbConcepts=cardinal(Ci)

The algorithm above ordered the results of the query:

- The growing order on rank is used which means that the resource with the lowest rank will be the first, and so on.
- If there are ex-aequo resources, the growing order on nbMax is used, which means that the resource with the lowest rank will be the first, and so on.
- If there are ex-aequo resources at this level, the descending order on nbConcepts is used, which means that the resource with the highest number of nbConcepts will be the first, and so on.

Figure 7 shows the graph which corresponds to a dataset annotated with several concepts (only hierarchical concept relations and MICA resources annotations are given).

From the example in Figure 7, the MICA resources annotation is:

micaresource: R1 annotated by micavocab:C1, micavocab:C2, micavocab:C3 & micavocab:C4
micaresource: R2 annotated by micavocab:C1, micavocab:C2 & micavocab:C3
micaresource: R3 annotated by micavocab:C1
micaresource: R4 annotated by micavocab:C1_1 & micavocab:C2
micaresource: R5 annotated by micavocab:C2_2 & micavocab:C3_1
micaresource: R6 annotated by micavocab:C4_2
micaresource: R7 annotated by micavocab:C1_2 & micavocab:C2_1_1
micaresource: R8 annotated by micavocab:C1_1_1
micaresource: R9 annotated by micavocab:C2_2_1, micavocab:C3_1_1 & micavocab:C4_1
micaresource: R10 annotated by micavocab:C3_2_1, micavocab:C4_2_1
micaresource: R11 annotated by micavocab:C5

The user query is the selection C1, C2, C3 and C4, and the result can be seen in Table 2.

Table 2 Ranking of the results of the 'C1, C2, C3 and C4' concept selection.

sheetURI	rank	nbmax	nbConceptURIs
micaresource:R1	"0"	"0"	"4"
micaresource:R5	"3"	"3"	"4"
micaresource:R2	"0"	"0"	"3"
micaresource:R9	"5"	"5"	"3"
micaresource:R4	"1"	"1"	"2"
micaresource:R7	"3"	"3"	"2"
micaresource:R10	"4"	"4"	"2"
micaresource:R3	"0"	"0"	"1"
micaresource:R6	"1"	"1"	"1"
micaresource:R8	"2"	"2"	"1"

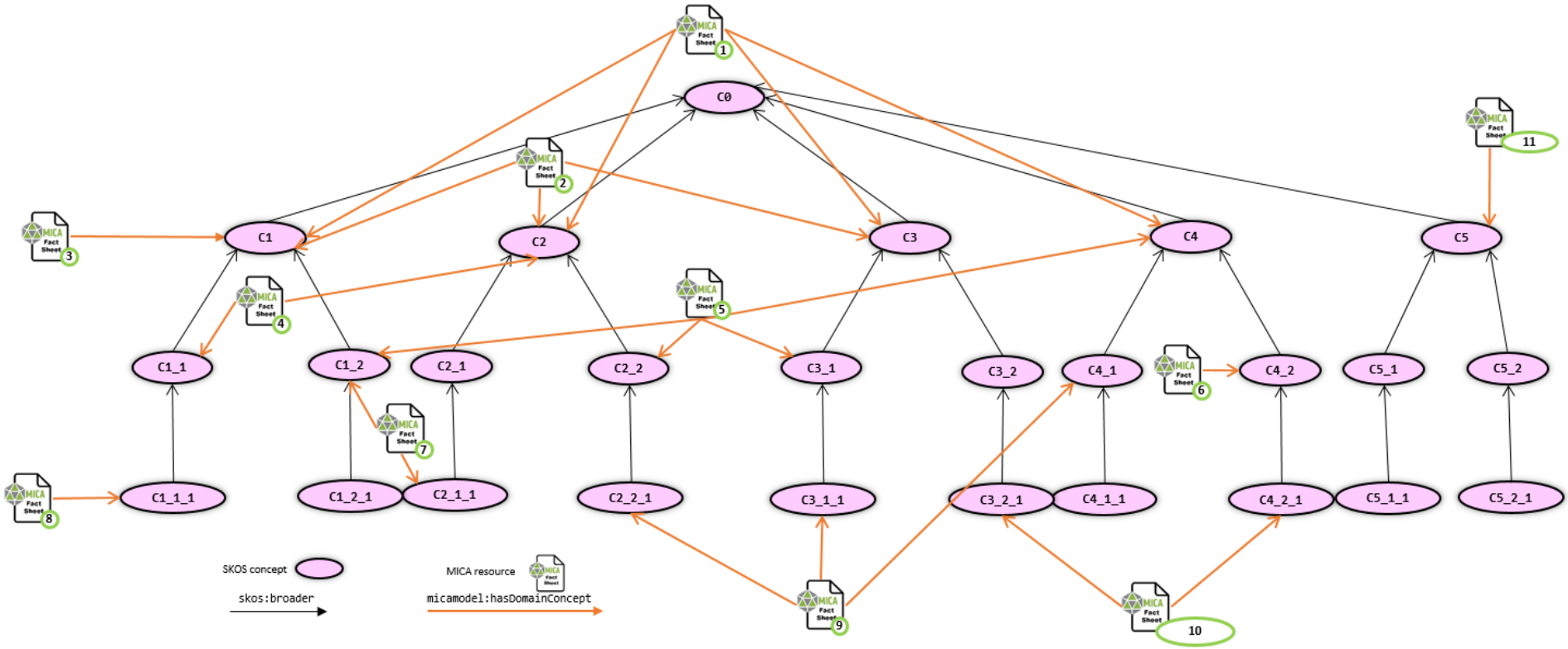

Figure 7 Ranking example with multiple-concept indexation.

Algorithm for ranking query results with a several-concept search:

We search for several concepts Ci. For every concepts Ci:
- we search the resources Ri which are annotated with sub-concepts Cii such as Cii≤Ci
- dii is a distance between Cii and Ci such as Cii-Ci

We compute for every resources Ri:
- ranki= min(dii)
- nbMaxi=max(dii)
- nbConceptsi=cardinal(Cii)

Then, for every resources Ri we compute:
- rankTotal=∑ranki
- nbMaxTotal=∑nbMaxi
- nbConceptsTotal=∑nbConceptsi

The algorithm above ordered the results:

Each resource is annotated with all searched concepts and/or sub-concepts, and its priority is defined by its order of apparition in each of all the searched concepts:

- If there are ex-aequo resources, the growing order on rankTotal is used which means that the resource with the lowest rankTotal will be the first, and so on.
- If there are ex-aequo resources, the growing order on nbMaxTotal is used, which means that the resource with the lowest nbMaxTotal will be the first, and so on.
- If there are ex-aequo resources at this level, the descending order on nbConceptsTotal is used, which means that the resource with the highest number of nbConceptsTotal will be the first, and so on.

5. Concluding remarks

The present note, accompanying the search and ranking module development, has been compiled by Danielle Ziébelin and Daniel Cassard. It summarizes all the work done by the LIG and BRGM on the development of the search and ranking modules. There is still some work for the finalization of the rules and this work will be checked, evaluated and possibly refined when a sufficient number of fact/flow/docSheets will be uploaded into the system, thus making full-scale tests possible.

6. References

Cassard D., Ziébelin D., Tomas, R. (2016) - Note accompanying the release of the first stabilized version of the MICA Dynamic Decision Graph (DDG). Deliverable D6.0. MICA Project Public Report, 24 p.