EasyChair Preprint

№ 6406

Pattern Matching for Mathematical Expressions with OpenMath

Ken Wenzel

August 26, 2021

Pattern matching for mathematical expressions with OpenMath

Ken Wenzel

Fraunhofer-Institute for Machine Tools and Forming Technology IWU,
Chemnitz, Germany
ken.wenzel@iwu.fraunhofer.de

Abstract

An OpenMath Content Dictionary with symbols for pattern matching of tree-like structures is presented. Furthermore, a mapping to RDF and SPARQL is introduced that allows to execute search queries against an OpenMath RDF representation.

1 Introduction

Structural search on mathematical expressions can be useful in the field of mathematical knowledge management, in different engineering disciplines and also in modeling and simulation environments, e.g., for refactoring of existing formulas.

Searching tree-like structures is usually not possible with common text-based search engines, so specialized systems were developed in the past. One of these systems is MathWebSearch that implements formula search by using substition tree indexing [KS06, KAJ+08, KMP12]. Another system is Sewelis, a tool for guided exploration and editing of RDF graphs, that was extended with query operators for tree-like structures to enable search in mathematical expressions [Fer12].

Both systems use some kind of pattern language to express search queries. MathWebSearch uses Content MathML as internal query language which is extended by additional tags (`mq:query`, `mq:and`, `mq:or` and `mq:not`) and attributes (`mq:target`, `mq:generic`, `mq:anyorder` and `mq:anycount`).

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply><csymbol cd="calculus1">int</csymbol>
   <bind><csymbol cd="fns1">lambda</csymbol>
    <bvar><ci mq:generic="x">x</ci></bvar>
    <apply><csymbol cd="arith1">power</csymbol>
      <ci mq:generic="x">x</ci><cn type="integer">0</cn>
    </apply>
   </bind>
  </apply>
</math>
```

Sewelis defines an extended Turtle [BBLPC14] and SPARQL [HSP13] syntax to express patterns for mathematical expressions including placeholders, containment relationships and complex filters as supported by SPARQL.

```
SELECT ?e WHERE {
  ?e is ...math:Power(?x,2)... .
  math:Integral(?e,?x) .
}
```

Both approaches use some kind of mathematical notation and combine it with boolean, binding or more complex operators, for example, to match containment relationships. While the resulting query languages are powerful, they use proprietary extensions for MathML or SPARQL to express patterns and search operations.

OpenMath and its extension mechanism allows to define mathematical symbols for the representation of search patterns as mathematical expressions. This makes it possible to exchange patterns as normal objects with OpenMath or Strict Content MathML.

The following section introduces the newly developed OpenMath Content Dictionary that defines a basic set of operators to represent search patterns as mathematical objects.

2 The patterns Content Dictionary

The `patterns` Content Dictionary defines symbols to encode simple queries for mathematical objects with OpenMath. The symbols and their proposed textual representations (extended POPCORN syntax) are shown in Table 1. Most symbols are application symbols and can be used in the form

$$S(O_1, \ldots, O_n) \tag{1}$$

where S is the particular symbol and $O_1, \ldots, O_n$ are one or more OpenMath objects that may also recursively contain patterns. The only exceptions are the symbol `any`, that can be used as a constant to match an arbitrary expression, and the attribution symbol `bind` for binding of result values to named variables.

Table 1: Symbols of the `patterns` Content Dictionary

Symbol	Shorthand	Description
all_of	`.&`	Matches if all of the given patterns are satisfied.
any	`?`	Matches an arbitrary expression.
any_of	`.|`	Matches if at least one of the given patterns is satisfied.
argument	`.,`	Matches argument lists that satisfy all of the given patterns.
descendant	`..+`	Matches subexpressions at any hierarchy level below an expression that satisfy all of the given patterns.
none_of	`.!`	Matches if none of the given patterns is satisfied.
root	`.^`	Matches expressions that are not part of any other expression and that satisfy all of the given patterns.
self_or_descendant	`...`	Matches the expression or any subexpression that satisfy all of the given patterns.
bind	`bind`	Binds values matched by the annotated patterns to a result variable. When used with an Openmath variable the value of bind is ignored, the variable is interpreted as placeholder and the target variable name is used for binding of result values. Multiple occurrences of a named variable defined with `bind` must match an equal expression.

By using symbols of the patterns Content Dictionary, for example, the following pattern can be defined:

```
patterns:root(patterns:descendant(
  patterns:any_of(arith1:sum, arith1:product)
))
```

This pattern finds mathematical expressions that contain sums or products at any nesting level. The rather long names of the symbols (like `descendant` or `any_of`) hinder the readability of such expressions. Therefore we propose shorthand notations as defined by Table 1 that would allow to express the pattern as:

```
.^(..+(.|(sum, product)))
```

Patterns with named placeholders as used by the examples given in the introduction can be expressed as follows:

```
patterns:root(
  calculus1:int($x{patterns:bind -> 1} -> patterns:self_or_descendant($x{patterns:bind -> 1}^2))
)
```

A possible short form of the previous pattern could be:

```
.^(calculus1:int(?x -> ...(?x^2)))
```

The term `?x` is the short form of `$x{patterns:bind -> 1}` and represents a placeholder that matches variables with arbitrary names.

3 Query execution

If the OpenMath objects are encoded in OpenMath-RDF then the query language SPARQL can be used to evaluate the patterns. This is similar to the approach followed by Sewelis. The transformation of the patterns to SPARQL reuses the transformation operator T that translates OpenMath-XML to OpenMath-RDF. This approach is possible since SPARQL uses the Turtle syntax for graph patterns whereby OpenMath-RDF can be directly represented in SPARQL.

Table 2 shows additional rules that are necessary to support all patterns as defined in Table 1. The SPARQL representation uses extended functions of the query language like complex path expressions and filters to traverse parent-child relationsships as well as argument lists.

By using shorthands for the symbols the previous example search pattern for sums and products can be compactly represented as:

```
.^(..+(.|(sum, product)))
```

This expression can be translated to SPARQL by applying the rules from Table 2. The result is the following SPARQL query:

```
SELECT ?result WHERE {
  {
    ?n0 (<>)? <http://www.openmath.org/cd/arith1#sum> .
  } UNION {
    ?n0 (<>)? <http://www.openmath.org/cd/arith1#product> .
  }
  ?result (math:arguments|math:symbol|...|rdf:rest)+ ?n0 .
  FILTER NOT EXISTS {
    [] math:arguments|math:symbol|...|rdf:rest ?result .
  }
}
```

The generated query represents the alternative (`.|`) between the symbols `sum` and `product` by using the `UNION` set operator. Furthermore, path expressions are used to determine the descendants (`..+`) and to filter any non-root nodes `.^` with the `FILTER NOT EXISTS` SPARQL operator. As can be seen, the OpenMath-based representation of the search pattern is considerably more compact and understandable as the equivalent SPARQL query.

Besides the unpredictable execution time for complex search patterns another limitation should be considered if OpenMath-RDF and SPARQL are used to execute search queries: SPARQL uses names (URIs or blank node identifiers) to compare nodes of an RDF graph and not their structural equality. Therefore it is necessary to preprocess the RDF data if patterns with named variables, that use the `patterns:bind` annotation, should be

Table 2: Transformation rules for search patterns to SPARQL

POPCORN – O	**SPARQL** – $T(O)$
Patterns	
`.&(A1, ..., An)`	`?v T(SELF_PATH) T(A1) ; ... ;` `  T(SELF_PATH) T(An) .`

`.|(A1, ..., An)`	`{ ?v T(SELF_PATH) T(A1) }`
	`  UNION { ... }`
	`  UNION { ?v T(SELF_PATH) T(An) } .`

`.,(A1, ..., An)`	`?v rdf:first|(rdf:rest+/rdf:first) T(A1) ; ... ;` `  rdf:first|(rdf:rest+/rdf:first) T(An) .`
`..+(A1, ..., An)`	`?v (T(COMPONENT_PATH))+ T(A1) ; ... ;` `  (T(COMPONENT_PATH))+ T(An) .`

`.!(A1, ..., An)`	`FILTER NOT EXISTS {`
	`  { ?v T(SELF_PATH) T(A1) }`
	`  UNION { ... }`
	`  UNION { ?v T(SELF_PATH) T(An) } .`
	`}`

`.^(A1, ..., An)`	`{ ?v T(SELF_PATH) T(A1) }`
	`  UNION { ... }`
	`  UNION { ?v T(SELF_PATH) T(An) } .`
	`FILTER NOT EXISTS { [] T(COMPONENT_PATH) ?v }`

`...(A1, ..., An)`	`?v (T(COMPONENT_PATH))* T(A1) ; ... ;` `  (T(COMPONENT_PATH))* T(An) .`
`?`	`?v T(SELF_PATH) [] .`
`?x`	`?x T(SELF_PATH) [] .`
`O1{patterns:bind -> "x"}`	`?x T(SELF_PATH) T(O1) .`
`$x{patterns:bind -> 1}`	`?x T(SELF_PATH) [] .`
Paths	
`SELF_PATH`	`<>?`

`COMPONENT_PATH`	`math:arguments|math:symbol|`
	`math:operator|math:target|math:variables|`
	`math:binder|math:body|math:attributeKey|`
	`math:attributeValue|rdf:first|rdf:rest`

matched via SPARQL. A possible algorithm is the *Universal RDF Dataset Canonicalization Algorithm 2015*[1] that generates stable content-based identifiers for blank (anonymous) nodes of an RDF graph. After applying the algorithm blank nodes of an RDF graph have the same name if they recursively share the same properties with the same values. This allows to use a simple comparison of node names to reason about their structural equality.

For verifying the functionality of the developed Content Dictionary a prototypical web application was created that allows to execute search queries on available OpenMath Content Dictionaries. A screenshot of the application is shown in Figure 1.

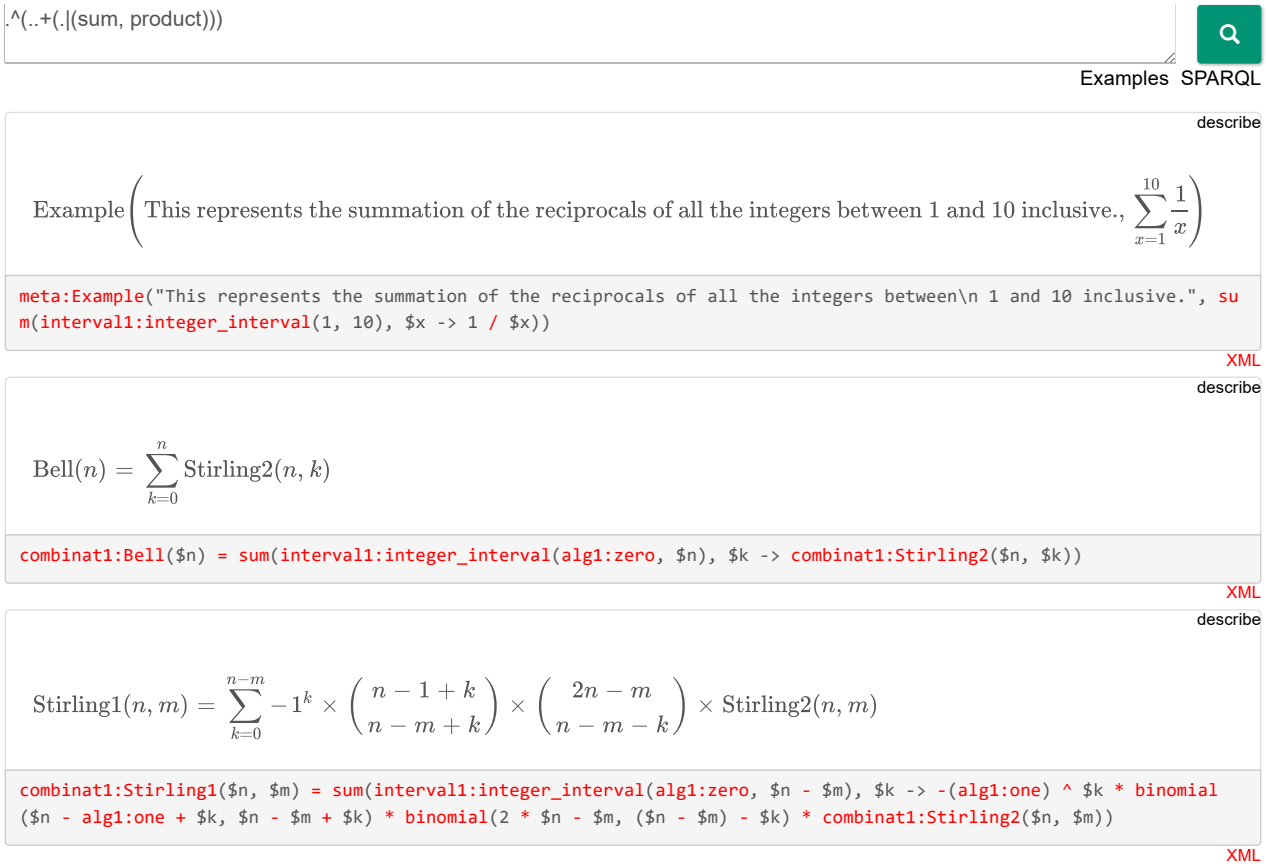

Figure 1: Search results visualized in the web application

The upper part contains an input field for the search pattern. This pattern is parsed into an OpenMath-XML representation and then converted to SPARQL by applying the rules from Table 2. This query is executed against the RDF database containing OpenMath objects. The results are displayed graphical via MathML as well as in their textual form.

As reference data the RDF version[2] of the Content Dictionaries available on the OpenMath website was used. For the example pattern `.^(..+(.|(sum, product)))` the generated SPARQL query finds multiple results as the OpenMath Content Dictionaries contain different matching definitions and examples. The first three results are shown in Figure 1 where the first is an example (`meta:Example`) while the others come from formal mathematical properties of symbols.

[1] `https://json-ld.github.io/rdf-dataset-canonicalization/spec/` (09.07.2021)
[2] The RDF version of the Content Dictionaries is available at `https://github.com/numerateweb/openmath-cd` (29.06.2021)

4 Conclusion

A Content Dictionary to express search patterns for mathematical expressions with OpenMath was presented. This enables the exchange of such patterns in a standardized way as OpenMath objects.

Furthermore, a mapping of patterns to RDF and SPARQL was given as an option to execute search patterns against an OpenMath RDF representation. Although the scalability of this approach has to be investiged it is a proof of concept to use general purpose graph databases for storage and retrieval of mathematical formulas.

References

[BBLPC14] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. RDF 1.1 Turtle. *W3C Recommendation*, 2014.

[Fer12] Sébastien Ferré. An RDF Vocabulary for the Representation and Exploration of Expressions with an Illustration on Mathematical Search, 2012.

[HSP13] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. SPARQL 1.1 Query Language. *W3C Recommendation*, 2013.

[KAJ+08] Michael Kohlhase, Stefan Anca, Constantin Jucovschi, Alberto González Palomo, and Ioan A Sucan. Mathwebsearch 0.4, a semantic search engine for mathematics. *Manuscript at http://mathweb.org/projects/mws/pubs/mkm08. pdf*, 2008.

[KMP12] Michael Kohlhase, Bogdan A. Matican, and Corneliu-Claudiu Prodescu. Mathwebsearch 0.5: Scaling an open formula search engine. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics*, pages 342–357, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[KS06] Michael Kohlhase and Ioan Sucan. A search engine for mathematical formulae. In *Proceedings of the 8th International Conference on Artificial Intelligence and Symbolic Computation*, AISC'06, pages 241–253, Berlin, Heidelberg, 2006. Springer-Verlag.