FunMap: Functional Mappings for Scaled-Up Knowledge Graph Creation

Samaneh Jozashoori[1], David Chaves-Fraga[2], Enrique Iglesias[1,3], Maria-Esther Vidal[1],Oscar Corcho[2]

[1] TIB Leibniz Information Center for Science and Technology & L3S, Germany
{samaneh.jozashoori,maria.vidal}@tib.eu
[2] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
{dchaves,ocorcho}@fi.upm.es
[3] University of Bonn
s6enigle@uni-bonn.de

We present FunMap, an interpreter of RML+FnO,that converts a data integration system defined using RML+FnO into an equivalent data integration system where RML mappings are function-free; Figure 1 depicts the FunMap architecture. FunMap resembles existing mapping translation proposals and empowers the knowledge graph creation process with optimization techniques to reduce execution time. Transformations of data sources include the projection of the attributes used in the RML+FnO mappings. They are supported on well-known properties of the relational algebra, e.g., the pushing down of projections and selections into the data sources, and enable not only the reduction of the size of data sources but also the elimination of duplicates. Additionally, FunMap materializes functions –expressed in FnO– and represents the results as data sources of the generated data integration system; the translation of RML+FnO into RML mappings that integrate the materialization of functions is performed by means of joins between the generated RML mappings. In this document, we present the ***FunMap experimental guideline*** for reproducing the experiments reported in the paper entitled "FunMap: Functional Mappings for Scaled-Up Knowledge Graph Creation" accepted in the research track of ISWC 2020.

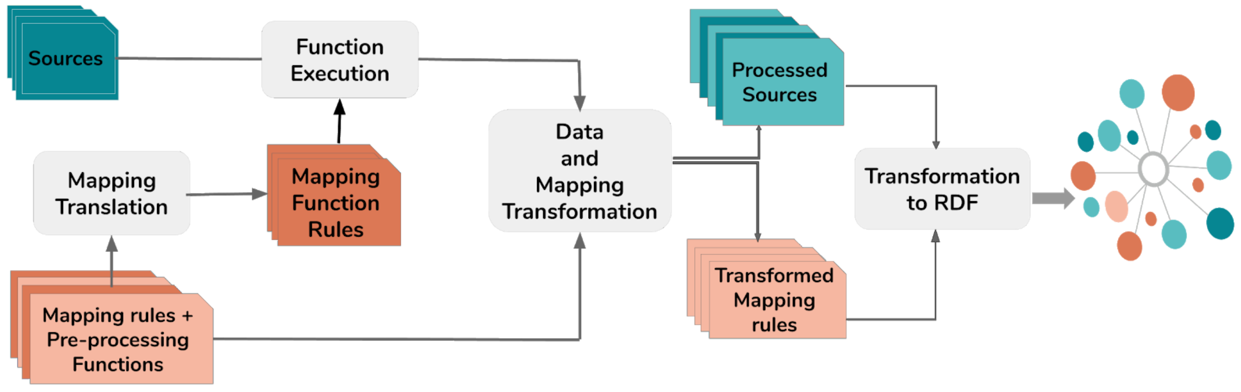

Figure 1. The Architecture of FunMap.

The **FunMap experimental guideline** comprise the instructions to reproduce our experiments and produce the figures reported in the paper; it is publicly available on GitHub (https://github.com/SDM-TIB/FunMap) under Apache 2.0 License and with a permanent DOI (https://doi.org/10.5281/zenodo.3993656).

Where is the FunMap experimental guideline to reproduce the ISWC 2020 experimental evaluation? Github repository:

The Github repository ((https://github.com/SDM-TIB/FunMap) contains an independent branch named *eval-iswc2020* (https://github.com/SDM-TIB/FunMap/tree/eval-iswc2020). The resource contains the code of the engines, the dataset, and RML+FnO mapping rules. Furthermore, it comprises the bash scripts to run the testbeds over the studied engines and generate the results, as well as the Python scripts to plot the figures presented in our paper.

What is required to run the FunMap experimental guideline?

Libraries

- Java - openjdk v1.8.0_265
- Docker - v19.03.7
- Docker-compose - v1.23.1
- Node - v12.18.3
- Python - v3.7
- Anaconda - (https://repo.anaconda.com/archive/Anaconda3-2020.07-Linux-x86_64.sh)

Bash commands

- bc
- wget

Computational requirements

Ubuntu 16.04 with Intel(R) Xeon(R) Platinum 8160, CPU 2.10GHz and 700Gb RAM.

The FunMap Experimental Evaluation:

Research questions

1. What is the impact of data duplication rate in the execution time of a knowledge graph creation approach?
2. What is the impact of different types of complexity over transformation functions during a knowledge graph creation process?
3. How does the repetition of the same function in different mappings affect the existing RML engines?
4. What is the impact of relational data sources in the knowledge graph creation process?

Datasets and Mappings.
To the best of our knowledge, there are no testbeds to evaluate the performance of a knowledge graph construction approach that applies functional mappings. Consequently, following the real-world scenario that initially motivated this research, we create our testbed from the biomedical domain. We generate a baseline dataset by randomly selecting 20,000 records from the coding point mutation dataset in COSMIC[1] GRCh37, version90, released August 2019 database. We keep all 39 attributes of the original dataset in the baseline dataset, while only five to seven of them are utilized in mappings. In total, four different mapping files are generated consisting of one **FunctionMap** and four, six, eight, or ten **TriplesMaps** with a **predicateObjectMap** linked to the function. To additionally validate FunMap in case of large-sized data, we create another dataset following the same criteria, with 4,000,000 records and the size of about 1.3GB.

RML+FnO Engines
The baselines of our study are three different open source RML-complaint engines that are able to execute RML+FnO mappings and have been extensively utilized in multiple applications and tested by the community:
1. SDM-RDFizer v3.0 (named SDM-RDFizer**(RML+FnO))
2. RMLMapper v4.7 (named RMLMapper**(RML+FnO))
3. RocketRML v1.1 (named RocketRML**(RML+FnO).

In order to evaluate the impact of transformation rules, we implement FunMap v1.0 on the top of the aforementioned engines with DTR2 optimization as an optional parameter. We refer to the approach which applies FunMap excluding DTR2 as FunMap$^-$ (in the experiment scripts and results it will appear as FunMap-Basic. The combination of each configuration is as follows:
1. FunMap+SDM-RDFizer
2. FunMap+RMLMapper
3. FunMap+RocketRML
4. FunMap$^-$+SDM-RDFizer
5. FunMap$^-$+RMLMapper
6. FunMap$^-$+RocketRML

Metrics
Elapsed time spent by an engine to complete the creation of a knowledge graph and also counts FunMap pre-processing; it is measured as the absolute wall-clock system time as reported by the **time** command of the Linux operating system. Each experiment was executed five times and average is reported.

Experimental Setups
Based on our research questions, we set up in overall 198 experiments as the combinations of the following scenarios. We create two datasets from our baseline with 25% and 75% duplicates which means in the 25% duplicate dataset, 25% and in the 75% duplicate dataset, 75% of the records are duplicated. Additionally, two functions with different levels of complexity are created.

[1] https://cancer.sanger.ac.uk/cosmic

We describe the complexity level of the functions based on the number of required input attributes and operations to be performed. Accordingly, "simple" function is defined to receive one input attribute and perform one operation, while a "complex" function receives two input attributes and completes five operations. In total, we create eight mapping files including four, six, eight, and ten **TriplesMap** and one **FunctionMap** to be either "simple" or "complex". Additionally, six experiments using 75% duplicate datasets of 20,000 and 4,000,000 records and a mapping file including ten complex functions are set up in order to be run over a relational database (RDB) implemented in MySQL 8.0.

How to execute the FunMap experimental guideline?

Follow these steps to reproduce the experiments shown in the paper:
1) Go to the RDB-Preparation folder and follow the instructions in the README.md file to prepare the RDBs.
2) Run experiments over SDM-RDFizer (to generate the Figure7.a,b and Figure8.a,b)

```
cd SDM-RDFizer
bash preparation.sh
bash run.sh
bash run-rdb.sh
```

Expected outputs:

Files: the following files compile the results obtained by running the testbeds on SDM-RDFizer
1. results-funmap-sdmrdfizer.csv: FunMap+SDM-RDFizer
2. results-funmap-basic-sdmrdfizer.csv: FunMap^{-}+SDM-RDFizer
3. results-sdmrdfizer.csv: SDM-RDFizer**(RML+FnO)
4. results-rdb-funmap-sdmrdfizer.csv: FunMap+SDM-RDFizer over RDB
5. results-rdb-sdmrdfizer.csv: SDM-RDFizer**(RML+FnO) over RDB

Figures: Further the figures Figure7.a,b and Figure8.a,b are generated.
1. Figure7_a.png
2. Figure7_b.png
3. Figure8_a.png
4. Figure8_b.png

3) Run experiments over RMLMapper (to generate the Figure7.c,d and Figure8.c,d)

```
cd RMLMapper
bash preparation.sh
bash run.sh
bash run-rdb.sh
```

Expected outputs:

Files: the following files compile the results obtained by running the testbeds on RMLMapper:

1. results-funmap-rmlmapper.csv: FunMap+RMLMapper
2. results-funmap-basic-rmlmapper.csv: FunMap$^-$+RMLMapper
3. results-rmlmapper.csv: RMLMapper**(RML+FnO)
4. results-rdb-funmap-rmlmapper.csv: FunMap+RMLMapper over RDB
5. results-rdb-rmlmapper.csv: RMLMapper**(RML+FnO) over RDB

Figures: Further the figures Figure7.c,d and Figure8.c,d are generated.

1. Figure7_c.png
2. Figure7_d.png
3. Figure8_c.png
4. Figure8_d.png

4) Run experiments over RocketRML (to generate the Figure7.e,f)

```
cd RocketRML
bash preparation.sh
bash run.sh
```

Expected outputs:

Files: the following files compile the results obtained by running the testbeds on RocketRML:

1. results-funmap-rocketrml.csv: FunMap+RocketRML
2. results-funmap-basic-rocketrml.csv: FunMap$^-$+RocketRML
3. results-rocketrml.csv: RocketRML**(RML+FnO)

Figures: Further the figures Figure7.e,f

1. Figure7_e.png
2. Figure7_f.png

Note that the files "results-rdb-funmap-sdmrdfizer.csv", "results-rdb-sdmrdfizer.csv", "results-rdb-funmap-rmlmapper.csv", and "results-rdb-rmlmapper.csv" include the data that support the following statement reported in the paper:

"*The experimental results on RDBs show even more significant improvement in the performance of both RMLMapper and SDM-RDFizer in the presence of FunMap. In FunMap+RMLMapper, applying join in the SQL queries that define the logicalSources instead of using joinConditions reduces execution time by up to a factor of 18. These results evidence that joinConditions are not efficiently implemented by RMLMapper, and explain why FunMap+RMLMapper is showing less improvement compared to FunMap+SDM-RDFizer in \autoref{fig:exp-complex}. Moreover, FunMap+SDM-RDFizer successfully performs on the large-sized relational dataset of 1.3GB in 5,670.67 seconds, while SDM-RDFizer**(RML+FnO) cannot create the KG and times out after 10,000 seconds.* "