Ademar Crotti Junior, Christophe Debruyne, and Declan O'Sullivan. Juma uplift: Using a block metaphor for representing uplift mappings. In 12th IEEE International Conference on Semantic Computing, ICSC 2018, Laguna Hills, CA, USA, January 31 - February 2, 2018, pages 211–218. IEEE Computer Society, 2018

Juma Uplift: Using a Block Metaphor for Representing Uplift Mappings

Ademar Crotti Junior
ADAPT Centre
Trinity College Dublin
Dublin 2, Ireland
Email: ademar.crotti@adaptcentre.ie

Christophe Debruyne
Odisee University College
100 Brussels
Belgium
Email: christophe.debruyne@odisee.be

Declan O'Sullivan
ADAPT Centre
Trinity College Dublin
Dublin 2, Ireland
Email: declan.osullivan@adaptcentre.ie

***Abstract*—A significant part of the Linked Data web is achieved by converting non-RDF resources into RDF. Even though several approaches and mapping languages have been proposed in the literature, the knowledge required for such a task is still substantial. In prior work, we proposed a visual representation based on the block metaphor and applied it to the W3C Recommendation R2RML. In this paper, we describe a new implementation of this method, called Juma Uplift, that is capable of generating mapping definitions for different uplift mapping languages while still being fully compliant to the particular uplift mapping specification. Preliminary findings indicate that Juma Uplift is expressive enough to generate accurate mappings for the two syntactically distinct mapping languages under examination, R2RML and SML.**

***Index Terms*—Visual Representation, Data Mapping, R2RML, SML.**

I. Introduction

Large amounts of data on the Web still resides in formats other than the Resource Description Framework[1] (RDF) data model, currently being advocated by the W3C community as the means to enable meaningful data exchange on the Web, and a variety of innovative applications, such as data integration and others [1]. For this reason, significant research has been invested by the Semantic Web community into the process of converting non-RDF resources into RDF. This process is typically called uplift.

Many approaches and mapping languages have been proposed to map non-RDF data into RDF in literature. Mapping languages are able to detach mapping definitions from the implementations that executes them [2]. These, however, still require knowledge of the mapping language and significant human effort on manually creating, editing and curating these mappings [3]. To mitigate the knowledge required by mapping languages one can avail of graphical editors. Nonetheless to date, these have mainly focused on Semantic Web experts, representing mappings as graphs, since the RDF data model is itself one [4].

In previous work, we have proposed a visual representation for mappings and applied it to the W3C's RDB to RDF mapping language (R2RML) [5], called Juma [6]. A user study showed that this visual representation was helpful in the creation of mappings with sufficient results in standard usability tests for different types of stakeholders (including experts and non-experts of Semantic Web technologies) [4].

Juma is based on the block (or jigsaw) metaphor that has become popular with visual programming languages where it is called the block paradigm such as Scratch[2]. In this metaphor, concepts are represented as blocks that can only be combined with other compatible blocks. The block metaphor targets different types of users, allowing them to focus on the logic instead of the languages syntax. In addition, it has been used successfully in other domains, such as programming [7] and on the creation of SPARQL queries [8].

In this paper, we describe a new implementation of Juma, called Juma Uplift. This implementation is able to generate (depending on preference) mappings compliant to the W3C Recommendation R2RML, an RDF-based mapping language, and compliant to the SML mapping language [9], a RDB2RDF SPARQL-based mapping language.

The main contributions of this paper can be summarized as follows: (i) a new implementation of the Juma method with support for two syntactically distinct mapping languages, called Juma Uplift; (ii) a comparison between Juma implementations and (iii) an evaluation of the expressiveness of Juma Uplift based on the supported mapping languages and accuracy of the RDF datasets generated.

The remainder of this paper is structured as follows: Section II reviews the related work. In Section III and IV we overview the R2RML and SML mapping languages, respectively. Section V describes Juma and the new implementation, Juma Uplift. A comparison of Juma implementations is presented in Section VI. Section VII describes the evaluation undertaken. Section VIII concludes the paper.

II. Related Work

In this section, we discuss the state-of-the-art in mapping languages and editors.

[1]https://www.w3.org/TR/rdf11-concepts/, accessed in September 2017.

[2]https://scratch.mit.edu/, accessed in September 2017.

A. Mapping Languages

Mapping languages are declarative languages used to express customized mappings defining how non-RDF data should be represented in RDF. An engine is usually associated with a mapping language, being a software processor that uses the mapping file together with the input data to generate an RDF dataset.

R2RML [5] is the W3C Recommendation mapping language to map relational databases into RDF. Examples of R2RML implementations are db2triples[3] and morph[10]. R2RML-F [11] extends R2RML to add support for data transformation functions. Other R2RML extensions are RML [12] and xR2RML [13]. These extensions added support for a wider set of input data formats, such as CSV and XML. D2R[4] is another RDF-based declarative language to map relational databases into RDF. It is supported by the D2R server and UltraWrap [14], with the latter also supporting R2RML.

Sparqlification Mapping Language (SML) [9] is a mapping language based on SQL CREATE VIEWS and SPARQL CONSTRUCT queries with support for relational databases and CSV files. SPARQL-Generate [15] is another SPARQL-based mapping language with support for multiple input data formats, like RML. XSPARQL [16] combines XQuery and SPARQL to map XML data into RDF. TARQL[5] is another mapping language based on SPARQL to convert CSV datasets into RDF.

Though useful, mapping languages require significant human effort in manually creating, editing and curating mappings [3]. To mitigate the knowledge required by mapping languages one can avail of graphical editors.

B. Graphical Editors

Graphical editors offer an interface or a visual representation to support the definition of mappings.

The fluidOps editor [17] is a web-based application that relies on a step-by-step workflow. Each step focuses on the creation of one part of the mapping. The mapping is only available at the end of this process and changes in the mapping restart the workflow. In [3], an extension of this editor was proposed. In this extension, the mapping process starts based on an existing ontology. In this sense, changes do not restart the workflow. OntopPro[6] [18] is a Protégé [19] plugin that uses a proprietary mapping language internally to create mappings. The Virtuoso Universal Server[7] has an extension where data can be converted into RDF by creating R2RML mappings or using a wizard that guides users in the creation such mappings. R2RML By Assertion (RBA) [20] uses a tree table structure to represent ontologies and RDF vocabularies, side by side with the input data. This interface allows users to match classes and properties to attributes. These assertions generate an R2RML mapping. Datalift [21] is a web editor where one needs to convert data into raw RDF in a first step. Following this step, it is possible to explore and transform the raw RDF representation. OpenRefine[8] is an ETL tool that supports cleaning and transformation functions for many input data formats. The uplift to RDF is available through RDF Refine[9], an extension to OpenRefine, allowing the mapping and interlinking of RDF datasets through its web interface.

Karma [22] is a web-based application where data is loaded before it can be mapped into RDF. The ontologies used during the mapping process are represented in a tree structure and the data as a table. The mapping is represented using a graph. The creation of mappings using Karma can be troublesome because of the data-centric approach, where every input is shown in a different table. This makes the interlinking between tables unnecessarily complex. Lembo et al. [23] uses a graph representation for R2RML mappings. However, the creation and/or editing of mappings are undertaken through text editing, which make the mapping process prone to errors. RMLeditor [2] has support for R2RML and RML mapping languages. The RMLeditor also uses a graph representation for the mapping. The input data and RDF output are shown as tables. MapOn [24] is yet another graph representation tool with support for R2RML mappings. SQuaRE [25] is a tool that provides a visual environment for the creation of R2RML mappings. This tool also uses a graph visual representation for mappings. In a first step, users need to select the tables that are going to be mapped. Ontologies and RDF vocabularies that will be used in the mapping process are shown as trees.

These tools either offer interfaces to guide users in the creation of mappings or represent them as graphs, since the RDF model is itself one. We argue that these approaches may not be as intuitive for non-expert users.

In contrast to the state of the art editors, the Juma approach aims to represent mappings using the block metaphor targeting different types of stakeholders. Our previous implementation, Juma R2RML, is closely tied to the R2RML mapping language, representing each R2RML construct as a block. In this paper, we describe a new implementation that has been developed, which abstracts away from R2RML to provide an uplift mapping block editor capable of generating a variety of mapping languages.

III. R2RML

In this section, we briefly explain the main concepts related to the W3C Recommendation R2RML for the purpose of this paper. For more information, we refer the reader to the W3C Recommendation [5]. Each R2RML mapping definition consists of one or more triples maps. Looking at Listing 1, we can see that a triples map has (1) one logical table, (2) one subject map and (3) zero or more predicate object maps, where:

1) **Logical Table:** the table or a SQL query from which RDF will be generated.

[3]https://github.com/antidot/db2triples, accessed in September 2017.
[4]http://d2rq.org/d2rq-language, accessed in September 2017
[5]https://github.com/tarql/tarql, accessed in September 2017.
[6]http://ontop.inf.unibz.it, accessed in September 2017.
[7]https://virtuoso.openlinksw.com, accessed in September 2017.
[8]http://openrefine.org/, accessed in September 2017.
[9]http://refine.deri.ie, accessed in September 2017

2) **Subject Map:** subject maps define the subjects of the RDF triples. These subjects can be IRIs or blank nodes. You may also specify zero or more URI class types.
3) **Predicate Object Map:** each predicate object map defines the predicates, using predicate maps, and objects, using object maps, of the RDF triples. Each predicate object map must have at least one predicate map and one object map. Predicates must be valid IRIs. Objects can be IRIs, blank nodes or literals. For literal values, it is possible to define a data type or a language. You may link triples maps using parent triples map. A parent triples map can have zero or more join conditions.

In Listing 1, we map the table (or view) *person*. A triples map defines subjects to have the IRI `http://example.org/person/{id}` and to be instances the class `foaf:Person`. A predicate object map relates the subjects with the predicate `foaf:name` to values in the column *name*.

Listing 1: Example of an R2RML mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<#TripleMap1>
  rr:logicalTable [ rr:tableName "person"; ];
  rr:subjectMap [
    rr:template "http://example.org/person/{id}";
    rr:class foaf:Person; ];
  rr:predicateObjectMap [
    rr:predicate foaf:name;
    rr:objectMap [ rr:column "name"; ]; ];.
```

IV. SML

In this section, we explain the main concepts related to Sparqlification Mapping Language (SML) (see [9] for more information).This mapping language is based on SQL CREATE VIEWS and SPARQL CONSTRUCT queries. An SML mapping is composed of the following parts:

- **Construct:** this clause consists of triple patterns, similar to SPARQL CONSTRUCT queries. These are used as templates for the construction of the triples;
- **From:** is used to define the logical table. As in R2RML, these can be table names or SQL queries;
- **Variable definition:** this clause is used to specify variables whose values are RDF terms from rows of the relation. These variables can be used in the *construct* clause;
- **Constraints:** these are optional constraints about the variables on the RDF level. It is used for query optimization.

Listing 2 shows the SML version of the mapping presented in Listing 1.

Listing 2: Example of an SML mapping

```
Prefix foaf: <http://xmlns.com/foaf/0.1/>

Create View view1 As Construct {
    ?s1  a foaf:Person.
    ?s1   foaf:name   ?o1. }
 With
  ?s1 = uri(concat('http://example.org/person/', ?id
      ))
  ?o1 = plainLiteral(?name)
From person
```

V. JUMA: JIGSAW PUZZLES FOR REPRESENTING MAPPINGS

In previous work, we have presented a method called **J**igsaw p**u**zzles for representing **ma**ppings, Juma[10]. The Juma method in general focuses on facilitating the creation, management, and "understandability" of mappings, by making the technology available to a wider set of stakeholders. In [6], we showed how we applied the approach to the R2RML mapping language, creating an implementation called Juma R2RML. A user evaluation of this implementation suggested that the visual representation was helpful to both expert and non-experts useres alike, and that accurate uplift mappings were created with sufficient quality demonstrated in standard usability tests [4]. Fig 1 shows the mapping presented in Listing 1 represented in Juma R2RML.

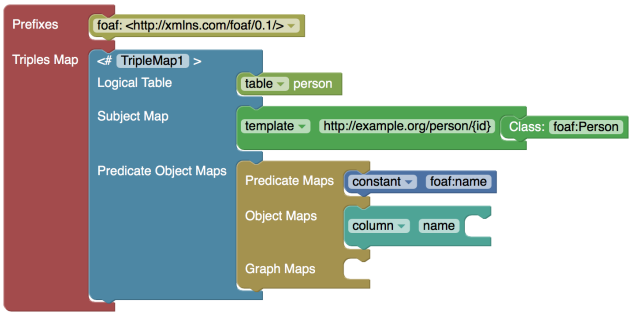

Fig. 1: Juma R2RML visual representation.

This paper proposes a new implementation of the Juma method for uplift mappings, called Juma Uplift[11]. Our new implementation has been developed to support a higher level of abstraction in order to have the capability to generate different mapping languages, as per user choice. The new implementation is also based on Google's Blockly API[12]. Google Blockly is a visual programming language that uses blocks to facilitate code creation. These blocks are shaped like jigsaw puzzle pieces that show how the language works by abstracting the languages syntax. Furthermore, Blockly has been successfully used in many projects, such as code.orgs[13] introduction courses to Computer Science and for the construction of SPARQL queries [8].

The main interface has menu options on the left-hand side and a workspace on the right-hand side. The menu option provide users with all blocks that can be used in our visual representation. From this menu, users can drag new blocks into the workspace. The workspace represents the uplift mapping. The visual representation guides users in the creation of valid

[10]Juma at https://www.scss.tcd.ie/~crottija/juma/
[11]Juma Uplift video at https://www.youtube.com/watch?v=Q97YeZtu_tA
[12]https://developers.google.com/blockly/, accessed in September 2017.
[13]https://code.org/about, accessed in September 2017.

mappings by highlighting and only allowing the connection of blocks that would create a valid mapping. The visual representation also uses colors to identity the type of structure that is being created. For example, subjects use the same colour and so on. Other features of the visual representation are the possibility of zooming in and out, duplication of blocks, disable/enable them (disabled blocks are visible in the workspace but not used in the generation of the mapping) or add comments to them. The menu options (see Fig. 2) are defined as follows:

- **Mapping**: this option defines a block to map an input source (a table, view or a SQL query). Each mapping block maps one input source to one or more subject definition blocks using zero or more vocabularies;
- **Vocabularies**: the vocabularies that are going to be used can be found under this option. There are two types of vocabulary blocks. One with common predefined vocabularies and another customizable one;
- **Subject**: the menu option for subjects define ways of generating the subject, defining it as a blank node, and as instances of classes. Subjects can be defined as templates, constants or columns. Subject blocks have an id (that is used to relate subjects) and are associated with the mapping block, which defines the input source;
- **Predicate/Object**: the block to define predicate and objects is defined under this menu option. Just like subjects, predicates and objects can be defined using using templates, constants or columns. Objects can also be defined as IRI's, blank nodes, literals, to have a datatype or a language tag. These blocks are associated with a subject definition block. Each predicate/object block defines a new triple for the associated subject;
- **Linking**: this option defines a block to link subjects defined within a mapping through their ids. This block also allows users to define how the subjects being linked are related. This relation translates to SQL joins in the mapping. Linking blocks can only be associated to subject ones;
- **Functions**: under this menu option, we have built-in functions. A few examples of functions available are concatenation, replace and summation. The loading of custom functions is also available. Moreover, others can be added in future implementations. Functions can be used to generate subjects, predicates or objects;
- **Graph**: this menu option defines an optional block to generate the RDF triples in a specific named graph (these can also be templates, constants or columns). Graph blocks are associated to subjects, defining in which named graph the triples associated to the subject will be generated.

To be compatible with the R2RML mapping language, we have decided to design the blocks for subjects, predicates, objects and graphs using the R2RML constructs types: templates, constants and columns. Templates are string templates. These can also refer to columns using curly brackets. For example, the template `http://example.org/person/{id}` refers to the column *id* of the input source being mapped. Constants are values that do not change; e.g. a property such as `foaf:name` or a string value ẗome value". Columns refer to attributes of the data source being mapped; e.g. *id* or *name*.

The reason for supporting R2RML (an RDF-based mapping language) being that it is a W3C Recommendation with many implementations available. We also support the SPARQL-based SML mapping language. SML has support for SPARQL to SQL query rewriting and has been used in many projects such as LinkedGeoData[14] and PanLex: RDF[15]. Another reason for supporting R2RML and SML is to demonstrate that the visual representation is expressive enough to support syntactically distinct mapping languages.

The support for functions within Juma Uplift is made available through R2RML-F [11]. R2RML-F defines an extension to the R2RML vocabulary to define functions as resources within the mapping. This R2RML-F implementation defines functions using JavaScript. In this sense, new functions can be added to Juma together with its JavaScript implementation. All functions defined in Juma are supported by this R2RML-F implementation. SML only has support for the concatenation function, however, the language is extensible to more functions.

In our tool, the interface used to create/edit mappings has 4 tabs (see Fig. 2). In the first tab, Mapping, we show the Juma visual representation. In Configuration, one can define the properties of the configuration file. The configuration file is used as input to an R2RML processor together with the R2RML mapping file. SML configurations are set through the command line. In the R2RML-Mapping tab, the user can see the R2RML mapping generated from the visual representation. In SML-Mapping tab, the same mapping is represented using the SML mapping language. These show a live view of the mapping. In other words, users can see changes in the mapping as blocks are being dragged and dropped. Nonetheless, disconnected blocks are not used to generate the mappings since these would result in an invalid mapping.

Fig 2 shows the same mapping in Juma Uplift implementation as that shown in Fig 1 Juma R2RML implementation. In this example, we are mapping the table (or view) "person" to the vocabulary FOAF[16]. The subjects of the triples are defined as a string template with the URI `http://example.org/person/{id}`. The block *with classes* define the triples as instances of `foaf:Person`. The mapping also defines a predicate to have the constant value `foaf:name` related to the column value from *name*.

[14]http://linkedgeodata.org, accessed in September 2017.
[15]http://ld.panlex.org/rdf.html, accessed in September 2017.
[16]http://xmlns.com/foaf/0.1/, accessed in September 2017.

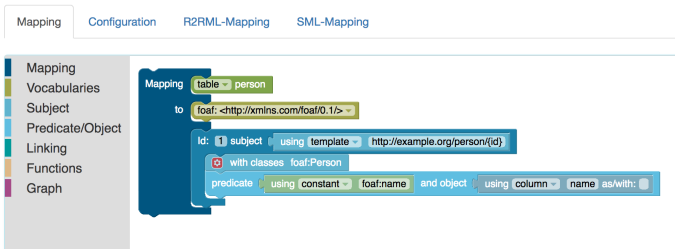

Fig. 2: Juma Uplift visual representation.

The SML engine and some R2RML implementations have support for CSV files. In this sense, the mappings created using Juma Uplift implementation can also be used to uplift CSV files as well as relational databases.

VI. Comparison of Juma Implementations

In this section, we will compare essential features of Juma R2RML and Juma Uplift.

The fundamental block of Juma R2RML relates prefixes (or vocabularies being used in the mapping) to triples maps. Juma Uplift's main block relates logical tables and vocabularies.

Fig. 3 shows a side-by-side mapping of both implementations representing the same mapping. This mapping converts two tables into RDF. The mapping is using the vocabularies FOAF and a fictional one (http://example.org/). The table person is generating subjects with the IRI `http://example.org/person/{id}`. This table is also being mapped as instances of the class `foaf:Person`. The mapping also defines the predicate `foaf:name` to be related to values of the column *name* from the logical table *person*. The predicate `foaf:based_near` is used to related these subjects to the subjects generated from the logical table *city*. The table *city* is mapped to have the IRI `http://example.org/city/{id'}` as subjects. These subjects are instances of the class `ex:City` with the predicate `foaf:name` related to objects from the column *name* of the logical table *city*.

A. Defining logical tables

The definition of logical tables is similar in both implementations. Users can define the logical table as a table (or view) or a SQL query. In Juma R2RML the logical table is related to a triples map. Juma Uplift defines the logical table as part of the mapping block. In Juma Uplift, it is possible to generate multiple subjects from the same logical table. To do the same in Juma R2RML, one needs to define a new triples map with a new subject map from the same logical table.

B. Defining the vocabularies

In Juma R2RML, vocabularies are defined in the main block, separated from triples maps. Juma Uplift defines vocabularies to each mapping block. In Juma Uplift, users map a logical table to vocabularies.

C. Creating RDF terms

Both implementations rely on R2RML's constructs for defining RDF terms from logical tables (templates, constants and columns). Juma R2RML uses distinct blocks for the definition of blank nodes, datatypes, languages and so on. In Juma Uplift this is contained within the RDF term block using the option *as/with* for the objects of the triples. Users can relate subjects to a block defining it as a blank node.

D. Forming triples

Juma R2RML uses subject maps to specify how to generate the subject. To define it as a blank node or to be instances of classes, users must related subject maps to its respective blocks. One for the definition of its type, and one other per class being defined. Juma Uplift has one block to define subjects as a blank node and one other for the definition of classes.

Juma R2RML relies on predicate object maps to define predicates and objects. Predicates and objects are defined using predicate object maps. Each predicate object map must define at least one predicate map and one object map. For example, if a predicate object map has two predicate maps and one object map then the RDF output will relate these two predicates to the same object. However, users need to understand the R2RML algorithm to be able to define this construct. For this reason, Juma Uplift defines one block that relates predicates and objects. This block only allows one predicate and one object definition. To relate two predicates to the same object, two predicate object blocks must be defined.

E. Functions

Juma Uplift has support for data transformation functions, while Juma R2RML does not. As mentioned before, functions are defined using R2RML-F's vocabulary. The implementation offers some built-in functions while still being extensible to others.

F. Linking

Both implementations allow for the definition of linking. As mentioned before, this allows users to relate subjects within a mapping. The underlying mapping uses SQL joins for the definition of these links. Juma R2RML uses parent triples map and join conditions (relating parent and child columns). Juma Uplift has a new block to define these links. Users can select the mapping using their id. The definition of parent and child columns is done with new labels: *from this table* or *from selected table*. We believe that these will help users in the definition of these SQL joins. It is worth noting that R2RML offers a model for the expressions of joins (parent triples map) while SML does not. To allow for the definition of linking using SML, Juma Uplift applies the R2RML algorithm and generates a corresponding SQL query in the SML mapping.

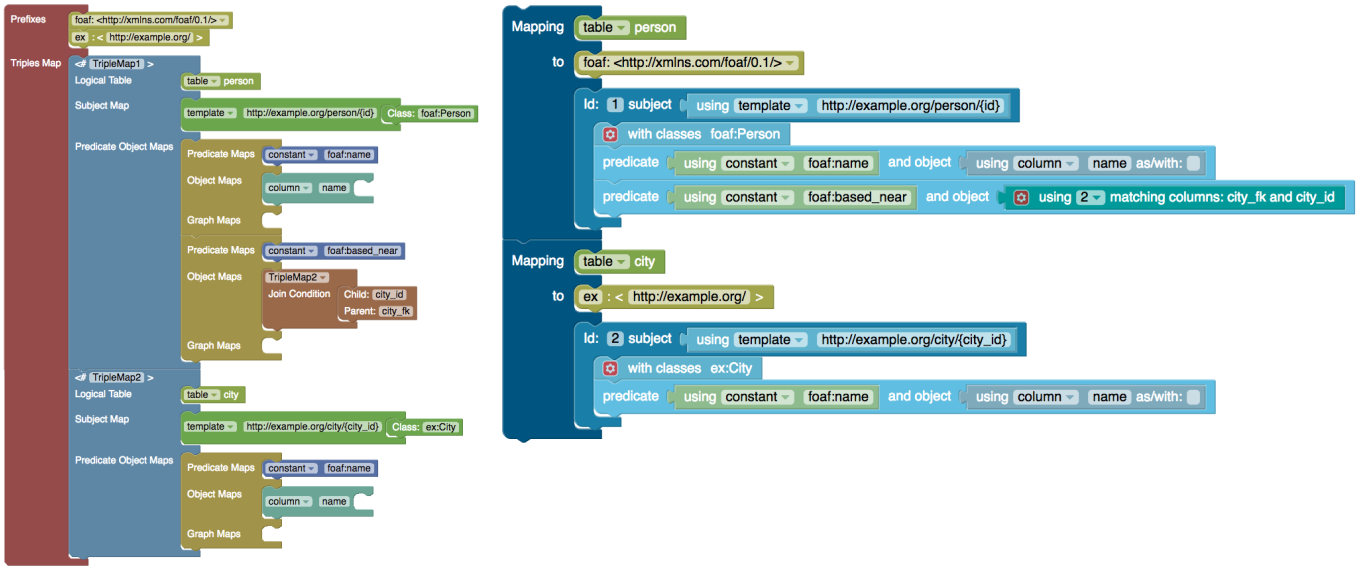

Fig. 3: Juma R2RML and Juma Uplift side-by-side.

G. Assigning Triples to Named Graphs

In Juma R2RML users can define in which named graph the triples will be generated for subjects and predicate object maps. Juma Uplift allows for the definition of named graphs for subjects only. We argue that this is more intuitive to users and for specific cases different mapping definitions can be used to generate triples in distinct named graphs.

VII. Evaluation

This section describes the evaluation of the expressiveness of Juma Uplift.

A. Hypothesis

The hypothesis related to this evaluation is: *It is possible to apply Juma Uplift to generate accurate mappings for the mapping languages R2RML and SML for common uplift use cases.*

B. Mapping use cases

For this experiment, we have defined 10 uplift mapping use cases. These use cases were based on the RDB mapping patterns presented in [26]. In this sense, these mappings were defined to cover common cases when uplifting data into RDF and also to explore the visual representation.

C. Database

We have created a relational database with two tables for this experiment (Tables II and III). Based on the mapping use cases presented we defined the expected RDF output.

TABLE II: Table person diagram

Person			
id	**name**	**age**	**city_fk**
1	Ana	29	100
2	John	25	100
3	Mary	30	200

TABLE I: Mapping use cases

Mapping	Description
#1	Mapping one table to one class
#2	Mapping one table to two classes
#3	Mapping two tables to one class each
#4	Mapping one table and one attribute
#5	Mapping one table and two attributes
#6	Mapping one table and one attribute with a language tag
#7	Mapping one table and one attribute as a resource
#8	Mapping two tables and one attribute each
#9	Mapping one table and one attribute as a data transformation function
#10	Mapping two tables with one class and one attribute each and a link between them

TABLE III: Table city diagram

City	
id	**name**
100	Dublin
200	London

D. Method

For each use case a mapping was created by the authors of this paper using Juma Uplift. The R2RML and SML mappings generated by the visual representation were then executed[17]. The generated RDF datasets were compared to an expected RDF output (based on the mapping use cases) using Jena[18]. The full experiment data is available[19].

E. Example: Mapping use case #10

Mapping use case #10 covers mapping of tables to classes and attributes with a link between them. The vocabularies used are FOAF and a fictional one (http://example.org/). Listing 3

[17]SML engine available at https://github.com/AKSW/Sparqlify. R2RML engine available at https://opengogs.adaptcentre.ie/debruync/r2rml.

[18]https://jena.apache.org/, accessed in September 2017.

[19]https://www.scss.tcd.ie/~crottija/juma-uplift/experiment-data

shows the expected RDF output. This use case represented in Juma R2RML and Juma Uplift can be seen in Fig. 3. Listing 4 and 5 show the corresponding R2RML and SML mappings.

Listing 3: RDF output of use case mapping #10

```
<http://example.org/person/1>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://example.org/city/100> ;
        <http://xmlns.com/foaf/0.1/name> "Ana" .
<http://example.org/person/2>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://example.org/city/100> ;
        <http://xmlns.com/foaf/0.1/name> "John" .
<http://example.org/person/3>
        a       <http://xmlns.com/foaf/0.1/Person> ;
        <http://xmlns.com/foaf/0.1/based_near>
                <http://example.org/city/200> ;
        <http://xmlns.com/foaf/0.1/name> "Mary" .
<http://example.org/city/100>
        a       <http://example.org/City> ;
        <http://xmlns.com/foaf/0.1/name> "Dublin" .
<http://example.org/city/200>
        a       <http://example.org/City> ;
        <http://xmlns.com/foaf/0.1/name> "London" .
```

Listing 4: R2RML use case mapping #10

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://example.org/> .

<#TriplesMap1>
  rr:logicalTable [ rr:tableName "person";];
  rr:subjectMap [
    rr:template "http://example.org/person/{id}";
    rr:class foaf:Person; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant foaf:name;];
    rr:objectMap [rr:column "name";]; ];
  rr:predicateObjectMap [
    rr:predicateMap [rr:constant foaf:based_near;];
    rr:objectMap [
      rr:parentTriplesMap <#TriplesMap2>;
      rr:joinCondition [
        rr:child "city_id";
        rr:parent "city_fk";
    ];]; ];.

<#TriplesMap2>
  rr:logicalTable [ rr:tableName "city"; ];
  rr:subjectMap [
    rr:template "http://example.org/city/{city_id}";
    rr:class ex:City; ];
  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant foaf:name;];
    rr:objectMap [rr:column "name";]; ];.
```

Listing 5: SML use case mapping #10

```
Prefix foaf: <http://xmlns.com/foaf/0.1/>
Prefix ex: <http://example.org/>

Create View view1 As Construct {
  ?s1  a foaf:Person.
  ?s1   foaf:name   ?o1. }
 With
  ?s1 = uri(concat('http://example.org/person/', ?id
      ))
  ?o1 = plainLiteral(?name)
 From person

Create View view2 As Construct {
    ?s1  a ex:City.
    ?s1   foaf:name   ?o1. }
 With
  ?s1 = uri(concat('http://example.org/city/', ?
      city_id))
  ?o1 = plainLiteral(?name)
 From city

Create View view3 As Construct {
  ?s1 foaf:based_near ?o1 . }
 With
 ?s1 = uri(concat('http://example.org/person/', ?id)
     )
 ?o1 = uri(concat('http://example.org/city/', ?
     city_id))
From
 [[  SELECT * FROM city AS child, person AS parent
     WHERE child.city_id = parent.city_fk ]]
```

F. Results and Analysis

It was found that Juma Uplift was able to create R2RML and SML mappings that generates the expected RDF output. This confirms our hypothesis that Juma is expressive enough to generate accurate R2RML and SML mappings for the designed use cases.

VIII. Conclusions and Future Work

In this paper, we have presented a new implementation of the Juma method, a visual representation based on the block metaphor, called Juma Uplift. This implementation has support for distinct mapping languages. We have decided to support the RDF-based W3C Recommendation R2RML and the SPARQL-based mapping language SML.

Through an experiment, we have defined common use cases when uplifting data from relational databases into RDF. For every use case a mapping was created using Juma Uplift. The visual representation automatically generates R2RML and SML mappings. These mappings were then executed using their respective uplift engines and compared to an expected RDF output. The results showed that Juma was able to create mappings that generate the expected RDF output. This confirmed our hypothesis that Juma Uplift implementation has the capability to generate accurate mappings for the R2RML and SML mapping languages.

The success of this initial evaluation has provided us with confidence that Juma Uplift is extensible to other mapping languages. The current implementation supports R2RML, an RDF-based mapping language. In this sense, a possible extension to undertake would be to support RML. RML supports heterogeneous input formats, such as XML and JSON. Juma Uplift also supports SML, a SPARQL-based mapping language, therefore, we believe that it should be possible to support SPARQL-Generate. Which is another SPARQL-based mapping language with support for various data formats.

Future development includes implementing the loading of mappings, extending the support for functions (for example, use of the Function Ontology [27]). We also plan on applying the Juma method in the representation of ontology mappings and interlinking of Linked Data.

Future evaluation includes an extensive user experiment to validate the usability and usefulness of Juma Uplift. As part of this evaluation we intend to assess and compare the cognitive load of our approach with others.

ACKNOWLEDGEMENTS

This paper was supported by CNPQ, National Counsel of Technological and Scientific Development Brazil and by the Science Foundation Ireland (Grant 13/RC/2106) as part of the ADAPT Centre for Digital Content Technology (http://www.adaptcentre.ie/) at Trinity College Dublin.

REFERENCES

[1] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[2] P. Heyvaert, A. Dimou, A. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, and R. V. de Walle, "RMLEditor: A graph-based mapping editor for linked data mappings," in *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, 2016, pp. 709–723. [Online]. Available: https://doi.org/10.1007/978-3-319-34129-3_43

[3] C. Pinkel, C. Binnig, P. Haase, C. Martin, K. Sengupta, and J. Trame, "How to best find a partner? an evaluation of editing approaches to construct R2RML mappings," in *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, 2014, pp. 675–690. [Online]. Available: https://doi.org/10.1007/978-3-319-07443-6_45

[4] A. C. Junior, C. Debruyne, and D. O'Sullivan, "Using a Block Metaphor for Representing R2RML Mappings," in *Proceedings of the Third International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 16th International Semantic Web Conference, VOILA@ISWC 2017, Vienna, Austria, October 22, 2017.*, 2017, pp. 1–12. [Online]. Available: http://ceur-ws.org/Vol-1947/paper01.pdf

[5] S. Das, S. Sundara, and R. Cyganiak, "R2RML: RDB to RDF mapping language," 2012.

[6] A. C. Junior, C. Debruyne, and D. O'Sullivan, "Juma: an Editor that Uses a Block Metaphor to Facilitate the Creation and Editing of R2RML Mappings," in *The Semantic Web - ESWC 2017 Satellite Events, Portotoz, Slovenia, May 28 - June 1, 2017*, 2017.

[7] N. Fraser, "Google Blockly: A visual programming editor," 2014.

[8] P. Bottoni and M. Ceriani, "SPARQL playground: A block programming tool to experiment with SPARQL," in *Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 11, 2015.*, 2015, p. 103. [Online]. Available: http://ceur-ws.org/Vol-1456/paper12.pdf

[9] C. Stadler, J. Unbehauen, P. Westphal, M. A. Sherif, and J. Lehmann, "Simplified RDB2RDF mapping," in *Proceedings of the Workshop on Linked Data on the Web, LDOW 2015, co-located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, May 19th, 2015.*, 2015. [Online]. Available: http://ceur-ws.org/Vol-1409/paper-09.pdf

[10] F. Priyatna, Ó. Corcho, and J. F. Sequeda, "Formalisation and experiences of r2rml-based SPARQL to SQL query translation using morph," in *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, 2014, pp. 479–490. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567981

[11] C. Debruyne and D. O'Sullivan, "R2RML-F: Towards Sharing and Executing Domain Logic in R2RML Mappings," in *Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016)*, 2016. [Online]. Available: http://ceur-ws.org/Vol-1593/article-13.pdf

[12] A. Dimou, M. V. Sande, J. Slepicka, P. A. Szekely, E. Mannens, C. A. Knoblock, and R. V. de Walle, "Mapping hierarchical sources into RDF using the RML mapping language," in *2014 IEEE International Conference on Semantic Computing, Newport Beach, CA, USA, June 16-18, 2014*, 2014, pp. 151–158. [Online]. Available: https://doi.org/10.1109/ICSC.2014.25

[13] F. Michel, L. Djimenou, C. Faron-Zucker, and J. Montagnat, "Translation of relational and non-relational databases into RDF with xr2rml," in *WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015*, 2015, pp. 443–454. [Online]. Available: https://doi.org/10.5220/0005448304430454

[14] J. F. Sequeda and D. P. Miranker, "Ultrawrap: SPARQL execution on relational data," *J. Web Sem.*, vol. 22, pp. 19–39, 2013. [Online]. Available: https://doi.org/10.1016/j.websem.2013.08.002

[15] M. Lefrançois, A. Zimmermann, and N. Bakerally, "A SPARQL extension for generating RDF from heterogeneous formats," in *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, 2017, pp. 35–50. [Online]. Available: https://doi.org/10.1007/978-3-319-58068-5_3

[16] S. Bischof, S. Decker, T. Krennwallner, N. Lopes, and A. Polleres, "Mapping between RDF and XML with XSPARQL," *J. Data Semantics*, vol. 1, no. 3, pp. 147–185, 2012. [Online]. Available: https://doi.org/10.1007/s13740-012-0008-7

[17] K. Sengupta, P. Haase, M. Schmidt, and P. Hitzler, "Editing R2RML mappings made easy," in *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013*, 2013, pp. 101–104. [Online]. Available: http://ceur-ws.org/Vol-1035/iswc2013_demo_26.pdf

[18] M. Rodriguez-Muro, J. Hardi, and D. Calvanese, "Quest: Effcient sparql-to-sql for RDF and OWL," in *Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA, November 11-15, 2012*, 2012. [Online]. Available: http://ceur-ws.org/Vol-914/paper_59.pdf

[19] N. F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen, "Creating semantic web contents with protégé-2000," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 60–71, 2001. [Online]. Available: https://doi.org/10.1109/5254.920601

[20] L. E. T. Neto, V. M. P. Vidal, M. A. Casanova, and J. M. Monteiro, "R2RML by assertion: A semi-automatic tool for generating customised R2RML mappings," in *The Semantic Web: ESWC 2013 Satellite Events - ESWC 2013 Satellite Events, Montpellier, France, May 26-30, 2013, Revised Selected Papers*, 2013, pp. 248–252. [Online]. Available: https://doi.org/10.1007/978-3-642-41242-4_33

[21] F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, S. Villata, B. Bucher, F. Hamdi, L. Bihanic, G. Képéklian, F. Cotton *et al.*, "Enabling linked-data publication with the datalift platform," 2012.

[22] C. A. Knoblock, P. A. Szekely, J. L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan, and P. Mallick, "Semi-automatically mapping structured sources into the semantic web," in *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, 2012, pp. 375–390. [Online]. Available: https://doi.org/10.1007/978-3-642-30284-8_32

[23] D. Lembo, R. Rosati, M. Ruzzi, D. F. Savo, and E. Tocci, "Visualization and management of mappings in ontology-based data access (progress report)," in *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, 2014, pp. 595–607. [Online]. Available: http://ceur-ws.org/Vol-1193/paper_77.pdf

[24] Á. Sicilia, G. Nemirovski, and A. Nolle, "Map-On: A web-based editor for visual ontology mapping," *Semantic Web*, vol. 8, no. 6, pp. 969–980, 2017. [Online]. Available: https://doi.org/10.3233/SW-160246

[25] M. Blinkiewicz and J. Bak, "SQuaRE: A visual support for OBDA approach," in *Proceedings of the Second International Workshop on Visualization and Interaction for Ontologies and Linked Data co-located with the 15th International Semantic Web Conference, VOILA@ISWC 2016, Kobe, Japan, October 17, 2016.*, 2016, pp. 41–53. [Online]. Available: http://ceur-ws.org/Vol-1704/paper4.pdf

[26] J. F. Sequeda, F. Priyatna, and B. Villazón-Terrazas, "Relational database to RDF mapping patterns," in *Proceedings of the 3rd Workshop on Ontology Patterns, Boston, USA, November 12, 2012*, 2012. [Online]. Available: http://ceur-ws.org/Vol-929/paper9.pdf

[27] B. D. Meester, A. Dimou, R. Verborgh, and E. Mannens, "An ontology to semantically declare and describe functions," in *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, 2016, pp. 46–49. [Online]. Available: https://doi.org/10.1007/978-3-319-47602-5_10