Improving Effectiveness of Knowledge Graph Generation Rule Creation and Execution

Pieter Heyvaert

PhD advisors: Ruben Verborgh and Anastasia Dimou

GHENT UNIVERSITY

IDLab
INTERNET & DATA LAB

imec

More and more devices and applications are connected to the Internet

Devices





More and more devices and applications are connected to the Internet

Devices



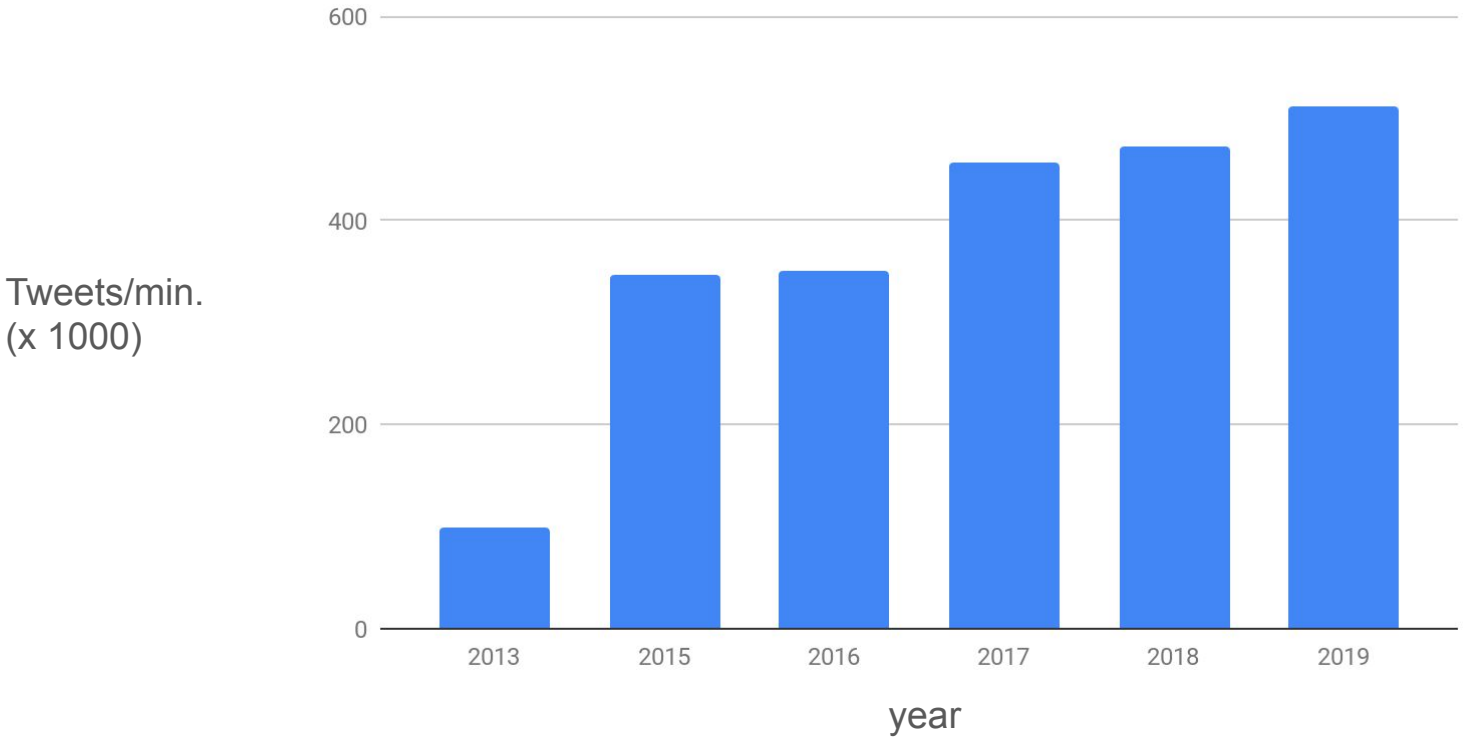


Applications





More and more data is generated



New techniques deployed using this data on the Web

Data can be analyzed, combined and shared.

→ Powerful, new techniques can be designed and deployed on the Web.

Examples are

- Artificial intelligence used by personal assistants (Siri, Alexa)
- Improved search engines (Google, Bing)

Overview

Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

Overview

Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

Original data representations on the Web are insufficient to exchange and reuse data

Originally data on the Web is represented using different data formats.

Extra data can be added without saying what this data means.

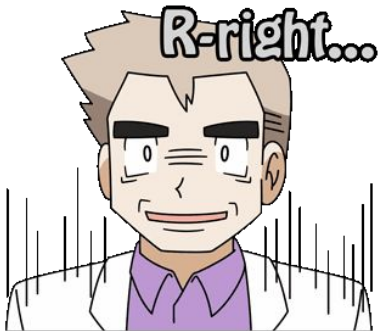
→ Makes it hard to exchange and reuse data on the Web.

Original data representations on the Web are insufficient to exchange and reuse data

Originally data on the Web is represented using different data formats.

Extra data can be added without saying what this data means.

→ Makes it hard to exchange and reuse data on the Web.

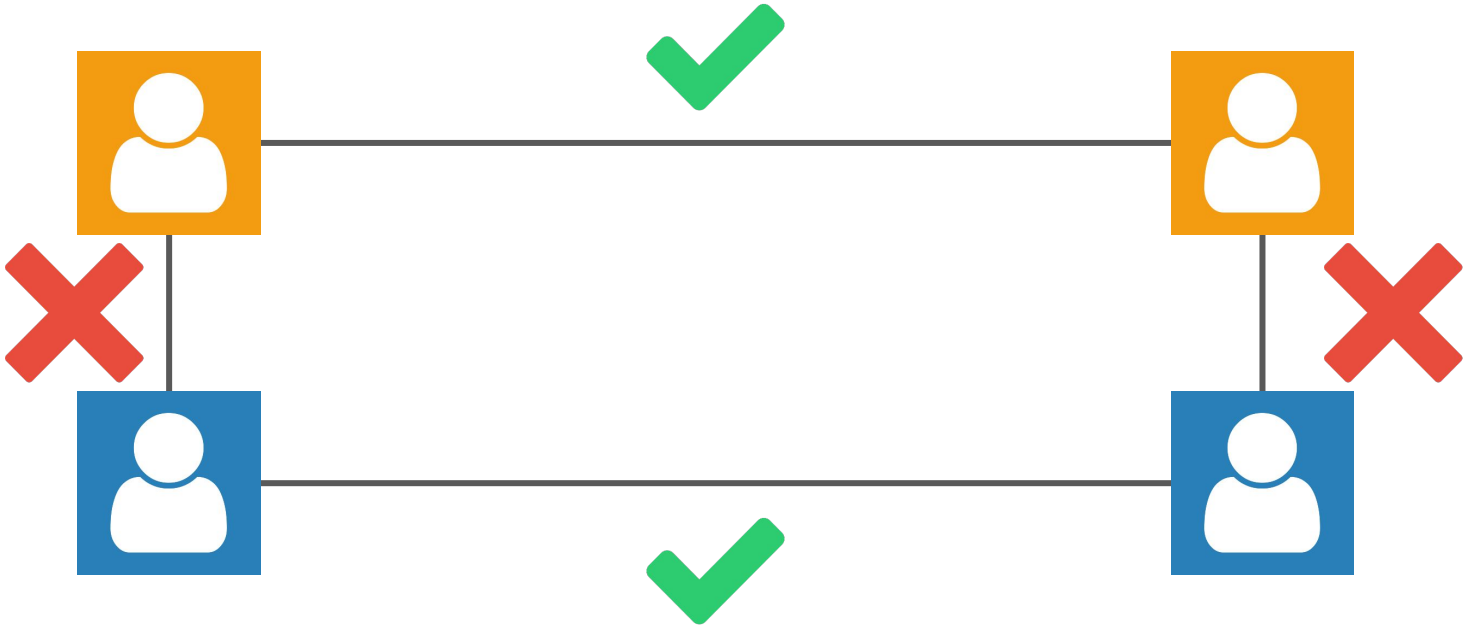


Example: able to share data between same contacts apps

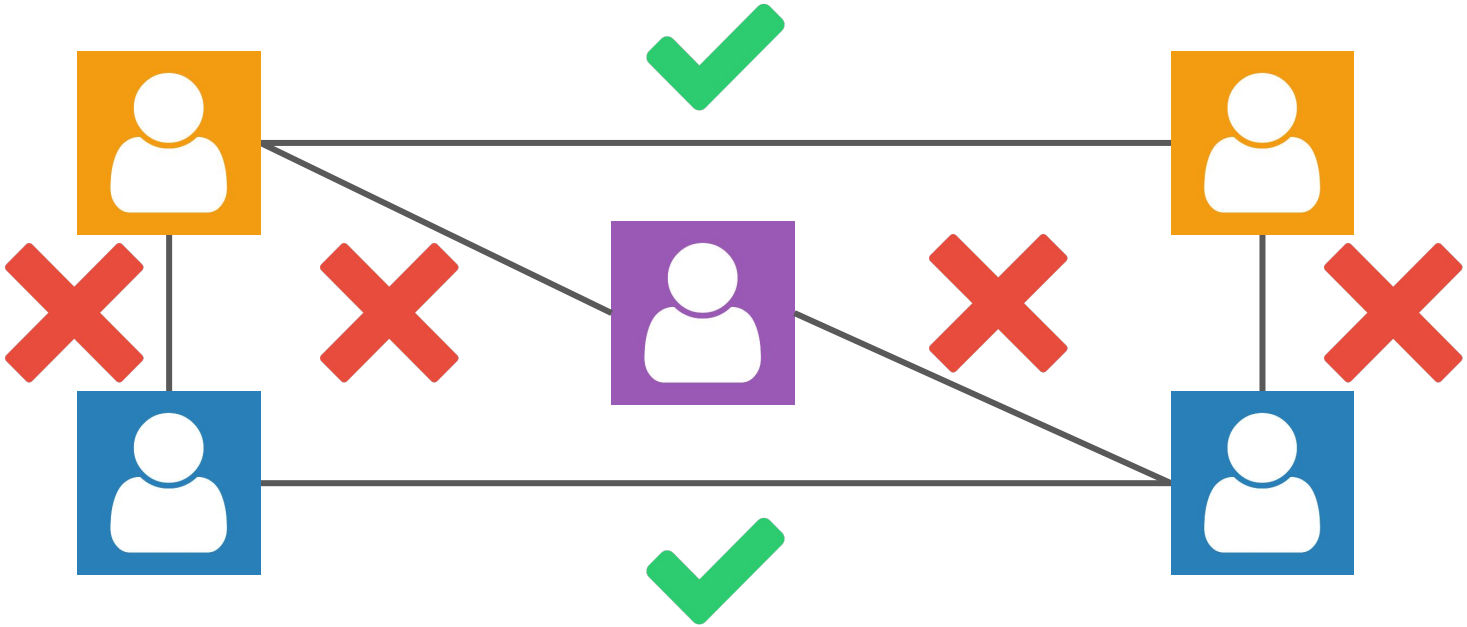
Example: able to share data between same contacts apps

Example: able to share data between same contacts apps

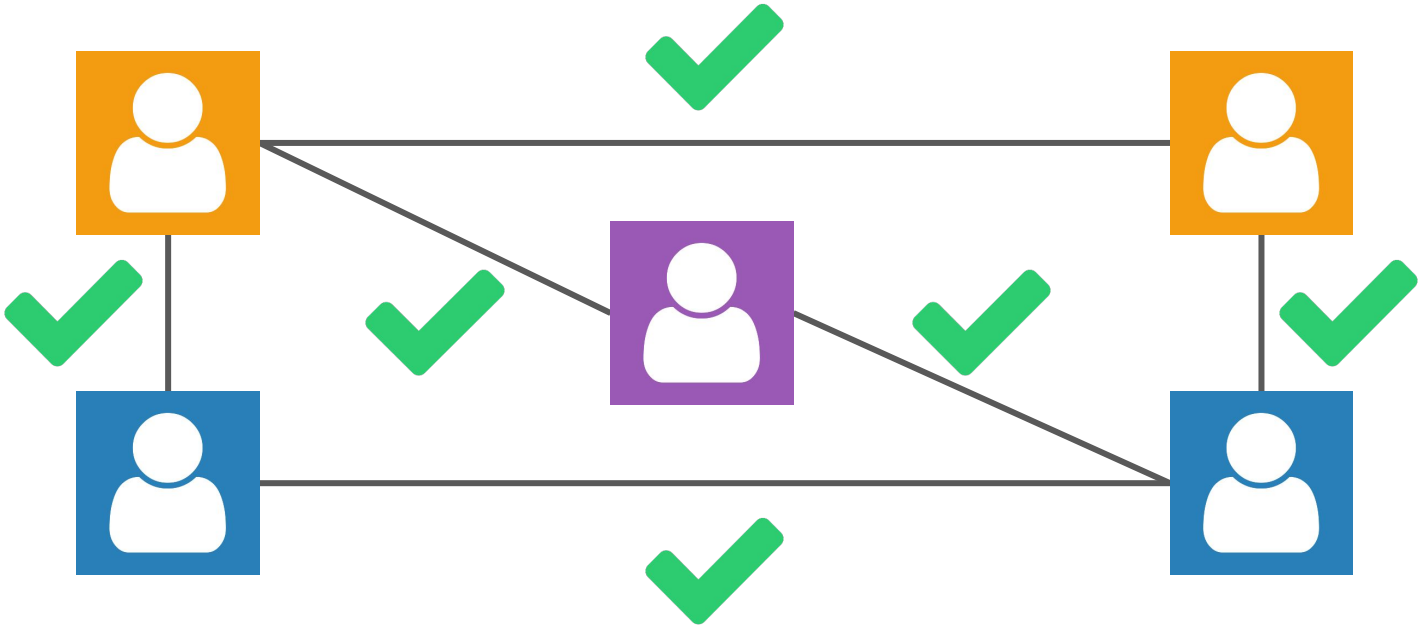
Example: not able to share data between different contacts apps



Example: not able to share data between different contacts apps



Semantic Web allows to easily exchange and reuse data



Overview

Moving from Web to Semantic Web

- **Knowledge graphs**
- Knowledge graph generation
- Knowledge graph inconsistencies
- Processors

Challenges

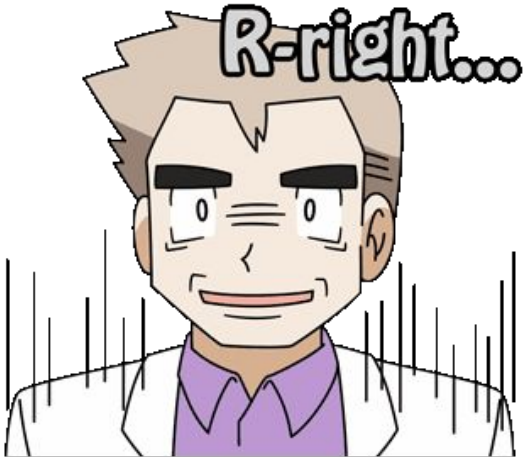
Contributions

The main Semantic Web technology is knowledge graphs

Knowledge graphs = graphs that provide semantic descriptions of entities and their relationships.

The main Semantic Web technology is knowledge graphs

Knowledge graphs = graphs that provide semantic descriptions of entities and their relationships.



Example: knowledge graph

Example: data about a person

has the name **Professor Oak**

Example: data about a person

This is already a graph.

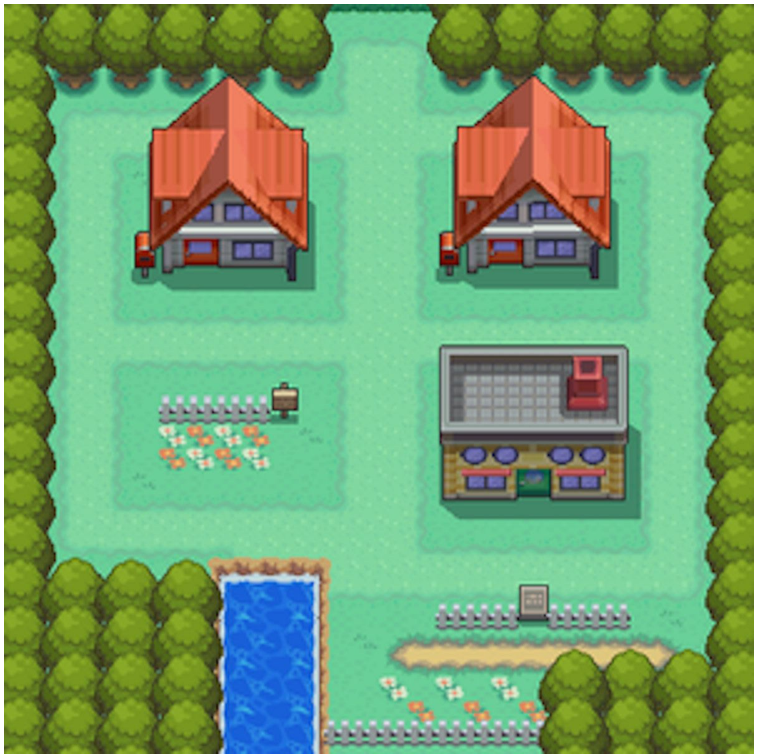
Example: data about a person

name — **Professor Oak**

location — 

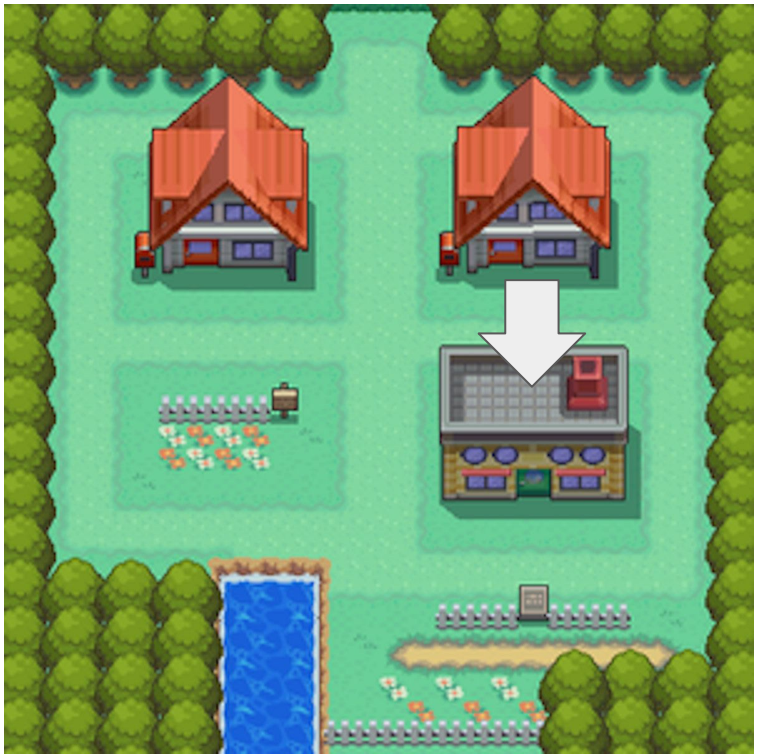
Example: data about his location

Pallet Town



Example: data about his location

Pallet Town



Laboratory

Example:combining data about person and location

name

Professor Oak

location

city

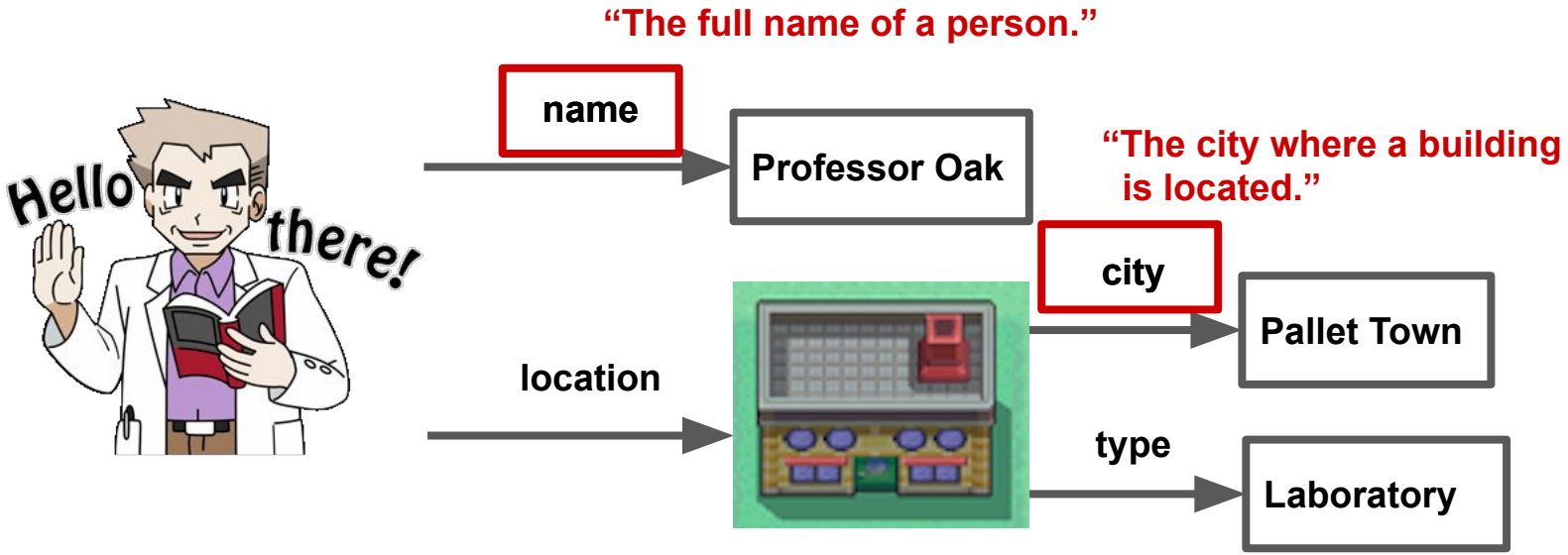
Pallet Town

type

Laboratory

Knowledge graph provides semantic descriptions

Semantic descriptions give the data meaning



The main Semantic Web technology is knowledge graphs

Knowledge graphs = **graphs** that provide **semantic descriptions** of entities and their relationships.

Overview

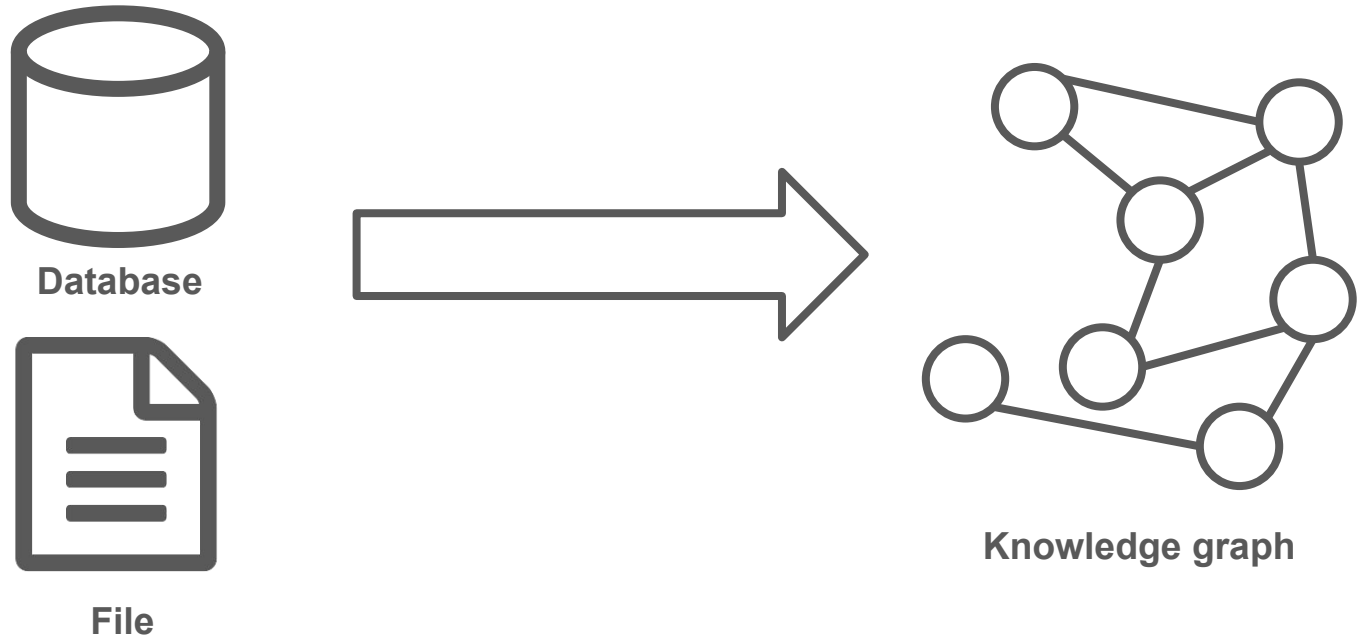
Moving from Web to Semantic Web

- Knowledge graphs
- **Knowledge graph generation**
- Knowledge graph inconsistencies
- Processors

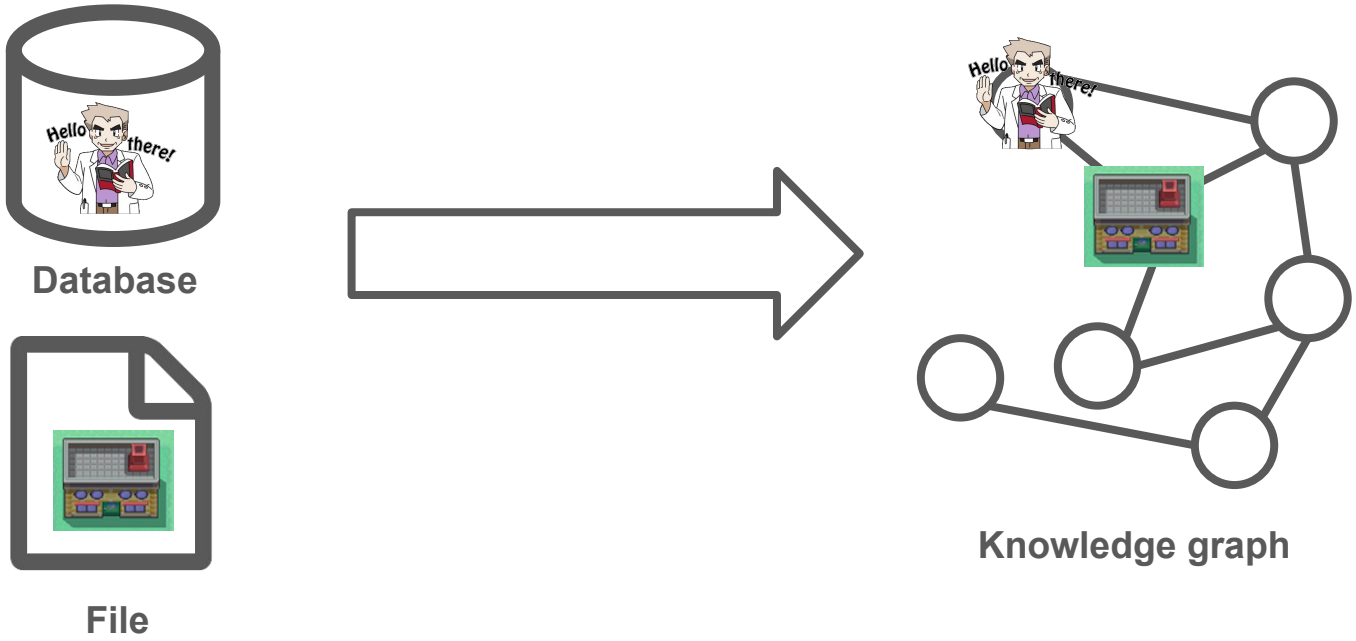
Challenges

Contributions

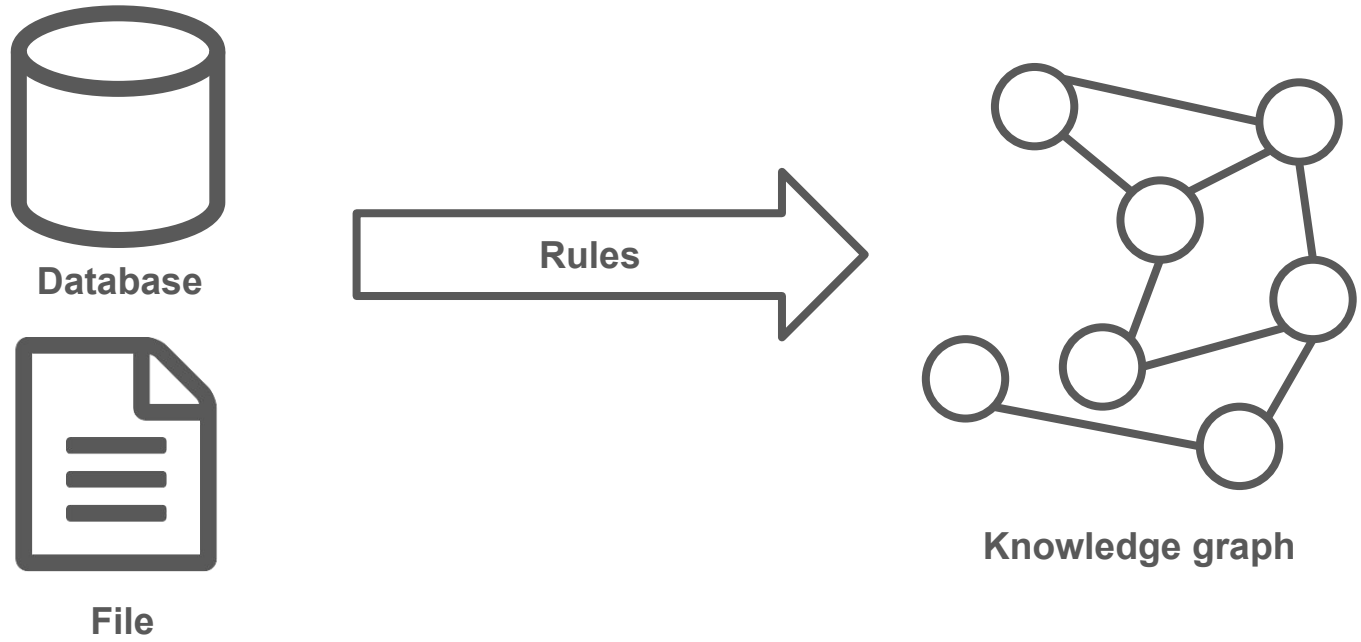
Knowledge graphs are often generated from other data sources



Example: knowledge graphs are often generated from other data sources



Knowledge graphs are often generated via rules



Rules add semantic descriptions to data

Example: rules for a person

id	name	place
oak	Professor Oak	lab
ash	Ash Ketchum	h1

Every row in the table is a person.

“name” is the official full name of a person.

“place” refers to a person’s current location.

Example: rules for a location

id	category	city
lab	laboratory	Pallet Town
h1	house	Pallet Town
h2	house	Pallet Town

Every row in the table is a location.

“category” is the type of a location.

“city” refers to a location’s city it is in.

Example: executing rules results in knowledge graph

id	name	place
oak	Professor Oak	lab
ash	Ash Ketchum	h1

id	category	city
lab	laboratory	Pallet Town
h1	house	Pallet Town
h2	house	Pallet Town

Every row in the table is a person.

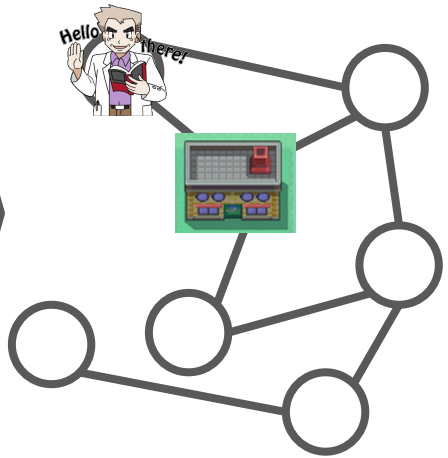
“name” is the offcial full name of a person.

“place” refers to a person’s location in other table.

Every row in the table is a location.

“category” is the type of a location.

“city” refers to a location’s city it is in.



Knowledge graph

Syntax and grammar of rules defined by knowledge graph generation language

Syntax and grammar: how to create rules correctly.

Similar to natural languages, such as English and Dutch.

Examples of knowledge graph generation languages:

R2RML: W3C recommendation for databases.

RML: Extension of R2RML for multiple data sources in different formats.

Overview

Moving from Web to Semantic Web

- Knowledge graphs
- Knowledge graph generation
- **Knowledge graph inconsistencies**
- Processors

Challenges

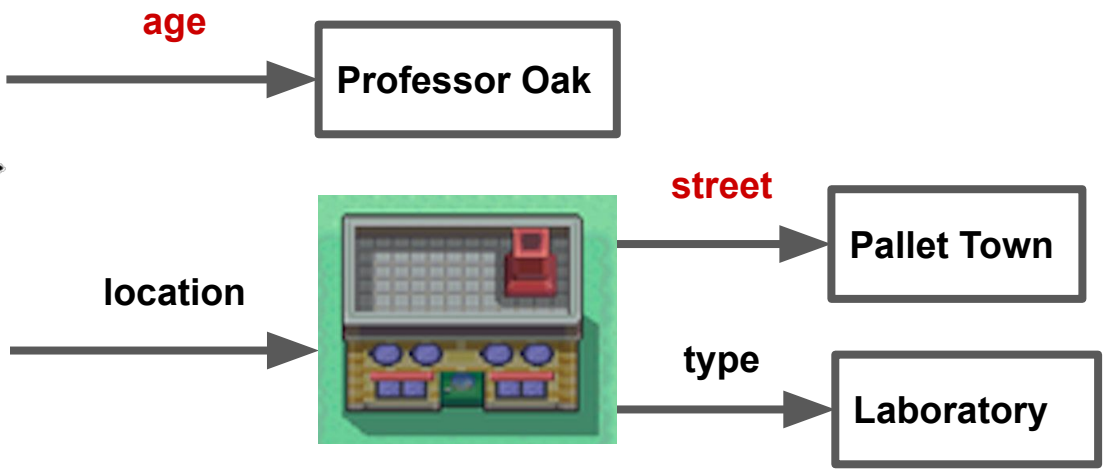
Contributions

Knowledge graphs can contain two types of inconsistencies

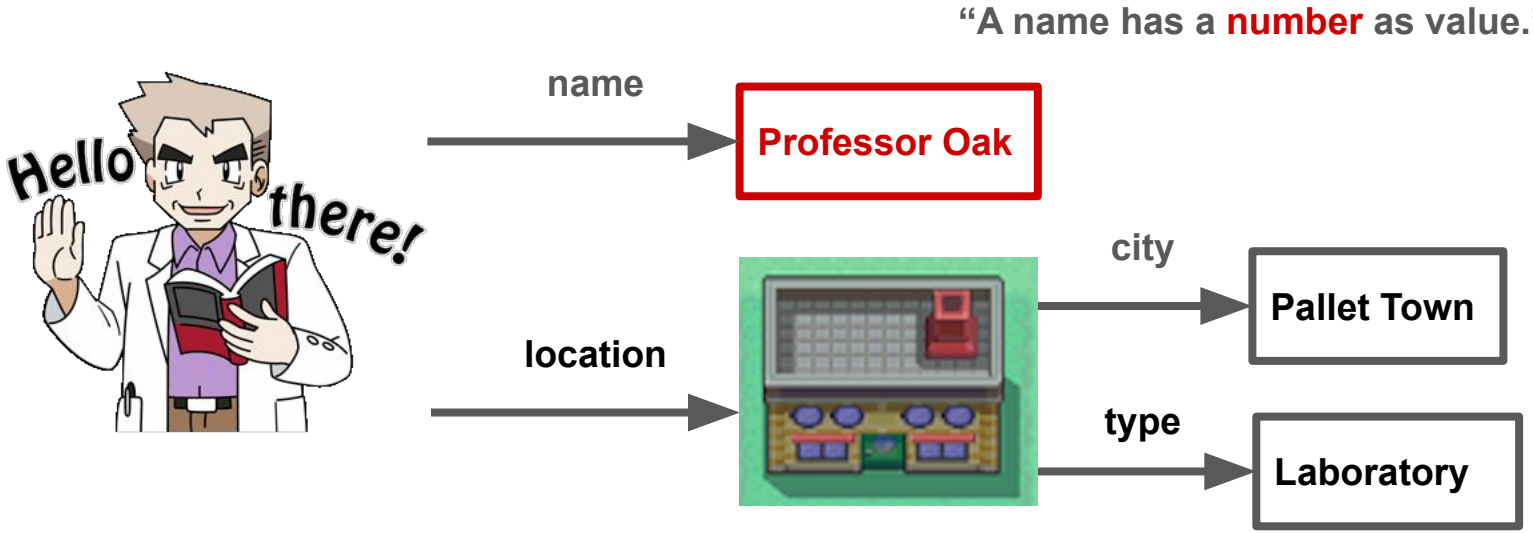
Not using the suitable concepts and relationships.

Concepts and relationships do not model the real world as desired.

Example: not using suitable concepts and relationships



Example: concepts and relationships do not model real world as desired



Possible causes for inconsistencies

Rules: wrong concepts and relationships are used.

For example, using the relationship “age” instead of “name”.

Definitions of concepts and relationships: incorrectly model real world.

For example, a name only consists of numbers.

Inconsistencies are fixed by updating the rules, definitions or both of them.

Overview

Moving from Web to Semantic Web

- Knowledge graphs
- Knowledge graph generation
- Knowledge graph inconsistencies
- **Processors**

Challenges

Contributions

Processor executes rules to generate knowledge graph

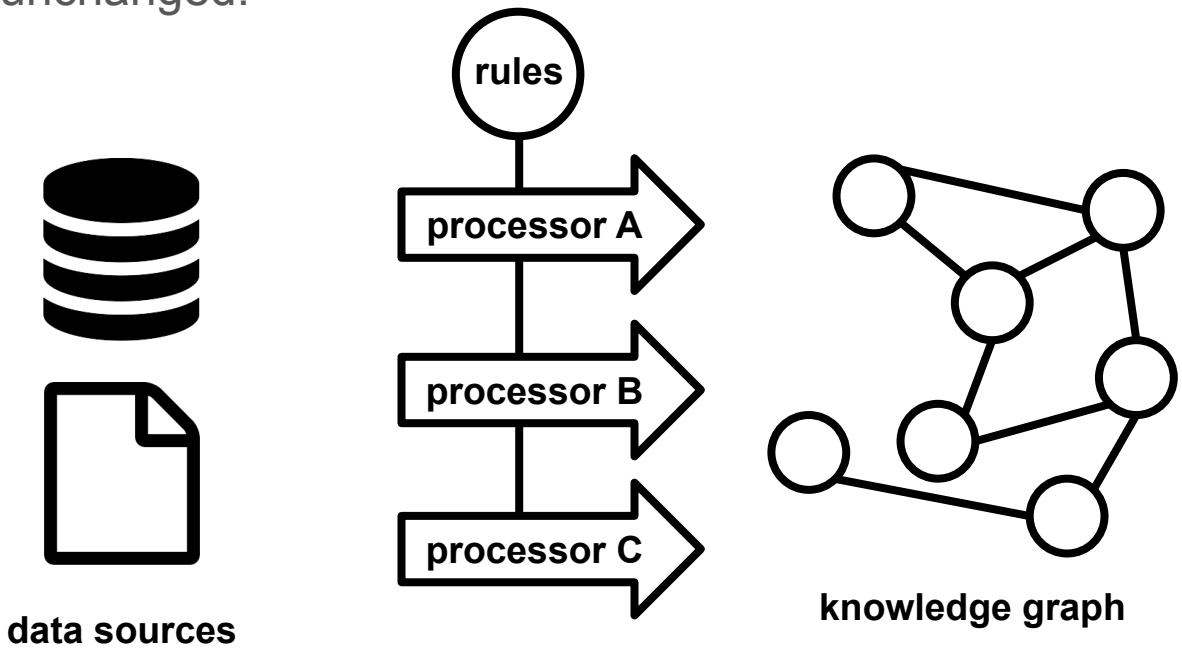
Processor = software tool that, given a set of rules and data sources, generates knowledge graphs.

There exists at least one processor for each language:

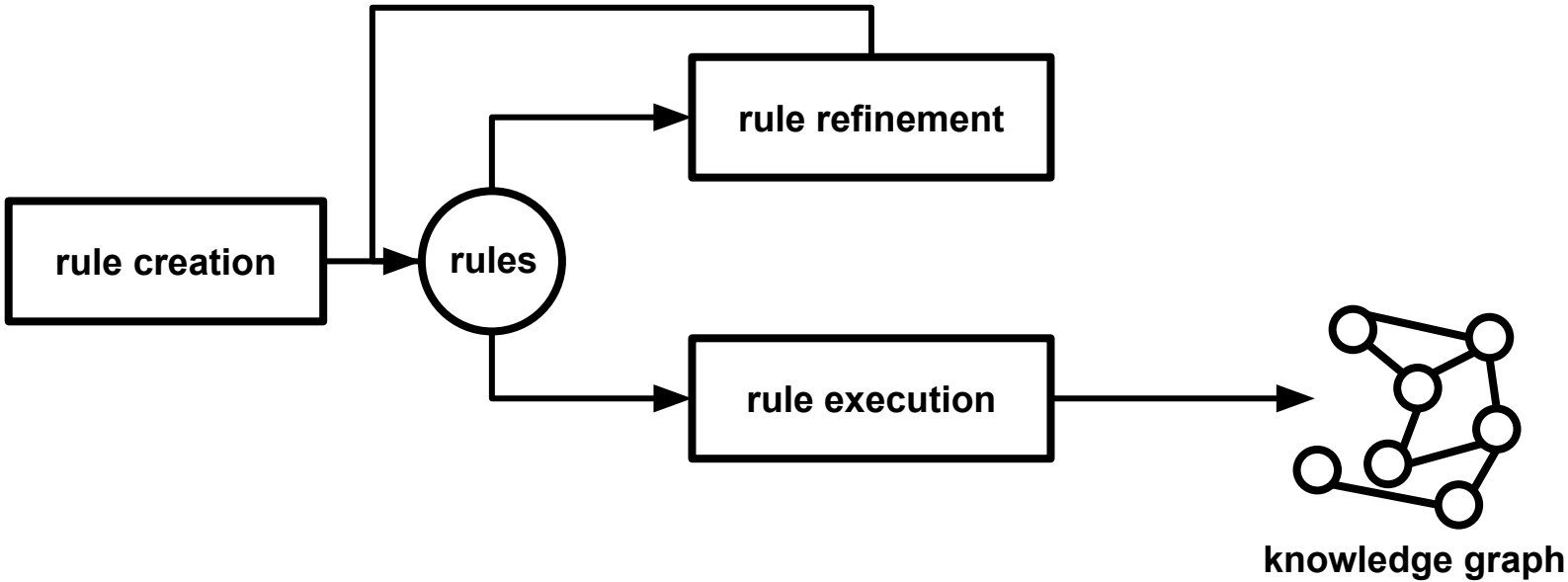
- RMLMapper and RMLStreamer for RML
- Morph-RDB and R2RML Parser for R2RML

Switch between different processors

Rules remain unchanged.



Putting it all together




Most
impre
ssive!

This already existed.
I didn’t do anything here.

Overview

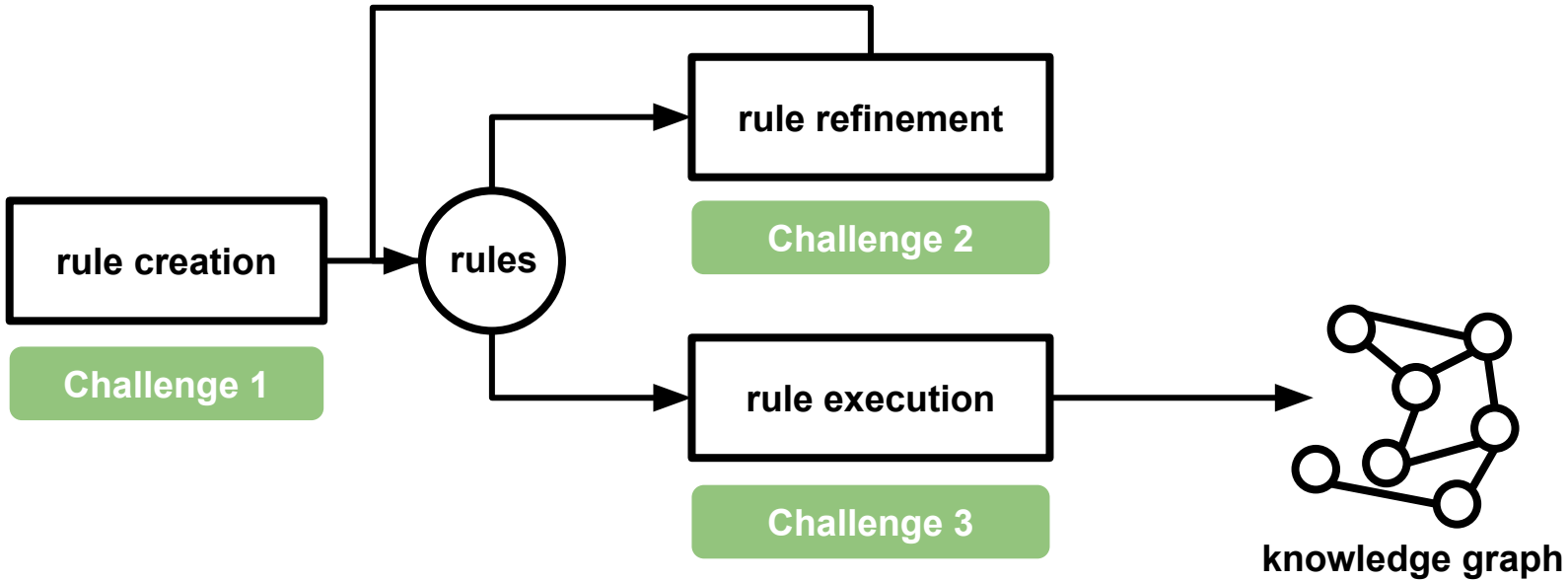
Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

Three research challenges



Creating rules requires lot of human effort

Components: data sources, rules, concepts, and relationships.

Data sources can be in different formats (databases, files).

Data might need to be transformed (capitalize names).

Lot of rules might be needed.

Rules are meant for machines, not humans.

Different concepts and relationships are used (how are they related).

Challenge 1: make it easier to create rules

Rules and definitions of concepts and relationships can lead to inconsistencies.

Rules: wrong concept and relationships are used.

For example, using the relationship “age” instead of “name”.

Definitions of concept and relationships: incorrectly model real world.

For example, a name only consists of numbers.

Challenge 2: make it easier to fix inconsistencies in knowledge graphs

Different processors have different features

Speed

Programming language

Follow syntax and grammar of knowledge graph generation language or not

Challenge 3: make it easier to select the best processor

Overview

Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

Overview

Moving from Web to Semantic Web

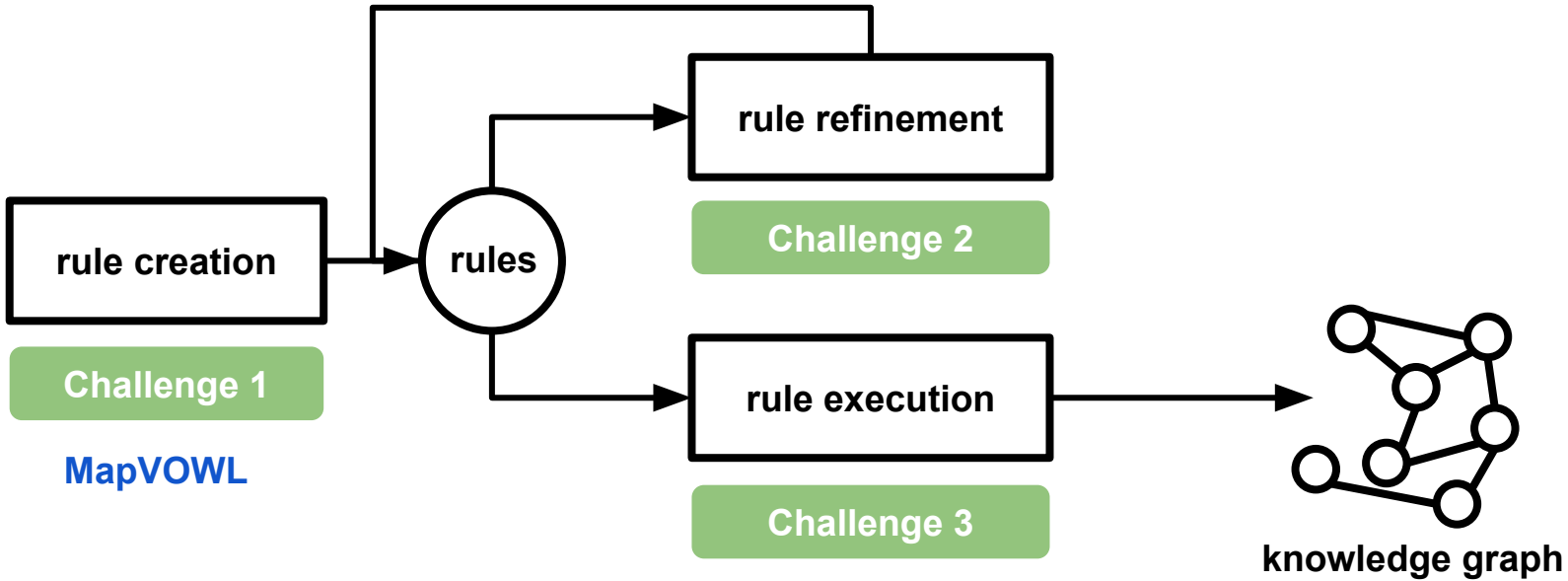
Challenges

Contributions

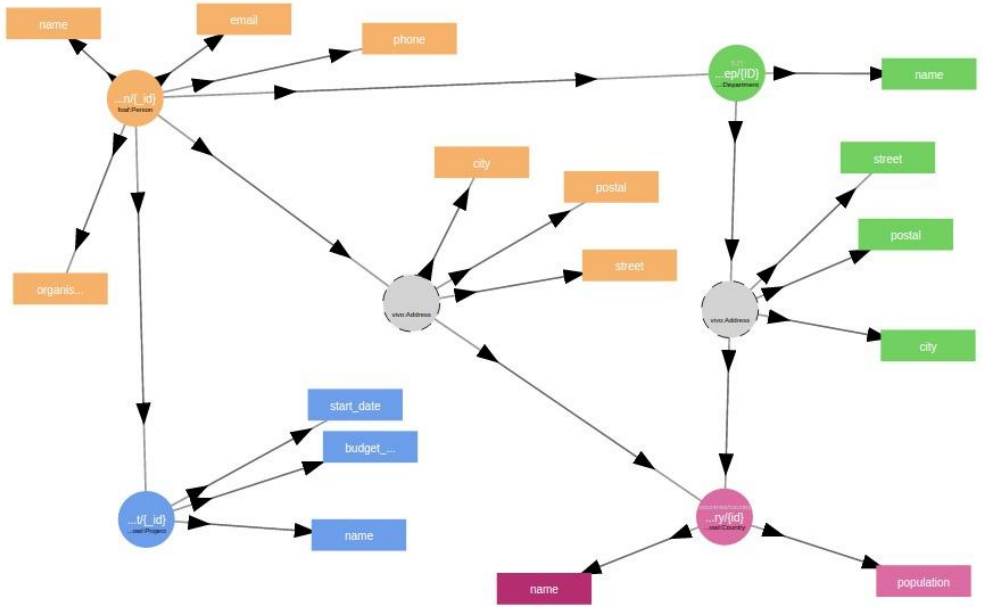
- **MapVOWL**
- RMLEditor
- Resglass
- RML test cases

Conclusions

MapVOWL contributes to Challenge 1



MapVOWL: visual notation for knowledge graph generation rules



Problem: visualizations not thoroughly investigated yet.

Different tools have different graph-based visualizations.

No specification of visualizations specified →

- Hard to compare with others tools/visualizations.
- Hard to implement visualizations in other tools.

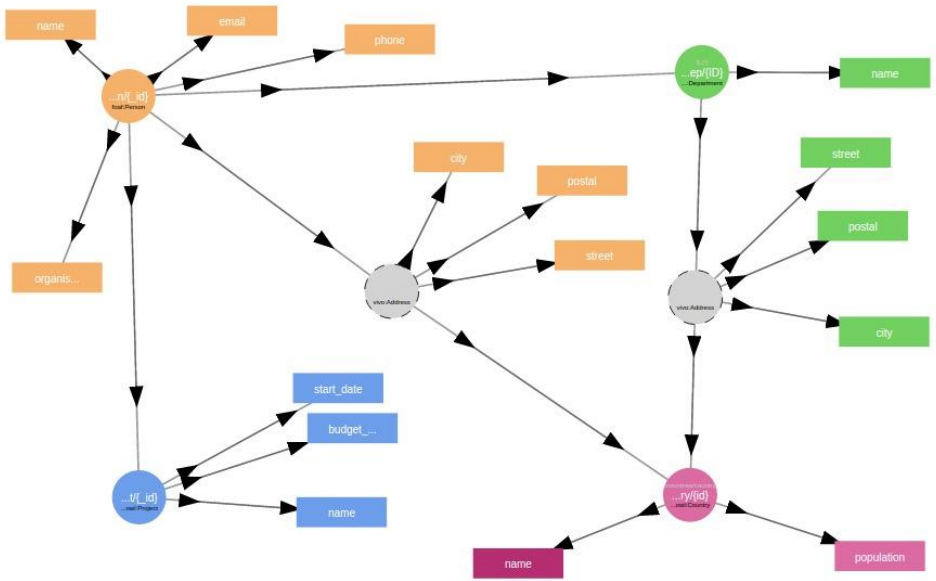
MapVOWL: visual notation for knowledge graph generation rules

Graph-based

Models knowledge graphs

Colors to denote data sources

Independent of knowledge graph generation language



Evaluation: compare MapVOWL with RML

9 participants

List of questions answered for both MapVOWL and RML:

- Number of relationships
- Number of data sources
- List relationships related to person

Example: RML rules

```
:map_anomaly_0 rml:logicalSource :source_0.
:source_0 a rml:LogicalSource;
      rml:source "input.json";
      rml:iterator "$";
      rml:referenceFormulation ql:JSONPath.
:map_anomaly_0 a rr:TriplesMap;
      rdfs:label "anomaly".
:s_0 a rr:SubjectMap.
:map_anomaly_0 rr:subjectMap :s_0.
:s_0 rml:reference "id".
:pom_0 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_0.
:pm_0 a rr:PredicateMap.
:pom_0 rr:predicateMap :pm_0.
:pm_0 rr:constant rdf:type.
:pom_0 rr:objectMap :om_0.
:om_0 a rr:ObjectMap;
      rr:template
"http://IBCNServices.github.io/Folio-Ontology/Folio.owl#
{anomaly.type}";
      rr:termType rr:IRI.
```

```
:pom_1 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_1.
:pm_1 a rr:PredicateMap.
:pom_1 rr:predicateMap :pm_1.
:pm_1 rr:constant dc:description.
:pom_1 rr:objectMap :om_1.
:om_1 a rr:ObjectMap;
      rml:reference "anomaly.description";
      rr:termType rr:Literal.
:pom_2 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_2.
:pm_2 a rr:PredicateMap.
:pom_2 rr:predicateMap :pm_2.
:pm_2 rr:constant folio:hasCauseDescription.
:pom_2 rr:objectMap :om_2.
:om_2 a rr:ObjectMap;
      rml:reference "anomaly.explanation";
      rr:termType rr:Literal.
:pom_3 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_3.
:pm_3 a rr:PredicateMap.
:pom_3 rr:predicateMap :pm_3.
:pm_3 rr:constant sosa:usedProcedure.
:pom_3 rr:objectMap :om_3.
```

Result: MapVOWL preferred over RML

Correctness when answering questions is same for MapVOWL and RML.

Participants **prefer MapVOWL over RML**.

BONUS: participants would like to use MapVOWL to edit rules.

Overview

Moving from Web to Semantic Web

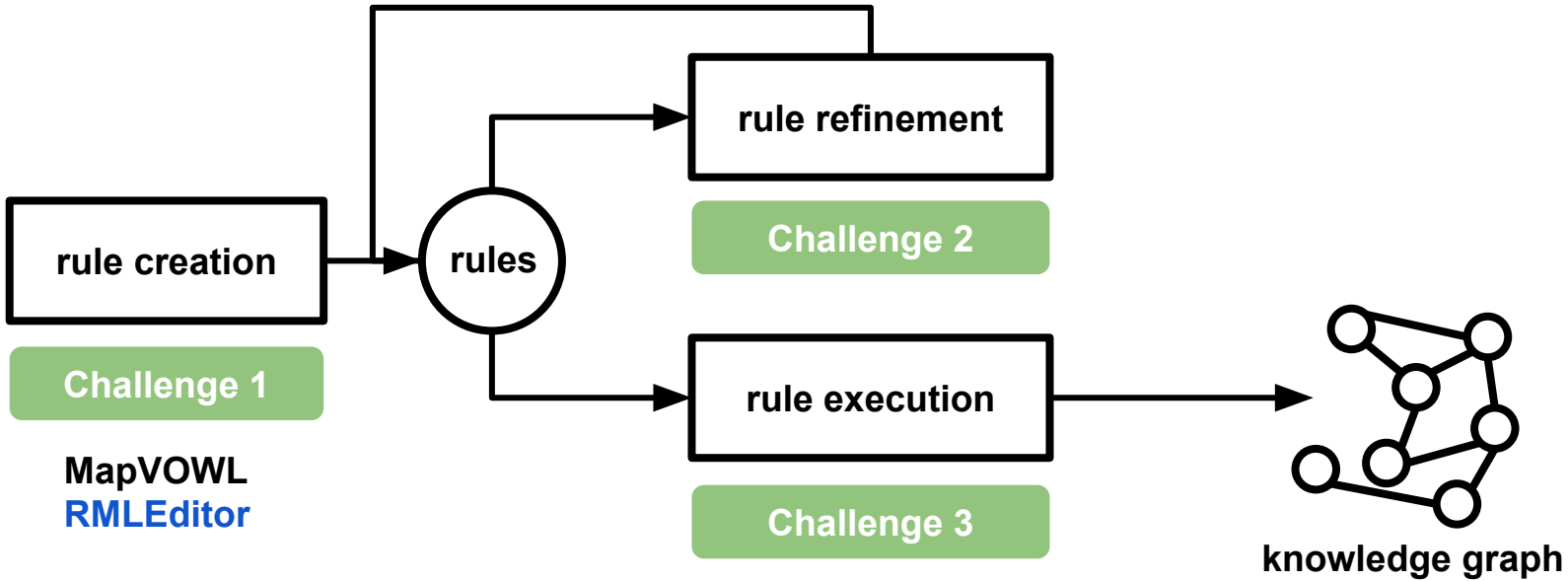
Challenges

Contributions

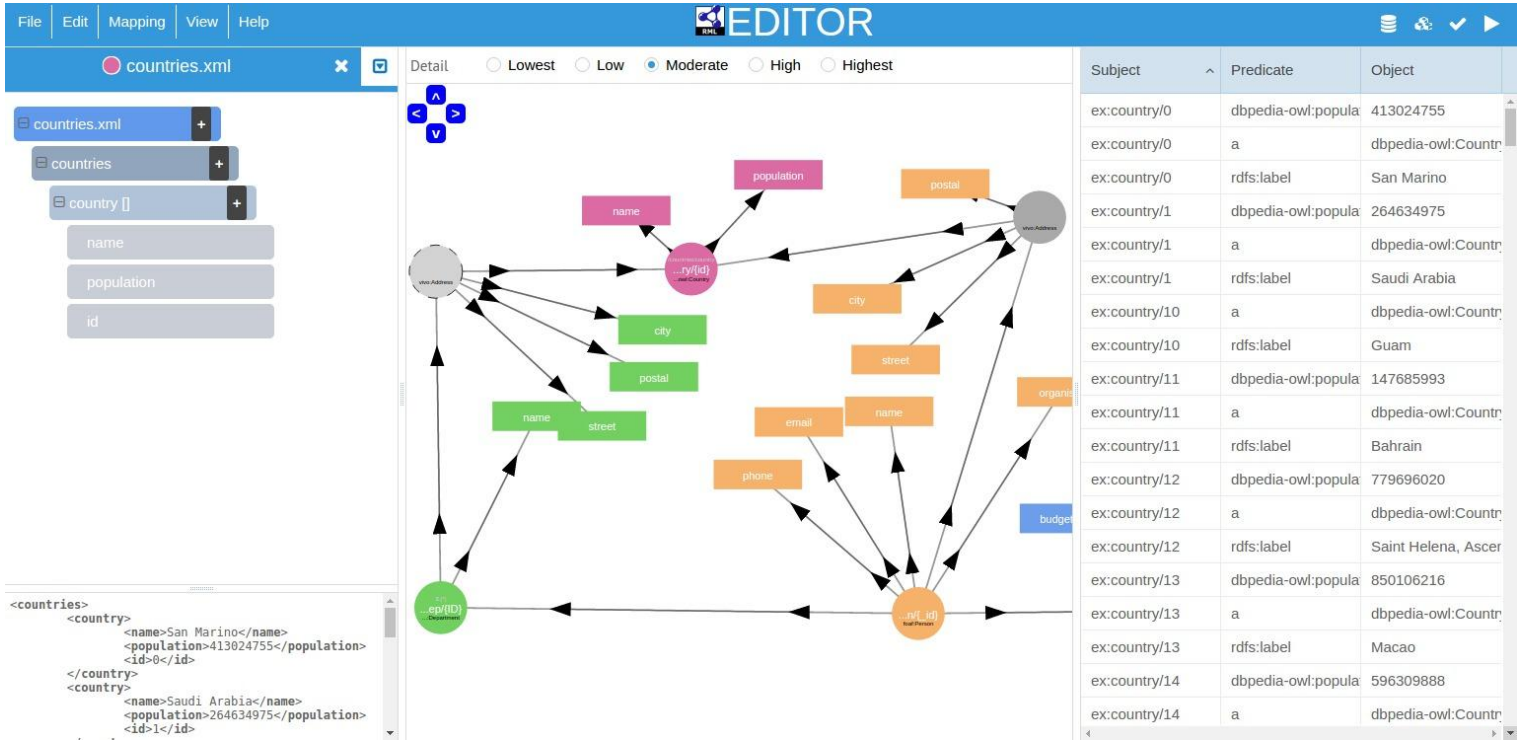
- MapVOWL
- **RMLEditor**
- Resglass
- RML test cases

Conclusions

RMLEditor contributes to Challenge 1



RMLEditor: graph-based rule editor



Problem: rule editors not thoroughly investigated yet

Different approaches

- Form-based
- Graph-based

Support different data sources

- Only databases
- Only files

(No) support for data transformations (e.g., capitalize names)

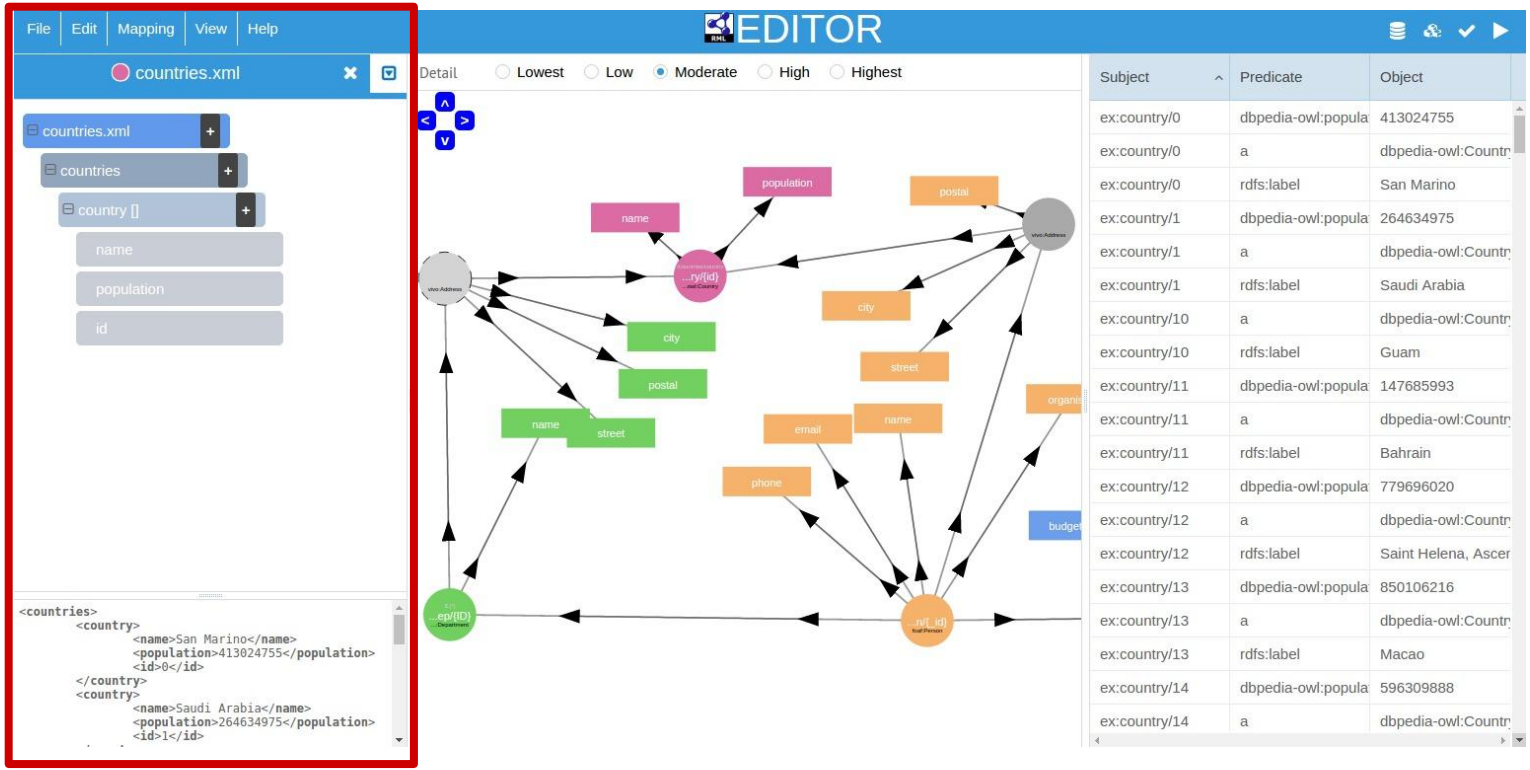
RMLEditor’s GUI has three panels

Input Panel

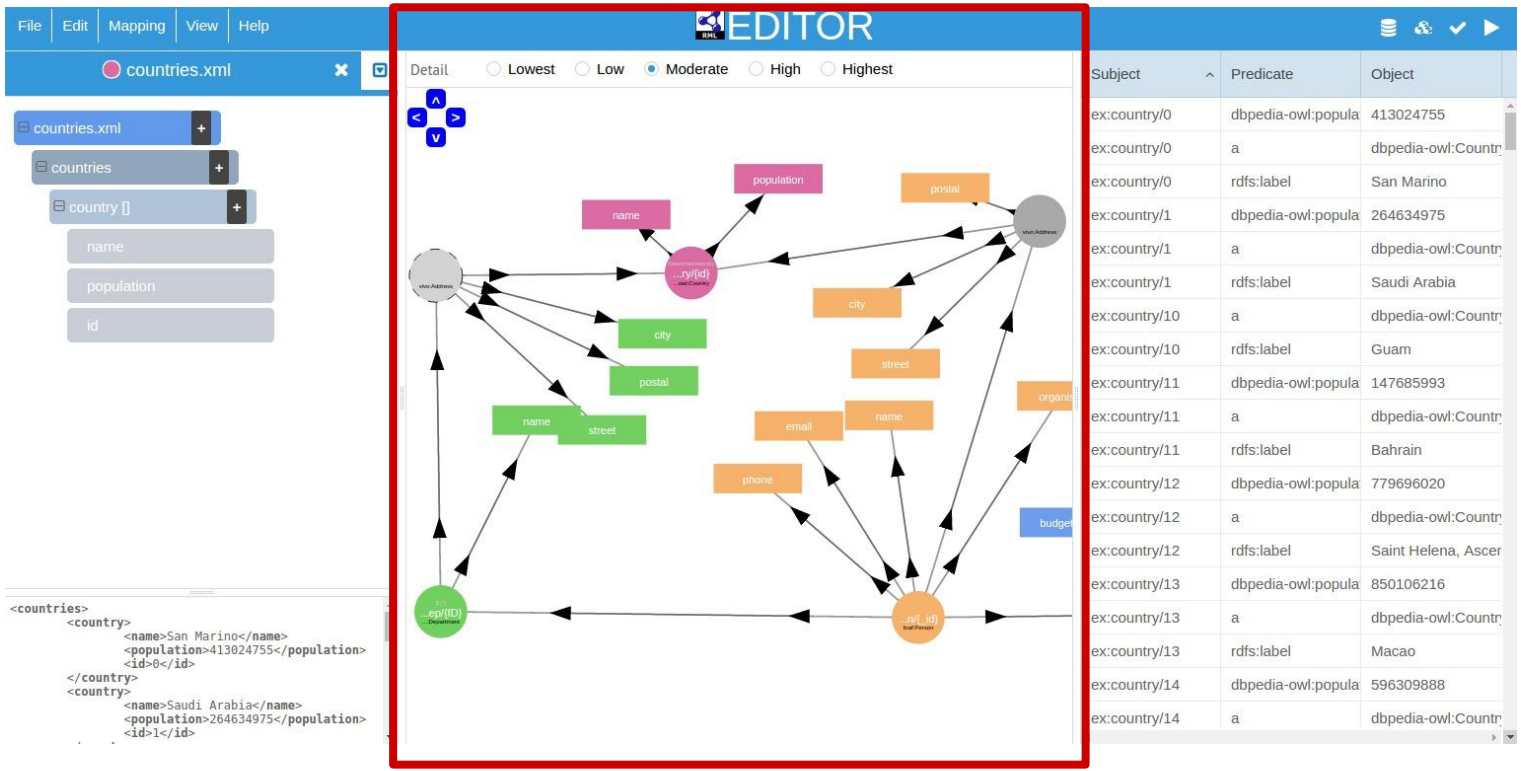
Modeling Panel

Results Panel

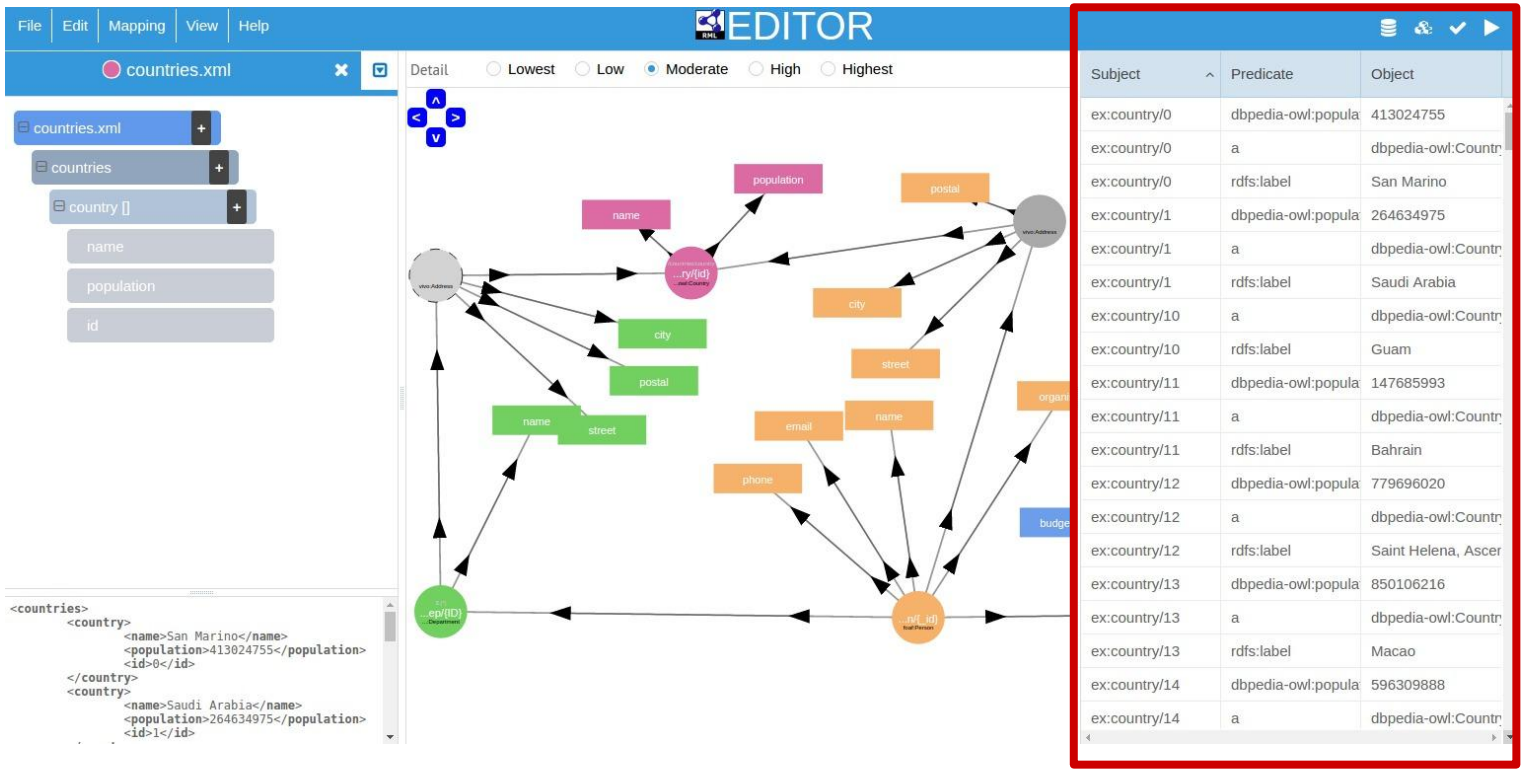
Input Panel shows data sources



Modeling Panel allows to create rules via MapVOWL



Results Panel shows generated knowledge graph



Evaluation: compare RMLEditor with RMLx

10 participants

Evaluation:

- Two use cases: either with RMLEditor or RMLx
- List of questions for RMLEditor
  - Large graphs
  - Data transformations

Result: RMLEditor is better than RMLx

Higher completeness of rules when using RMLEditor.

Participants rated **RMLEditor as good, while RMLx as poor**.

Challenge 1 

MapVOWL + RMLEditor = easier to create rules.

Overview

Moving from Web to Semantic Web

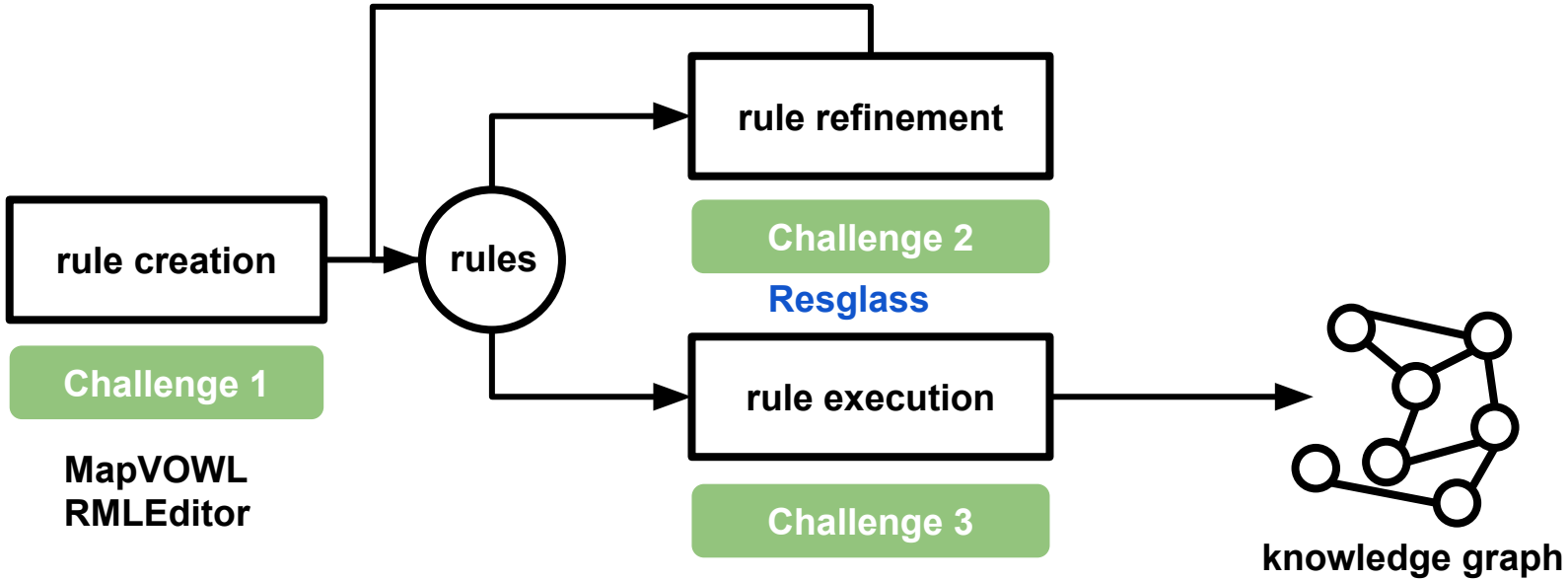
Challenges

Contributions

- MapVOWL
- RMLEditor
- **Resglass**
- RML test cases

Conclusions

Resglass contributes to Challenge 2



Resglass: rule-driven method to resolve inconsistencies

Resglass: rule-driven method to resolve inconsistencies

Due to the lack of a screenshot here's a dog riding a skateboard.

Problem: removal of inconsistencies

Introduced by rules.

Introduced by definitions of used concepts and relationships.

Resglass consists of 5 steps

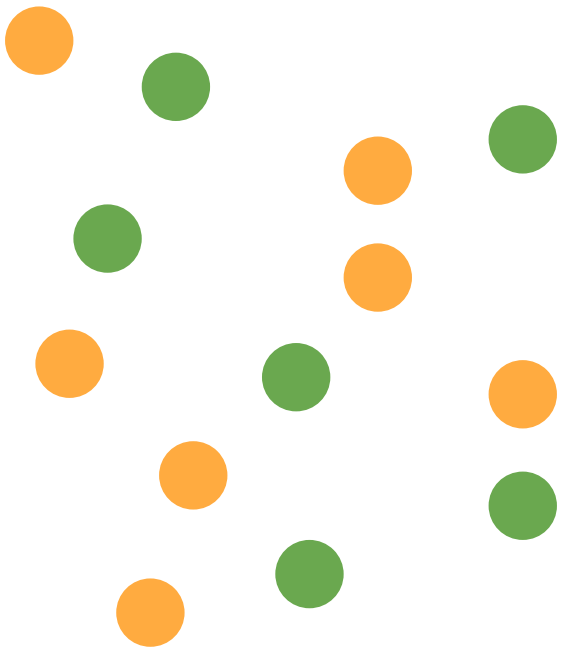
1. Rule inconsistency detection
2. Rules and definitions refinement
3. Knowledge graph generation
4. Knowledge graph inconsistency detection
5. Rules and definitions refinement

Step 1: Rules inconsistency detection

Person rules

Location rules

Definitions



name

location

city

person

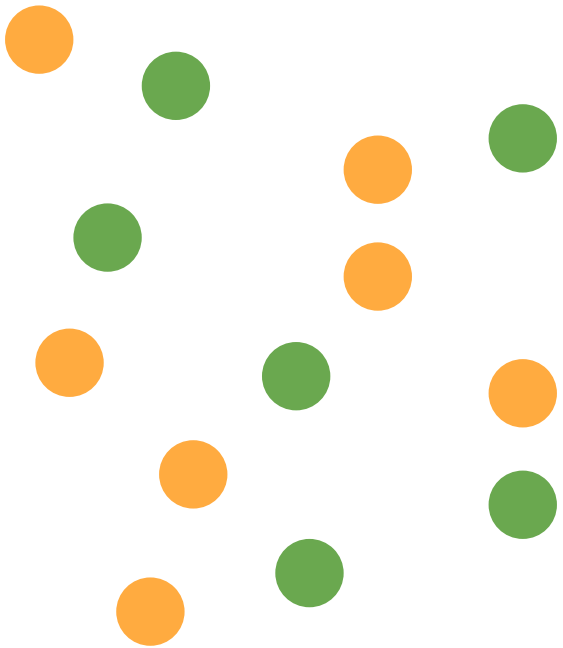
Step 1: Rules inconsistency detection

Person rules

Location rules

Definitions

Inconsistency



name

location

city

person

Name is not a number

Location is a person

Step 1: Rules inconsistency detection

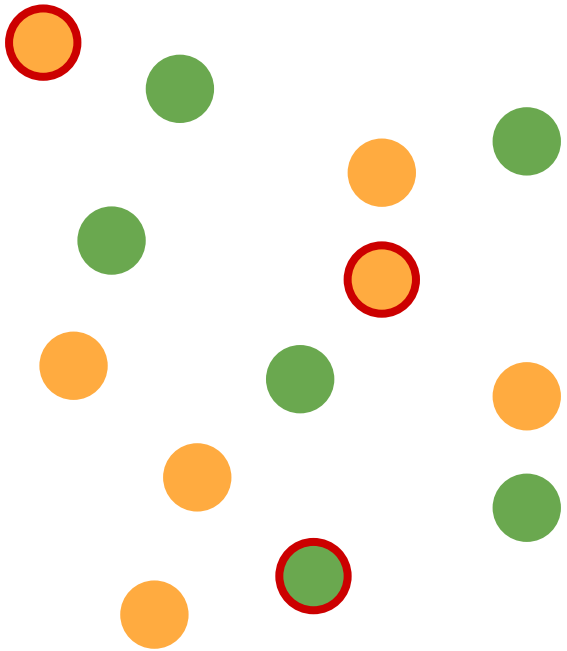
Person rules

Location rules

Definitions

Inconsistency

Rule/definition contributing to at least 1 inconsistency



name

location

city

person

Name is not a number

Location is a person

Step 2: Rules and definitions refinement

2.1 Rules clustering

2.2 Rules and definitions ranking

2.3 Rules and definitions refinement

Step 2.1: Rules clustering based on entity

Why? Rules concerning single entity impact each other → their refinements too.

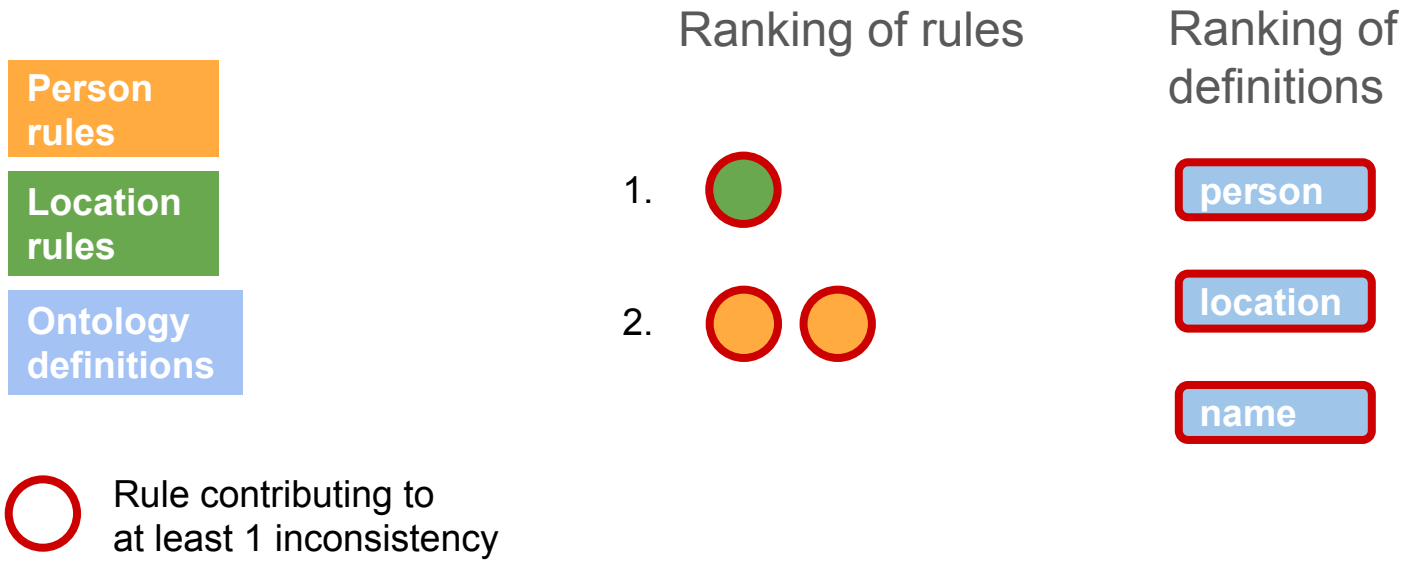
Person rules

Location rules

Rule contributing to at least 1 inconsistency

Step 2.2: Rules and definitions ranking via score

Why? So we know which rules and definitions to inspect first.



Step 2.3: Rules and definitions refinement to resolve inconsistencies

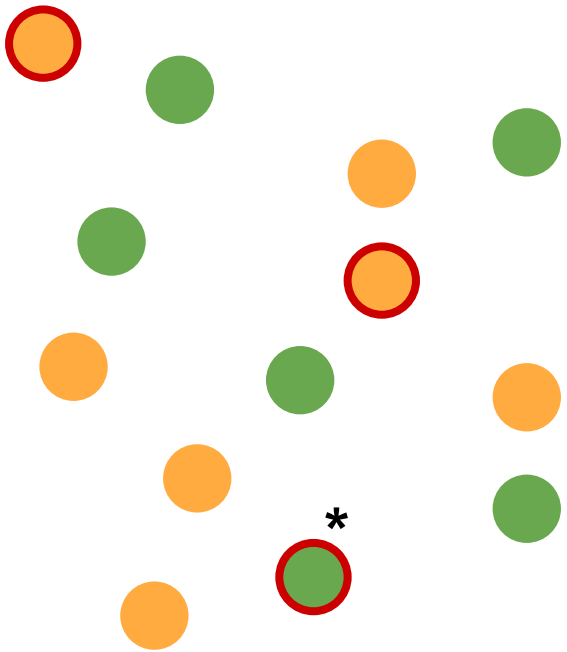
Person rules

Location rules

Ontology definitions

Rule contributing to at least 1 inconsistency

* Refinement



*

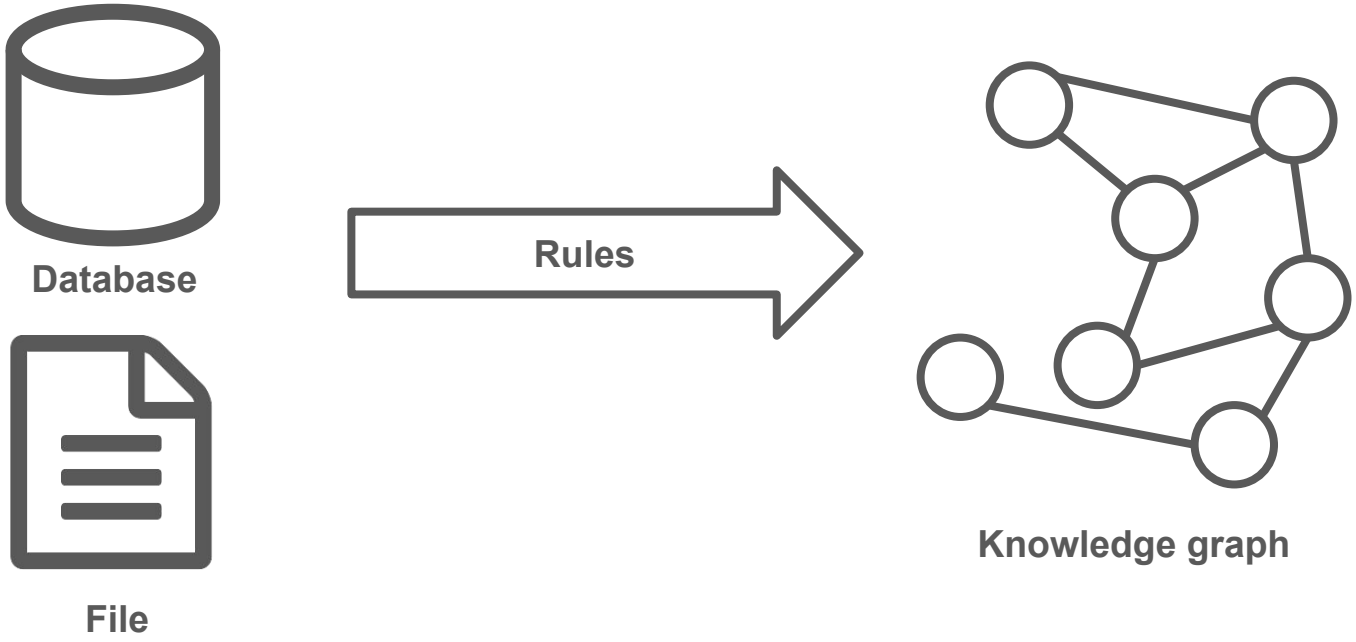
name

location

city

person

Step 3: Knowledge graph generation

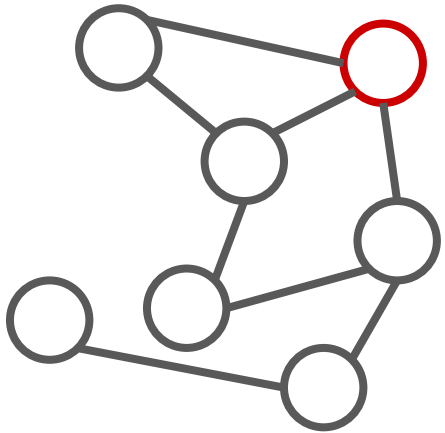


Step 4: Knowledge graph inconsistency detection

Why? Certain inconsistencies cannot be detected via the rules alone.
The complete knowledge graph is required.

Name is not capitalized.

“professor oak”
instead of “Professor Oak”



Knowledge graph

Step 5: Rules and definitions refinement

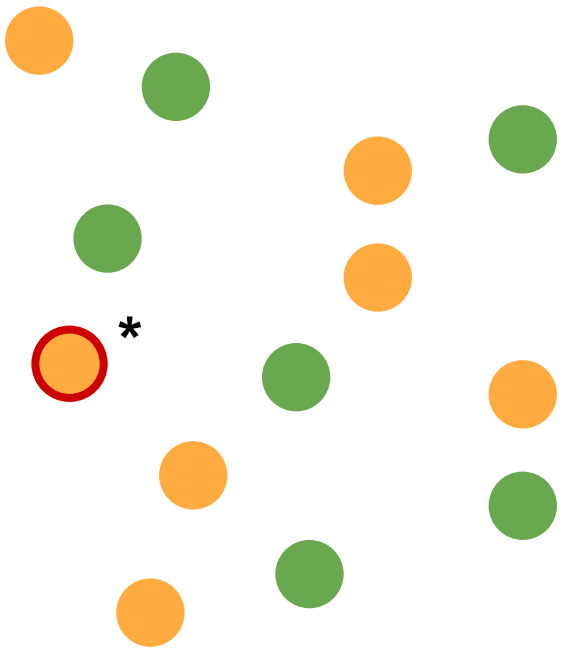
Person rules

Location rules

Definitions

Rule contributing to at least 1 inconsistency

* Refinement



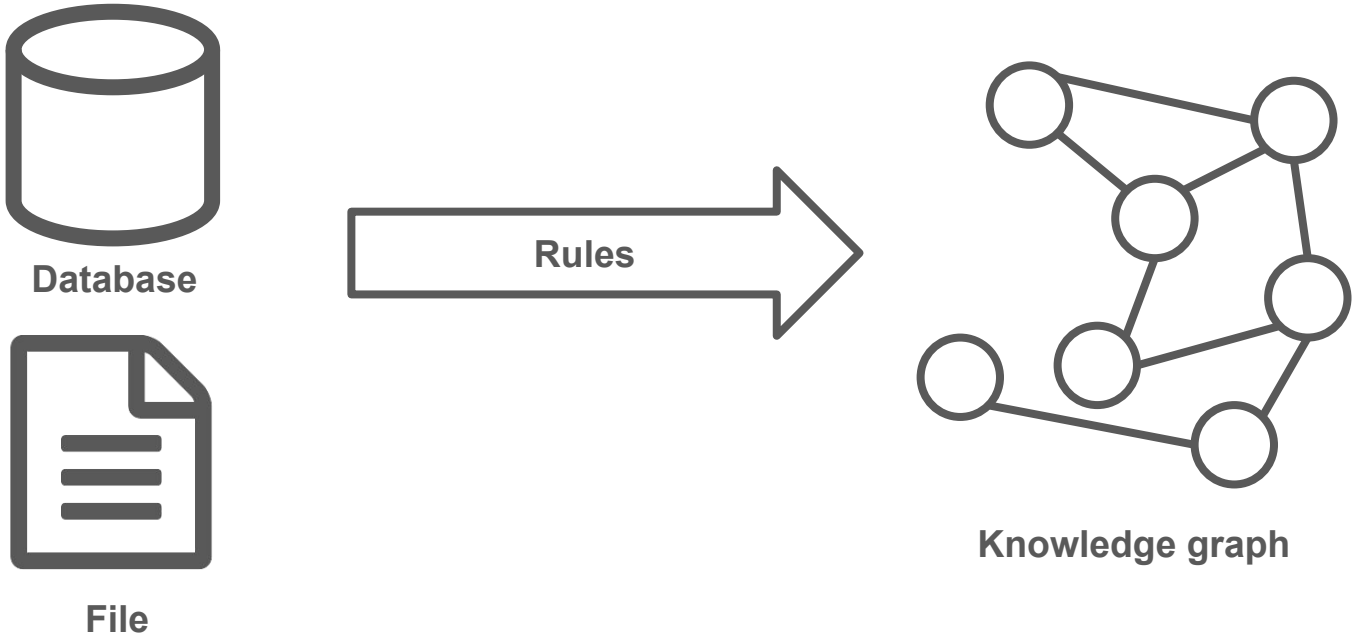
name

location

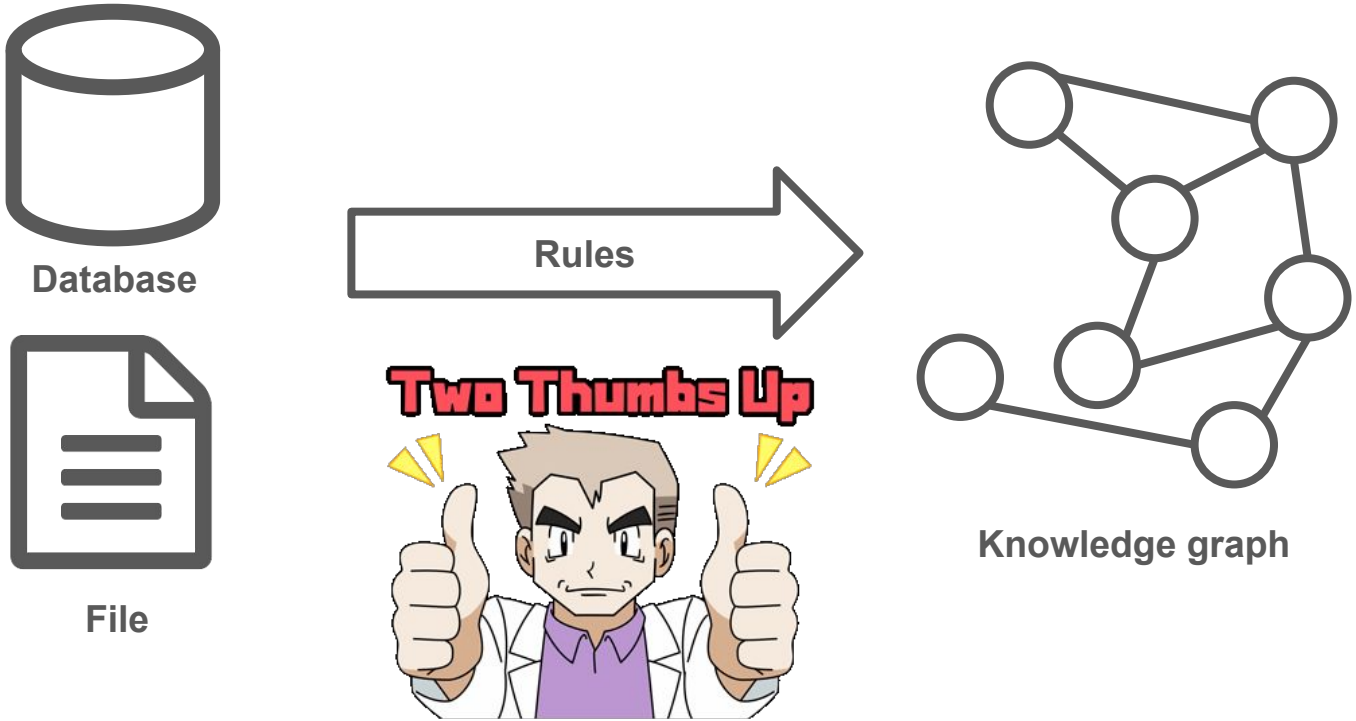
city

person

Result: knowledge graph without inconsistencies



Result: knowledge graph without inconsistencies



Evaluation: compare ranking of Resglass with manual ranking of experts

3 participants

Evaluation:

- Participants rank rules and definitions for manually.
- Comparison between ranking of Resglass and participants' rankings.

Result: Resglass can help experts

80% overlap with ranking of experts for rules and definitions.

Challenge 2 

Resglass makes it easier to fix inconsistencies.

Overview

Moving from Web to Semantic Web

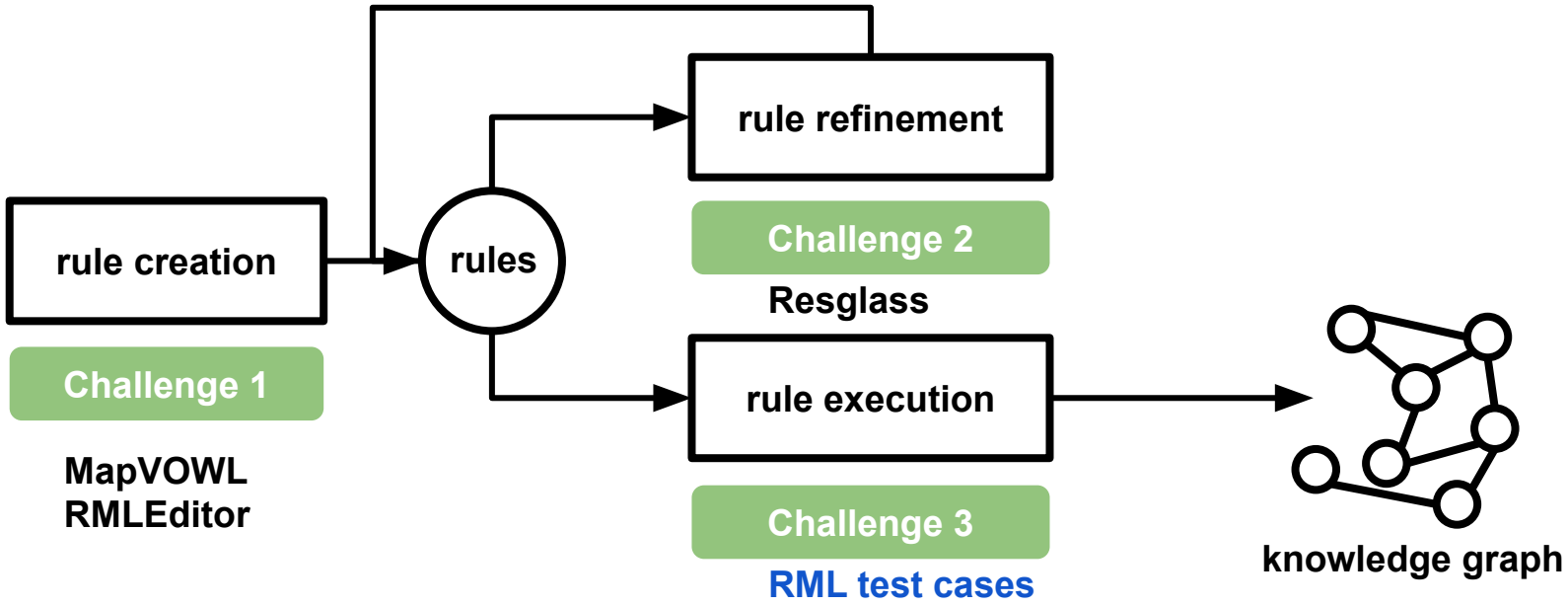
Challenges

Contributions

- MapVOWL
- RMLEditor
- Resglass
- **RML test cases**

Conclusions

RML test cases contributes to Challenge 3



Example: test case

Application that calculates the sum of two numbers: `sum`

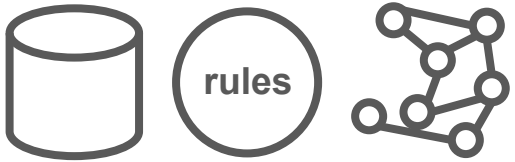
```
sum(x, y)
    x + y
```

Test cases

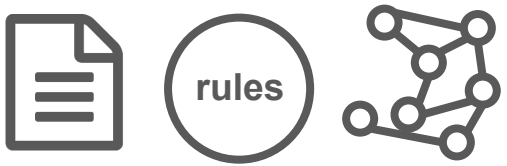
sum(1, 1) = 2
sum(2, 2) = 4
sum(1, 2) = 3

If all these cases succeed then the application works correctly.

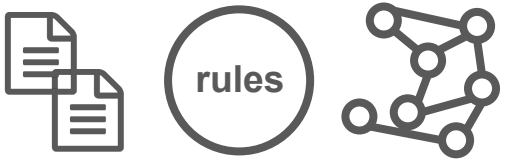
Test cases to determine conformance of processors to languages



Test if correct knowledge graph is generated from one database.



Test if correct knowledge graph is generated from one file.



Test if correct knowledge graph is generated from two files.

Problem: what are the characteristics of test cases for knowledge graphs languages that support different data sources?

No test cases for such languages (RML).

Only test cases for databases (R2RML).

Create initial test cases for RML

RML supports different data sources.

No test cases existed for RML.

Remember that RML is an extension of R2RML.

RML test cases based on R2RML test cases.

Benefits of RML test cases for developers and users

Developers determine how conformant processors are to RML specification.

Users can use test case results to select most suitable processor for use case.

Challenge 3 ✅

The RML test cases make it easier to select the best RML processor.

Overview

Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

Overview

Moving from Web to Semantic Web

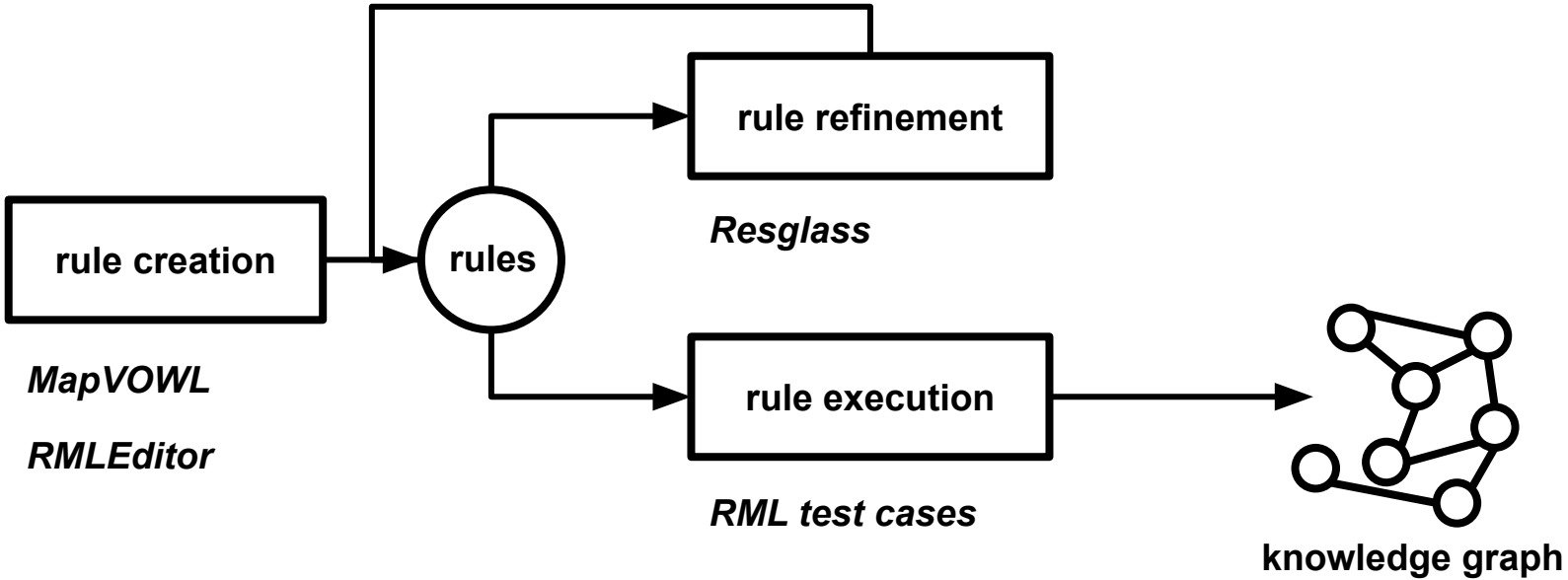
Challenges

Contributions

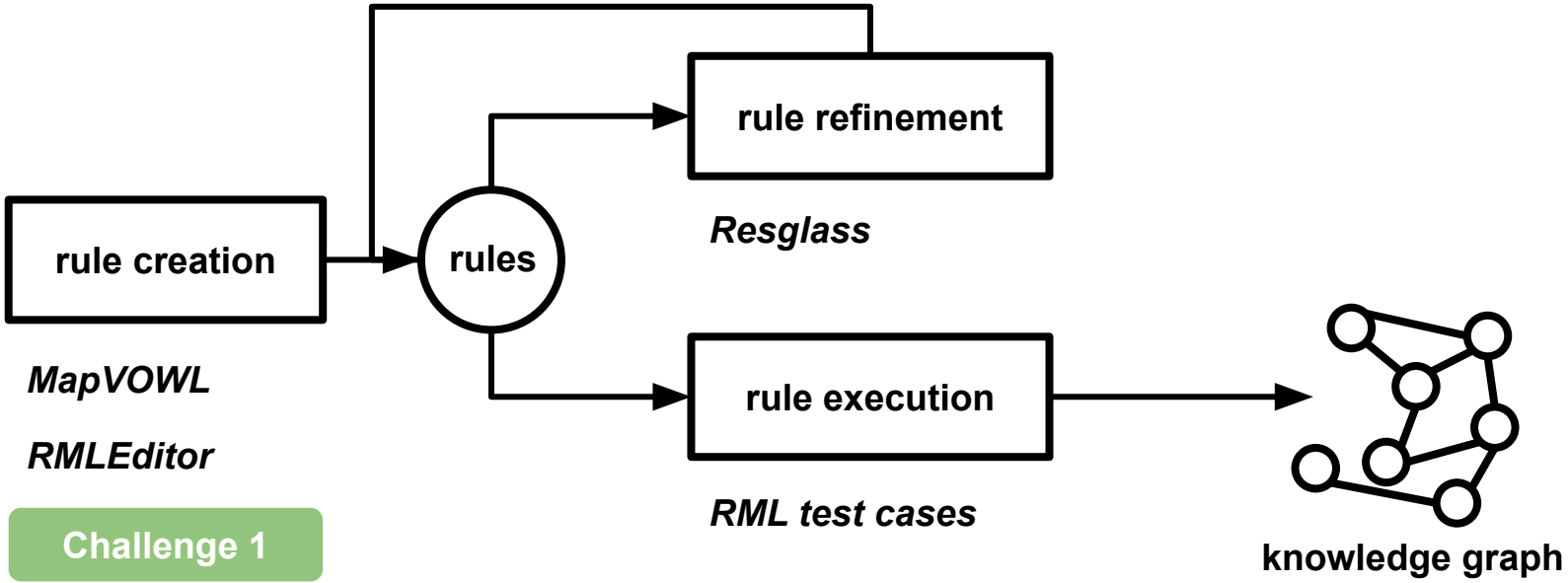
Conclusions

- **Impact of contributions**
- Remaining challenges and future directions

Impact of contributions

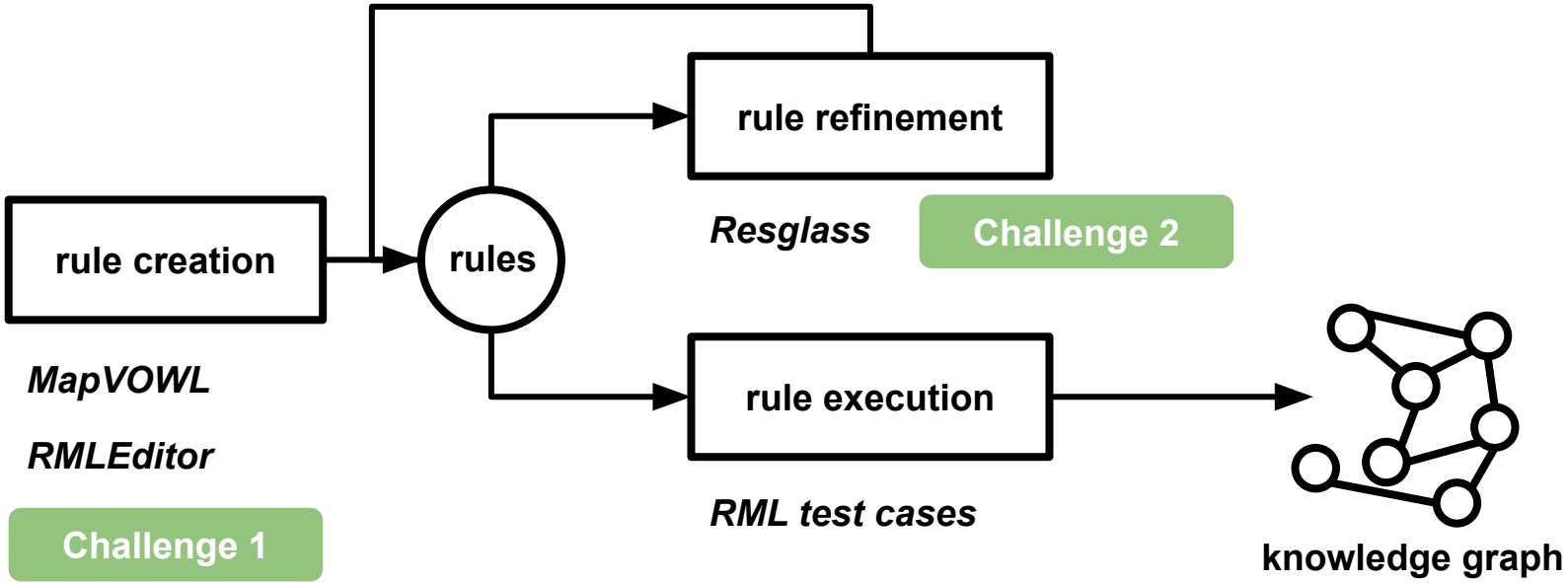


MapVOWL and RMLEditor → Challenge 1



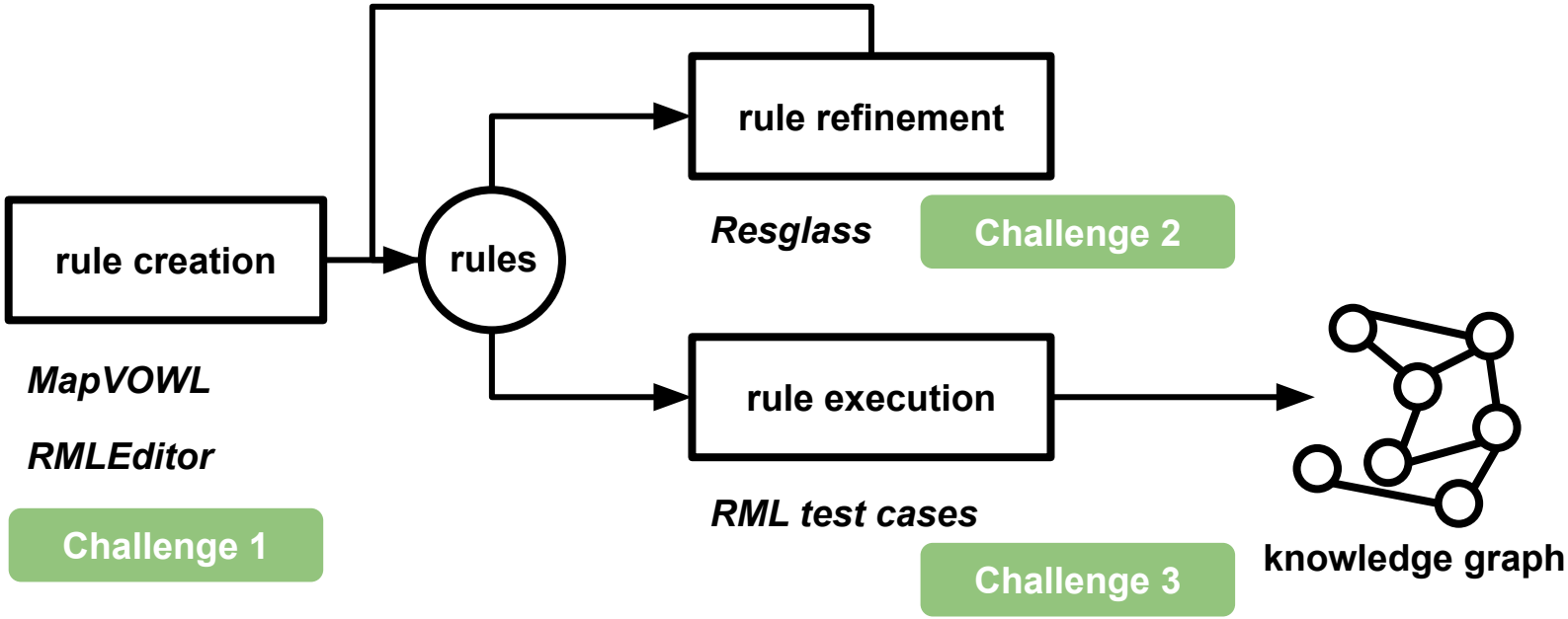
Challenge 1: make it easier to create rules.

Resglass→ Challenge 2



Challenge 2: make it easier to fix inconsistencies in knowledge graphs.

RML test cases → Challenge 3



Challenge 3: make it easier to select the best processor.

Overview

Moving from Web to Semantic Web

Challenges

Contributions

Conclusions

- Impact of contributions
- **Remaining challenges and future directions**

Rule creation: YARRRML

Some users prefer text-based solution over visualizations.

Solution: YARRRML

Rule refinement: Resglass can further assist users in resolving inconsistencies

Suggest possible resolutions based on used and other concepts and relationships.

Suggest possible resolutions based on rules of other use cases.

Rule execution: define abstract test cases for languages

Beyond RML

What needs to be tested?

Define what is part of knowledge graph language

And maybe more importantly what is *not*.

- Pre-processing
- Post-processing
- Use case-specific functions

Beyond RML

But RML (and its test cases) can serve as inspiration.

Improving Effectiveness of Knowledge Graph Generation Rule Creation and Execution

Pieter Heyvaert

PhD advisors: Ruben Verborgh and Anastasia Dimou

Overview

Moving from Web to Semantic Web

- Knowledge graphs
- **Ontology**
- Knowledge graph generation
- Knowledge graph inconsistencies
- Processors

Challenges

Research questions and hypotheses

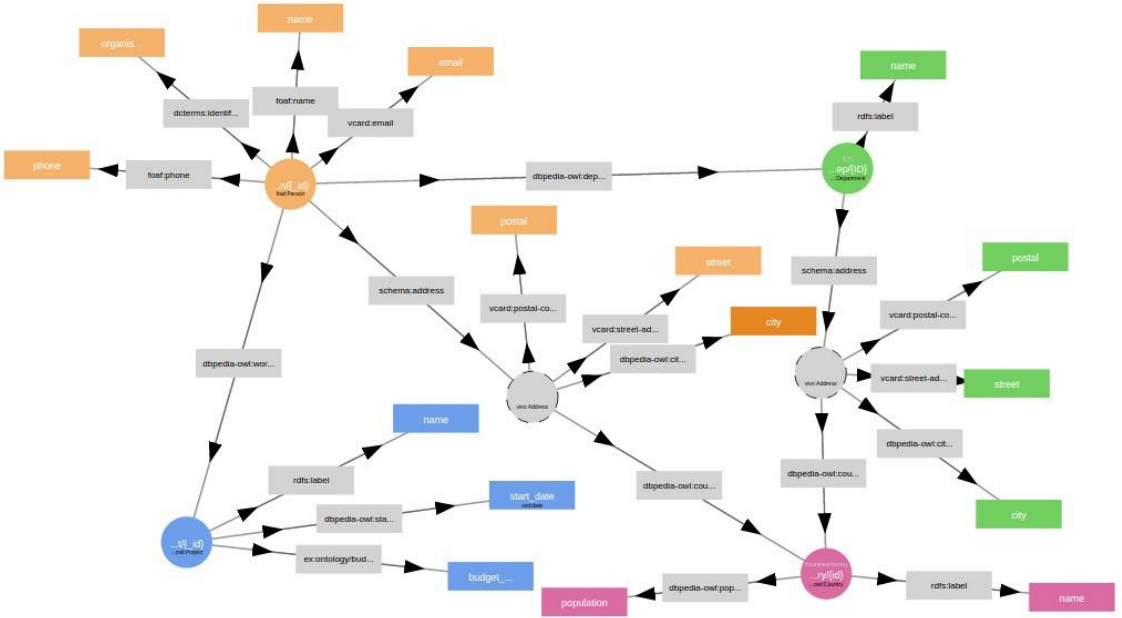
Contributions

Large graphs become complex

Difficult to keep overview of rules.

Difficult to find specific rules.

Difficult to edit rules.



Solution RMLEditor: 5 detail levels

Highest

High

Moderate

Low

Lowest

Solution RMLEditor: 5 detail levels

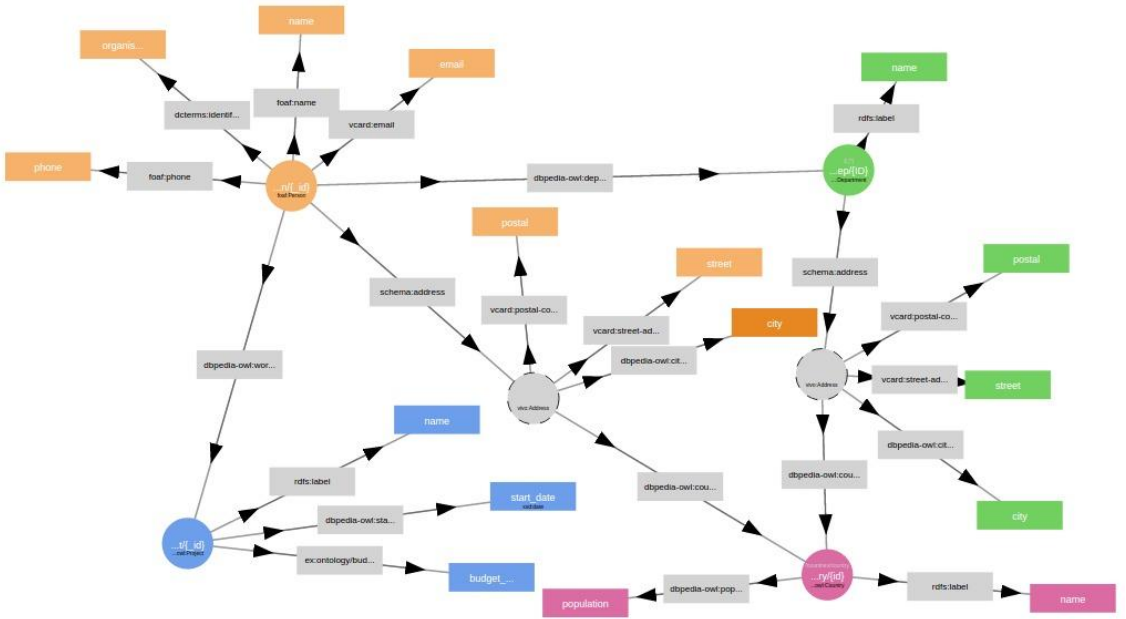
Highest

High

Moderate

Low

Lowest



Solution RMLEditor: 5 detail levels

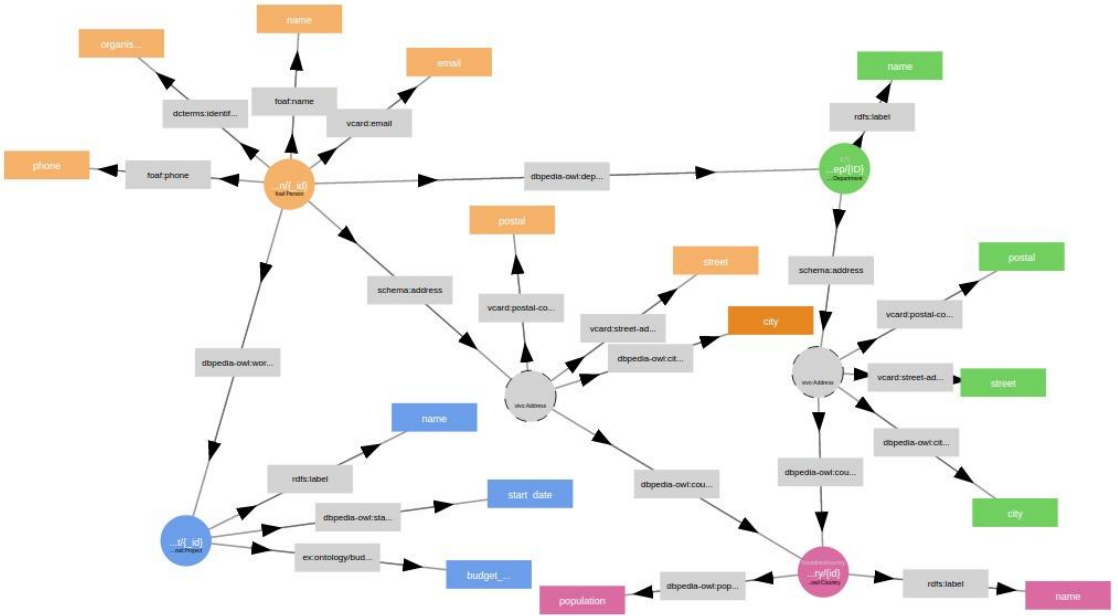
Highest

High

Moderate

Low

Lowest



Solution RMLEditor: 5 detail levels

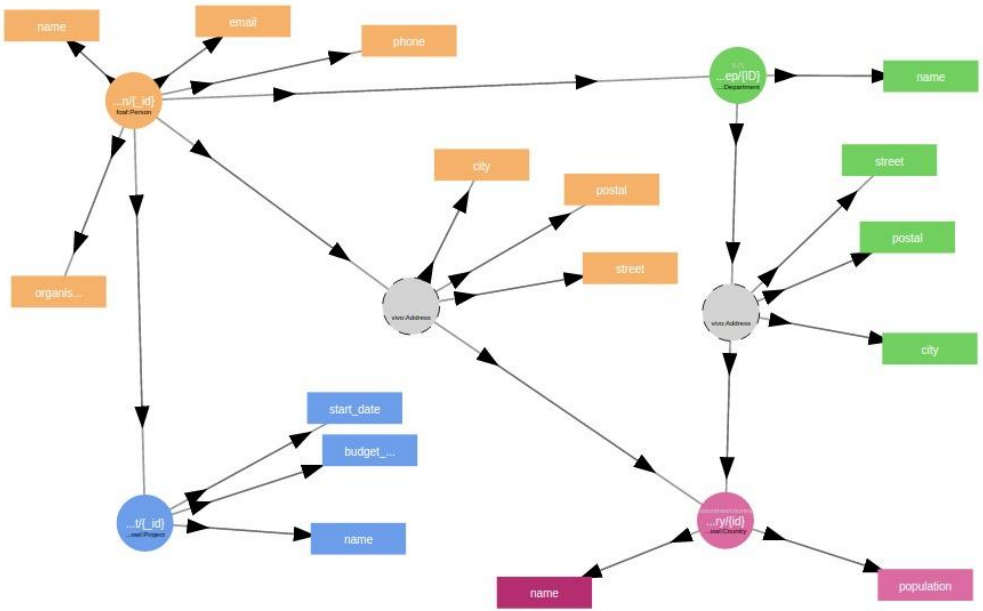
Highest

High

Moderate

Low

Lowest



Solution RMLEditor: 5 detail levels

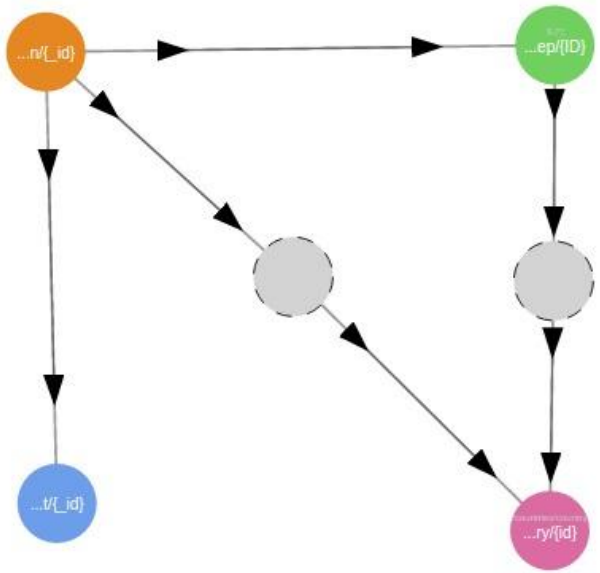
Highest

High

Moderate

Low

Lowest



Solution RMLEditor: 5 detail levels

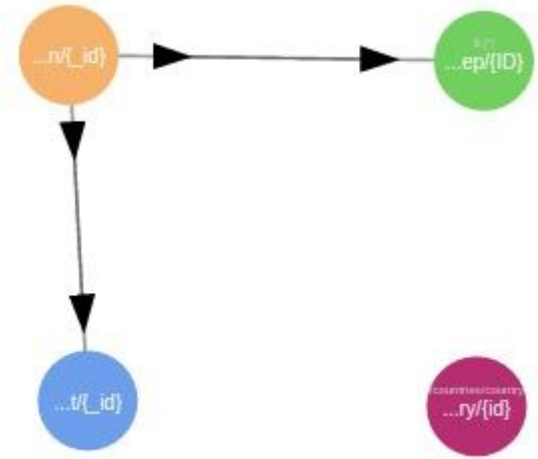
Highest

High

Moderate

Low

Lowest



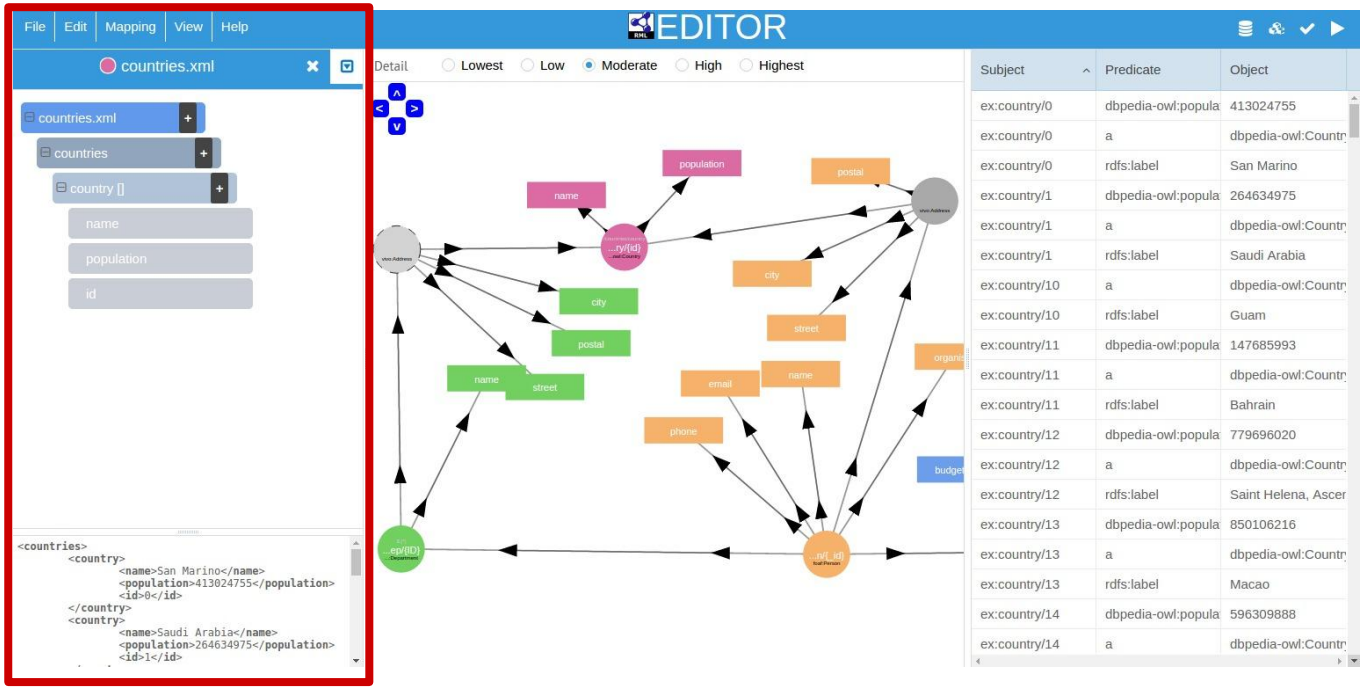
Data values might need to be transformed

Original name: professor oak

Preferred name: **P**rofessor **O**ak

Data transformation needed to fix the capitalization.

Solution RMLEditor: define transformation in Input Panel

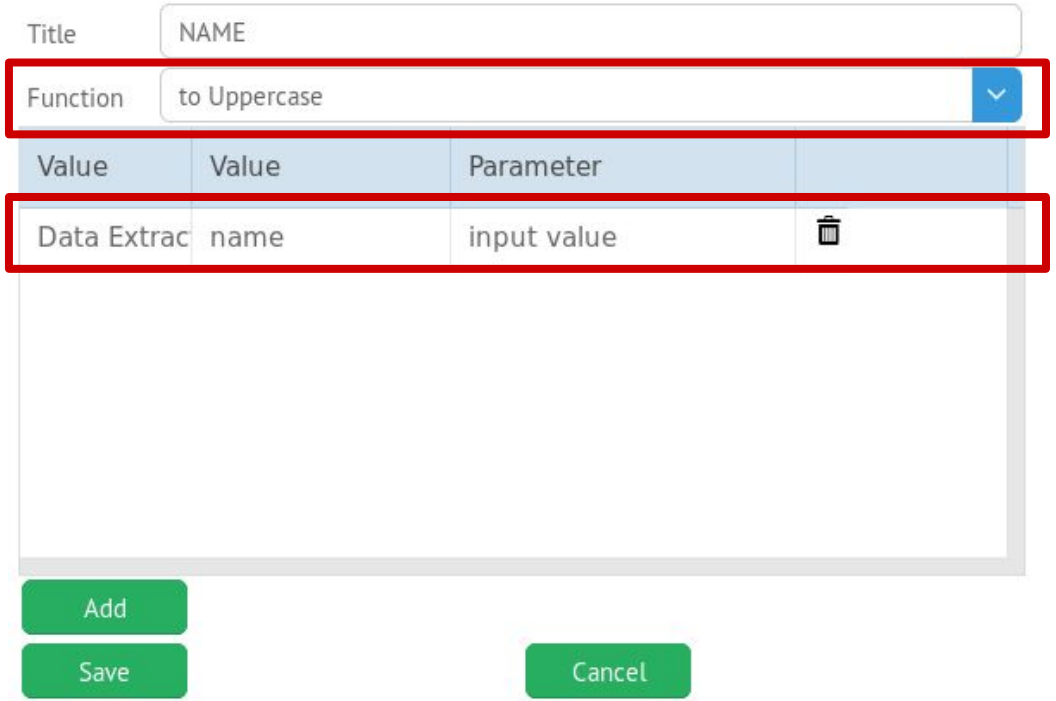


Solution RMLEditor: define transformation in Input Panel

Solution RMLEditor: define transformation in Input Panel

Transformation to use

Values to use



Use Resglass’ ranking for (semi-)automatic solution

Ranking is use case-specific.

(Semi)-automatic solution can do better than using a predefined set of refinements.

Evaluation: compare RMLMapper and CARML

RML processors:

- RMLMapper
- CARML

Ran RML test cases against both processors.

Test case results are either

- Passed
- Failed
- Inapplicable: not implemented → not tested

RMLMapper passes more test cases than CARML

	RMLMapper	**CARML**
Passed	**277**	84
Failed	**20**	33
Inapplicable	**0**	180

RMLMapper and CARML are not perfect

	RMLMapper	**CARML**
Passed	**277**	84
Failed	**20**	33
Inapplicable	**0**	180

RMLMapper fails for some database test cases: automatic typing

CARML fails for core language aspects

CARML does not support databases

	RMLMapper	**CARML**
Passed	**277**	84
Failed	**20**	33
Inapplicable	**0**	180

CARML does not support databases.

The RMLMapper does.

More research to create new test cases

Specific of hierarchical data formats (JSON, XML)

Impact of data source language (SQL, JSONPath, XPath)

Remaining challenges and future directions

Rule creation: are visualizations the only way?

Rule refinement:

- What can we automate?
- Can we avoid inconsistencies?

Rule execution:

- What is really part of the test cases?
- What is really part of the language?

Example: data about a person

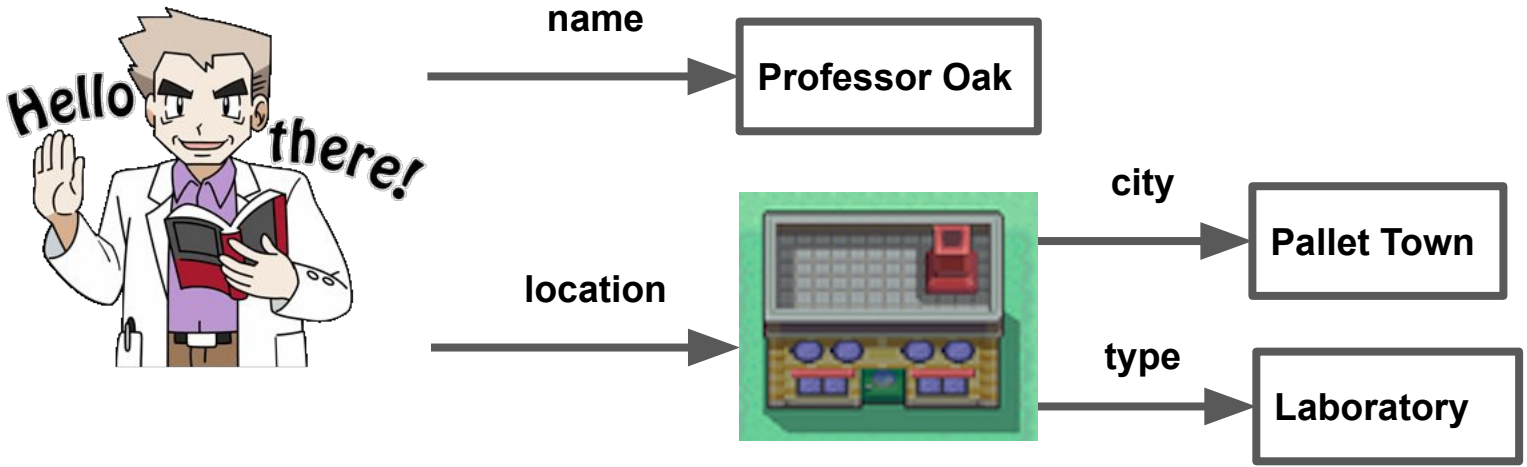
has the name

Professor Oak

subject **predicate** **object**

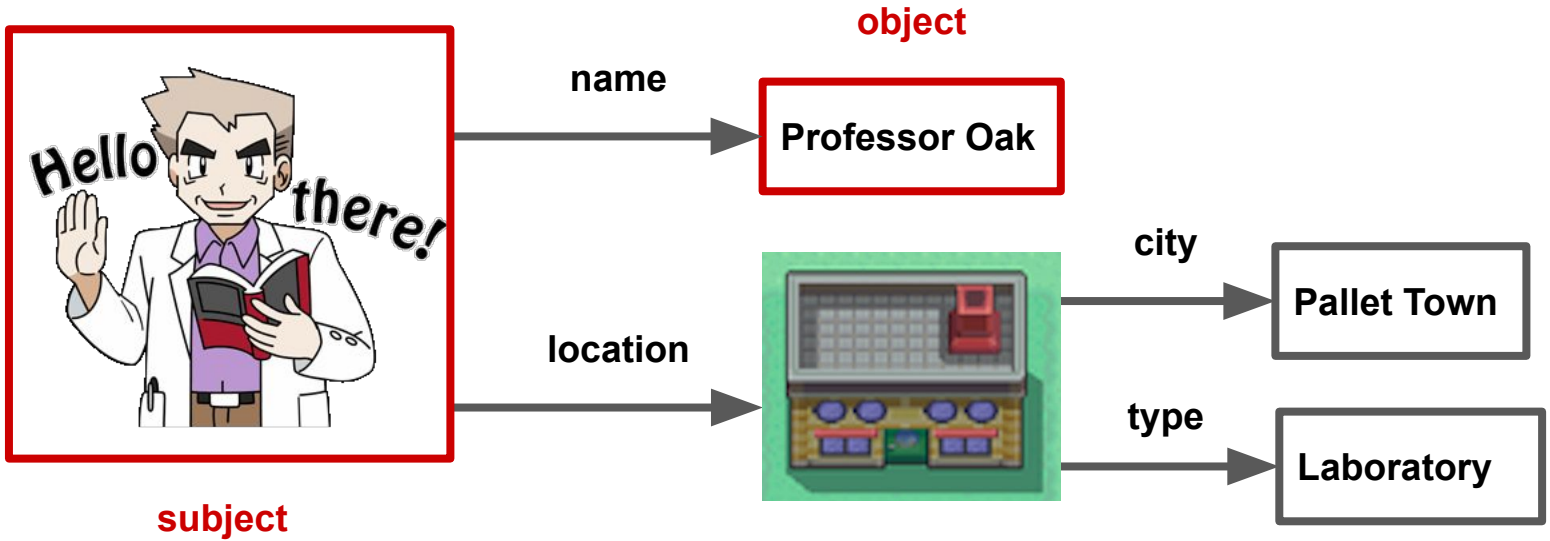
Knowledge graph is directed

Arrows denote the subject and object of a relationship.



Knowledge graph is directed

Arrows denote the subject and object of a relationship.



Ontology includes machine-understandable definitions of concepts and relationships

Concepts and relationships model real world.

For example: location, name, city

Their definitions are written in machine-understandable way.

Definitions are part of an ontology.

→ Applications can work with these concepts and relationships.

Improving Effectiveness of Knowledge Graph Generation **Rule Creation and Execution**

Summary of MapVOWL

Research Question 1: How can we design visualizations that improve the cognitive effectiveness of visual representations of knowledge graph generations rules?

MapVOWL is visual notation for knowledge graph generation rules.

Hypothesis 1: MapVOWL improves the cognitive effectiveness of the generation rule visual representation to generate knowledge graphs compared to using RML directly.

MapVOWL is preferred over RML.

Summary of Resglass

Research Question 3: How can we score and rank rules and ontology terms for inspection to improve the manual resolution of inconsistencies?

RMLEditor is a rule-driven method to resolve inconsistencies.

Hypothesis 3: The automatic inconsistency-driven ranking of Resglass improves, compared to a random ranking, by at least 20% the overlap with experts' manual ranking.

Resglass can improve resolution of inconsistencies, partly due to its ranking.

Differences between R2RML and RML on test case-level

	R2RML	**RML**
Data errors	Process stops	Process doesn't stop: best effort
Inverse expressions: to optimize execution	Tested, but can't see difference in output	Not tested
SQL-specific features	Tested	Depending on data source language
Null values	Tested	Not supported for CSV and XML
Spaces in columns	Tested	Not supported for XML

Summary of RML test cases

Research Question 4: What are the characteristics of test cases for processors that generate knowledge graphs from heterogeneous data sources independent of languages' specifications?

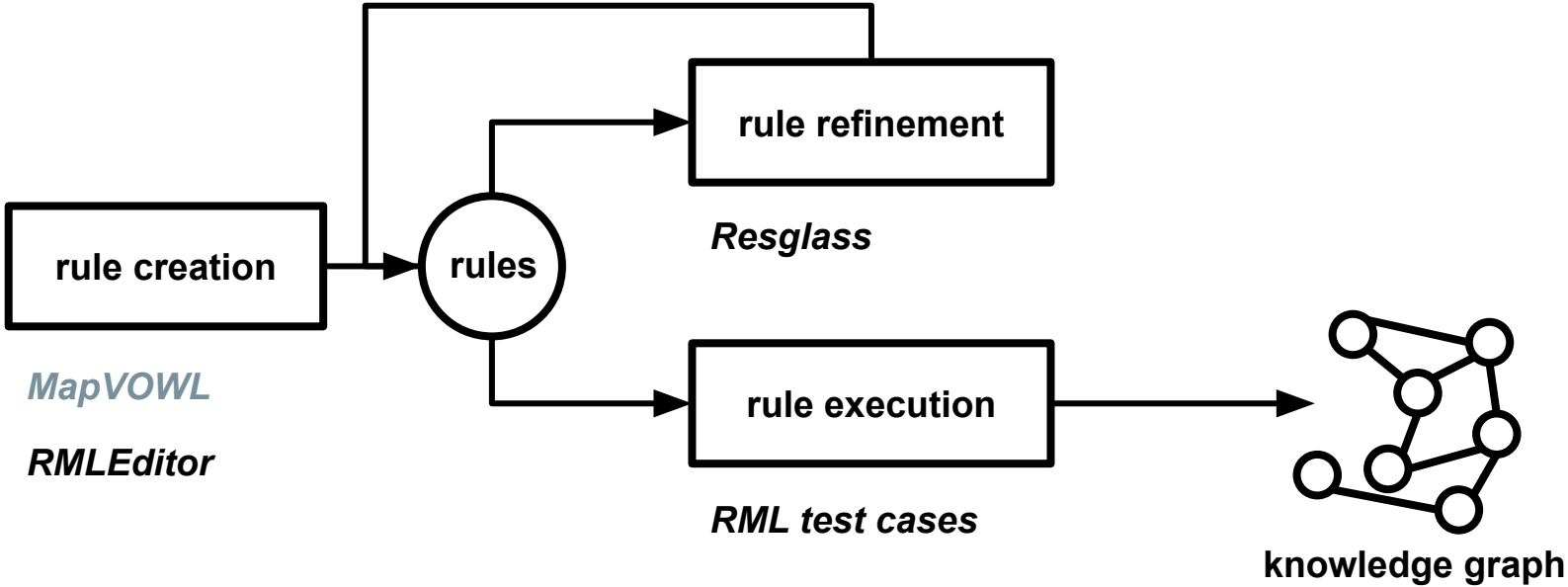
Databases vs other data sources

Tabular vs hierarchical data formats

SQL vs other data source languages

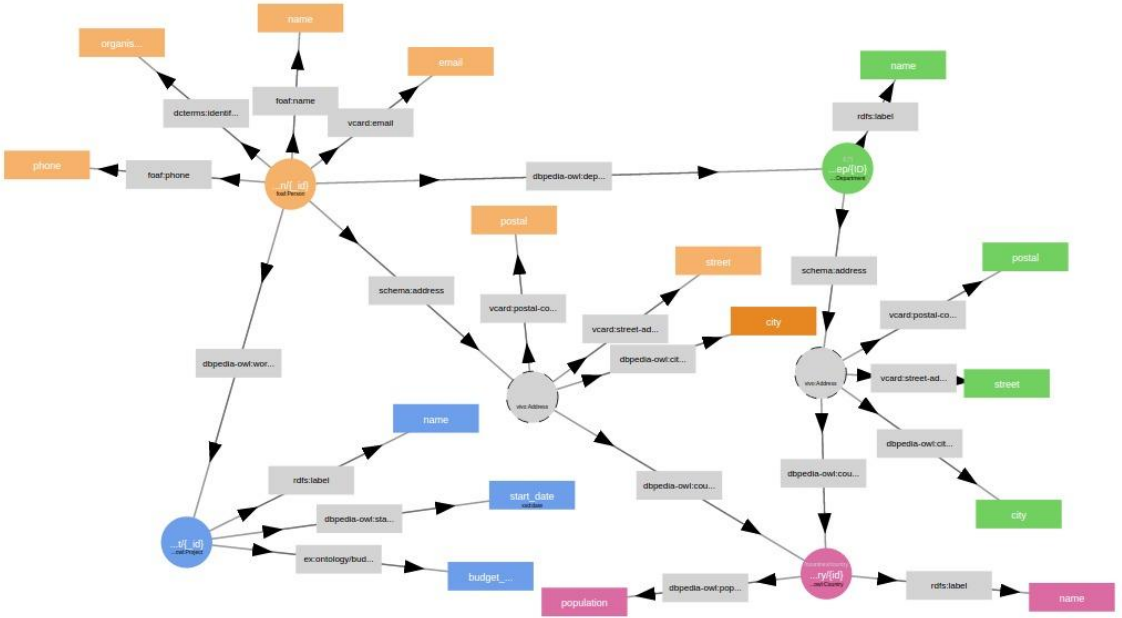
RML test cases help define and explore these characteristics.

Overview of contributions



Highest level

Shows everything

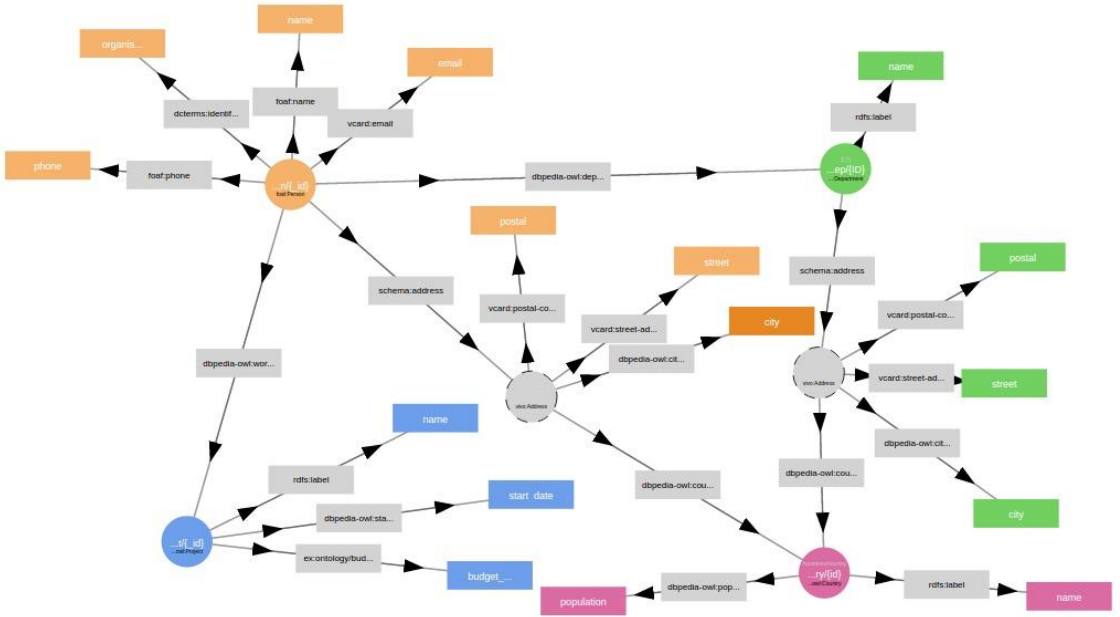


High level

Shows everything

Except

- Datatype of literal
- Language of literal



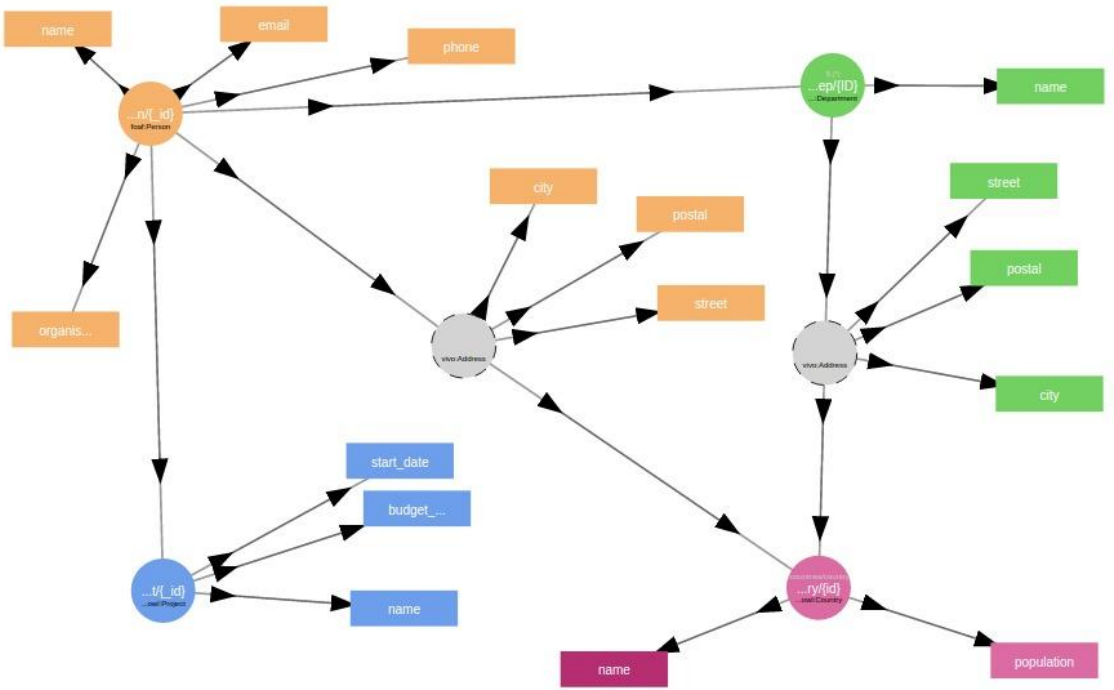
Moderate level

Shows everything

Except

- Datatype of literal
- Language of literal

- Relationships on arrows



Low level

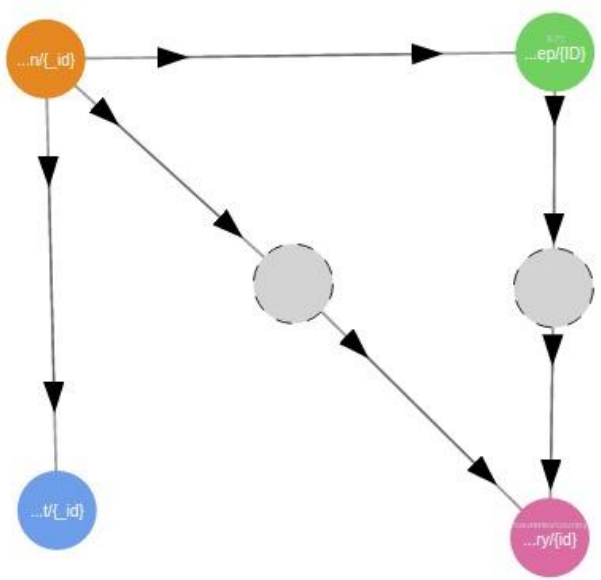
Shows everything

Except

- Datatype of literal
- Language of literal

- Relationships on arrows

- Relationships that are not between two entities



Lowest level

Shows everything

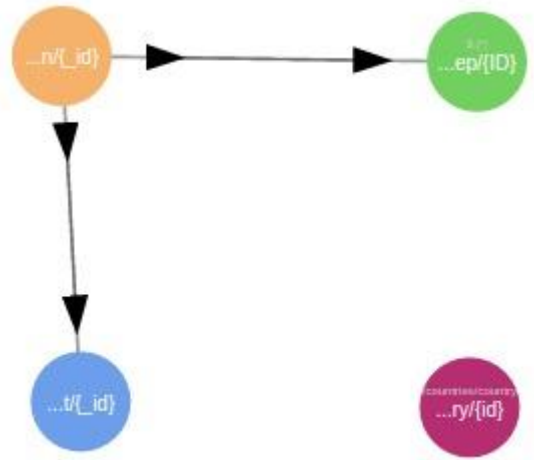
Except

- Datatype of literal
- Language of literal

- Relationships on arrows

- Relationships that are not between two entities

- Entities without an links



Summary of RMLEditor

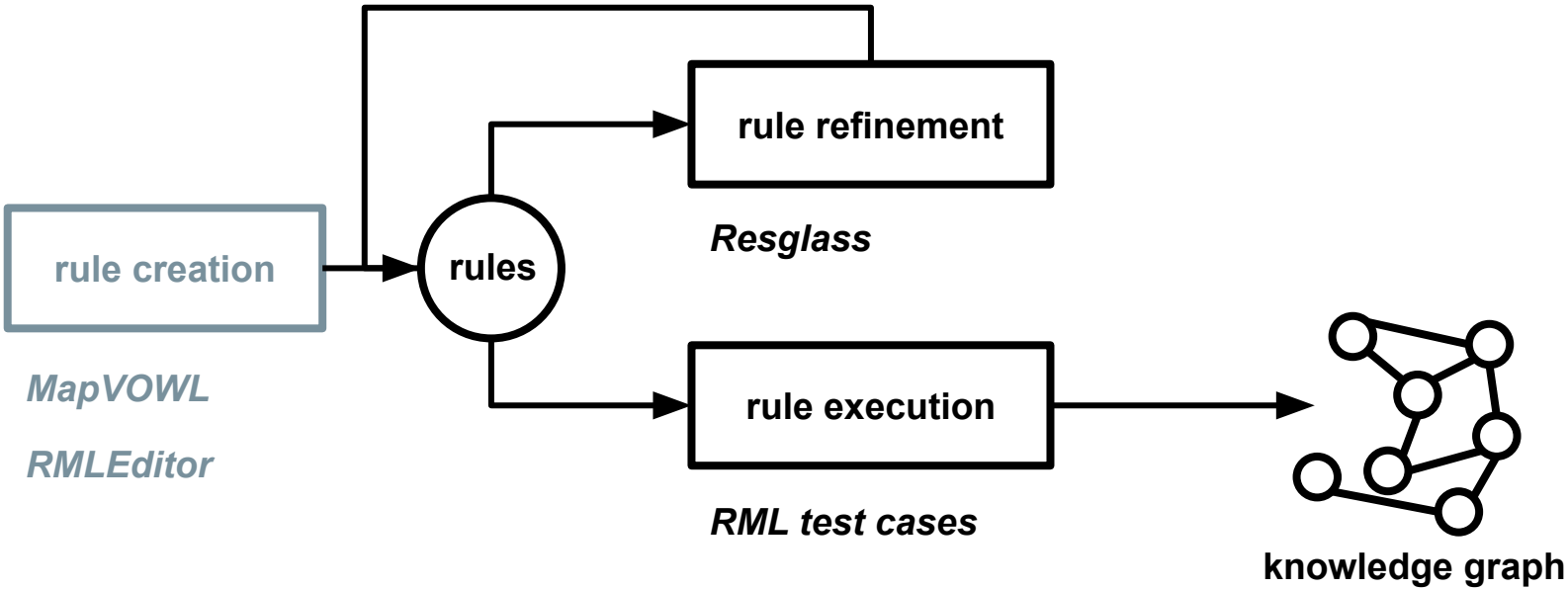
Research Question 2: How can we visualize the components of a knowledge graph generation process to improve its cognitive effectiveness?

RMLEditor is graph-based rule editor.

Hypothesis 2: The cognitive effectiveness provided by the RMLEditor’s GUI improves the user’s performance during the knowledge graph generation process compared to RMLx.

RMLEditor is preferred over RMLx.

Overview of contributions



Example: rules for a person and location combined

id	full-name	place
001	Professor Oak	lab

id	category	city
lab	Laboratory	Pallet Town
h1	house	Pallet Town
h2	house	Pallet Town

Every row in the table is a person.

“name” is the official full name of a person.

“place” refers to a person’s current location **in other table**.

Every row in the table is a location.

“category” is the type of a location.

“city” refers to a location’s city it is in.

Example: RML rules

```
:map_anomaly_0 rml:logicalSource :source_0.
:source_0 a rml:LogicalSource;
      rml:source "input.json";
      rml:iterator "$";
      rml:referenceFormulation ql:JSONPath.
:map_anomaly_0 a rr:TriplesMap;
      rdfs:label "anomaly".
:s_0 a rr:SubjectMap.
:map_anomaly_0 rr:subjectMap :s_0.
:s_0 rml:reference "id".
:pom_0 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_0.
:pm_0 a rr:PredicateMap.
:pom_0 rr:predicateMap :pm_0.
:pm_0 rr:constant rdf:type.
:pom_0 rr:objectMap :om_0.
:om_0 a rr:ObjectMap;
      rr:template
"http://IBCNServices.github.io/Folio-Ontology/Folio.owl#
{anomaly.type}";
      rr:termType rr:IRI.
```

```
:pom_1 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_1.
:pm_1 a rr:PredicateMap.
:pom_1 rr:predicateMap :pm_1.
:pm_1 rr:constant dc:description.
:pom_1 rr:objectMap :om_1.
:om_1 a rr:ObjectMap;
      rml:reference "anomaly.description";
      rr:termType rr:Literal.
:pom_2 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_2.
:pm_2 a rr:PredicateMap.
:pom_2 rr:predicateMap :pm_2.
:pm_2 rr:constant folio:hasCauseDescription.
:pom_2 rr:objectMap :om_2.
:om_2 a rr:ObjectMap;
      rml:reference "anomaly.explanation";
      rr:termType rr:Literal.
:pom_3 a rr:PredicateObjectMap.
:map_anomaly_0 rr:predicateObjectMap :pom_3.
:pm_3 a rr:PredicateMap.
:pom_3 rr:predicateMap :pm_3.
:pm_3 rr:constant sosa:usedProcedure.
:pom_3 rr:objectMap :om_3.
```

Overview

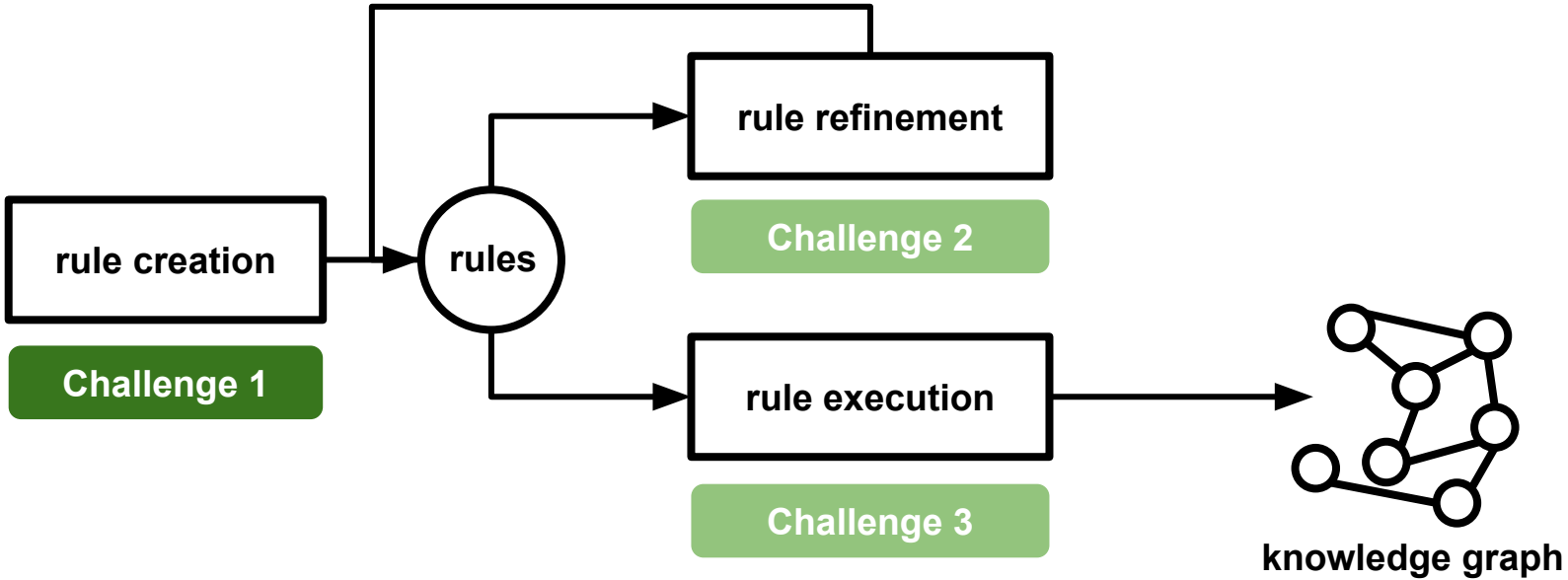
Moving from Web to Semantic Web

Challenges

Research questions and hypotheses

Contributions

Conclusions


rule refinement
Challenge 2
rule creation
rules
Challenge 1
rule execution
Challenge 3
knowledge graph

Different tools to create rules

Different approaches
- Forms
- Graph-based visualizations

Support different data formats
- Only databases
- Only files

(No) support for data transformations (e.g., capitalize names)

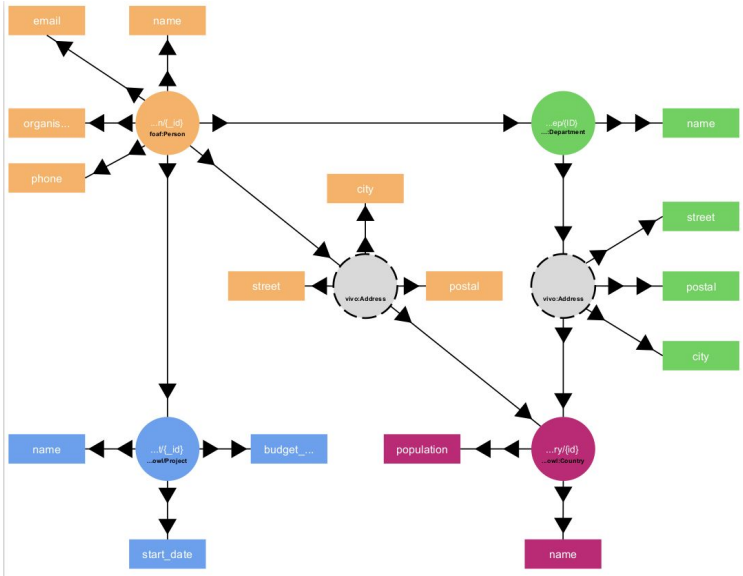
Design of tools and GUIs not thoroughly investigated

Research Question 1:
How can we design **visualizations** that **improve** the **cognitive effectiveness** of visual representations of **knowledge graph generation rules**?

Human words: how can we make the visualizations better?

Introduce MapVOWL to address Research Question 1

Visual notation for knowledge generation rules



Hypothesis 1:
MapVOWL improves the **cognitive effectiveness** of the generation rule **visual representation** to generate knowledge graphs **compared to using RML directly**.

Research Question 2:
How can we **visualize the components** of
a knowledge graph generation process to **improve**
its **cognitive effectiveness**?

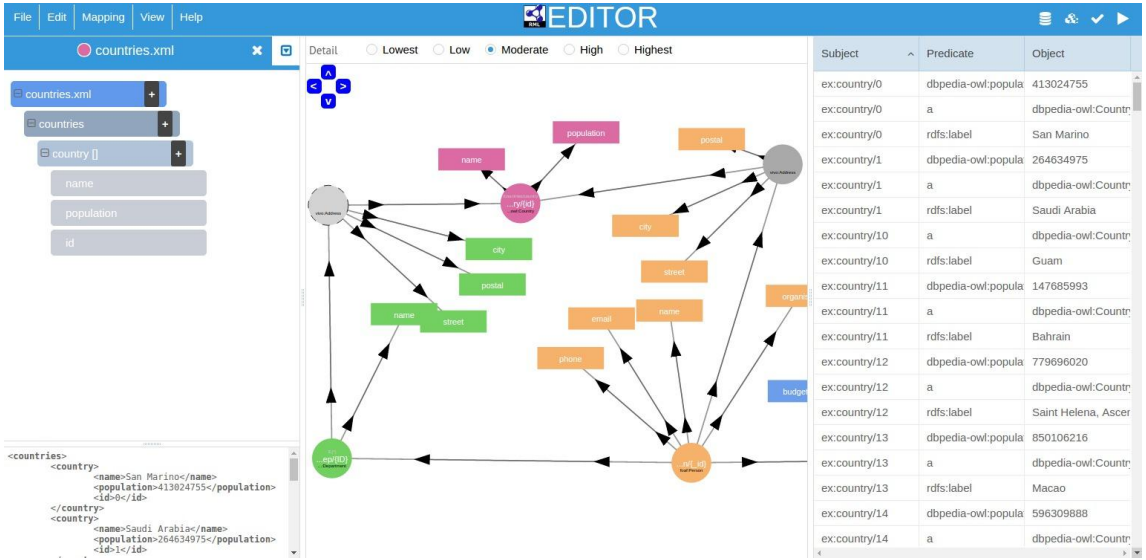
Human words: how can we make the GUIs better?

Hypothesis 1:
MapVOWL improves the **cognitive effectiveness** of the generation rule **visual representation** to generate knowledge graphs **compared to using RML directly**.

Human words: we hope MapVOWL is better than RML.

Introduce the RMLEditor to address Research Question 2

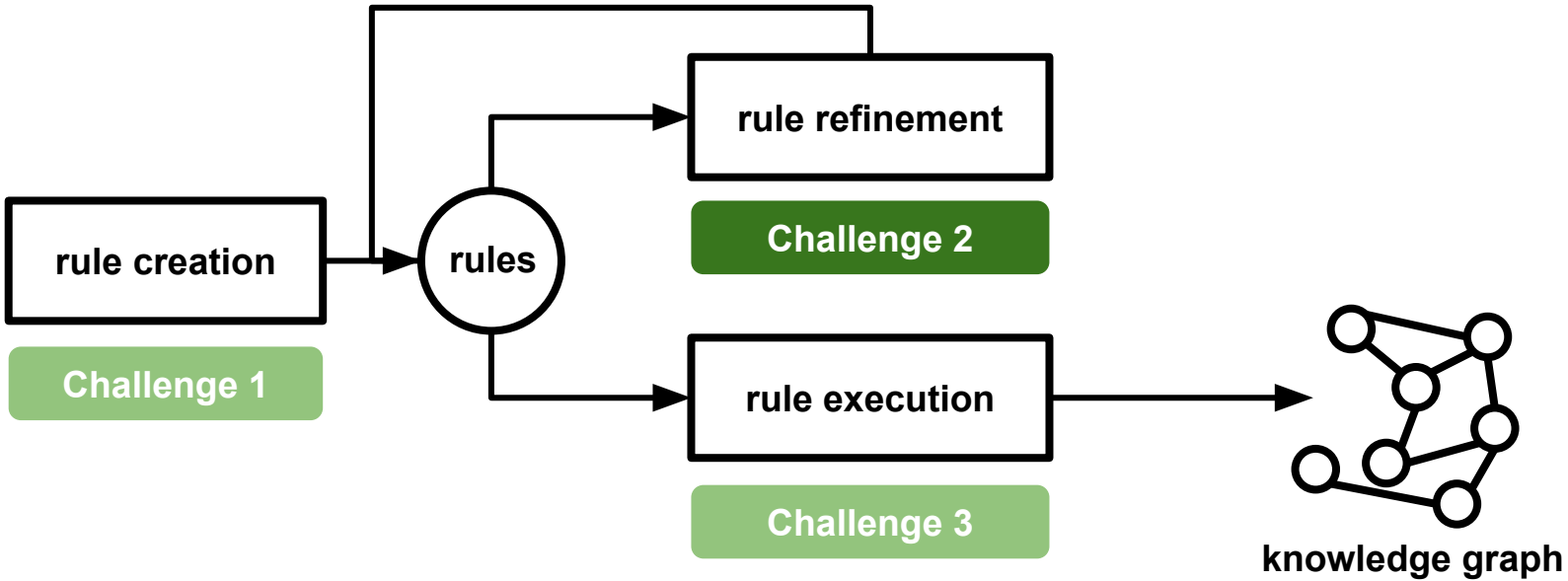
Graph-based rule editor



Hypothesis 2:
The **cognitive effectiveness** provided by
the **RMLEditor's GUI improves the user's performance**
during the knowledge graph generation process
compared to RMLx.

Hypothesis 2:
The **cognitive effectiveness** provided by
the **RMLEditor's GUI improves the user's performance**
during the knowledge graph generation process
compared to RMLx.

Human words:
we hope the RMLEditor is better than RMLx.


rule refinement
Challenge 2
rule creation
rules
Challenge 1
rule execution
Challenge 3
knowledge graph

Resolving inconsistencies is not straightforward

Use case: DBpedia (knowledge graph generated from Wikipedia)

- Total # rules: > 1,200,000
- # inconsistencies: > 2,000
- # rules involved in at least 1 inconsistency: > 1,300

Where do we start?

Which rules do we check first?

Which inconsistencies do we check first?

Research Question 3:
How can we **score and rank rules and ontology terms** for inspection to **improve the manual resolution of inconsistencies**?

Human words:
where do we start when fixing the problems?

Introduce Resglass
to address Research Question 3

Rule-driven method for the resolution of inconsistencies in rules and ontologies.

Introduce Resglass
to address Research Question 3

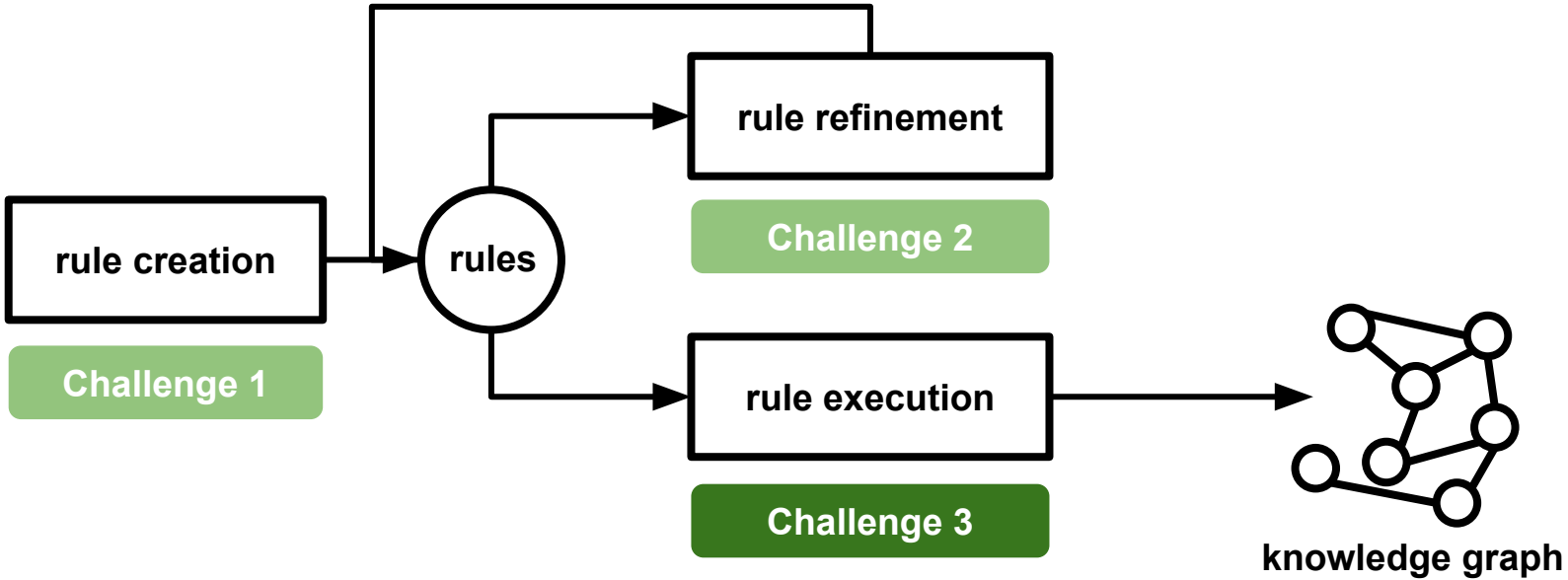
Rule-driven method for the resolution of inconsistencies in rules and ontologies.

Due to the lack of a screenshot here's a dog riding a skateboard.

Hypothesis 3:
The **automatic inconsistency-driven ranking of Resglass** improves, compared to a random ranking, by at least **20% the overlap with experts' manual ranking**.

Hypothesis 3:
The **automatic inconsistency-driven ranking of Resglass** improves, compared to a random ranking, by at least **20% the overlap with experts' manual ranking**.

Human words: we hope Resglass is better than doing something random.

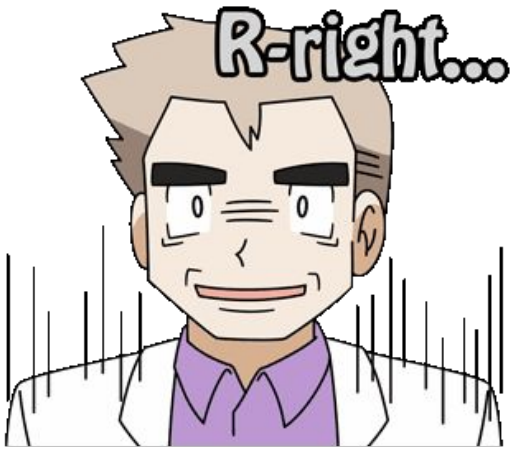

rule refinement
Challenge 2
rule creation
rules
Challenge 1
rule execution
Challenge 3
knowledge graph

Test cases to determine conformance of processors to languages

Test case = a set of actions executed to verify a particular feature or functionality of the software application.

Test cases to determine conformance of processors to languages

Test case = a set of actions executed to verify a particular feature or functionality of the software application.



No test cases for all languages

Only for languages that work with databases.

No for languages that work with different data sources and formats.

→ Processors that work with different data sources and formats not tested.

Research Question 4:
What are the **characteristics of test cases** for processors that generate knowledge graphs from **heterogeneous data sources independent of languages' specifications**?

Human words: what is part of a test case and what not?

Introduce initial set of conformances test cases for RML to address Research Question 4

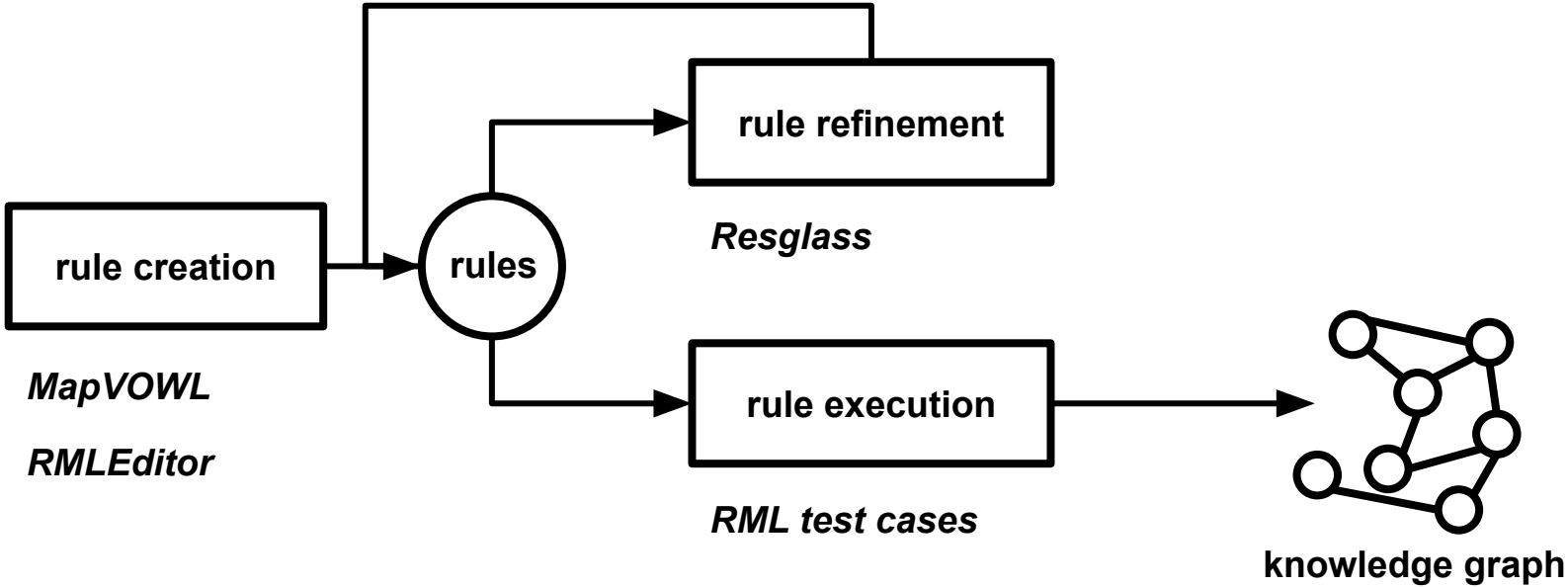
Databases (like R2RML)

CSV files

XML files

JSON files

Overview of contributions



Adding meaning in practice

Knowledge graph consists of zero or more triples.

One triple consists of a subject, predicate, and object.

Subject *and* predicate *and* object have explicit meaning defined.

Example: triple

subject

has the name

predicate

Professor Oak

object

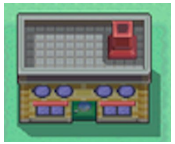
Subjects are in most cases links

https://google.com/

https://ugent.be

https://kanto.edu/oak

https://kanto.map/lab



Why links?

Uniquely identify a resource.

Possible to follow a link → get more information.

Predicates are links

name → http://schema.org/name

location → http://schema.org/location

city → http://dbpedia.org/ontology/city

Objects are in most cases links and literals

Links

- https://kanto.edu/oak
- https://kanto.map/lab

Literals

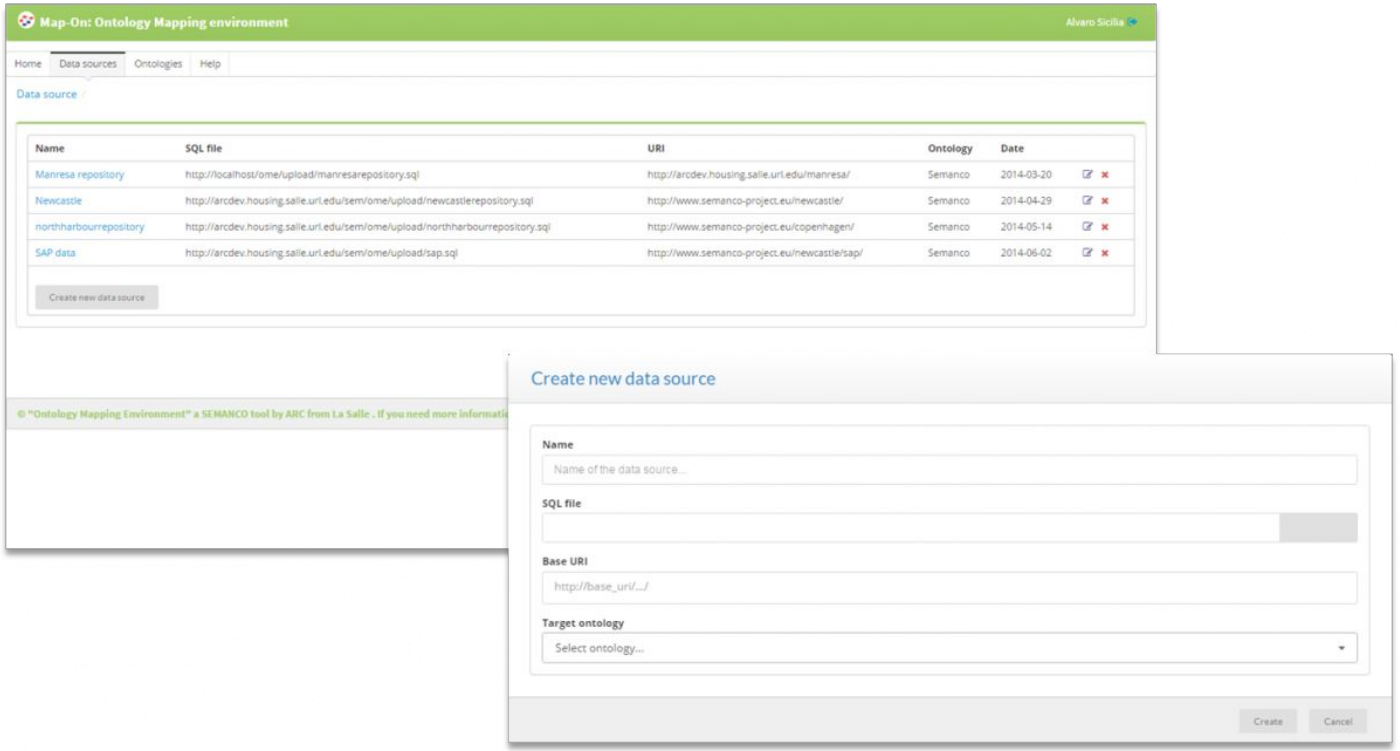
- Text: “Professor Oak”, “Pallet Town”
- Numbers: 1, 100, 1000

Example: triple

has the name **Professor Oak**

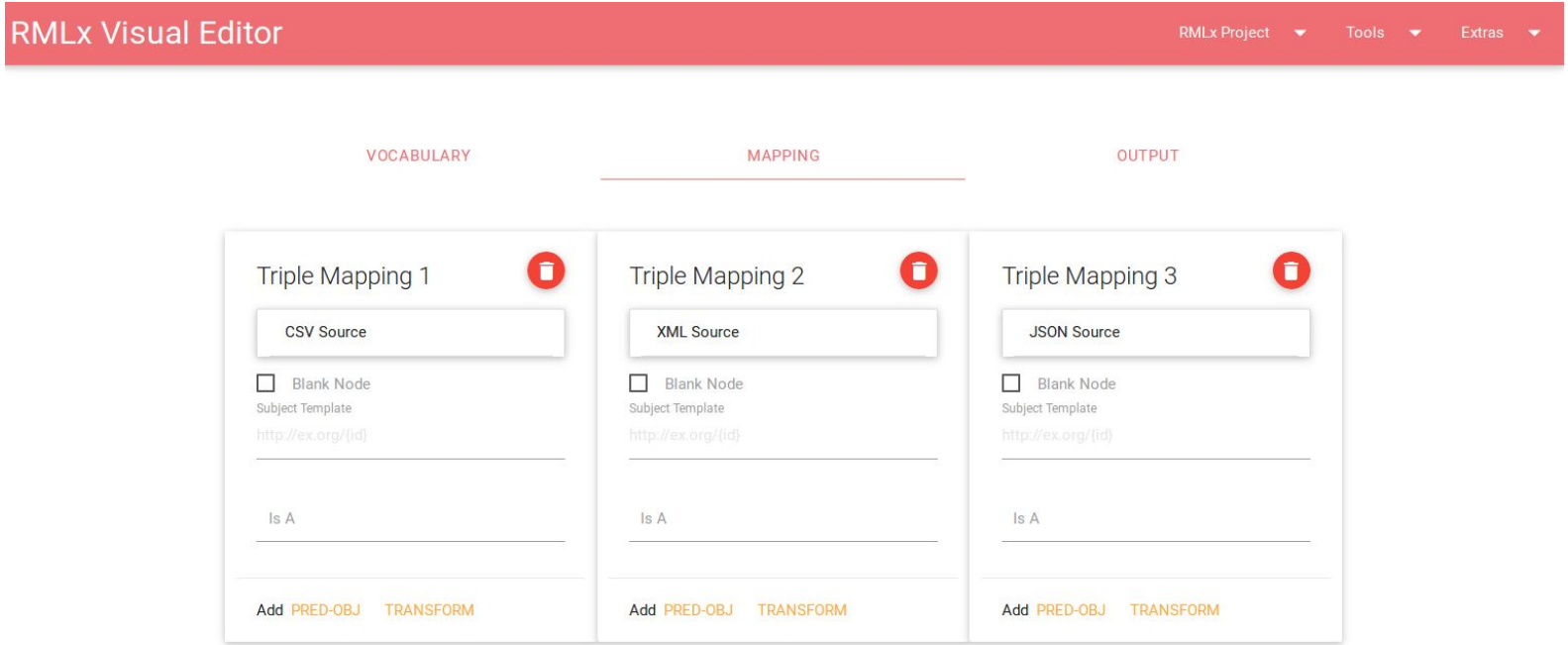
https://kanto.edu/oak http://schema.org/name “Professor Oak”

Tools to create rules: Map-On



Tools to create rules: Juma

Tools to create rules: RMLx



Example: rules for person via MapVOWL

Example: rules for person via MapVOWL

id	full-name	place
oak	Professor Oak	lab

Every row in the table is a person.

Link of person is https://kanto.edu/ + id.

“full-name” is the name of a person.

“place” refers to a person’s location.

Example: rules for person via MapVOWL

id	full-name	place
oak	Professor Oak	lab

Every row in the table is a person.

Link of person is https://kanto.edu/ + id.

“full-name” is the name of a person.

“place” refers to a person’s location.

Example: rules for person via MapVOWL

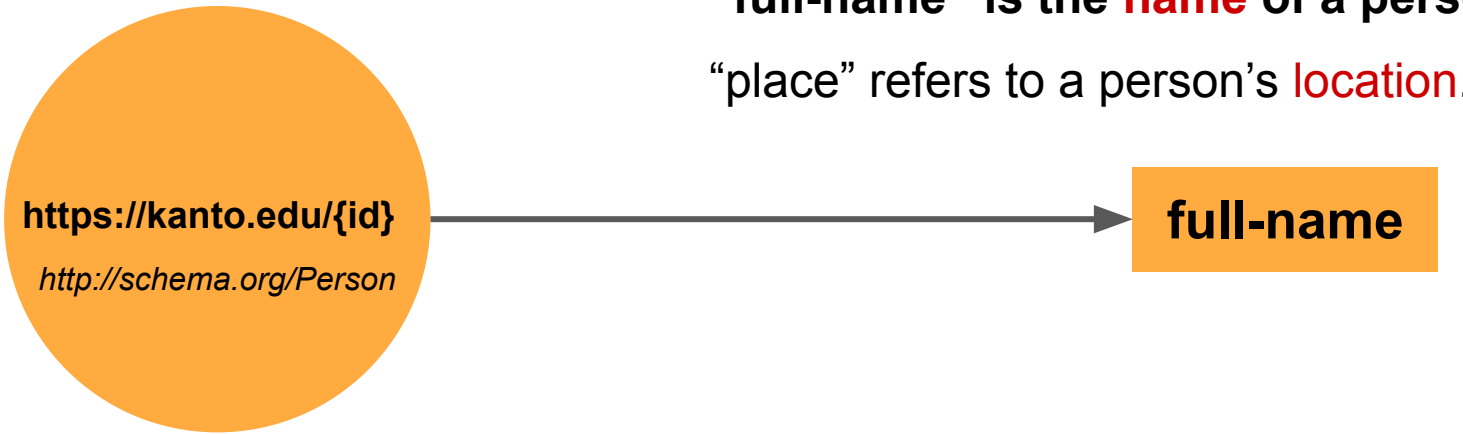
id	full-name	place
oak	Professor Oak	lab

Every row in the table is a person.

Link of person is https://kanto.edu/ + id.

“full-name” is the name of a person.

“place” refers to a person’s location.



Example: rules for person via MapVOWL

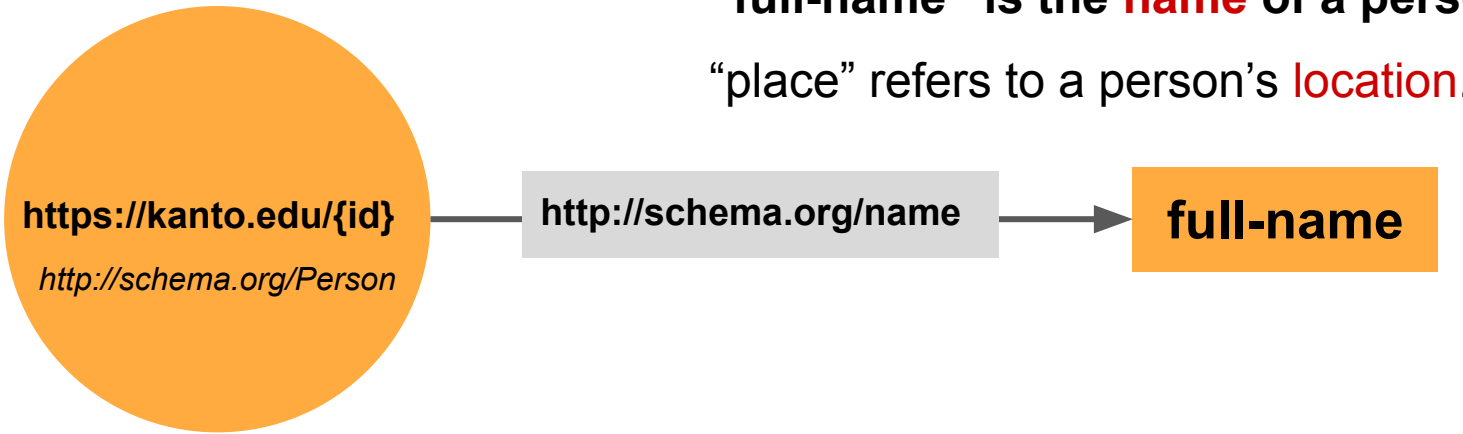
id	full-name	place
oak	Professor Oak	lab

Every row in the table is a person.

Link of person is https://kanto.edu/ + id.

“full-name” is the name of a person.

“place” refers to a person’s location.



Example: rules for person via MapVOWL

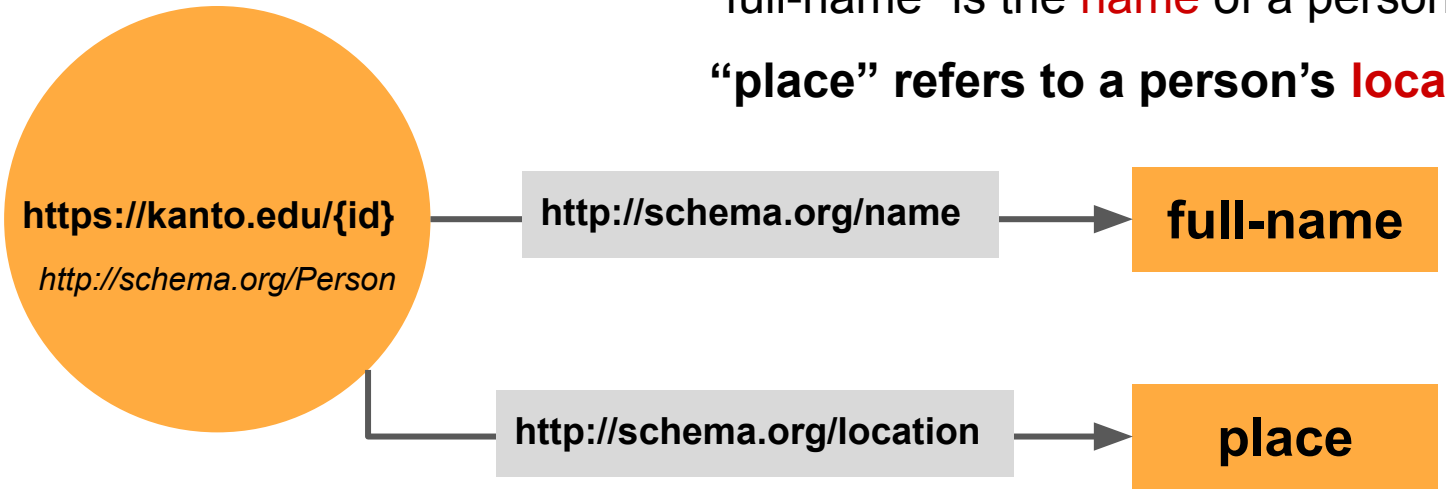
id	full-name	place
oak	Professor Oak	lab

Every row in the table is a person.

Link of person is https://kanto.edu/ + id.

“full-name” is the name of a person.

“place” refers to a person’s location.



Example: rules for location via MapVOWL

id	category	city
lab	Laboratory	Pallet Town

Every row in the table is a location.

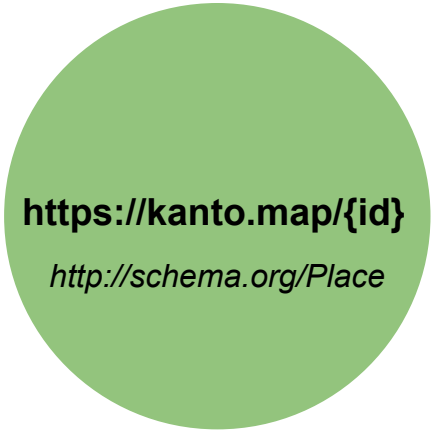
Link of location is https://kanto.map/ + id.

“category” is the type of a location.

“city” refers to a location’s city it is in.

Example: rules for location via MapVOWL

id	category	city
lab	Laboratory	Pallet Town



Every row in the table is a location.

Link of location is https://kanto.map/ + id.

“category” is the type of a location.

“city” refers to a location’s city it is in.

Example: rules for location via MapVOWL

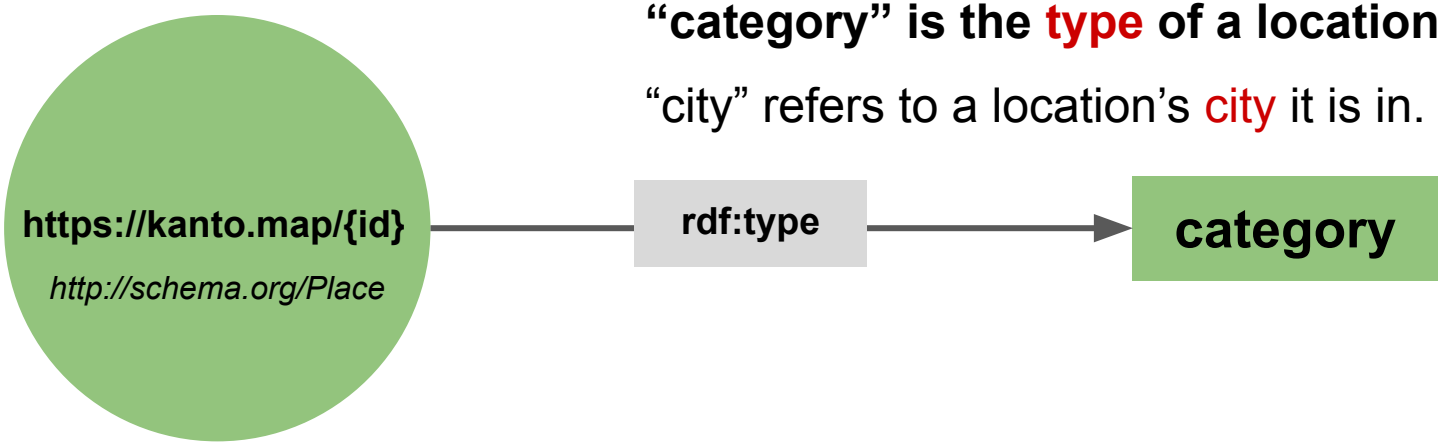
id	category	city
lab	Laboratory	Pallet Town

Every row in the table is a location.

Link of location is https://kanto.map/ + id.

“category” is the type of a location.

“city” refers to a location’s city it is in.



Example: rules for location via MapVOWL

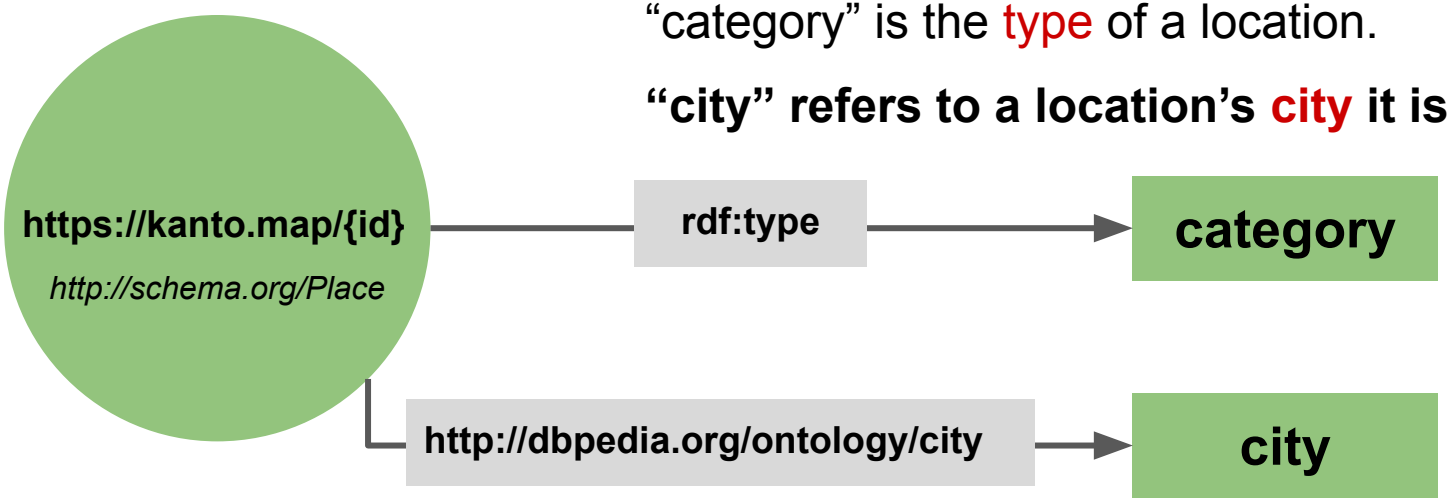
id	category	city
lab	Laboratory	Pallet Town

Every row in the table is a location.

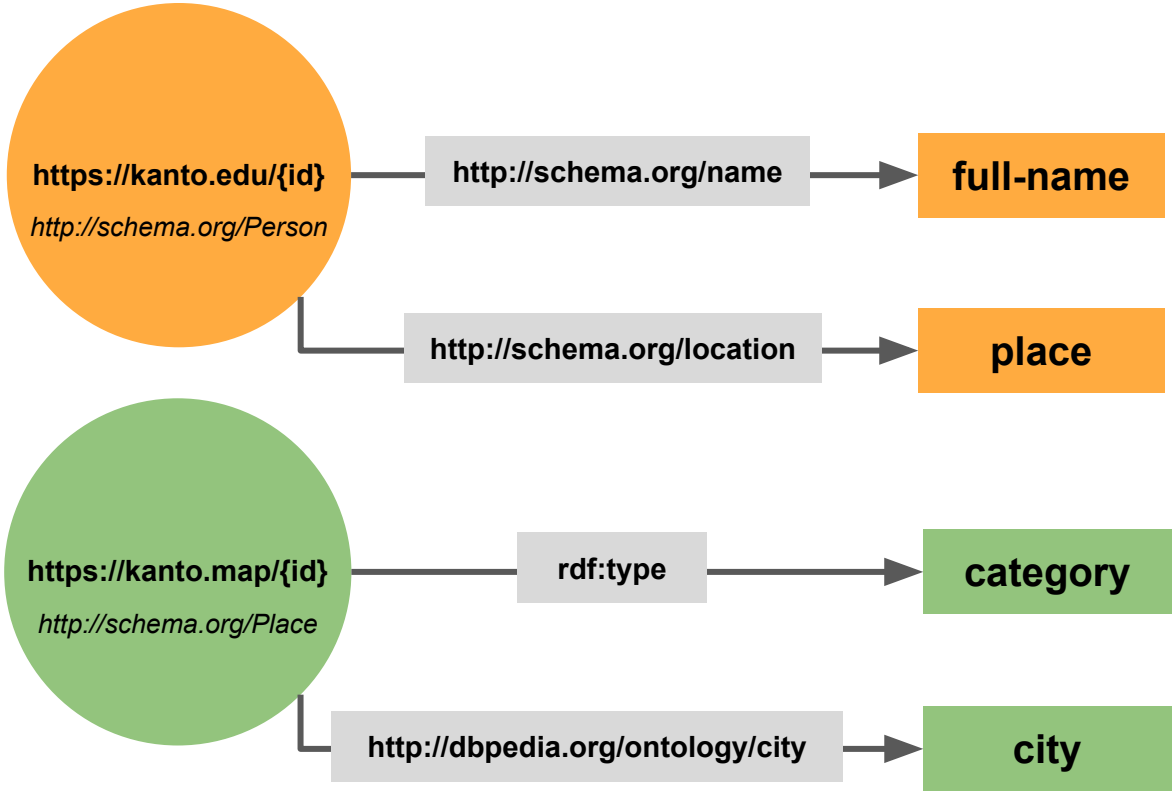
Link of location is https://kanto.map/ + id.

“category” is the type of a location.

“city” refers to a location’s city it is in.



Example: no link between persons and locations



Example: add link between persons and locations