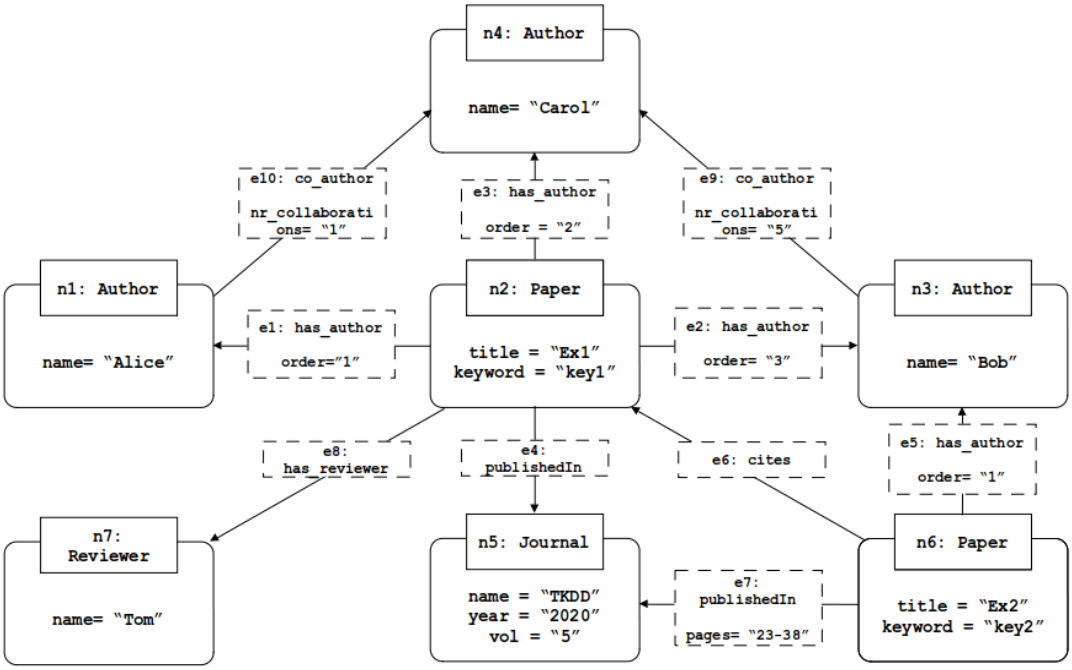
OPEN DATA / SEMANTIC DATA MANAGEMENT EXAM

5th of June 2020. *The exam will take **2 hours and a half**. Answer each question in the provided space.* Answers out of such space will not be considered. Further, clearly read the instructions how to answer. Answers not following the format set might not be considered.

Name:

QUESTION 1. PROPERTY GRAPHS (25%)

Given the property graph in Figure 1, answer the following questions.



a. **How and Why** Q1 and Q2 would differ in their result when using Cypher?

```
Q1. MATCH (x1:Author)<-[:has_author]-(:Paper{title:"Ex1"})-[:has_author]->(x2:Author)
    RETURN x1,x2
```

alice bob
bob alice

```
Q2. MATCH (x1:Author)<-[:has_author]-(x3:Paper{title: "Ex1"})
    MATCH (x2:Author)<-[:has_author]-(x3)
    RETURN x1,x2
```

alice bob
bob alice
alice alice
bob bob

Write your answer in the available space right below

b. In the research article publication domain, the heads of the research labs typically are the last ones signing a paper (i.e., they are the last authors in the list of authors of a paper). Write the Cypher query that identifies the research heads.

Write your answer in the available space right below

c. Assuming the graph in Figure 1.

Find a graph algorithm that once instantiated can find some potentially suspicious behavior in the domain of article publication. Instantiate the algorithm providing its parameters (you can either use Neo4j signature or your own algorithm signature but then clearly state the meaning of each parameter), and then intuitively interpret the output of the algorithm under the domain considered.

Write your answer in the available space allocated for each item

- Algorithm signature (in the form `algorithm1(param1, param2, … , paramN))`. Explain the algorithm chosen as well as the meaning of each parameter

- Instantiation of the algorithm signature

- Intuitive interpretation of the analysis achieved (not further than the end of this page)

QUESTION 2. LARGE PROPERTY GRAPHS PROCESSING (25%)

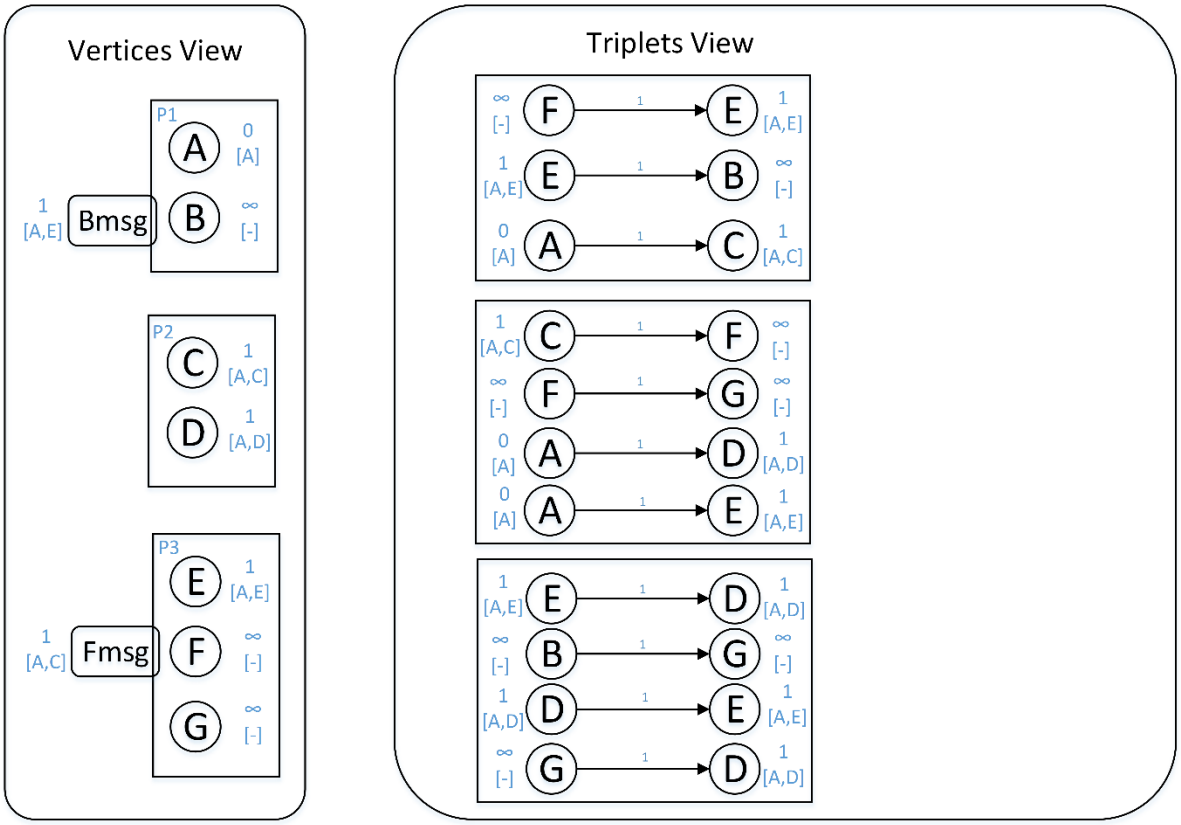
Consider the vertex and triplets views **in Pregel implementation** of the TLAV algorithm and their corresponding partitions depicted below for a given superstep.

Important notes:

- Each **vertex** is depicted with its current state (i.e., **the current shortest path length** and **the corresponding path**)
- Each **message** is depicted with its content (i.e., **the current shortest path length** and **the corresponding path**).
- Each **edge** has a weight depicted in the triplets view.
- The program benefits from the optimization achieved by using **triplets view** in Pregel.

a. Taking into account that the node kernel (in Pregel, the Apply function) runs the shortest path algorithm and that the incoming messages are the ones shown the figure below, you must specify:
   - The communication between **vertex partitions** and **triplets partitions** generated in this superstep.
   - Messages **produced** and **sent** from each triplet partition to the subsequent superstep. With each message sent, **indicate the corresponding partition (P1/P2/P3)** where the message is sent.

<u>Draw your solution over the figure below</u>



b. Now, compare the previous execution in Pregel with the theoretical TLAV algorithm.

What is the number of messages sent from vertices to edges/triplets partitions in the theoretical TLAV algorithm and Pregel implementation?

Theoretical TLAV: _____

Pregel: _____

Will this ratio always hold for any execution? **Justify your answer using the allocated space**.

_____

_____

_____

_____

_____

What is the number of messages sent from edges/triplets partitions to the subsequent superstep in the theoretical TLAV algorithm and Pregel implementation?

Theoretical TLAV: _____

Pregel: _____

Will this ratio always hold for any execution? **Justify your answer using the allocated space**.

_____

_____

_____

_____

_____

QUESTION 3. KNOWLEDGE GRAPHS AND DATA INTEGRATION (50%)

We aim at developing an integrated system to analyse the impact of co-morbidities (i.e., the presence of one or more additional conditions co-occurring with a primary condition or disease) in the Covid-19 diagnosis and treatment. Since reacting fast is key, we aim at developing a **graph-based virtual data integration system** spanning several health-related data sources. Specifically, we choose to implement the **ontology-mediated querying system** seen in the lectures.

a. Use RDFS to define the target schema of the integration system:

The two key elements we want to analysis are the concepts of diagnosis and treatment. On the one hand, a diagnosis event relates a patient with a disease. This event must include the following information: date of diagnosis, place of diagnosis, symptoms considered to make the diagnosis and by whom it was diagnosed (i.e., a doctor or health practitioner). On the other hand, a treatment event relates a patient with a medication plan. It must include the following information: the patient, the medicine given and the disease for what it was prescribed.

Patients are identified by their health-system card id (HSId) and we must record their complete name (as a single element), birthplace and birthdate. Medicines are identified by their international code (MIC) and besides that, they have a name and belong to a family of medicines. For diseases, we have an international code (DIC) that identifies them and a common name used to name it. Symptoms are identified by a common international code (SID) and a common name to refer to them.

This RDFS graph must be compliant with the constraints imposed in the **ontology-mediated querying** strategy discussed and **meet all of them**. Use colours to distinguish concepts and features so that you do not need to include the metamodel in your solution. Use green for concepts and yellow for features, as we did in the lectures.

Use the available space in the next page to embed your solution.

b. Using the IRIs you created for your RDFS graph, express the following information in Description Logics first, and as OWL statements later. If you do not know how to insert the corresponding mathematical symbols, replace them by capital keywords (e.g., SUBSUMPTION).

Patients must be either registered patients or unregistered patients. But it is mandatory everybody is of one or another type. Further, if someone is registered s/he cannot be unregistered and vice versa. Further, it is mandatory a patient has a HSId.

- Your DL statements to express these constraints:

- Your equivalent OWL statements expressing the DL statements above:

c. Consider an excerpt of the sources. For this exercise, we will **just** consider the following wrappers:

`W1. Plan(HSId, MIC, unit, quantity)`

W1 exposes data from the medicine plan system.

(…)

`W3. Diagnosis(HSId, DIC, diagnosedBy, when, where)`

W3 exposes data from the ERP set up in the country health facilities.

(…)

`W5. Diagnosis(patientId, Covid19, diagnosedBy, when, where)`

W5 exposes data from a system set up to trace Covid-19.

And the following function known to be true: F_{HSId}(patientId) = HSId

Let us build the graph-based virtual data integration system:

Draw the LAV mapping for W3. Do as follows. On the left, place the source diagrammatic graph representing W3 (like shown in the lectures). On the right, represent the target schema subgraph defining the LAV for W3 over the target schema (you may want to paste the corresponding diagrammatic subgraph or paste it all and circle the subgraph as we did in the lectures). Use red nodes to represent wrappers and blue nodes to represent attributes.

d. Now, we can query the system looking for patients taking anti-coagulant medicines (known to be problematic for Covid-19) to provide a different treatment for those patients. The query to pose is *retrieve all patients (i.e., its HSId) diagnosed with Covid-19 (whose DIC is who:Covid19) that were prescribed one of the following medicines for any previous patology*: *rivaroxaban (MID: who: rivaroxaban) or simtrom (MID: who: simtrom)*.
   - Write the SPARQL query[1] over your target schema to answer this query. The query must compile.

   - Express the query over the wrappers resulting from applying the LAV rewriting algorithm explained in the lectures. The rewriting must be complete. The result must be an algebraic query over the wrappers (you may want to use JOIN, SELECTION, PROJECTION, etc. to express the operations instead of greek letters).

[1] You can ignore the temporal aspect of the query