University of Central Lancashire
UCLan

CO3409
Distributed Enterprise Systems

Ontology/schema design
Querying linked data
Concurrent process models

Where opportunity creates success

Ontology/schema design

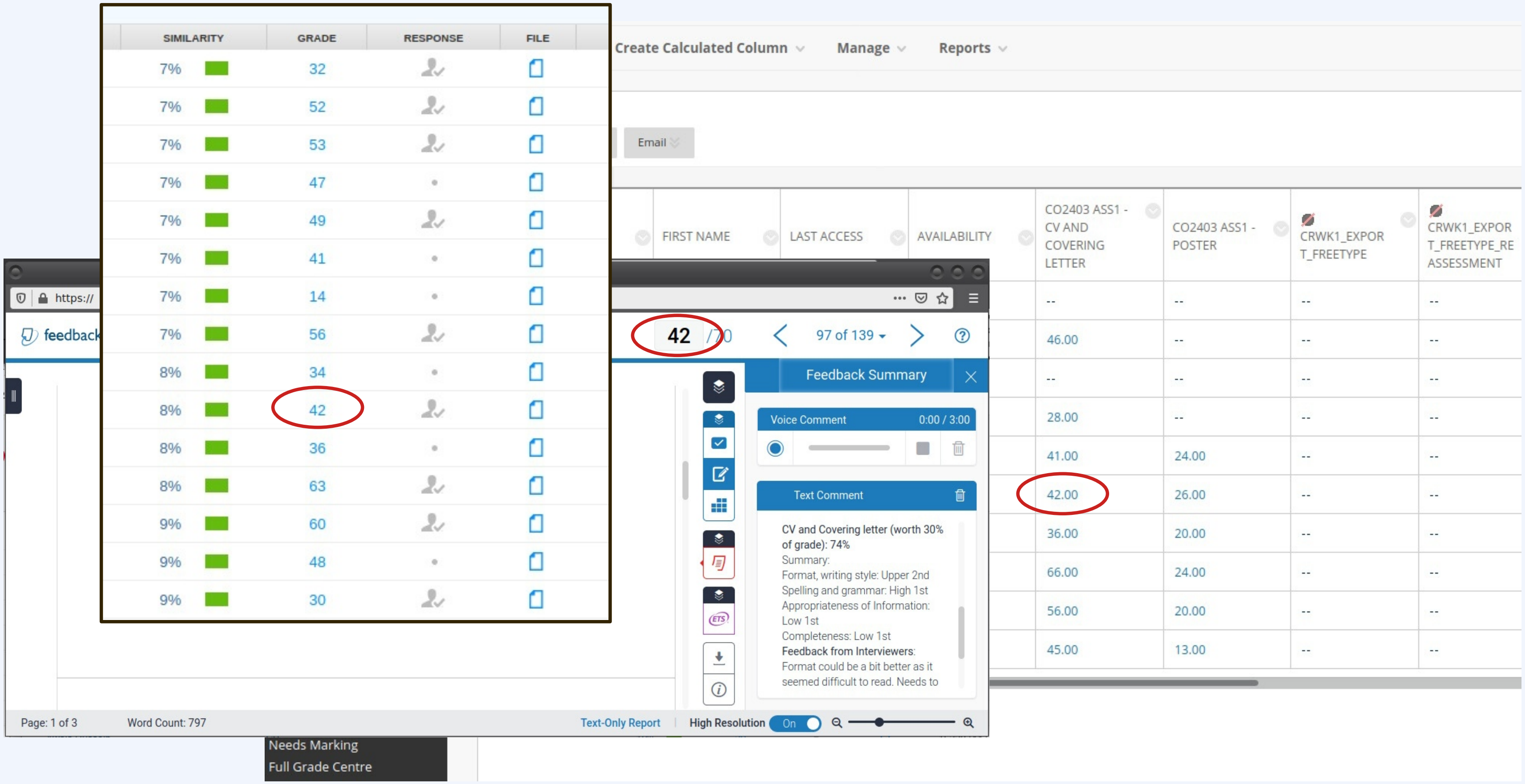
University of Central Lancashire
UCLan

CO3409

10th March 2022

University of Central Lancashire
UCLan

Schema/ontology design via competency questions

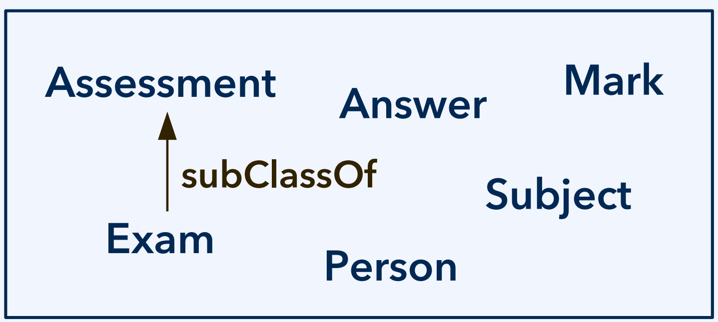


Another strategy for building an ontology consists in gathering **competency questions** and including the employed concepts and relations in the ontology.

Schema/ontology design via competency questions

Example **competency questions** from discussion in previous lecture:

- *(What are the numerical values of the marks? "42" in the example case.)*
- What **exams** and **subjects** are the **marks** for?
  *(Maybe reformulate as: What **assessments** are the **marks** for? What **subjects** are the **assessments** about?)*
- What are the maximum (numerical values of) marks for the **assessments**?
- Who are the **marks** for? (What **person** are the **marks** attributed to?)
- What is the submission time stamp of the **answers**/submitted material?

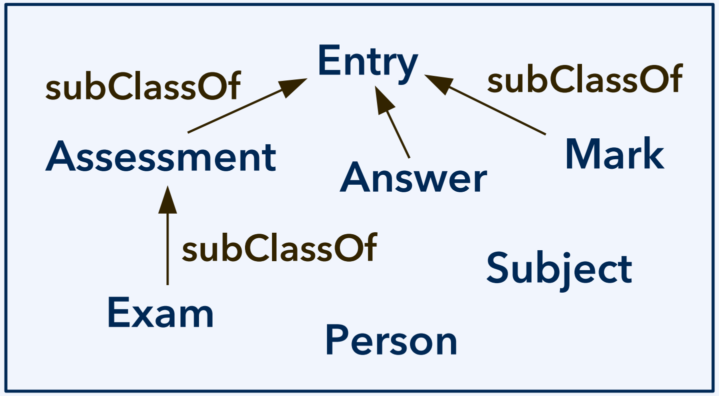


concepts

Schema/ontology design via competency questions

Object properties and **datatype properties** (XML schema data types[1]).

- *(What **are the numerical values of the marks**? "42" in the example case.)*
- What exams and subjects are the marks for?
  *(Maybe reformulate as: What assessments **are the marks for**? What subjects **are the assessments about**?)*
- What are the ***maximum (numerical values of) marks*** for the assessments?
- Who are the marks for? (What person are the marks ***attributed to***?)
- What ***is the submission time stamp of*** the answers/submitted material?



concepts

isMarkFor (Mark → Assessment)
isAssessmentAbout (Assessment → Subject)
isAttributedTo (Mark → Person)

isNumericalMark (Mark → decimal)
hasMarkMaximum (Assessment → decimal)
hasSubmissionTimestamp (**Entry** → dateTime)

[1]https://www.w3.org/TR/xmlschema11-2/

Schema/ontology design based on scenarios

What did you see?

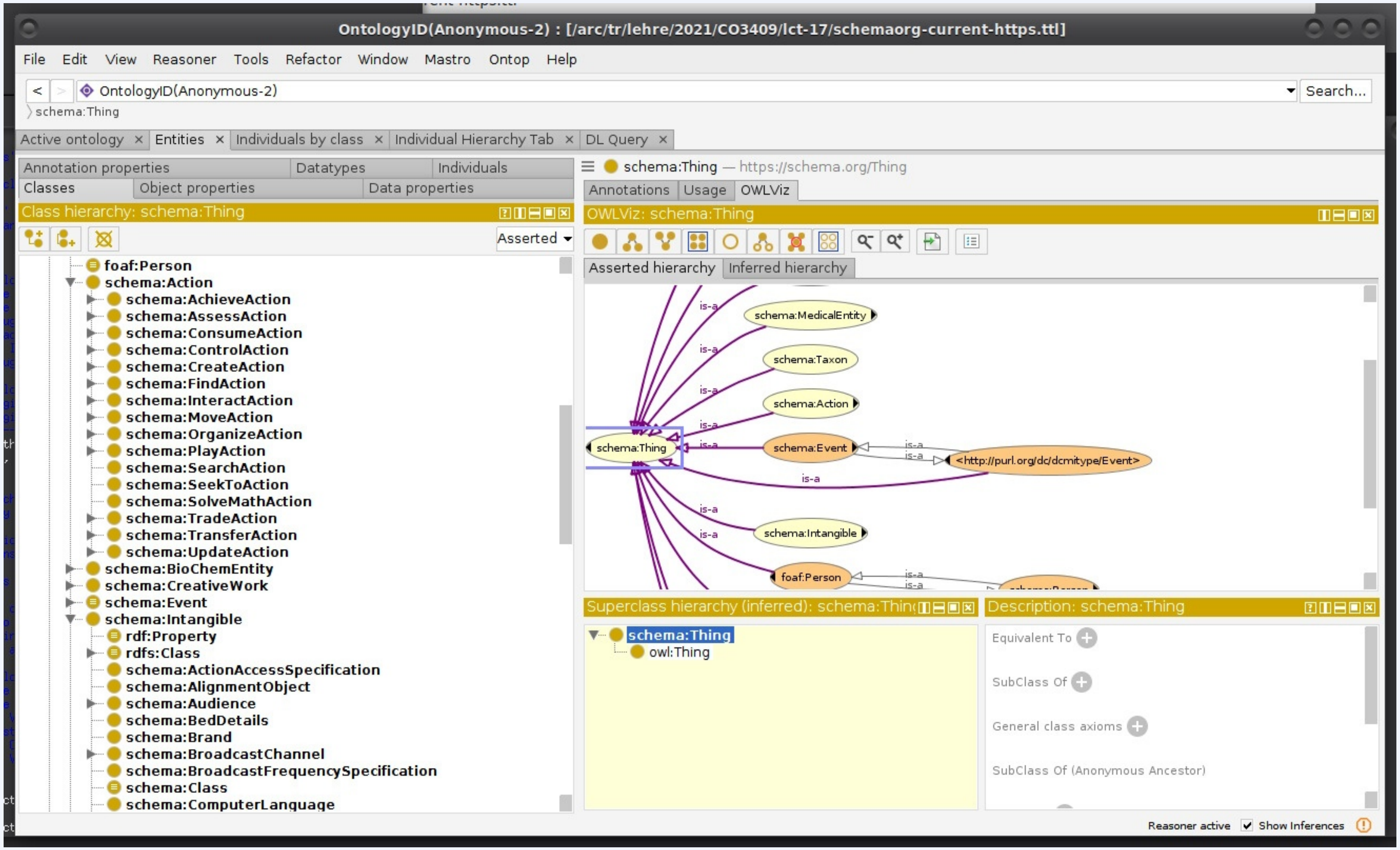
One approach to designing ontologies/schemas consists in **describing example scenarios**.

Usually, different people describe the same scenario in different terms, causing **semantic heterogeneity**.

Concepts instantiated by individuals should be in the ontology/RDF schema. **Relations** occurring as edges in the knowledge graph should be included; the **domains and ranges** of these relations should be included as concepts.

Different platforms may use different metadata schemas. To facilitate interoperability, an alignment is needed (*e.g.*, an **ontology alignment**).

Schema.org: A metadata schema used by Google[1, 2]



[1]Schema.org definitions and documentation: https://schema.org/docs/full.html.

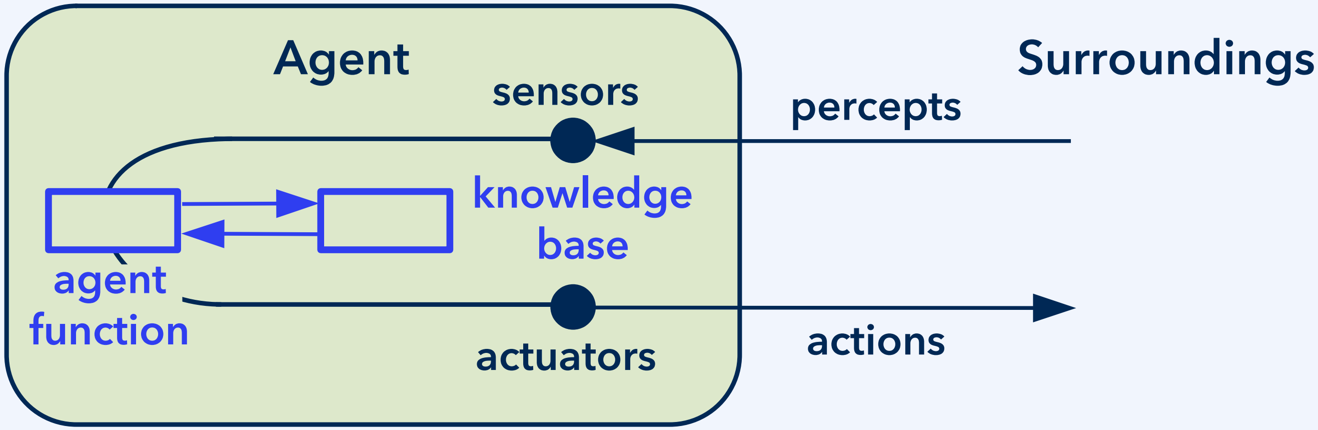
[2]Ontology in TTL format at https://schema.org/version/latest/schemaorg-current-https.ttl.

Querying linked data

CO3409

10th March 2022

Basic functionality of a knowledge base



The agent function interacts with the knowledge base (KB) in three ways:

1) First, the agent function **ingests** relevant percepts into the KB.
2) Second, it **queries** the KB for information needed in decision making.
3) Third, it **ingests** information about its own actions into the KB."

Interactions with the knowledge base take two forms:

- **Data ingest** ("tell") to extend or update the information about the world.
- **Data retrieval** based on **querying** ("ask").

From SQL to SPARQL

Querying relational databases: SQL (Structured Query Language)

category to be selected

model_parameters

id	name	value	unit	category	reference
1	“sigma”	3.7504	Å	pair_variable	
2	“epsilon”	59.71	$K \cdot k_B$	pair_variable	
3	“site mass”	13.019	*u*	object_variable	
4	“quadrupole”		DÅ	tensor	276
5	“site 1 position”		Å	vector	178
6	“site 2 position”		Å	vector	179

IDs from other tables

SELECT name, value, unit
FROM model_parameters
WHERE category = pair_variable;

→

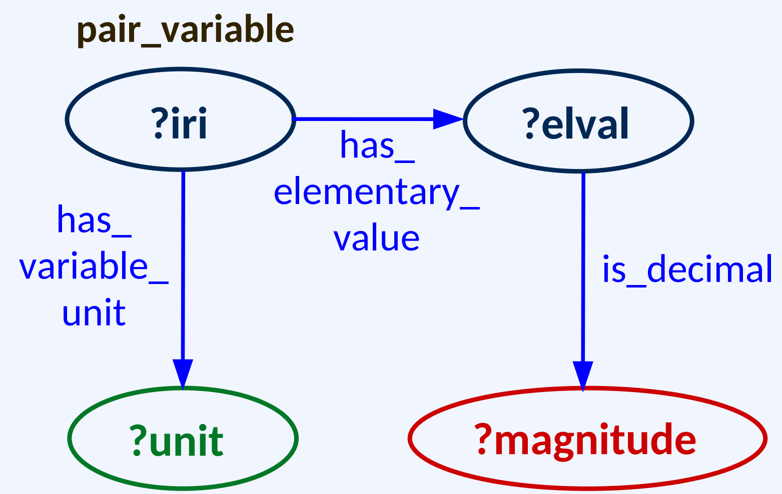
„sigma“	3.7504	Å
„epsilon“	59.71	$K \cdot k_B$

From SQL to SPARQL

SPARQL („SPARQL Protocol and RDF Query Language“) for linked data:[1]

SELECT ?x ?y … WHERE {sequence of triples involving ?x, ?y, …}

```
SELECT ?magnitude ?unit
WHERE {
  ?iri rdf:type :pair_variable.
  ?iri :has_elementary_value ?elval.
  ?elval :is_decimal ?magnitude.
  ?iri :has_variable_unit ?unit.
}
```



Observation: The WHERE clause consists of RDF triples with free variables.

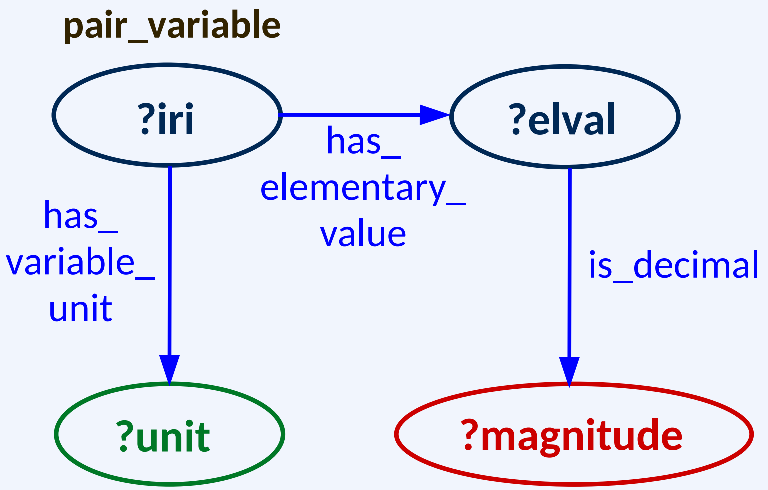
[1] W3C recommendation, https://www.w3.org/TR/sparql11-query/, **2013**.

From SQL to SPARQL

SPARQL („SPARQL Protocol and RDF Query Language“) for linked data:

SELECT ?x ?y … WHERE {sequence of triples involving ?x, ?y, …}

```
SELECT ?magnitude ?unit
WHERE {
  ?iri rdf:type :pair_variable.
  ?iri :has_elementary_value ?elval.
  ?elval :is_decimal ?magnitude.
  ?iri :has_variable_unit ?unit.
}
```



Relational databases can use SQL both for data extraction and for data ingest. In graph databases, JSON-LD, RDF/XML, TTL (or “NTriples”) files are often for ingesting linked data. (But there are also INSERT/DELETE queries in SPARQL.)

SPARQL and competency questions

Observation: Competency questions are queries that the system is required to answer competently. Therefore, the ontology (RDF schema) must be sufficiently powerful to express all the competency questions in SPARQL.

What are the numerical values of the marks?

```
SELECT ?mark ?value WHERE {
  ?mark uni:isNumericalMark ?value.
}
```

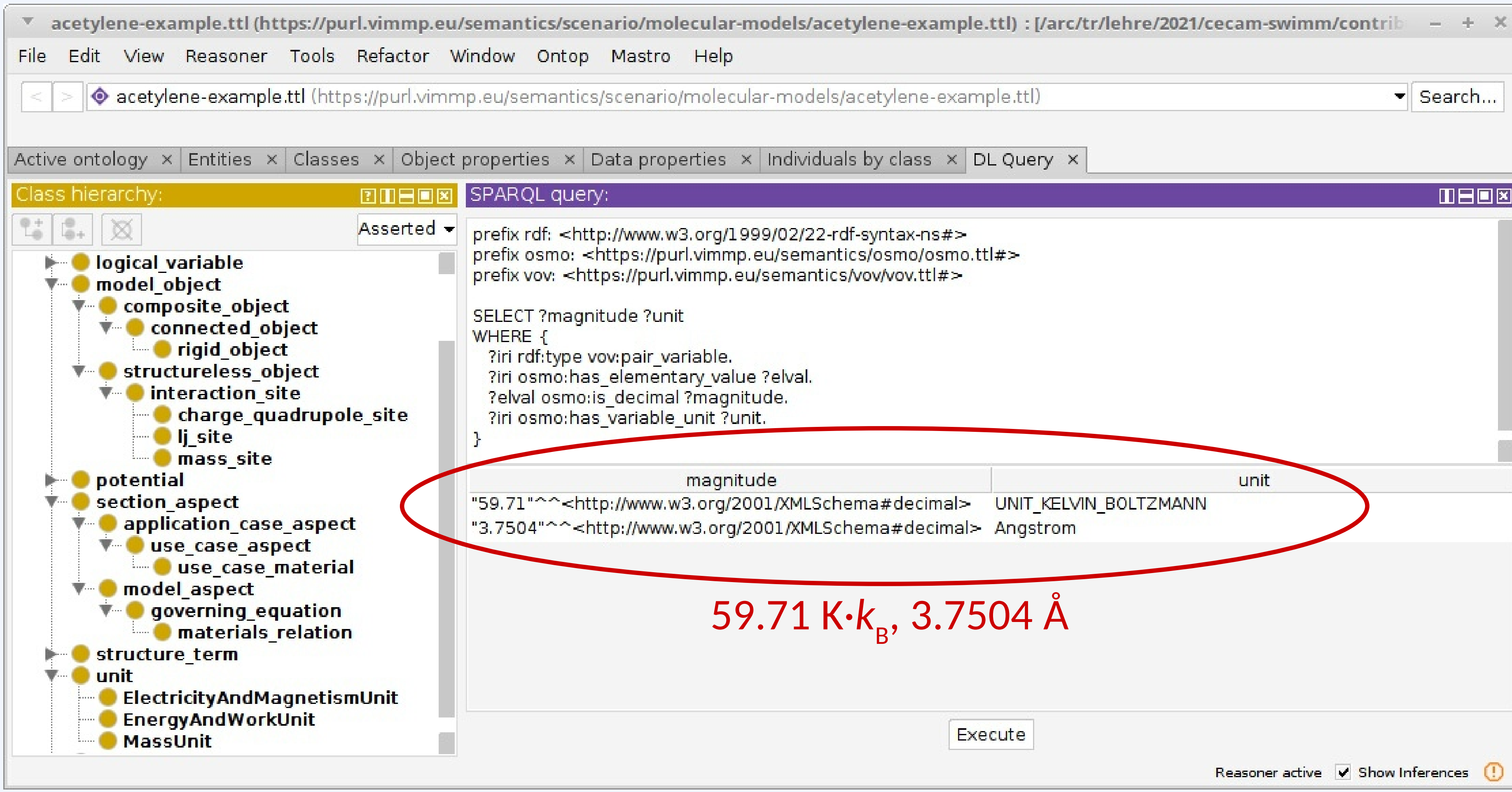
What exams are there marks for? What subjects are the exams on?

```
SELECT DISTINCT ?exam ?subject WHERE {
  ?mark uni:isMarkFor ?exam.
  ?exam rdf:type uni:Exam.
  ?exam uni:isAssessmentAbout ?subject.
}
```

What are the maximum marks for all the assessments?

```
SELECT ?assessment ?maxval WHERE {
  ?assessment uni:hasMarkMaximum ?maxval.
}
```

SPARQL in Protégé



Selected libraries for working with linked data

Jena (Java)

https://jena.apache.org/

Developer: Apache Software Foundation

Functionality: TTL and RDF/XML I/O, SPARQL querying (used by fuseki), reasoning („Inference API")

License: Apache Software License

owlready2 (Python)

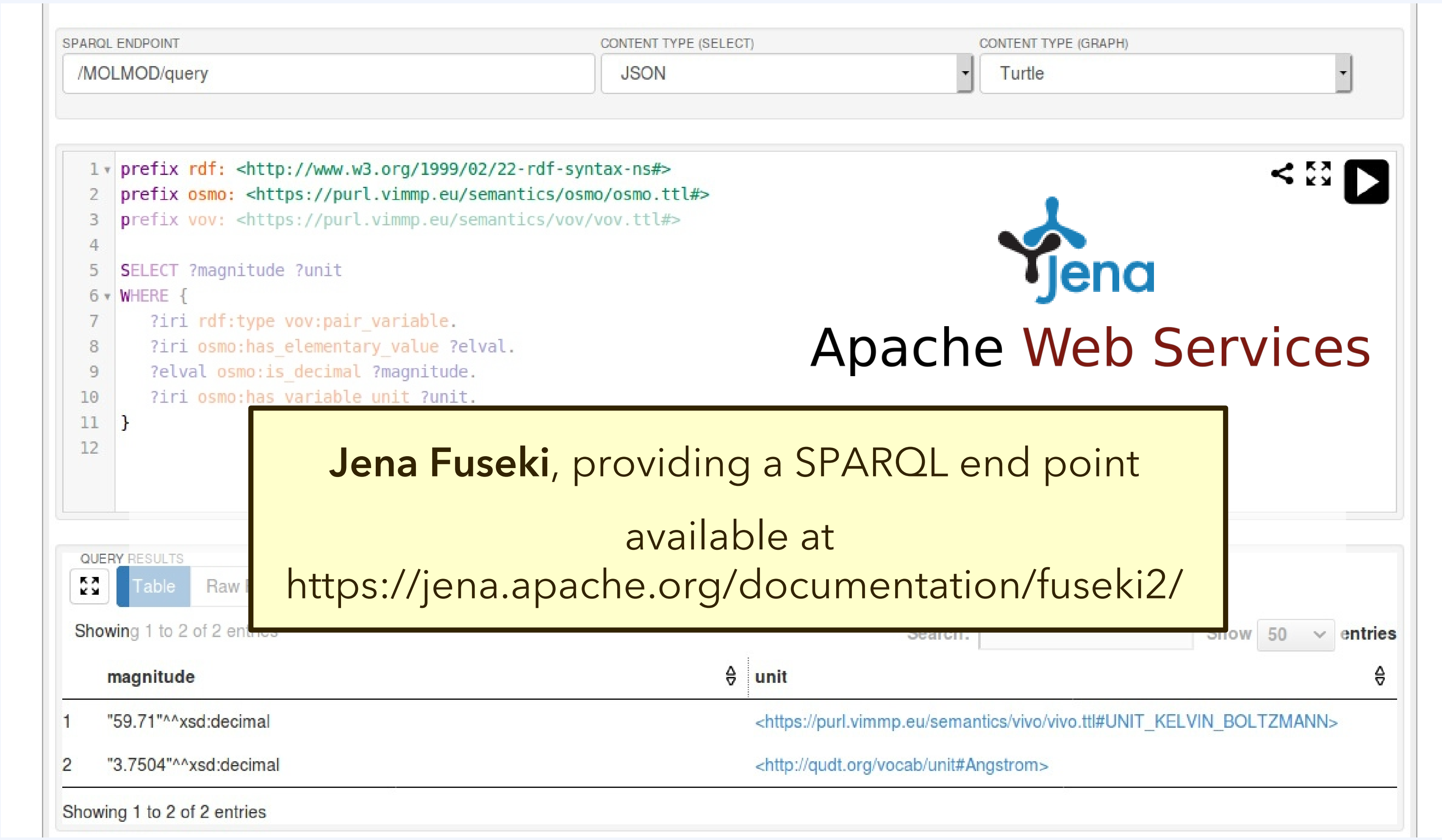
https://pypi.org/project/owlready2/

Functionality: manipulate OWL 2.0 ontologies as Python objects; reasoning; NTriples and RDF/XML I/O, OWL/XML as I, Reasoning, RDF quadstore

Developer: Jean-Baptiste Lamy

License: GNU Lesser General Public License v3 or later (LGPLv3+)

Example: Jena Fuseki SPARQL end point



Jena Fuseki, providing a SPARQL end point

available at
https://jena.apache.org/documentation/fuseki2/

Python based example

owlready2-example Jupyter notebook

```
from owlready2 import *

myonto = get_ontology("vimmp-ontology-fragment.owl")
myonto.load()

classes_list = list(myonto.classes())
op_list = list(myonto.object_properties())

my_ind = (my_class)("my_individual", label="my_label")
myonto.save(file="owlready2_test_outcome.ttl", format="ntriples")
```

A toy example to …

- **load** an ontology
- **navigate** it (classes and properties)
- **modify it** (create an individual)
- **save** the modified ontology **in a file**

Python based example

owlready2-sparql-example Jupyter notebook

```
graph = default_world.as_rdflib_graph()
```

```
In [64]:
sparql_query = "SELECT DISTINCT ?person ?address WHERE {\n" \
    + "   ?person <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#teachesAt> ?institution.\n" \
    + "   ?institution <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#hasCampusIn> ?address.\n" \
    + "   ?person <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#livesIn> ?address.\n" \
    + "}"

print(sparql_query)
```

```
SELECT DISTINCT ?person ?address WHERE {
   ?person <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#teachesAt> ?institution.
   ?institution <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#hasCampusIn> ?address.
   ?person <http://home.bawue.de/~horsch/teaching/co3409/semantics/uni#livesIn> ?address.
}
```

```
In [68]:
query_response = list(graph.query(sparql_query))

for i in range(len(query_response)):
    print("Match #", i, sep="")
    print("\tPerson:\t", query_response[i][0].n3())
    print("\tAddress:\t", query_response[i][1].n3())
```

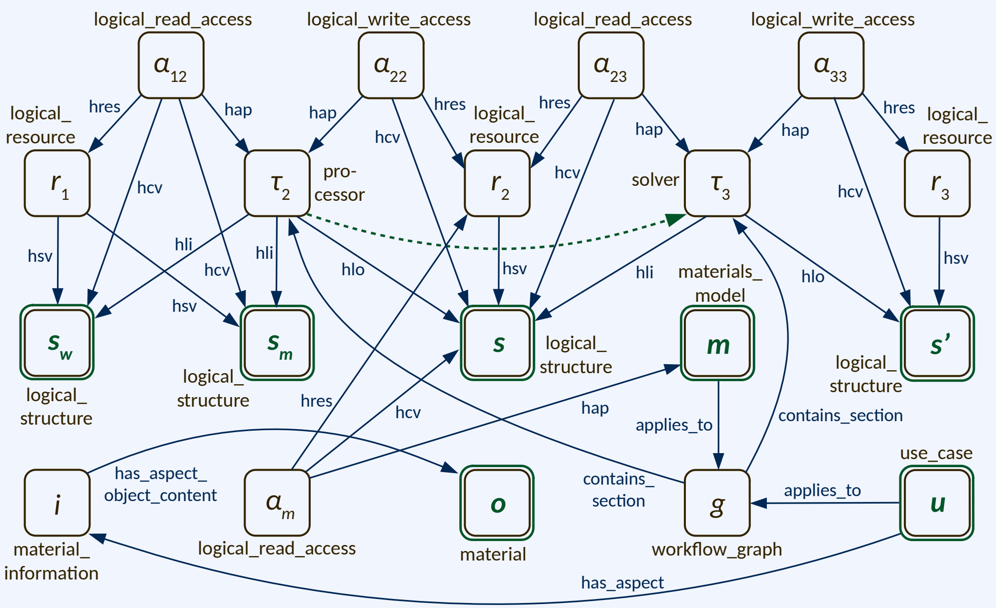
```
Match #0
	Person:	<http://home.bawue.de/~horsch/teaching/co3409/semantics/uni-scenario#martin>
	Address:		<http://home.bawue.de/~horsch/teaching/co3409/semantics/uni-scenario#preston>
```

Querying linked data as an NP-hard problem

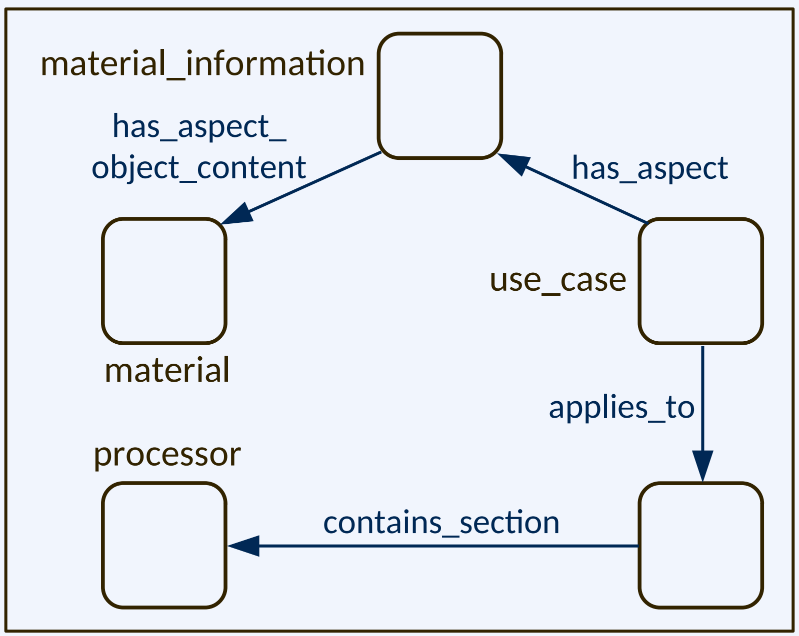
Subgraph matching problem (NP-complete):

Given a graph *G* and a pattern *H*, does G contain a subgraph isomorphic to *H*?

graph *G*



pattern *H*



(example from Klein et al., in Proceedings of JOWO 2021)

Concurrent process models

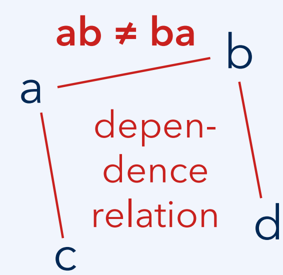
CO3409

10th March 2022

Concurrent process models

Dependence/independence between actions & events in an enterprise system:

a) Updated raw sensory data ingested into knowledge base
b) Data analysis on raw sensory data, creating aggregated data
c) Read access to raw sensory data by a user
d) Read access to aggregated data by a user



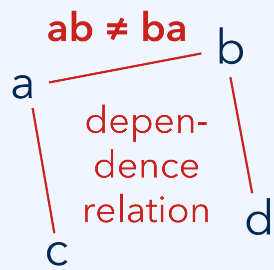
Some of these events are **independent**, for example b and c.
Events are independent if they are **commutative**: bc = cb.

- a and b are dependent because "update first, analysis second" (ab) in general has a different outcome from "analysis first, update second" (ba).
- a and d are independent because "ad" in immediate sequence will have the same outcome as "da". For "ad", the user will read aggregated data that are no longer up to date, but that is immaterial to this definition.

Concurrent process models

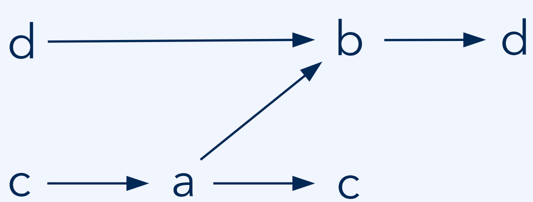
Dependence/independence between actions & events in an enterprise system:

a) Updated raw sensory data ingested into knowledge base
b) Data analysis on raw sensory data, creating aggregated data
c) Read access to raw sensory data by a user
d) Read access to aggregated data by a user



Events that are **dependent** can *never* occur *concurrently*.
Events are independent if they are **commutative**: bc = cb.

In a particular execution or process, if it is unsubstantial in what order two events occur, they are **concurrent**: Below, *e.g.*, the first and second c-d pairs:

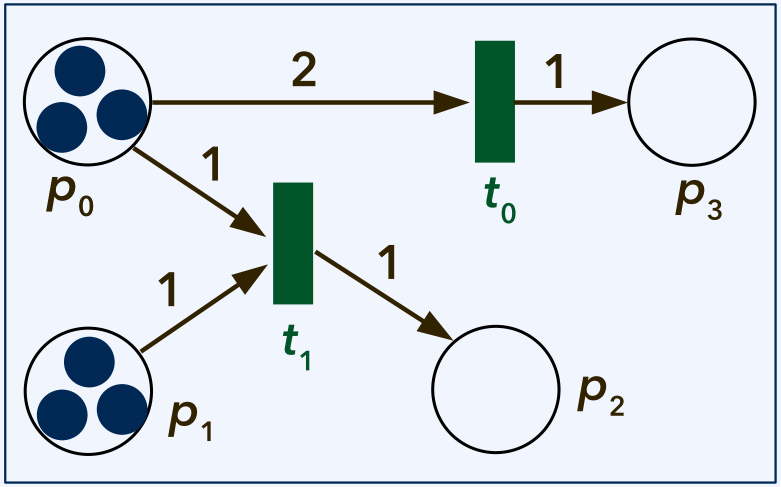


Hasse diagram for the process (“trace”)

cacdbd = cdacbd = dcabdc = …

Concurrent process models: Petri nets

Components of a Petri net: **places** **transitions** **tokens** **arc**



Semantics of this net:

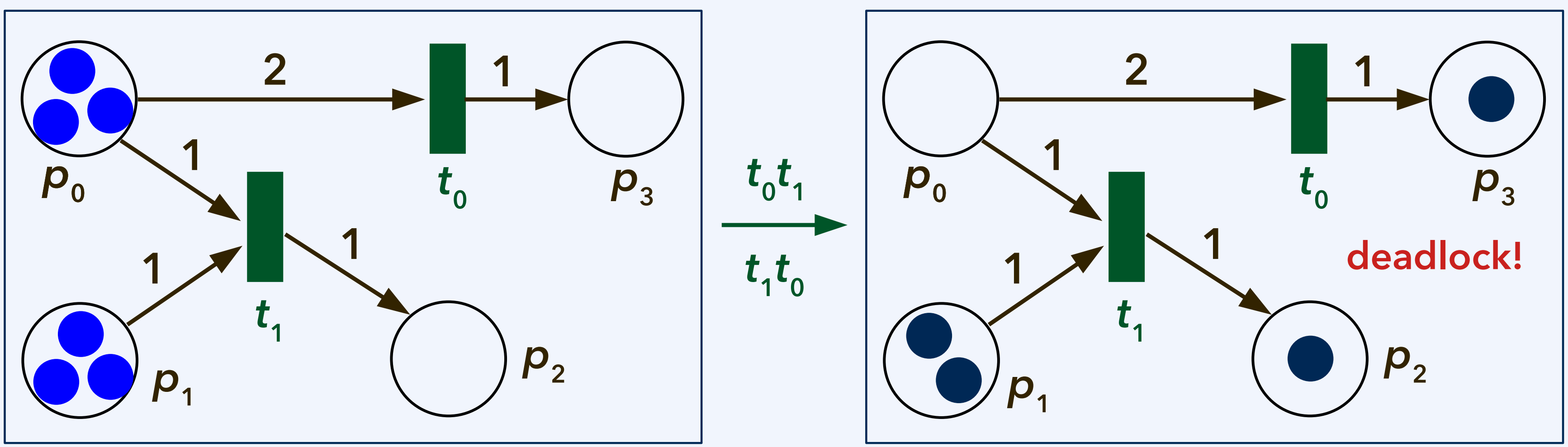
Transition t_0 can only be **fired** if place p_0 contains at least two tokens. Firing t_0 will take away two tokens from p_0 and add one token to p_3.

Transition t_1 can only be fired if both p_0 and p_1 each contain at least one token. It removes one token from each, and adds one token to place p_2.

Concurrent process models: Petri nets

Components of a Petri net: **places** **transitions** **tokens** **arc**

Remark: The initial state is part of the specification.

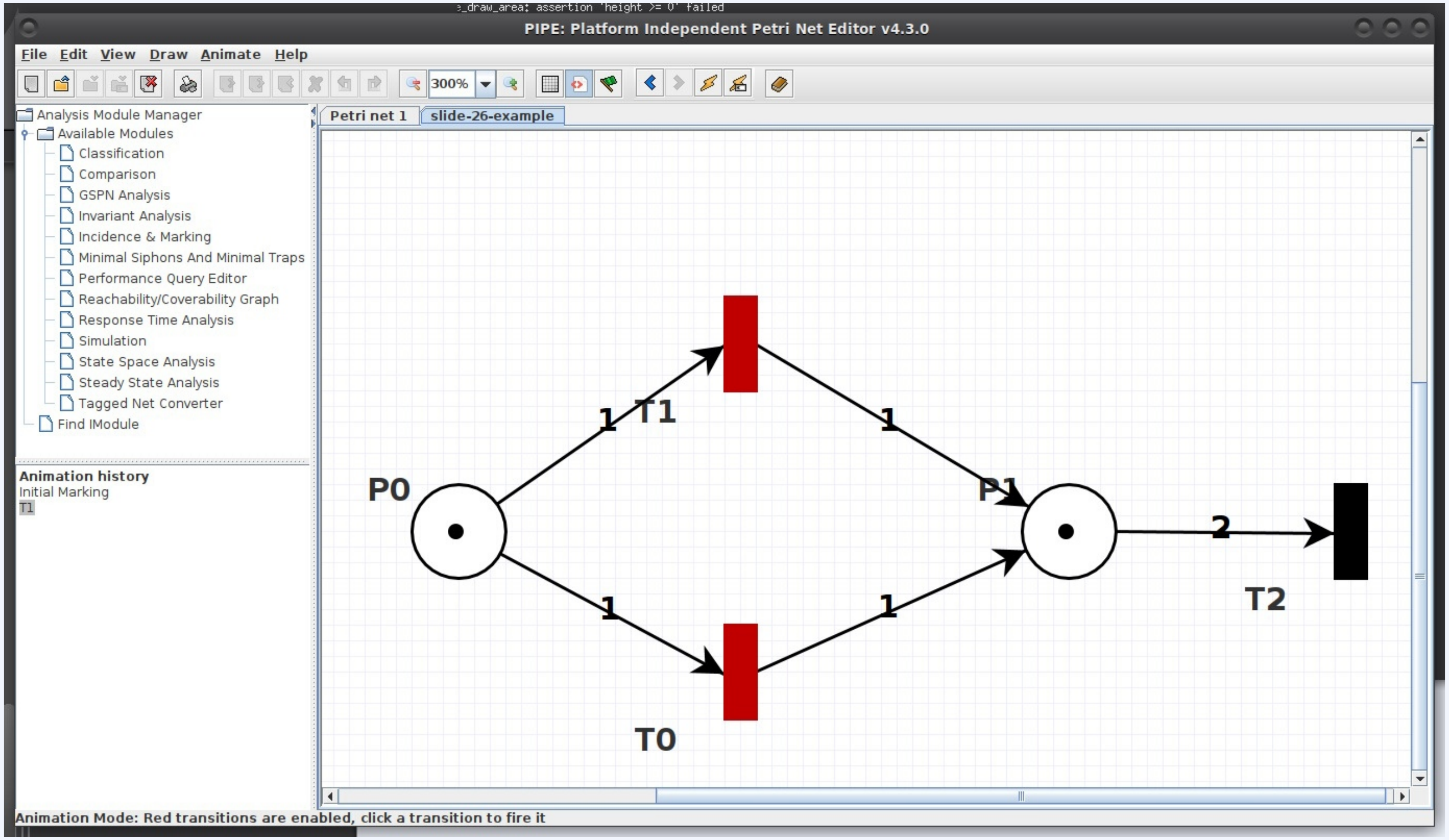


Possible **firing sequences** until a **deadlock** is reached:

- $t_0t_1 = t_1t_0$ (where t_0 and t_1 are concurrent)
- $t_1t_1t_1$ (where multiple occurrences of t_1 are also concurrent)

Concurrent process models: Petri nets

PIPE tool for editing/simulating Petri nets: http://pipe2.sourceforge.net/



Concurrent process models: Petri nets

Example from Wiśnewski *et al.*:[1]

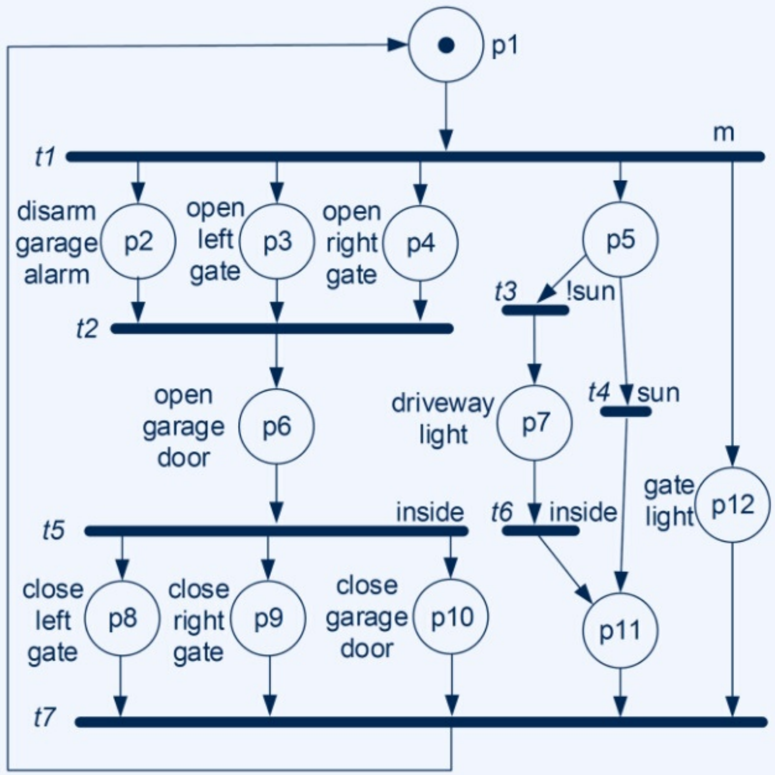


FIGURE 3. A Petri net-based specification of the smart-home system.

How do t_1 and t_2 relate to each other?

How do t_2 and t_3 relate to each other?

How do t_3 and t_4 relate to each other?

Is there a terminal state – a deadlock?

[1]R. Wiśnewski *et al.*, *IEEE Access* 7: 13510–13522, doi:10.1109/ACCESS.2019.2893284, **2019**.

Concurrent process models: Petri nets

Example from Wiśnewski *et al.*:[1]

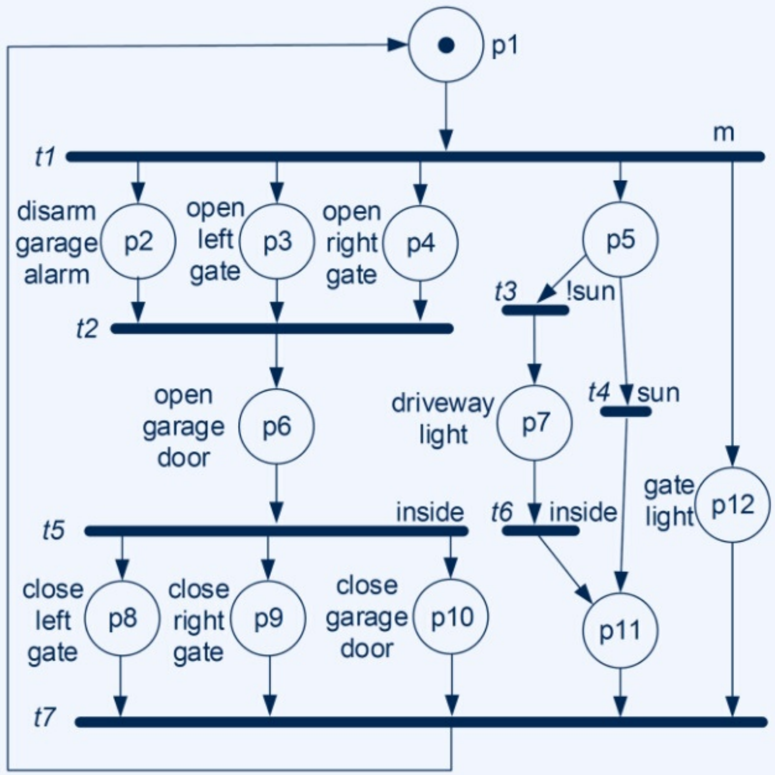


FIGURE 3. A Petri net-based specification of the smart-home system.

How do t_1 and t_2 relate to each other?

How do t_2 and t_3 relate to each other?

How do t_3 and t_4 relate to each other?

Is there a terminal state – a deadlock?

- t_1 and t_2 cannot occur concurrently.
- t_2 and t_3 can occur concurrently.
- t_3 and t_4 cannot occur concurrently; they mutually exclude each other (in each iteration of the process).
- There is no deadlock ("live Petri net").

[1]R. Wiśnewski *et al.*, *IEEE Access* 7: 13510–13522, doi:10.1109/ACCESS.2019.2893284, **2019**.

Concurrent process models: Petri nets

Example from Wiśnewski *et al.*:[1]

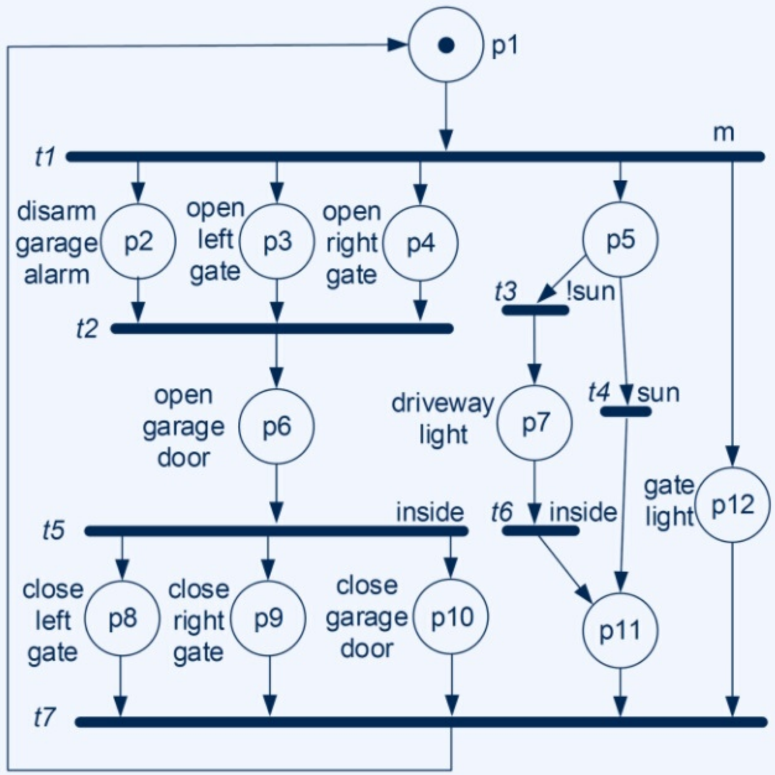


FIGURE 3. A Petri net-based specification of the smart-home system.

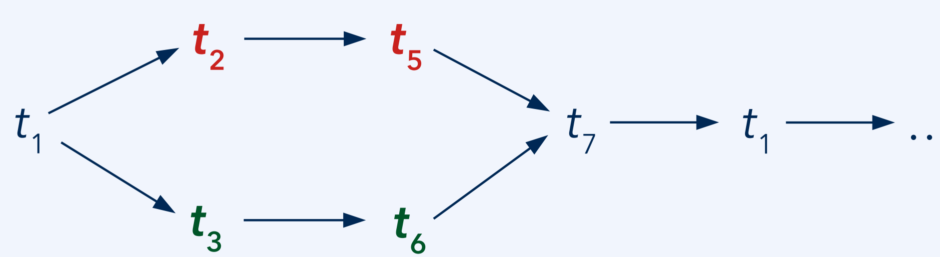
Firing sequence with concurrent transitions:

$$t_1\; \mathbf{t_2}\; \mathbf{t_3}\; \mathbf{t_6}\; \mathbf{t_5}\; t_7\; t_1\; \ldots$$

$$= \quad t_1\; \mathbf{t_3}\; \mathbf{t_2}\; \mathbf{t_5}\; \mathbf{t_6}\; t_7\; t_1\; \ldots$$

$$= \quad t_1\; \mathbf{t_2}\; \mathbf{t_3}\; \mathbf{t_5}\; \mathbf{t_6}\; t_7\; t_1\; \ldots$$

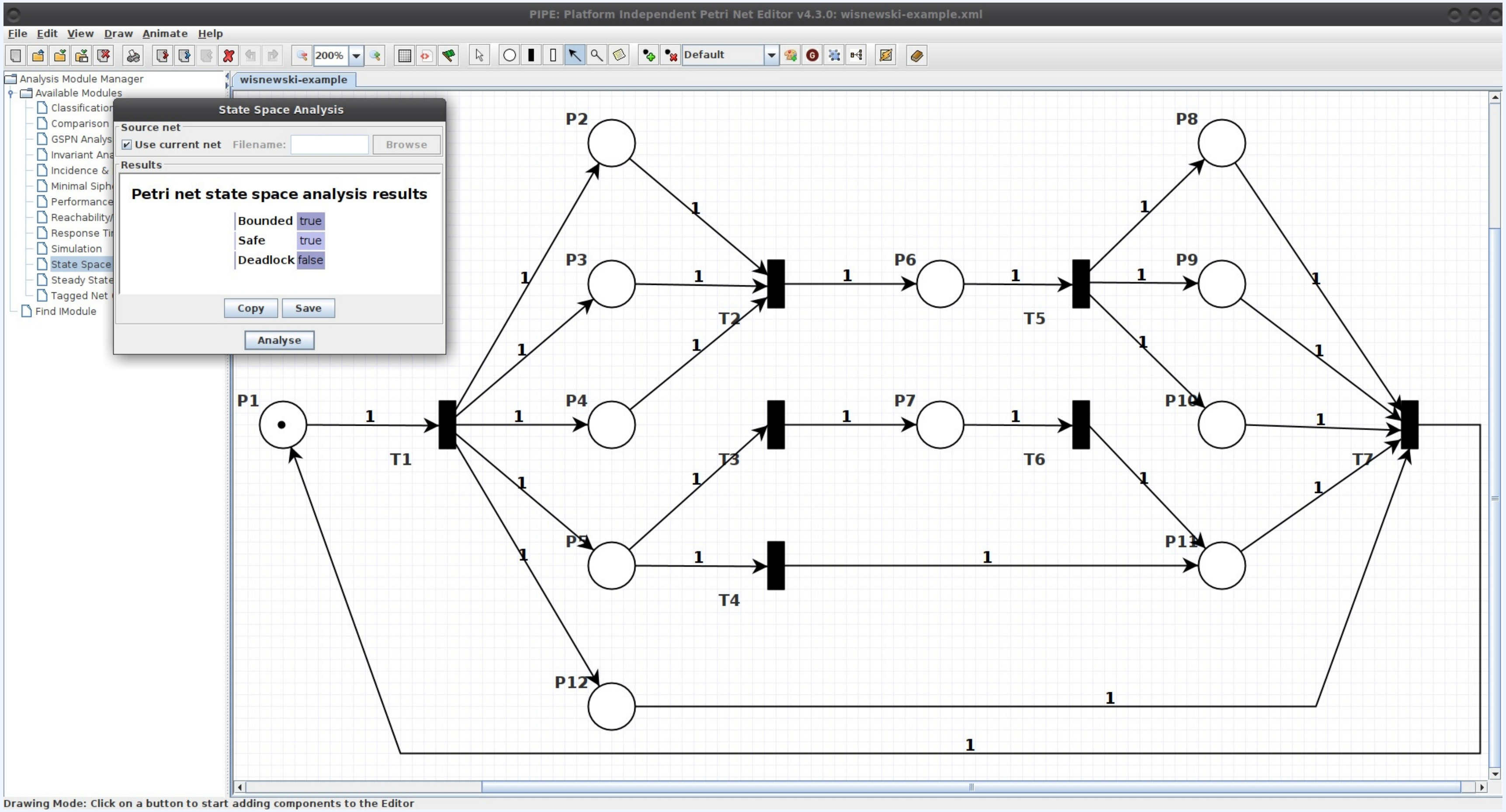
Trace representation of the sequence:



(Also called a **Hasse diagram**.)

[1]R. Wiśnewski *et al.*, *IEEE Access* 7: 13510–13522, doi:10.1109/ACCESS.2019.2893284, **2019**.

Concurrent process models: Petri nets



Safe: No place can contain more than 1 token. (**Bounded:** Some *k* instead of 1.)

University of
Central Lancashire
UCLan

CO3409
Distributed Enterprise Systems

Ontology/schema design
Querying linked data
Concurrent process models

Where opportunity creates success