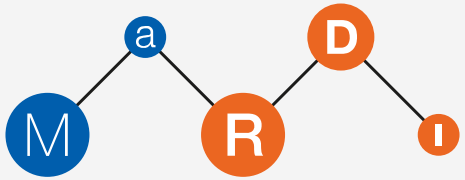


AlgoData – Algorithm Knowledge Graph

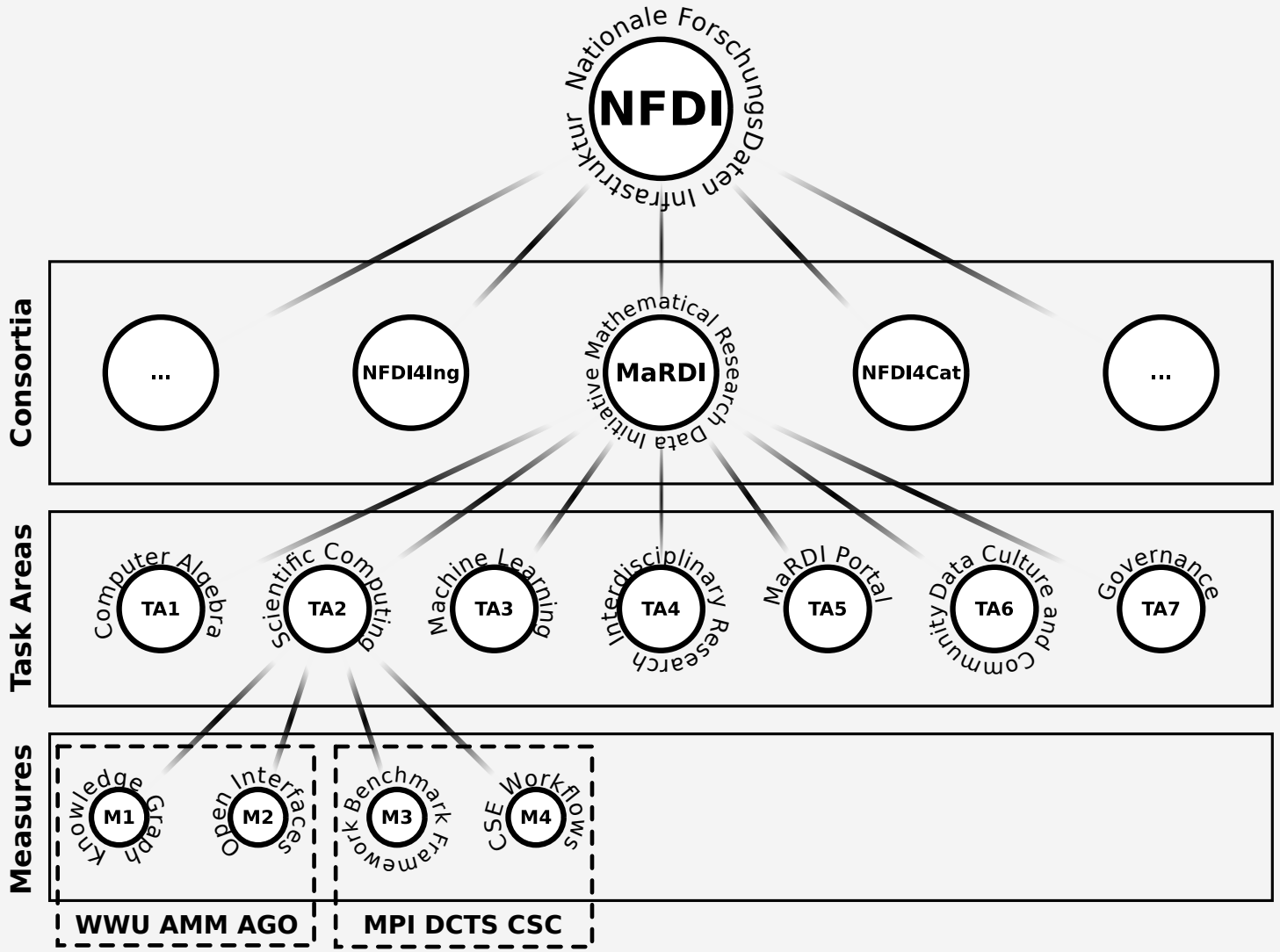
René Fritze, Christian Himpe, Hendrik Kleikamp, Mario Ohlberger, Stephan Rave

2022–01–18

WWU MÜNSTER

MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

MaRDI


Nationale Forschungsdaten Infrastruktur
NFDI
Consortia
...
NFDI4Ing
Mathematical Research Data Initiative
MaRDI
NFDI4Cat
...
Task Areas
Computer Algebra
TA1
Scientific Computing
TA2
Machine Learning
TA3
Interdisciplinary Research
TA4
MaRDI Portal
TA5
Data Culture and Community
TA6
Governance
TA7
Measures
Knowledge Graph
M1
Open Interfaces
M2
Benchmark Framework
M3
CSE Workflows
M4
WWU AMM AGO
MPI DCTS CSC

Motivation

- Some question are not answerable by full-text search.
- Not: Here is a reference potentially answering,
- rather: Here is an answer based on this reference.
- Externalization of learned knowledge and research results.
- Faster onboarding of researchers from other fields or fresh PhDs.

→ **Knowledge needs machine readable and human queryable encoding.**

Outline

1. Data-store
2. Back-end
3. Front-end

Knowledge Graph Database

- What is knowledge? A set of facts.
- How to represent knowledge? As a list of statements.
- What is a statement? A sentence comprised of subject-predicate-object.

	Relational Database	**Knowledge Graph**
Structure	Schema	Ontology
Data Items	Rows	Statements
Uniqueness	Keys	URIs
Query	SQL	SPARQL

Encoding Subject-Predicate-Object Triplets

- **Syntax:** RDF - Resource Description Framework
  - W3C Specification (Version 1.1, 2014)
  - Industry Standard (no relevant alternatives)
  - Directed graph of triple statements.
- **Serialization:** Turtle - Terse RDF Triple Language
  - W3C Recommendation (Version 1.1, 2014)
  - Human-Readable (and actually readable)
  - URI-based and similar to SPARQL syntax.

Closely Related W3C Semantic Web Standards:

$$\underbrace{\text{RDF} + \text{Turtle}}_{\text{Database}} + \underbrace{\text{RDFS} + \text{OWL}}_{\text{Ontology}} + \underbrace{\text{SPARQL}}_{\text{Query}}$$

It's Turtles All the Way Down

Ontology = Controlled Vocabulary + Grammar

- What are admissible classes ($= \{\text{subjects} \cup \text{objects}\}$) ?
- What are admissible properties ($=$ predicates) ?

- What classes may be subjects for what properties?
- What classes may be objects for what properties?

Comparable to a defintion of a mapping:

$$\text{predicate} : \underbrace{\text{subjects} \subset \text{classes}}_{\text{Domain}} \rightarrow \underbrace{\text{object} \subset \text{classes}}_{\text{Range}}$$

The ontology itself is a knowledge graph!

- (see: RDFS - RDF-Schema, and: OWL - Web Ontology Language)

AlgoData Ontology

- `:algorithm a owl:Class .`
- `:problem a owl:Class .`
- `:software a owl:Class .`
- `:publication a owl:Class .`
- `:benchmark a owl:Class .`

- `:solves a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :problem .`
- `:variant-of a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :algorithm .`
- `:modification-of a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :algorithm .`
- `:extension-of a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :algorithm .`
- `:defined-in a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :publication .`
- `:analyzed-in a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :publication .`
- `:studied-in a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :publication .`
- `:used-in a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :publication .`
- `:reviewed-in a owl:ObjectProperty ; rdfs:domain :algorithm ; rdfs:range :publication .`
- `:specializes a owl:ObjectProperty ; rdfs:domain :problem ; rdfs:range :problem .`
- `:implements a owl:ObjectProperty ; rdfs:domain :software ; rdfs:range :algorithm .`
- `:tests a owl:ObjectProperty ; rdfs:domain :software ; rdfs:range :benchmark .`
- `:documented-in a owl:ObjectProperty ; rdfs:domain :software ; rdfs:range :publication .`
- `:instance-of a owl:ObjectProperty ; rdfs:domain :benchmark ; rdfs:range :problem .`
- `dc:hasIdentifier a owl:ObjectProperty ; rdfs:domain :publication , :benchmark , :software .`

Server Back-End

SPARQL Protocal And RDF Query Language

- Uses namespaces (same as knowledge graph)
- Uses triplets (same as knowledge graph)
- Very powerful

SPARQL endpoint

- Allow query and update
- Provides SOH (SPARQL-Over-HTTP)
- Serves also JSON (Javascript Object Notation)

SPARQL server

- Triplet storage
- SPARQL endpoint
- i.e. Apache Jena Fuseki

Asking (the Back-End) in Chinese

- Statement: $\underbrace{\text{Alonzo Church}}_{\text{subject}\subset\text{humans}}$ $\underbrace{\text{is a}}_{\text{predicate}}$ $\underbrace{\text{mathematician}}_{\text{object}\subset\text{types}}$.
- English Question:
- Chinese Question:

Asking (the Back-End) in Chinese

- Statement: Alonzo Church (subject⊂humans) is (predicate) a mathematician. (object⊂types)
- English Question: *What* (question word (formerly object)) is (predicate) Alonzo Church? (object (formerly subject)) A mathematician.
- Chinese Question: 阿隆佐·邱奇 (subject) 是 (predicate) 什么 (question word object) ? 他 是 数学家。

Asking (the Back-End) in Chinese

- Statement: Alonzo Church (subject⊂humans) is (predicate) a mathematician. (object⊂types)
- English Question: *What* (question word (formerly object)) is (predicate) Alonzo Church? (object (formerly subject)) A mathematician.
- Chinese Question: Alonzo Church (subject) is (predicate) what (question word object) ? He is mathematician.

Asking (the Back-End) in Chinese

- Statement: Alonzo Church (subject⊂humans) is (predicate) a mathematician. (object⊂types)
- English Question: *What* (question word (formerly object)) is (predicate) Alonzo Church? (object (formerly subject)) A mathematician.
- Chinese Question: Alonzo Church (subject) is (predicate) what (question word object) ? He is mathematician.

In a sense, the front-end has to translate english to and fro chinese grammar.

Web Query Front-End

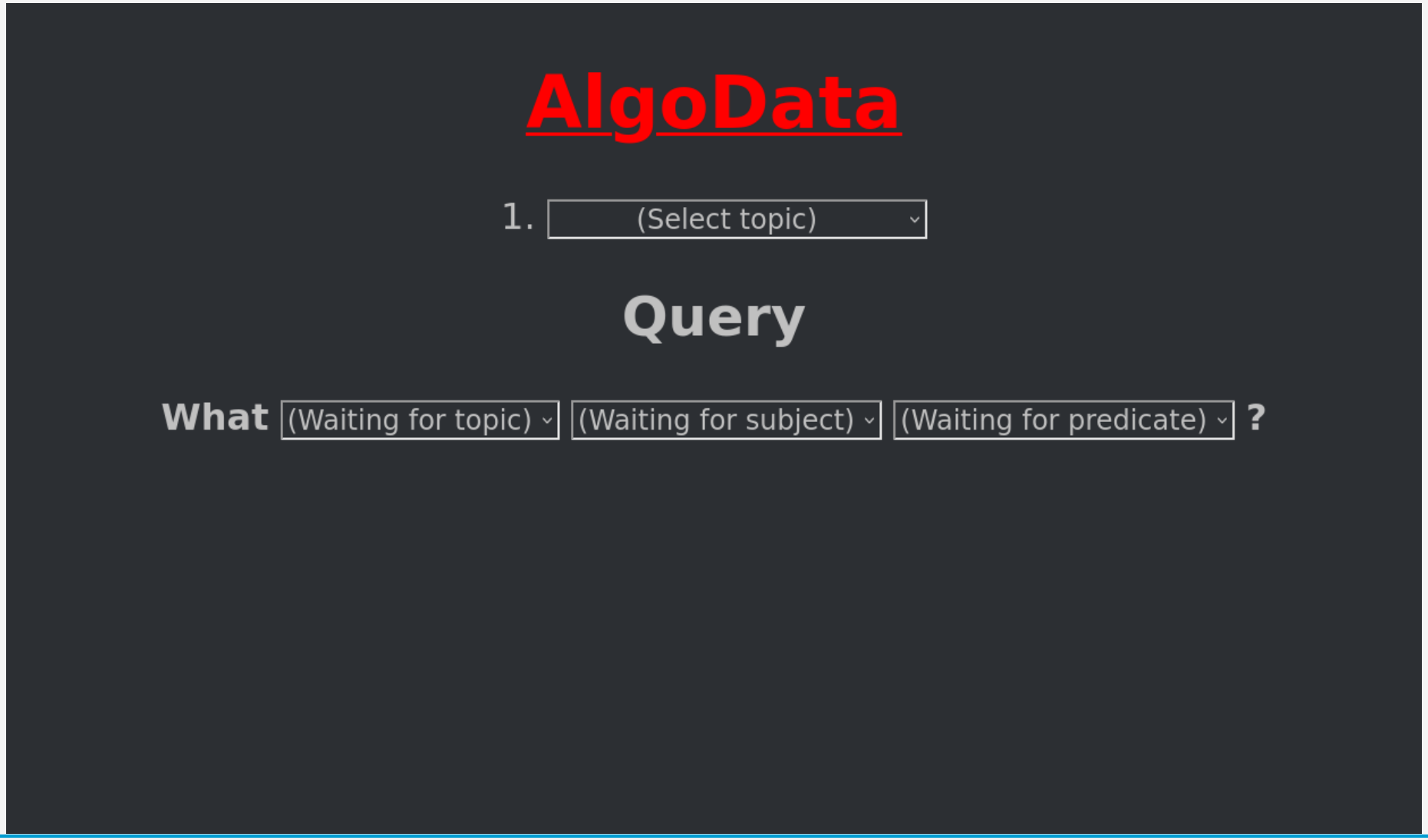
User Interface:

- Formulated as an english-language question.
- Automatically limit predicates and objects.
- Return URI as clickable link.
- Return human-readable name.
- As simple as possible.

Technically:

- JavaScript's built-in *Fetch* API for client-side asynchronuous loading.
- Fuseki server response in JSON (JavaScript Object Notation).
- Overall: *Fetch* queries via SOH receiving JSON.

AlgoData Query Demo



AlgoData Query Demo

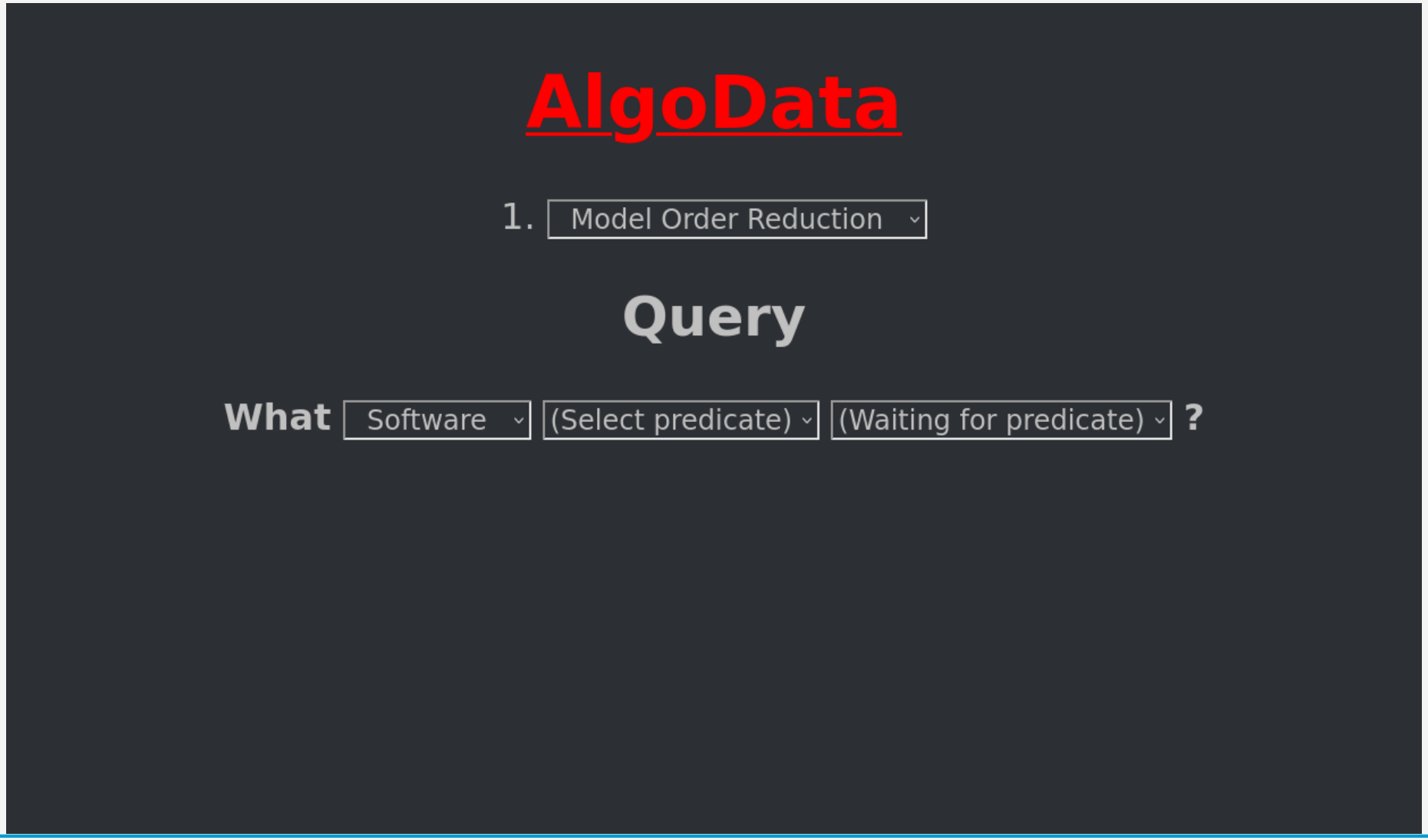
AlgoData

1. Model Order Reduction

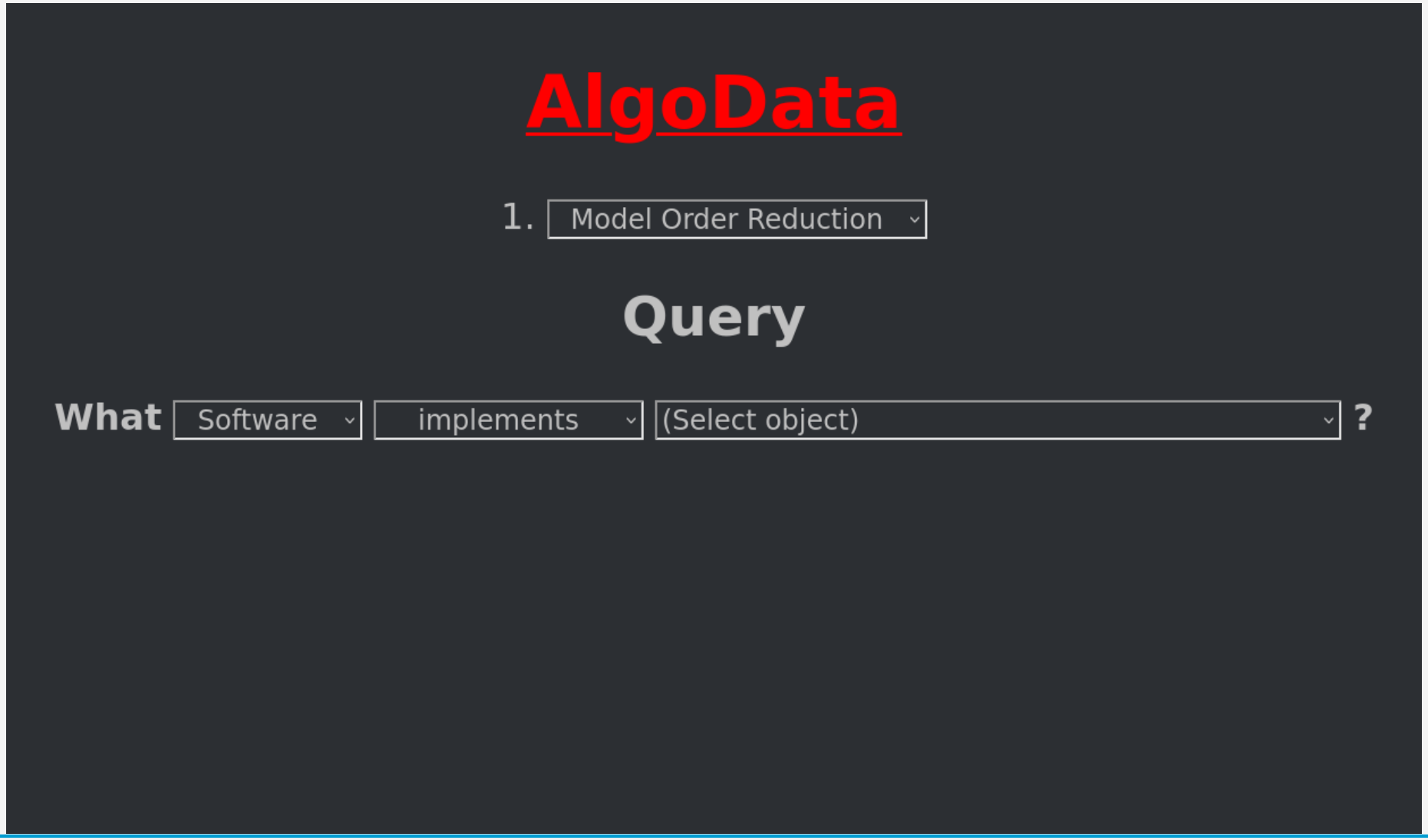
Query

What (Select subject) (Waiting for subject) (Waiting for predicate) **?**

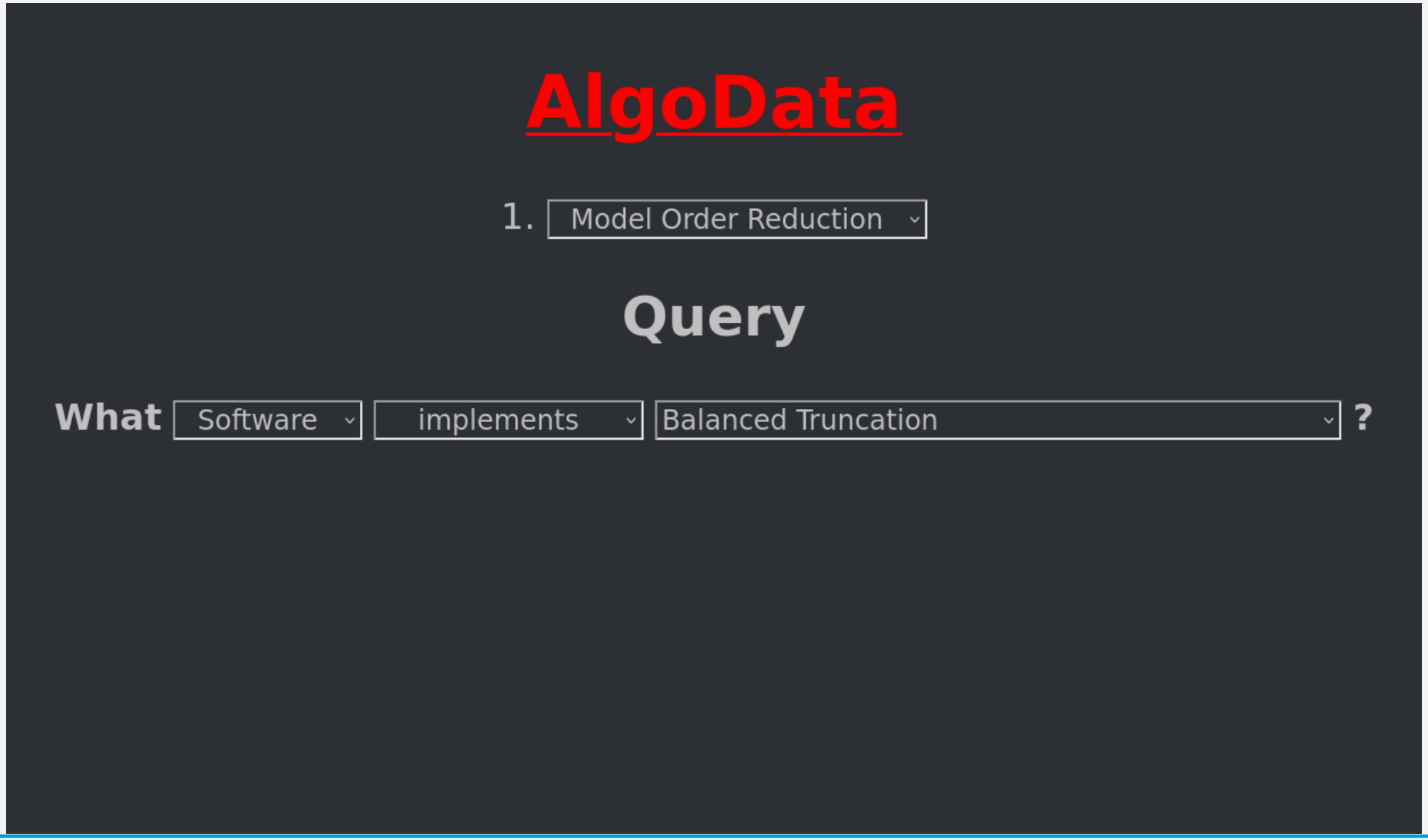
AlgoData Query Demo



AlgoData Query Demo



AlgoData Query Demo



AlgoData Query Demo

AlgoData

1. Model Order Reduction

Query

What Software implements Balanced Truncation ?

- MORE - MOdel REduction Toolbox
- pyMOR - Model Order Reduction with Python
- emgr - EMpirical GRamian Framework
- M.E.S.S. - Matrix Equation Sparse Solver
- SIMPLIFY - Structured ModeL reductIon Toolbox
- Python Control Systems Library
- MOREMBS - Model Order Reduction of Elastic Multibody Systems
- SSSMOR - Sparse State-Space Model Order Reduction Toolbox
- Control System Toolbox
- Computer-Aided Control System Design Tools for GNU Octave
- MORLAB - Model Order Reduction LABoratory

One Query to Rule Them All

The query front-end dispatches five SPARQL queries:

1. onChange topic: fetch all classes.
2. onChange class: fetch all subject-class predicates.
3. onChange class: fetch all object-class predicates.
4. onChange predicate: fetch all matching subjects or objects respectively.
5. onChange object: fetch answer.

All queries have the same structure:

```
SELECT DISTINCT ?answer ?label ?id
  WHERE { subj pred ?answer .
          { ?answer rdfs:label ?label . }
          OPTIONAL { ?answer dc:hasIdentifier ?id . } }
```

One Query to Rule Them All

The query front-end dispatches five SPARQL queries:

1. onChange topic: fetch all classes.
2. onChange class: fetch all subject-class predicates.
3. onChange class: fetch all object-class predicates.
4. onChange predicate: fetch all matching subjects or objects respectively.
5. onChange object: fetch answer.

All queries have the same structure:

```
SELECT DISTINCT ?answer ?label ?id
  WHERE { ?answer pred obj .
          { ?answer rdfs:label ?label . }
          OPTIONAL { ?answer dc:hasIdentifier ?id . } }
```

Propose & Curate

Plan:

- Users can propose statements (i.e. for new publications),
- which will be held in a staging graph.
- Editors from a topic get assigned round robin.
- If three editors vote yes, the statement is included in the topical graph.
- Editor (and if needed user) management will also be graphs.

State of Measure 1

Topics:

- Model Order Reduction (Ongoing)
- Numerical Linear Algebra (Started)
- Runge-Kutta Methods (Planning)
- Dynamic Mode Decomposition? (Possibly)
- Decentralized Control? (Possibly)

Outlook:

- Assess predicates (and extend if necessary)
- Optional second-level deducing in queries
- Proposal and curation interfaces
- Outreach to communities
- Super graph

Summary

- **Structure:** Prototype Ontology (RDFS/OWL)
- **Database:** Algorithm Knowledge Graph (RDF/Turtle)
- **Backend:** Open-Source Fuseki Server (SPARQL/Fuseki)
- **Frontend:** Query Web Interface (JS/JSON)

`https://himpe.science`