

Alex Ivliev, Lukas Gerlach, Simon Meusel, Jakob Steinberg, Markus Krötzsch

Knowledge-Based Systems, TU Dresden

Nemo: A Scalable and Versatile Datalog Engine

Introducing Nemo

- Datalog engine written in Rust
- Free and open source
- More than 10 contributors



Nemo
Graph Rule Engine

Design Goals

Introducing Nemo

- Datalog engine written in Rust
- Free and open source
- More than 10 contributors



Nemo
Graph Rule Engine

Design Goals



Scalability

Introducing Nemo

- Datalog engine written in Rust
- Free and open source
- More than 10 contributors



Nemo
Graph Rule Engine

Design Goals



Scalability



Versatility

Introducing Nemo

- Datalog engine written in Rust
- Free and open source
- More than 10 contributors



Nemo
Graph Rule Engine

Design Goals



Scalability



Versatility



Usability

Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1  head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3  @import data :- csv {resource="data.csv"}.
4  @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6  age("Alice", 24).
7
8  result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


Language Features

Basis: Recursive Datalog

Import/Export
(CSV, TSV, N-Triples, Turtle)

Datatypes

Arithmetic Expressions

Builtin Functions

Stratified Negation

Stratified Aggregation

Value Invention

```
1 head(?X, ?Z) :- body(?X, ?Y), head(?Y, ?Z).
2
3 @import data :- csv {resource="data.csv"}.
4 @import web :- rdf {resource="http://abc.com/data.ttl"}.
5
6 age("Alice", 24).
7
8 result(?X + ?Y) :- data(?X, ?Y), ?X < ?Y.
9
10 message(CONCAT("Result: ", STR(?X))) :- result(?X).
11
12 negation(?X) :- data(?X, ?Y), ~result(?Y).
13
14 summed(?X, #sum(?Y)) :- data(?X, ?Y).
15
16 human(!P), parent(?H, !P) :- human(?H) .
```


The Nemo Toolkit

```
> Rule
  Rule
  SubExprResult(?nr, INT(?val))
  SubExprResult
  ?nr
  INT(?val)
  ?val
  IntLiteral(?nr, ?val)
  IntLiteral
  ?nr
  ?val
  Rule
  SubExprResult(?nr, "MinusExpr")
  UnExpr
  ?nr
  "MinusExpr"
  Parent(?subNr, ?nr)
  Parent

Reasoning completed in 10512ms. Derived 2464723 facts.
Data import: 2032ms
Reasoning: 7760ms
Data export: 719ms

Timing report:
nemo [system/process/thread (ms): 10512/10510/10482]
- Reading & Preprocessing [0.7%, 70ms, 1x]
- Reasoning [92.5%, 9694ms, 1x]
  - Rules [100.0%, 9694ms, 1x]
    - Rule 37 [41.2%, 3998ms, 50x]
    - Rule 39 [14.7%, 1424ms, 28x]
    - Rule 0 [9.9%, 955ms, 2x]
    - Rule 41 [9.5%, 920ms, 154x]
    - Rule 36 [5.7%, 548ms, 37x]
    - Rule 38 [4.0%, 390ms, 16x]
    - Rule 43 [1.9%, 180ms, 2x]
    - Rule 42 [1.7%, 165ms, 15x]
    - Rule 11 [1.5%, 140ms, 2x]
    - Rule 21 [1.4%, 136ms, 14x]
    - Rule 33 [0.8%, 81ms, 16x]
    - Rule 20 [0.7%, 63ms, 8x]
    - Rule 25 [0.6%, 62ms, 2x]
    - Rule 2 [0.4%, 42ms, 2x]
    - Rule 19 [0.4%, 42ms, 8x]
```

The screenshot shows the Nemo web application interface. At the top, there's a navigation bar with the Nemo logo, version 'v0.5.2-dev', and links to 'Nemo on Github', 'Web Interface on Github', and 'Docs'. A 'Give feedback!' button is also present. The main area is split into two panels. The left panel is a 'Code editor' with tabs for 'Examples', 'Open file', and 'Save file'. It contains Prolog code for calculating distances and project rejection. The right panel is 'Program execution', featuring a 'Run program' button, an 'Add local file as input' button, and a list of input files: 'endangered.csv' (322.0 B) and 'locations.csv' (105.6 KB). Below this, the 'Results' section shows 'Derived 621 facts in 0.1 seconds (11+25+8 ms)'. A table of results is displayed, including 'CIRC_EARTH (1)', 'PI (1)', 'TRUE (1)', 'closest_house (3)', 'coordinates (122)', 'coordinates_m (122)', 'distance (366)', 'endangered (21)', 'environmental_impact (2)', 'locations (781)', 'project (3)', 'project_approved (1)', 'project_height_m (3)', 'project_location (3)', 'project_location_m (3)', 'project_reject (2)', and 'project_species (4)'. At the bottom, it shows 'project_reject (2 rows)' with a 'Save all rows as CSV' button and two rows of data: '1 "Wind Turbine A"' and '2 "Wind Turbine B"'. Each row has a small icon to its right.

VSCode Extension and CLI

Web Application

User Feedback

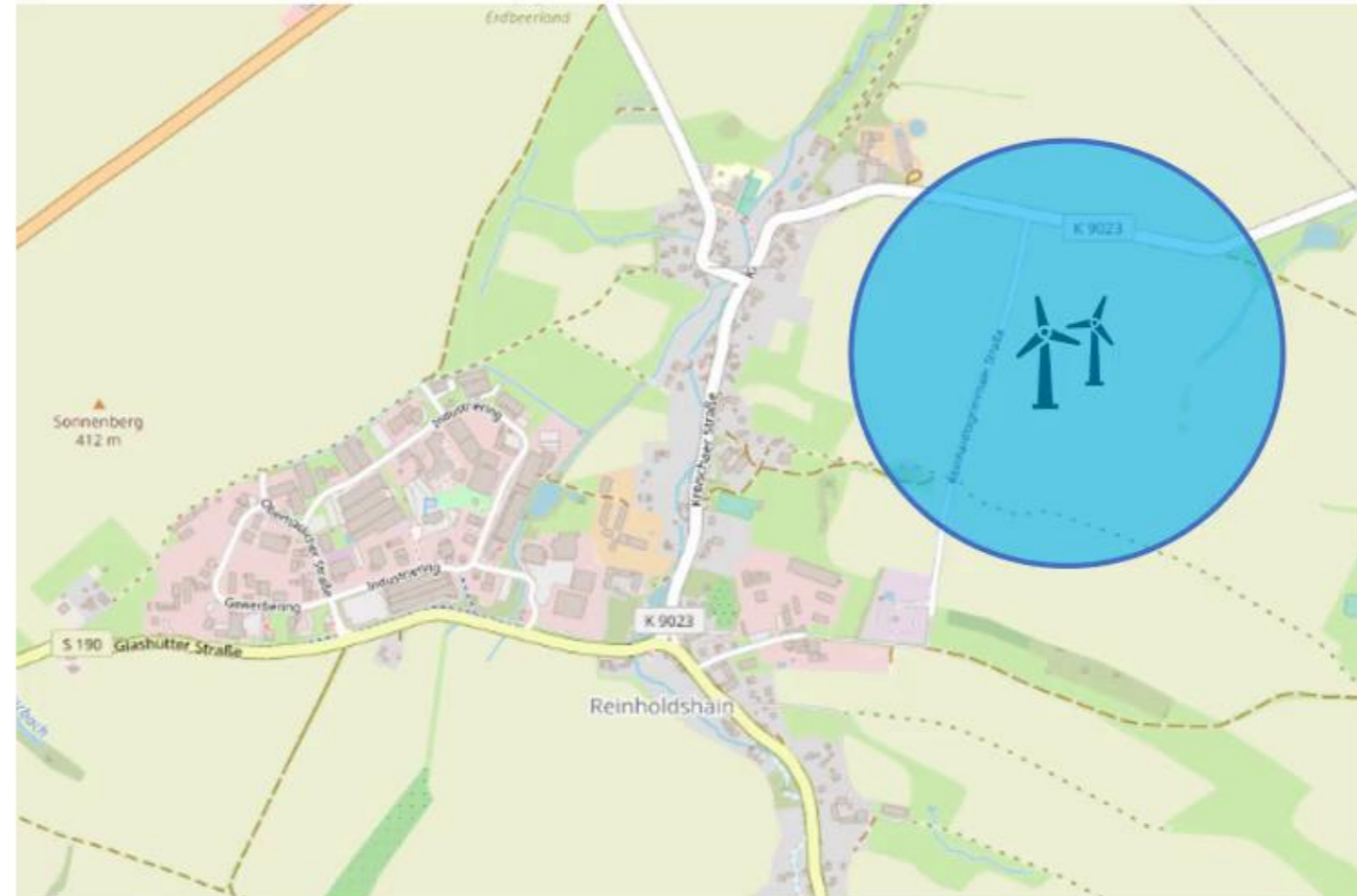
```
[202] Error: unsafe variable used in rule head: `?projekt`  
[test.rls:77:22]  
77 environmental_impact(?projekt) :- project_height_m(?project, ?height), ?height >= 50.  
                                     unsafe variable used in rule head: `?projekt`  
Help: a variable with a similar name exists: `project`  
Note: every universal variable in the head must occur at a safe position in the body
```



Nemo Demo: Wind Turbines

Rules for placing wind turbines

- Minimal distance from buildings
- If certain height: no endangered species



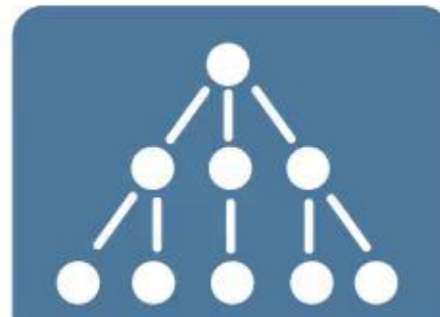
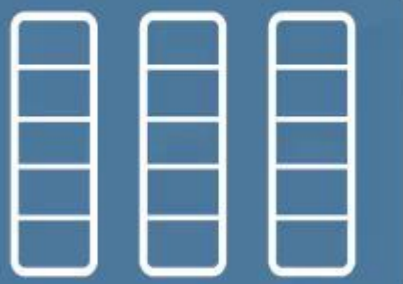
Architecture

Physical

Query Engine

Dictionary

Data Structures

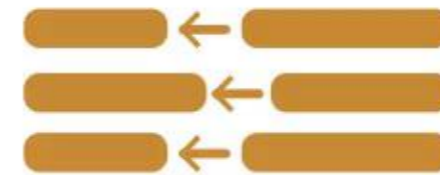


Operations

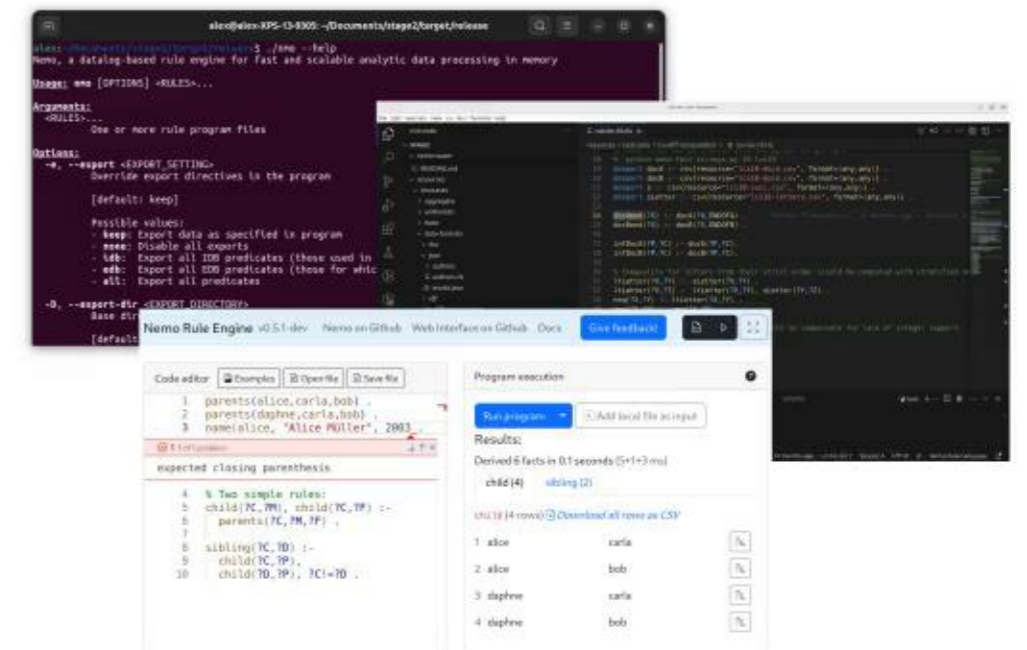
Logical

Planner

Execution Engine



Application



Data structure

Eve	Chuck	Alice
Eve	Alice	Bob
Alice	Chuck	David
Eve	Frank	Chuck
Alice	Chuck	Bob
Alice	Bob	David
Eve	Bob	Chuck

Data structure

5	3	1
5	1	2
1	3	4
5	6	3
1	3	2
1	2	4
5	2	3

Data structure

5	3	1
5	1	2
1	3	4
5	6	3
1	3	2
1	2	4
5	2	3

Alice → 1

Bob → 2

Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure

1	2	4
1	3	2
1	3	4
5	1	2
5	2	3
5	3	1
5	6	3

Alice → 1

Bob → 2

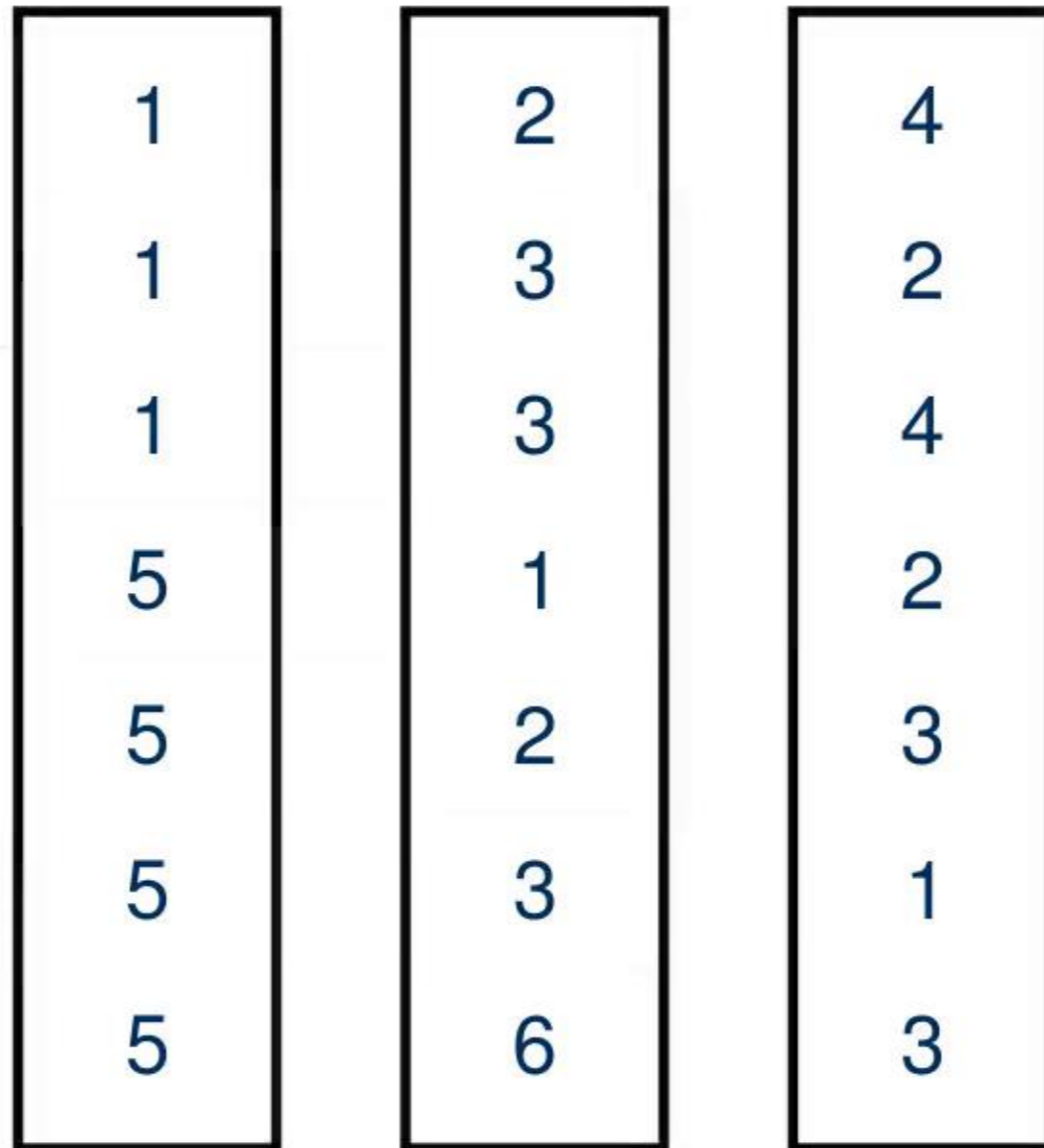
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

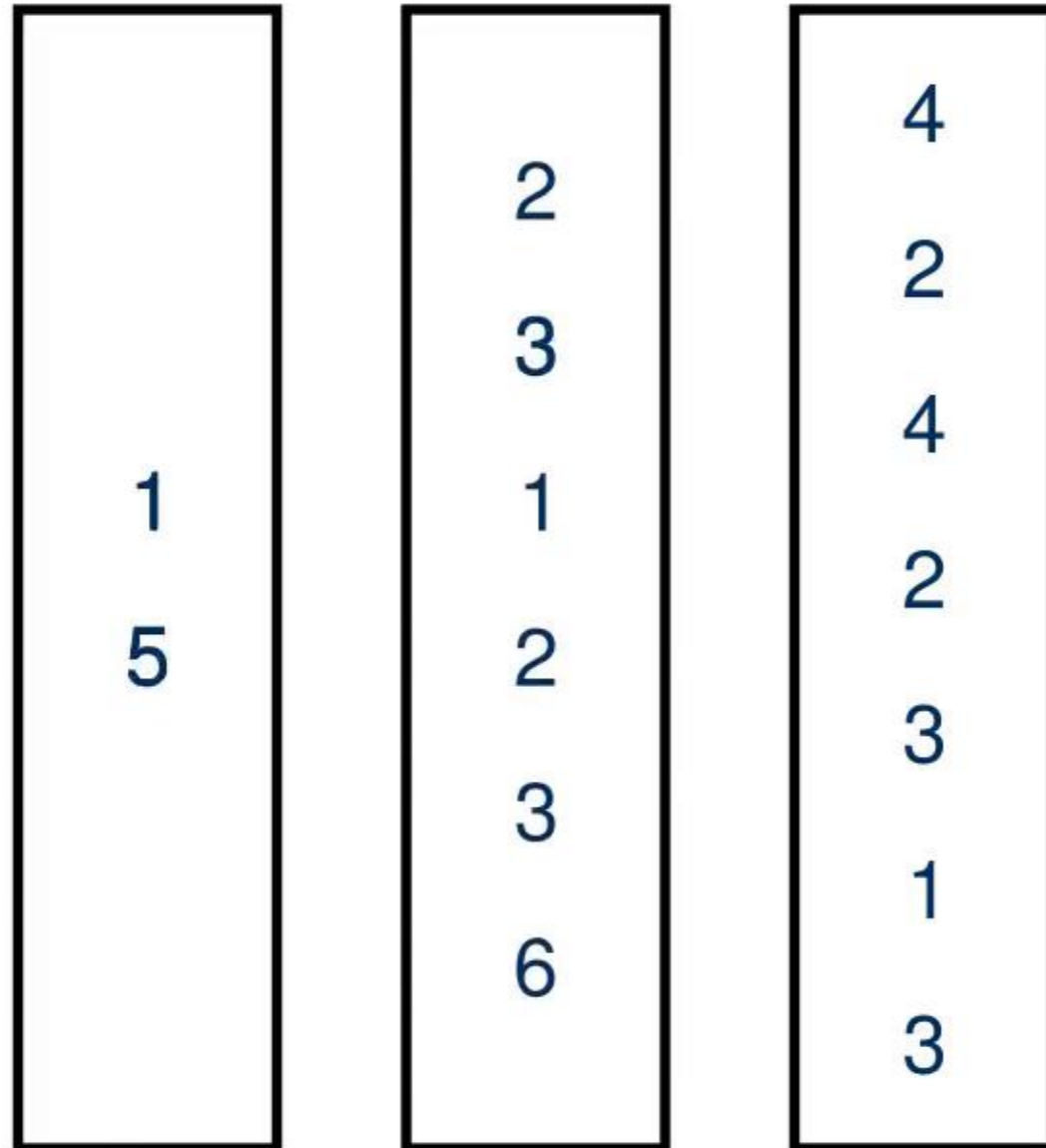
Chuck → 3

David → 4

Eve → 5

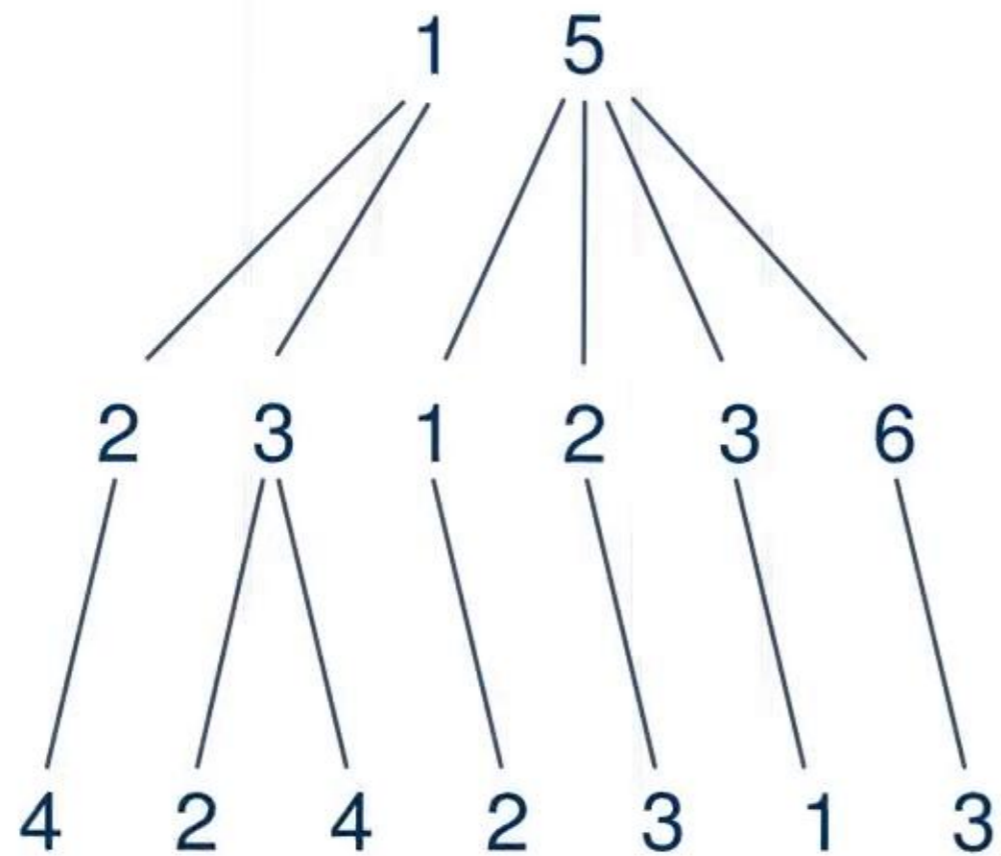
Frank → 6

Data structure



Alice → 1
Bob → 2
Chuck → 3
David → 4
Eve → 5
Frank → 6

Data structure



Alice → 1

Bob → 2

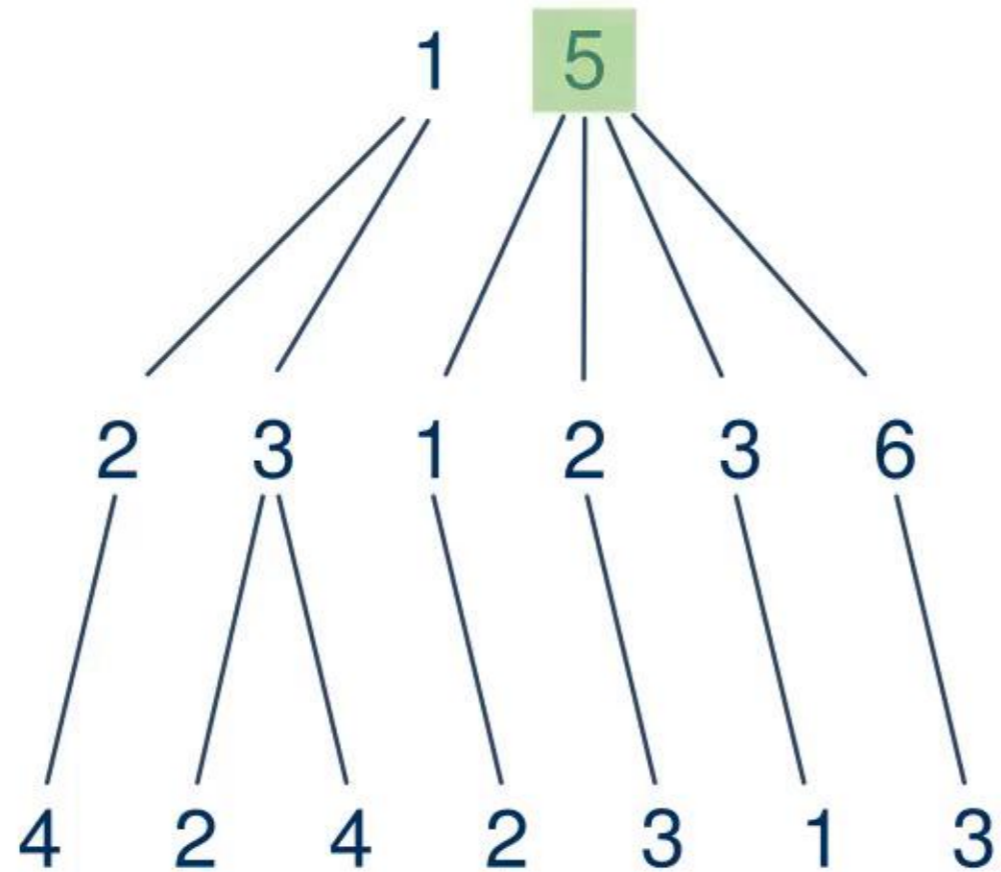
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

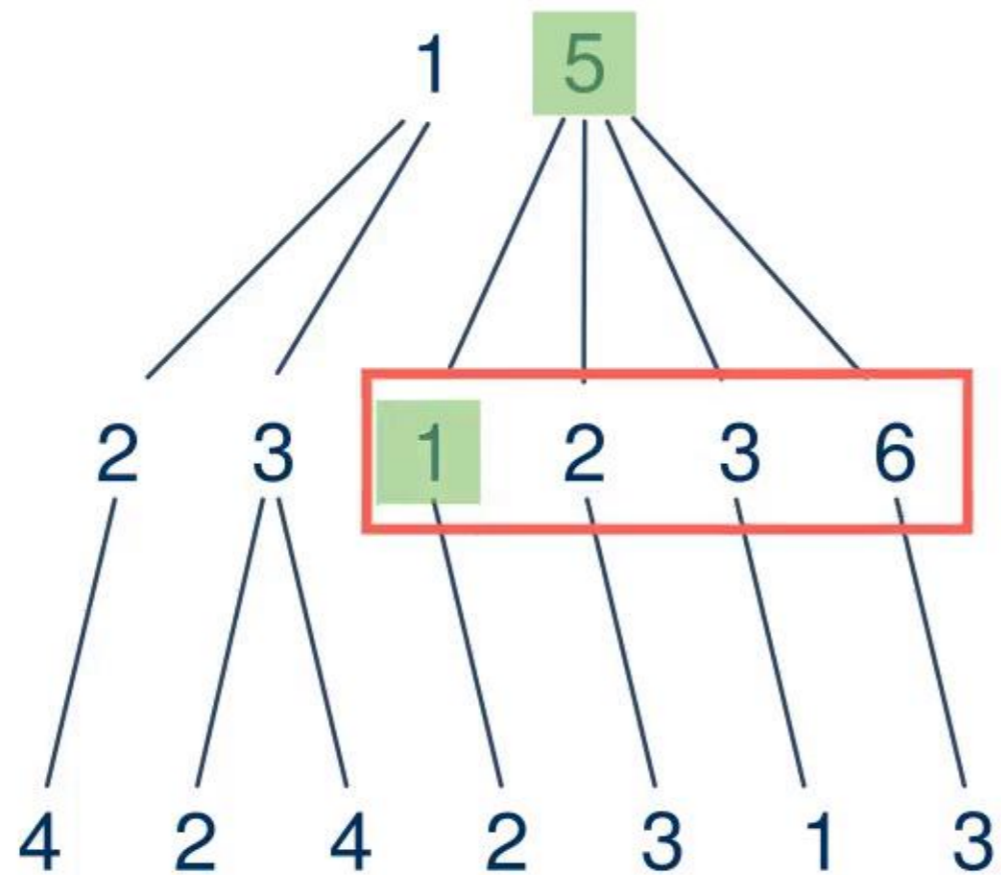
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

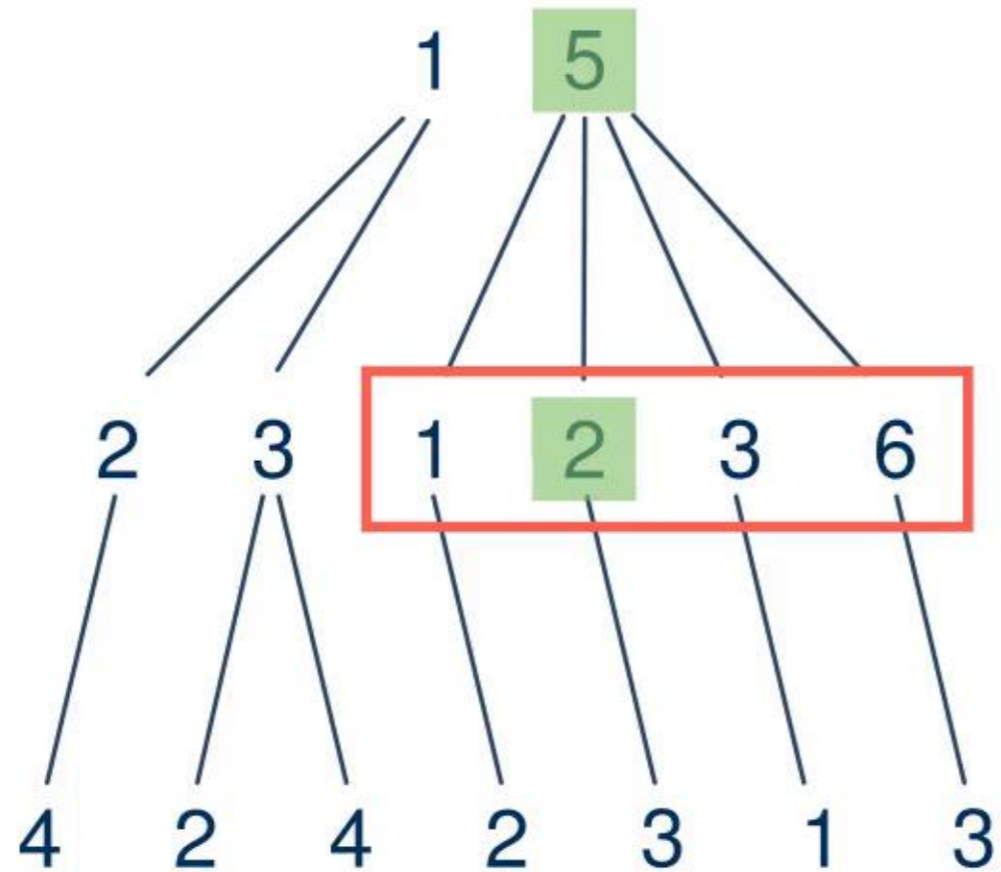
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

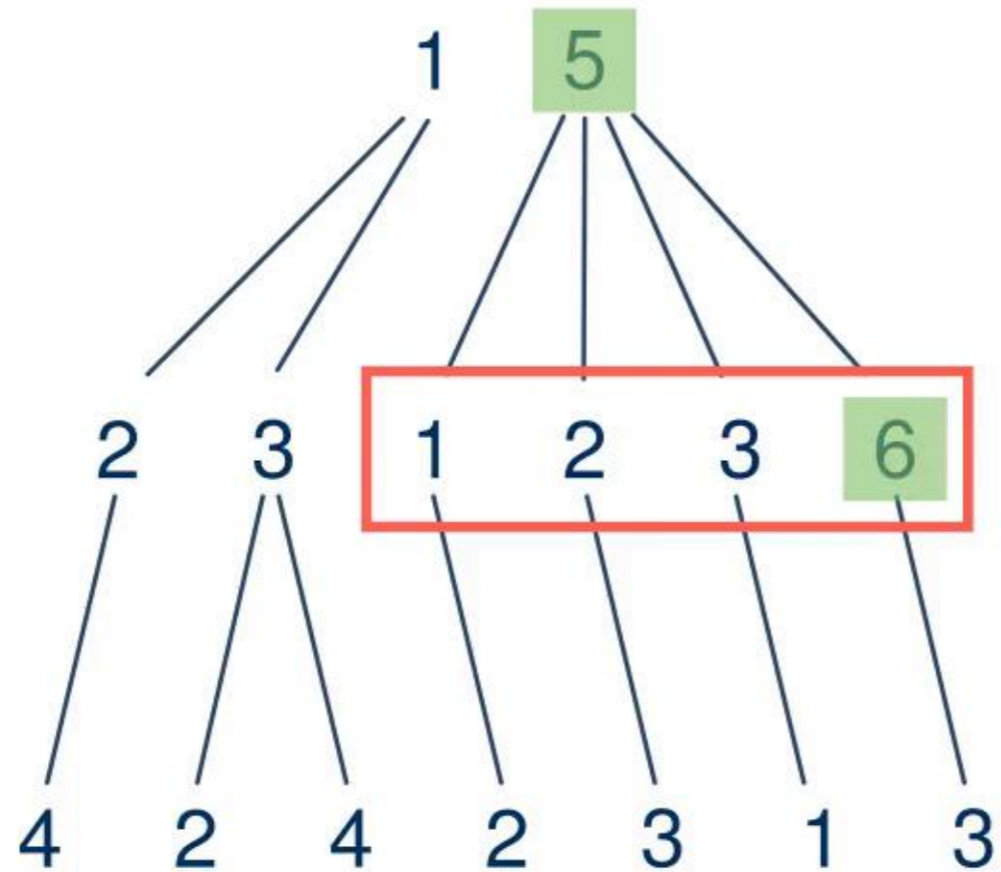
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

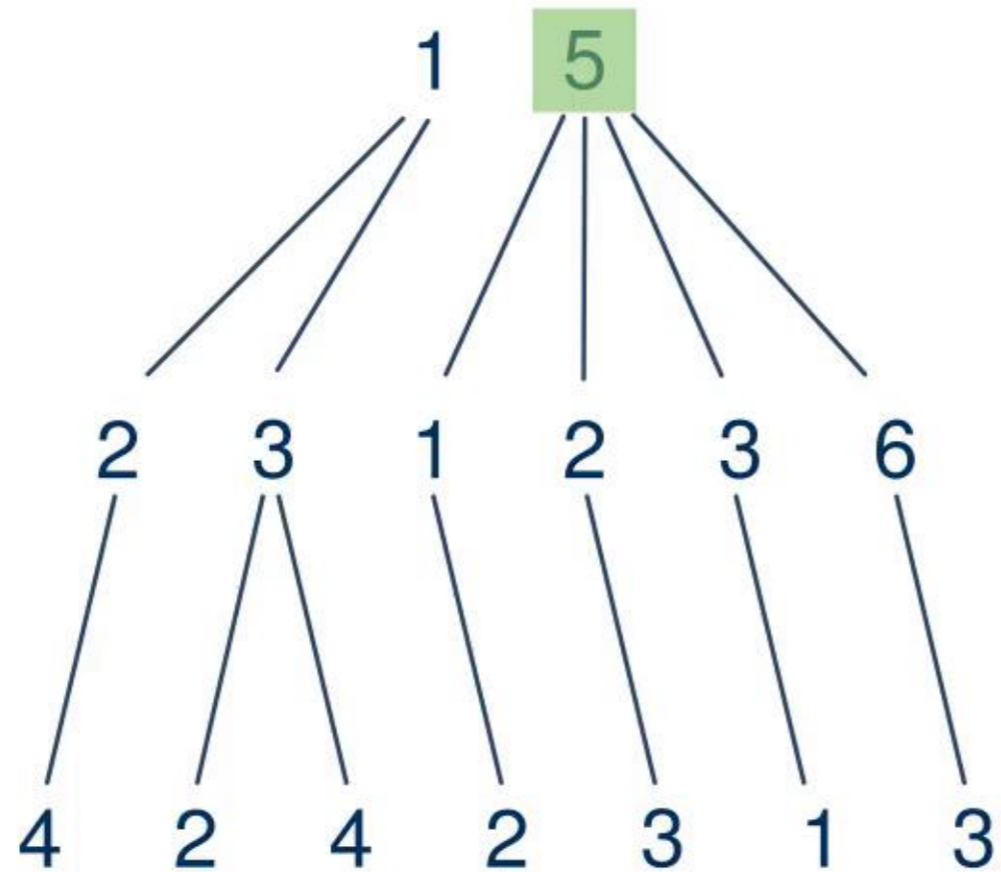
Chuck → 3

David → 4

Eve → 5

Frank → 6

Data structure



Alice → 1

Bob → 2

Chuck → 3

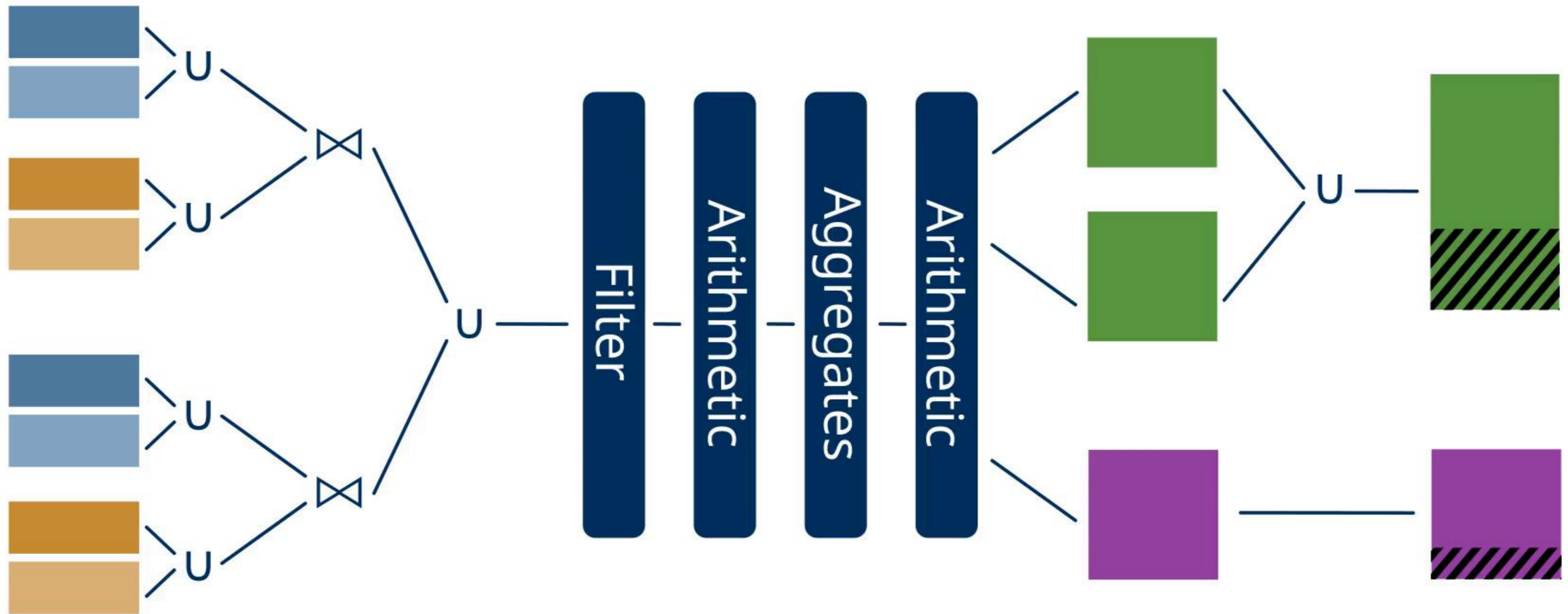
David → 4

Eve → 5

Frank → 6

Pipeline

$a(x, y)$, $b(y, x)$, $y > 2x$, $z = x+y$, $a = \#min(z)$, $r = a/2 \rightarrow p(x, r)$, $p(y, r)$, $q(z)$



Performance Evaluation

Compared Engines



VLog

Runtime

- EL Reasoning on medical ontologies
- Existential Rule Benchmarks: Benchmarking the Chase

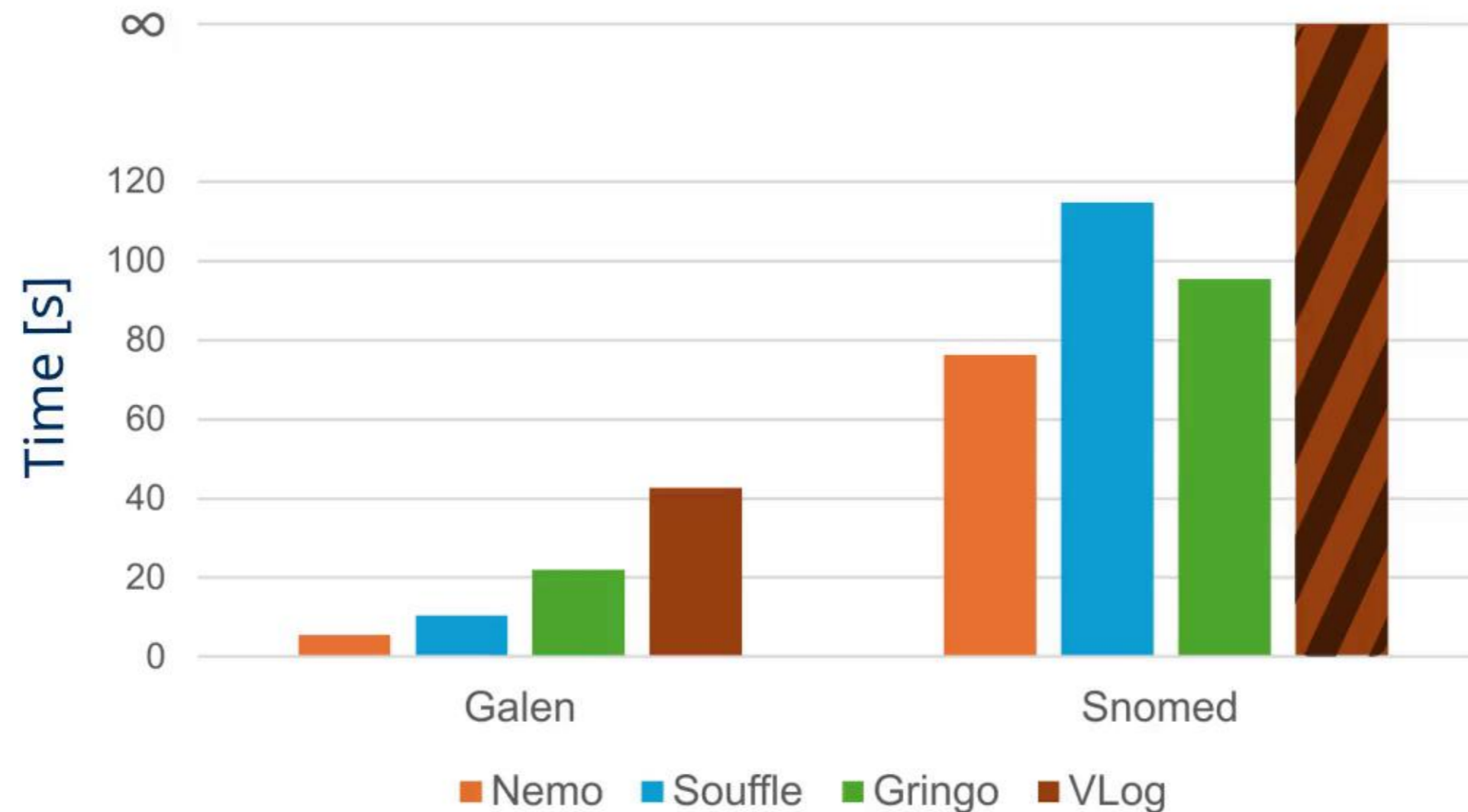
Scalability

- Reasoning Task on Wikidata

Evaluation: EL Reasoning

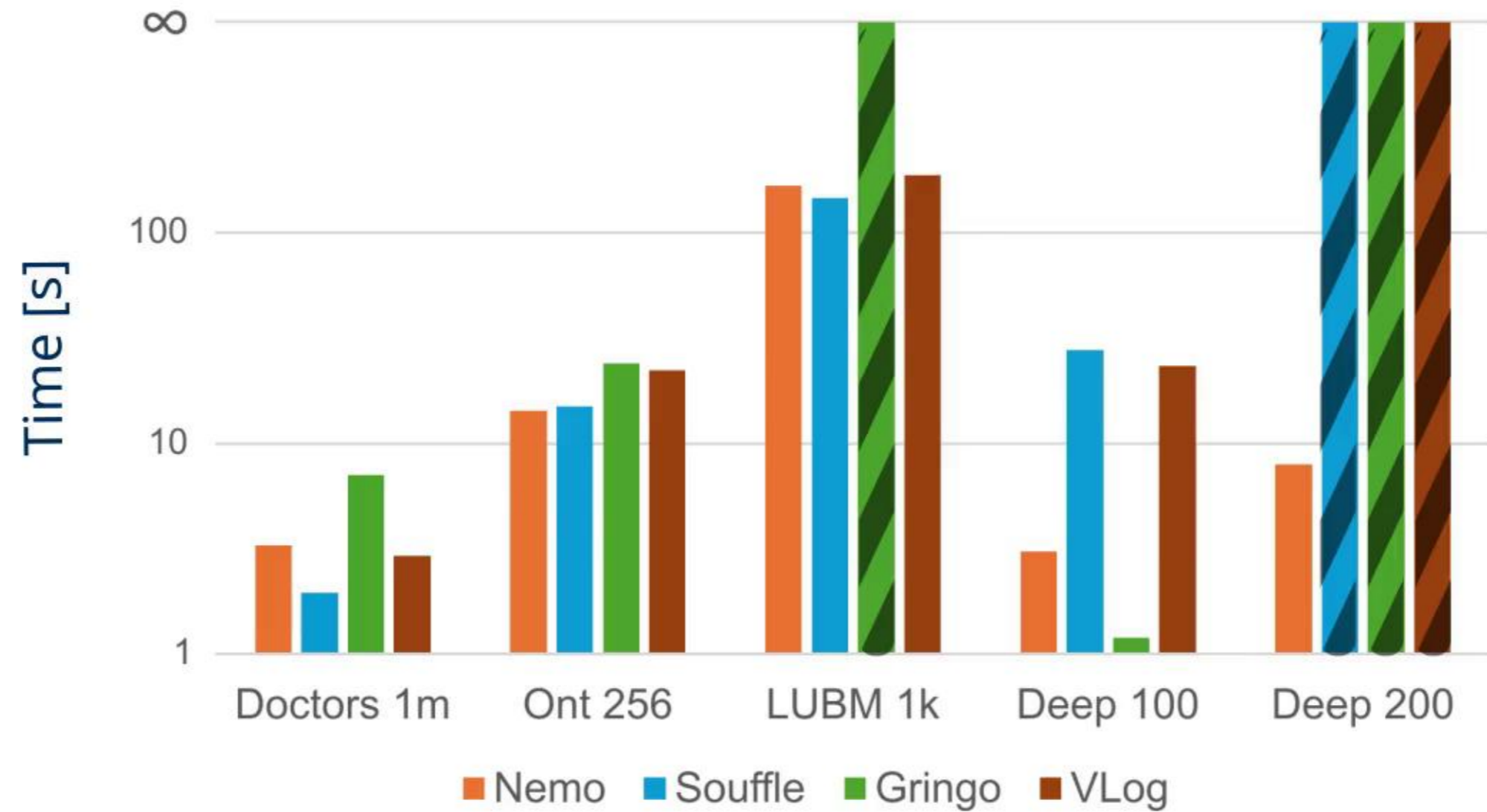
EL Reasoning with Datalog (12 rules, ≤ 5 body atoms, arity ≤ 3)

Input: Galen (facts: 140K) and SNOMED (facts: 1.5M)



Evaluation: Benchmarking the Chase

Established Benchmarks for the Chase [Benedikt et al., PODS'2017]



Evaluation: Wikidata

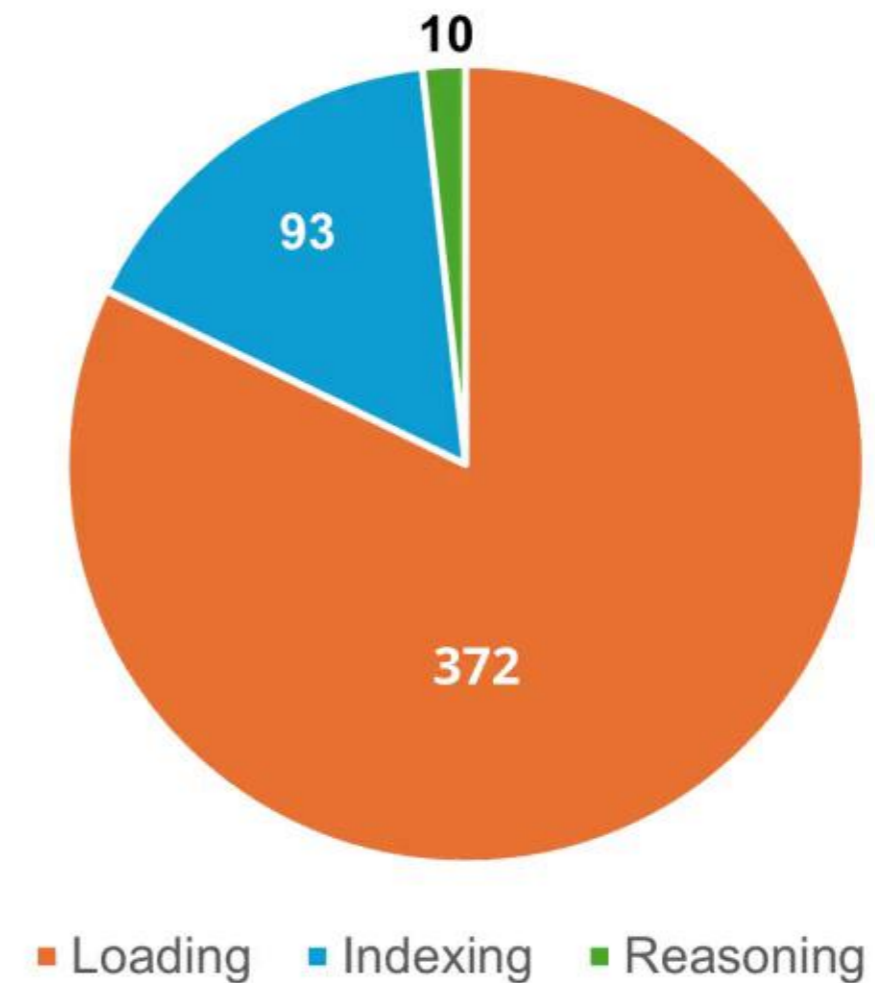
Large-Scale reasoning on (Truthy) Wikidata

Input

- Processing task in creating YAGO KB [Suchanek et. al., SIGIR'24]
- 8 billion triples
- 61GB (compressen N-Triples)

Result

- Time: 8 hours
- Memory: 220GB



Summary

Nemo

- Versatile and scalable Datalog engine
- Free and open-source
- Advanced tool support



Nemo
Graph Rule Engine

Upcoming

- Language extensions
 - List, Sets, Maps
 - Modules and user-defined functions
- Better explainability
- Performance improvements



<https://tools.iccl.inf.tu-dresden.de/nemo/>