Regular Path Query with SQLite Implementation

QIWEN WANG[1], HUI LYU[2], YANG CAO[2,3], BERTRAM LUDÄSCHER[1,2,3]

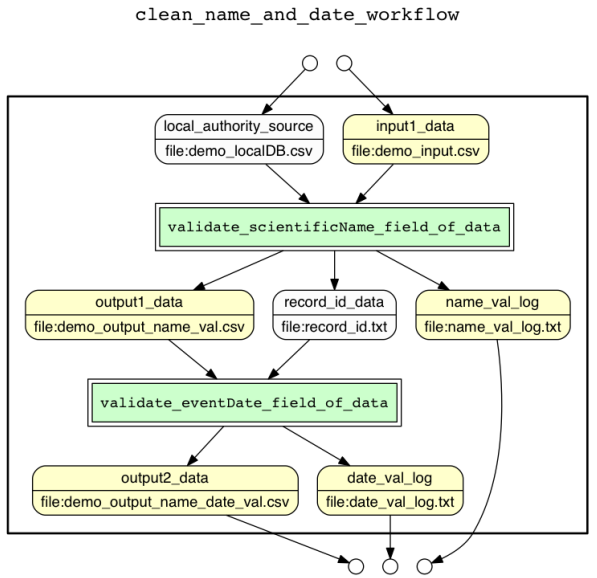
[1]DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN,

[2]SCHOOL OF INFORMATION SCIENCES, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN,

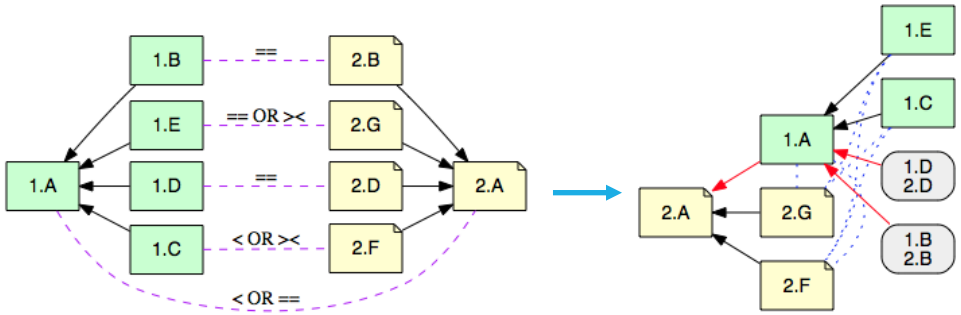
[3] NATIONAL CENTER FOR SUPERCOMPUTING APPLICATIONS

Our Group

- Lead by Bertram Ludäscher
- Data and knowledge management
- Design and optimize scientific workflow
- Data provenance



YesWorkflow Project/
Scientific workflow modeling



Euler Project/
Taxonomic alignment

Motivation

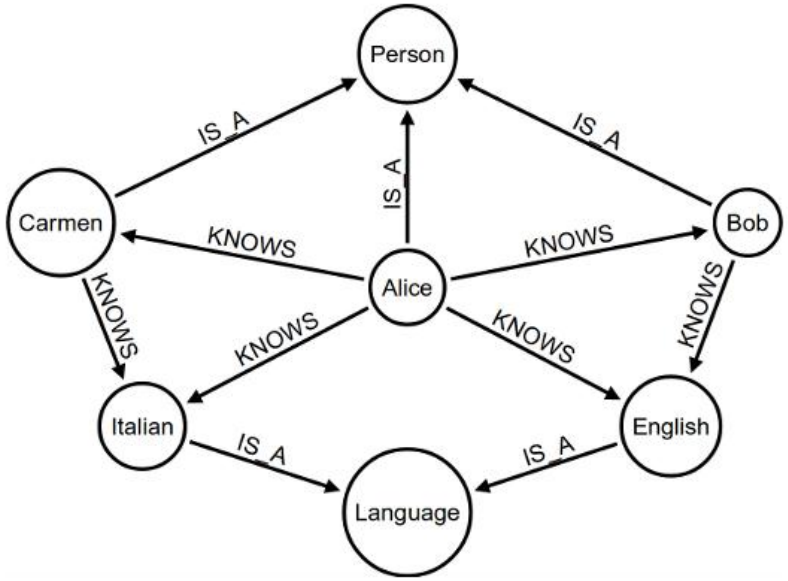
- **SQLite** is used more than all other database engines combined.
- **Regular Path Queries** (RPQ) make many recursive graph queries easy.

$\Longrightarrow$ Goal: Develop a simple **RPQ-to-SQLite compiler**

- Many possible applications, e.g. querying of provenance graphs, workflow graphs, social network graphs, etc.

What is Regular Path Query(RPQ)

- A Data Graph $G = \langle V, E, \rho \rangle$, where:
  - V is a finite set of nodes;
  - $E \subseteq V \times \Sigma \times V$ is a set of labeled edges, Σ is the set of labels
  - ρ is a function that assigns a data value to each node
- Regular Path Query:
  - A query in the form of regular expression
  - Find matching in the data graph to the regular expression
  - Return pairs of nodes in a graph



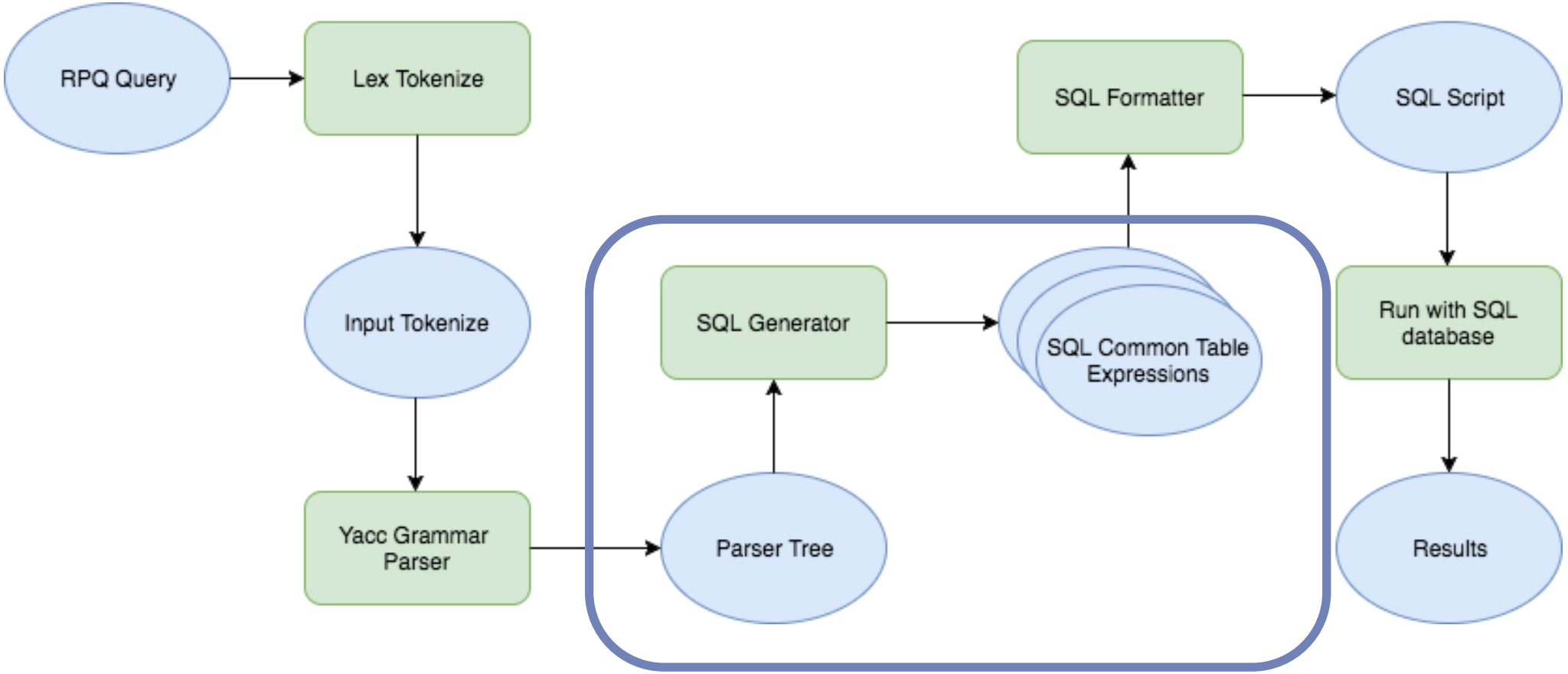
Question: Who is a person?
Query: IS_A.[Person]
Result: (Alice , Person)
(Bob , Person)
(Carmen, Person)

Challenge

- Should allow **conjunctive RPQs** (CRPQs) to support more queries.
- A variant of RPQs are part of SPARQL and can also be easily implemented in Datalog/Prolog, but widespread deployment of SQLite makes it a more desirable platform.
- Earlier RPQ implementations on top of PostgreSQL were not very efficient.

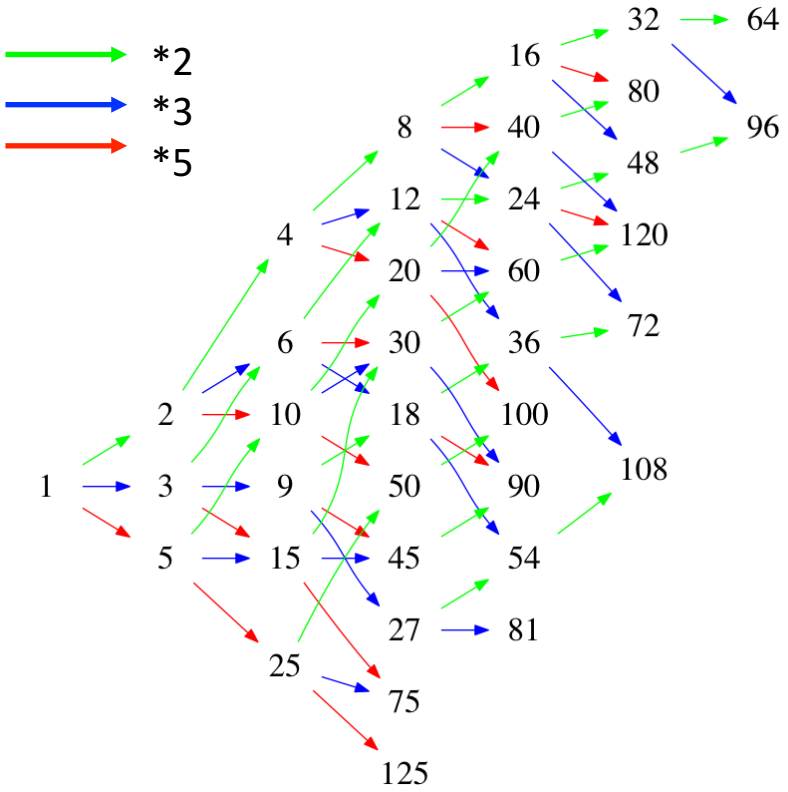
$\Longrightarrow$ Aiming for a **lightweight, reasonably efficient** approach for SQLite.

Approach – a SQLite Compiler



Hamming Numbers

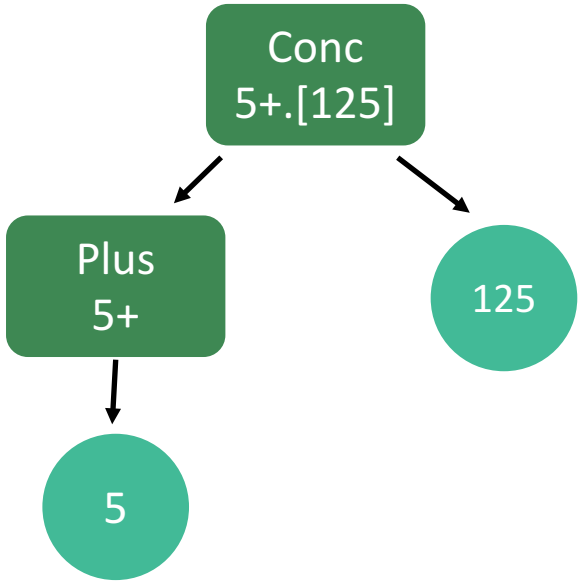
- **English query**: What are the Hamming numbers have at least one "*5" along the way to the Hamming number 125?
- **Graphically**: Find all X, such that $[X] - 5+ \ -> \ [125]$
- **RPQ query**: $5+.[125]$



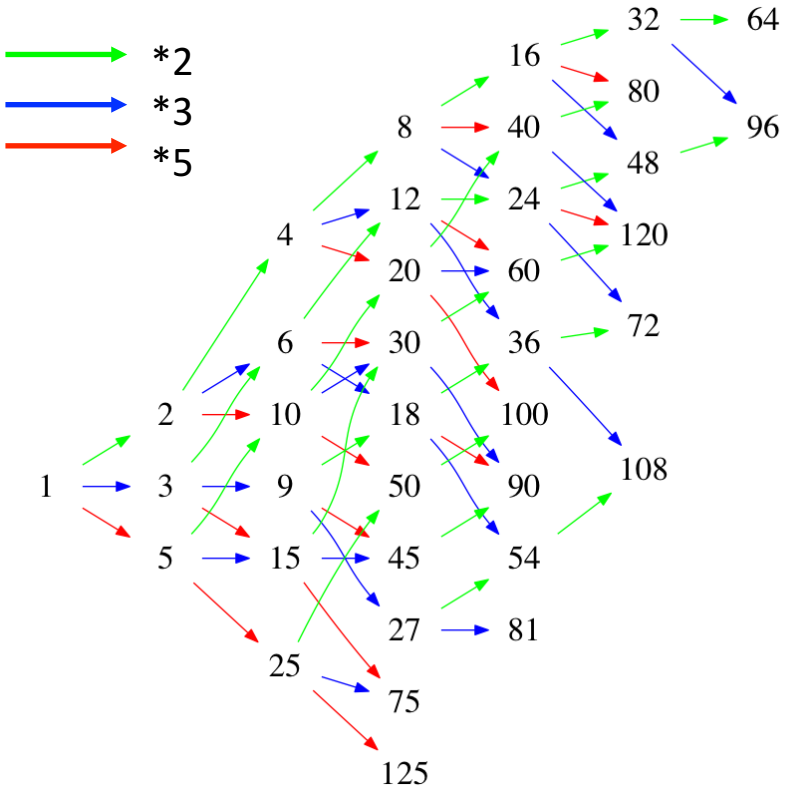
Fish

Hamming Numbers

- **RPQ query**: $5+.[125]$



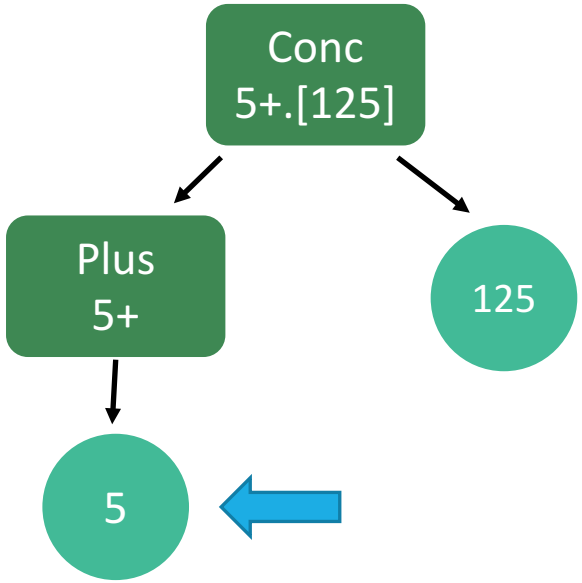
Traverse with Post-order traversal



Fish

Hamming Numbers

- **RPQ query** $5+.[125]$ to **SQLite**



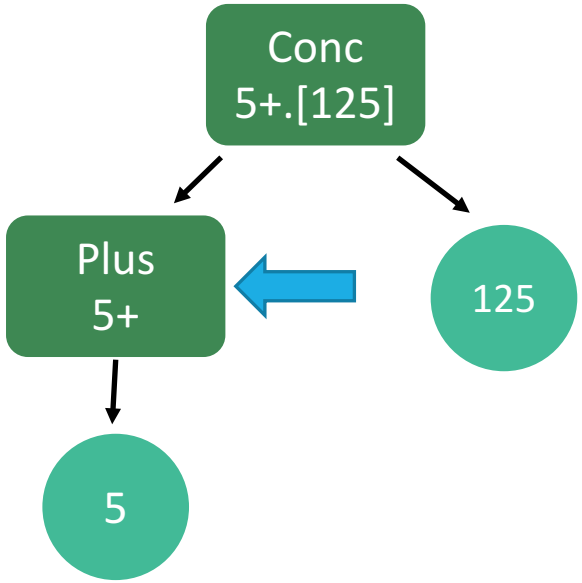
Traverse with Post-order traversal

Template to Generate CTE for a label:

```
Temp1(start, end) AS (
   SELECT fish.startNode, fish.endNode
   FROM fish
   WHERE fish.label = "5")
```

Hamming Numbers

- **RPQ query** $5+.[125]$ to **SQLite**



Traverse with Post-order traversal

Template to Generate CTE for PLUS:

```
temp2(start, end) AS (
   SELECT * FROM temp1
   UNION
   SELECT a.start, b.end
   FROM temp2 AS a, temp1 AS b
   WHERE a.end = b.start)
```

Note: {0} is the new temp table,
{1} is the previously generated temp table corresponds to label 5

Hamming Numbers

- **RPQ query** $5+.[125]$ to **SQLite**

```
WITH RECURSIVE
  Temp1(start, end) AS (
  ...
  WHERE fish.label = "5"),

  temp2(start, end) AS (
  ...
  WHERE a.end = b.start),

  temp3(start, end) AS (
  …
  WHERE a.endNode = "125”),

  temp4(start, end) AS (
  …
  WHERE a.end = b.start
```

select * from temp4;

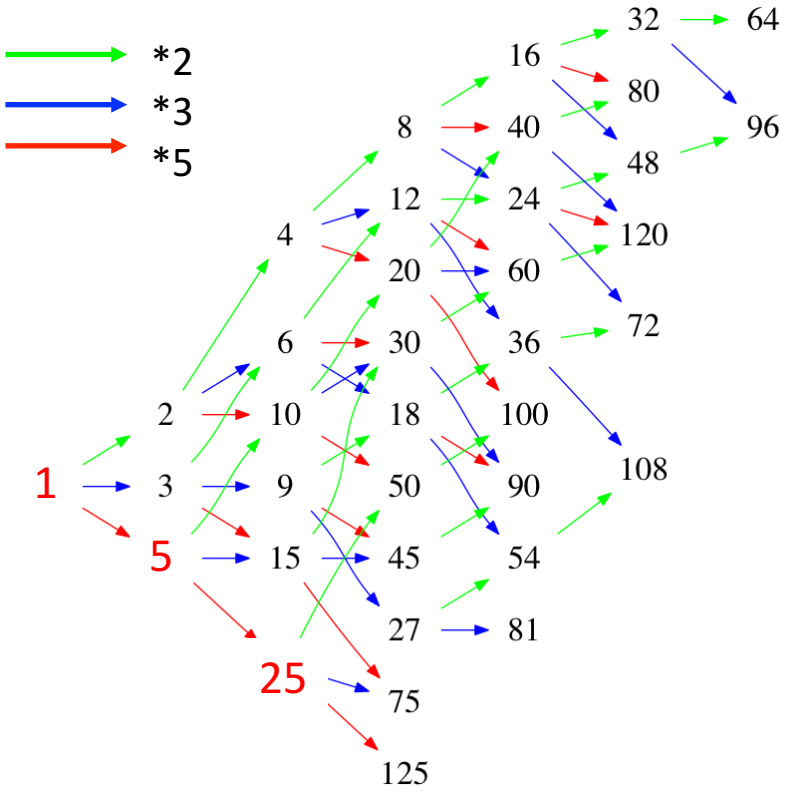
Hamming Numbers

Run SQLite representing RPQ query 5+.[125]

Results:

StartNode	EndNode
1	125
5	125
25	125

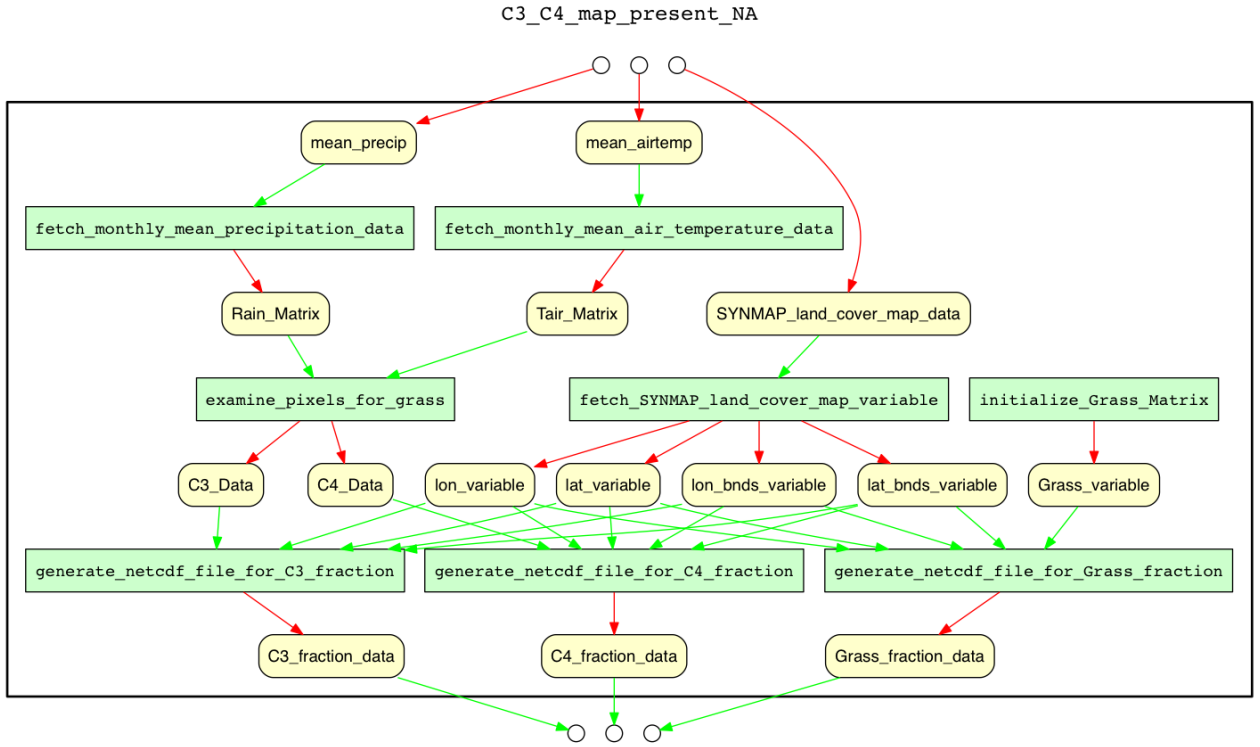
Can you find the corresponding path on the fish graph?



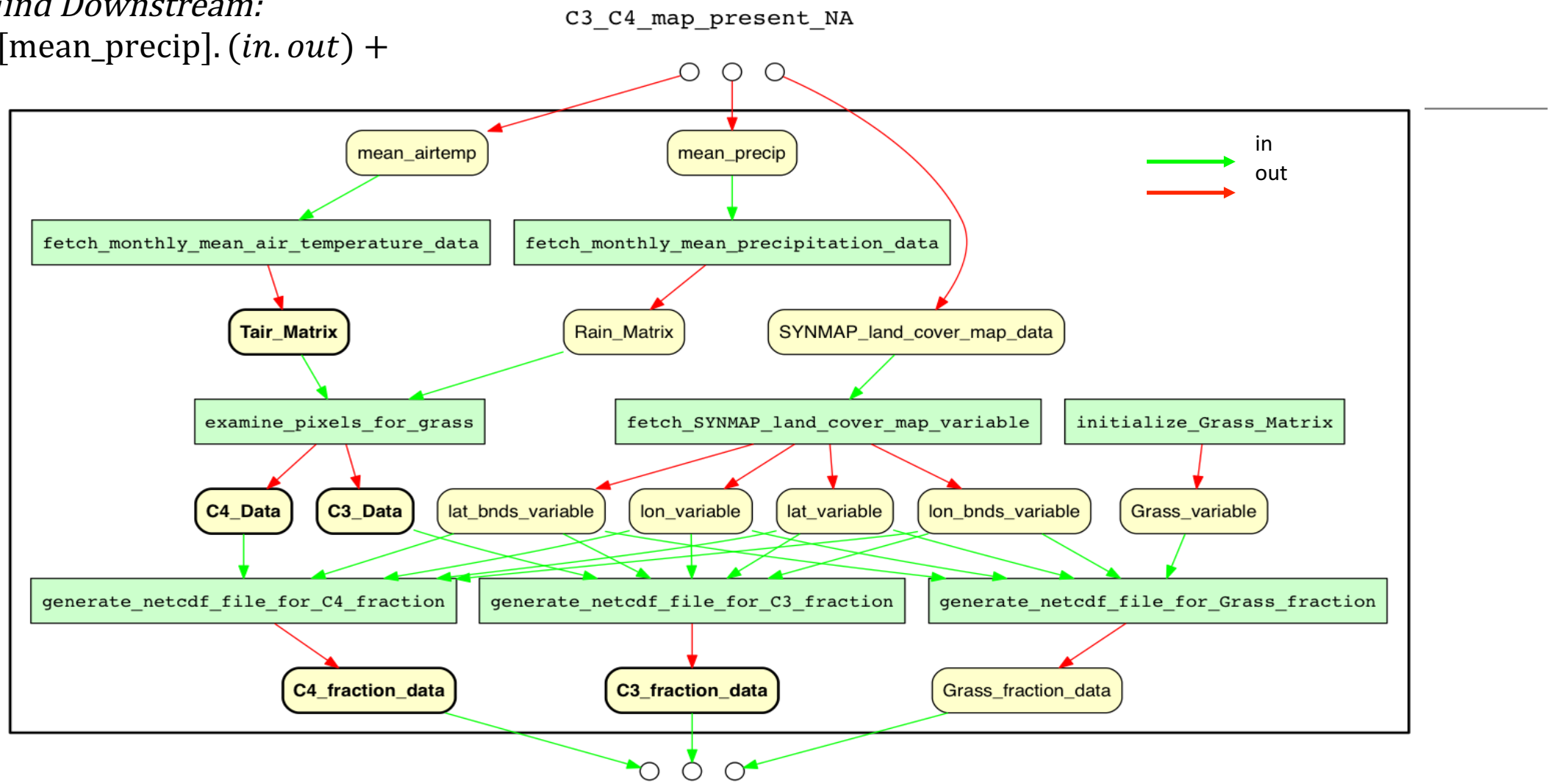
Fish

YesWorkflow(YW) Use Case

- **English query**: what are script outputs that are downstream of input data **mean_precip**?
- **RPQ query**: $[\text{mean\_}precip].(in.out) +$



Find Downstream:
[mean_precip]. (*in*. *out*) +



Conclusion

- In this research, we implemented an **RPQ to SQLite compiler** that uses CTEs to express recursive queries.
- This engine can be run on any system that comes with SQLite.
- It also supports complex traversals on any type of graph, but is especially useful in answering **data dependency queries** over workflow and provenance graphs.
- Git Repo: https://github.com/qwang70/rpq-engine-project