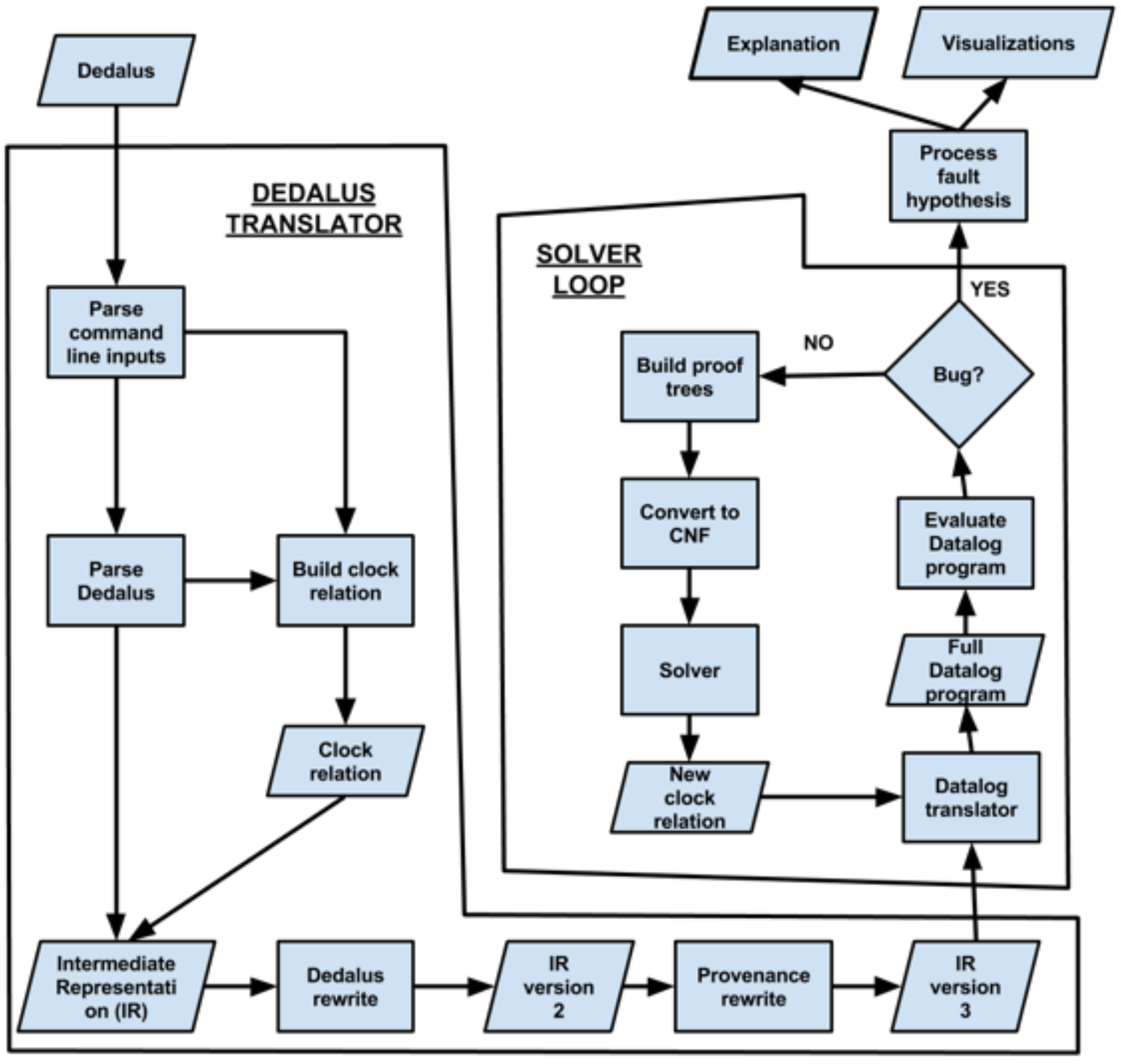
PyLDFI Update

Kathryn Dahlgren
Disorderly Labs
UCSC
May 2, 2017

Architecture



~ State of the Code ~
Address

POSITIVE NEWS

1. Can read Dedalus files adhering to some subset of the syntax supported by Molly.
2. Can run Dedalus models using C4.
3. Can produce provenance trees based on the results.
4. Can convert provenance trees to CNF formulas.
5. Can solve CNF formulas for fault hypotheses.

~ State of the Code ~
Address

POSITIVE NEWS

1. Can read Dedalus files adhering to some subset of the syntax supported by Molly.
2. Can run Dedalus models using C4.
3. Can produce provenance trees based on the results.
4. Can convert provenance trees to CNF formulas.
5. Can solve CNF formulas for fault hypotheses.

NEGATIVE NEWS

1. Correctness bugs in solver.
   > Does not recognize valid fault hypotheses.

Open Issues

Obstacles toward processing Dedalus programs:

1. Solver correctness.
2. Syntax parsing.

Elasticsearch obstacles

1. Syntax parsing
2. Type conversion.

Neo4J

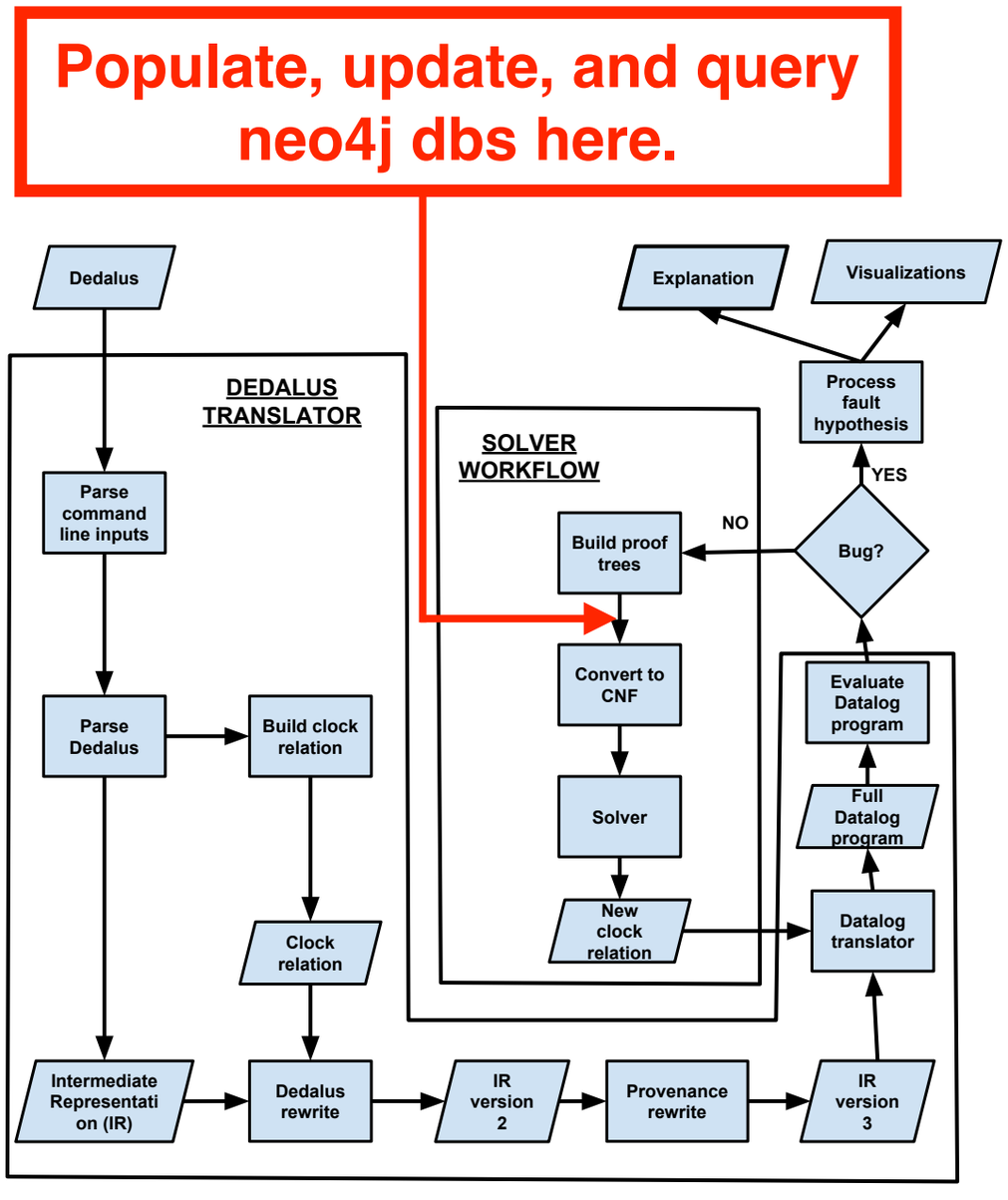
BIG IDEA

1. Store a long-lived version of the provenance tree in a graph database.
2. Use cool graph database features to ease the process of building simplified graphs.
3. Augment graph with new lineage after each “good” run.

Integration with PyLDFI

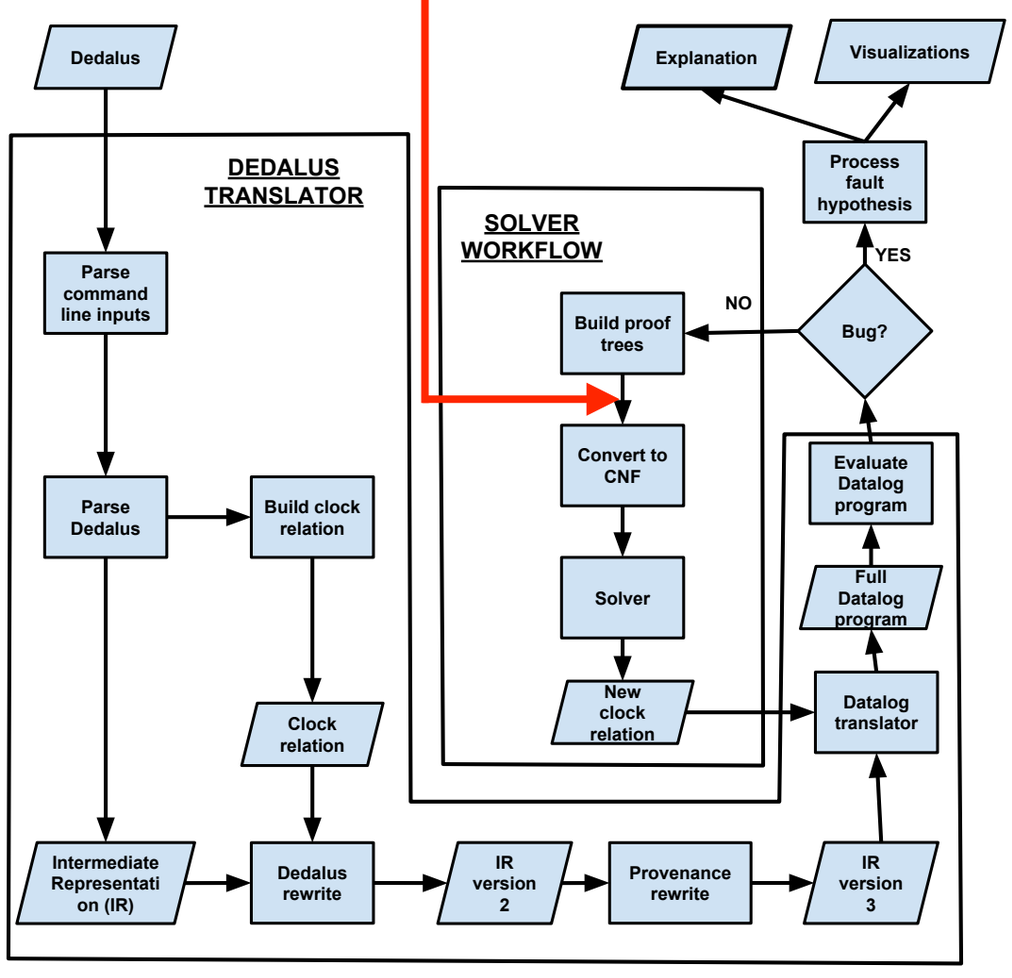
1. Need to create the neo4j db in the driver (see IR db creation) (driver1.py).
2. Need to grab the initial good provenance tree (LDFICore.py).
3. Need to output simplified provenance tree for conversion into CNF (LDFICore.py).
4. Need to augment provenance tree with lineage of any subsequent good runs (LDFICore.py).

Neo4J


Populate, update, and query neo4j dbs here.
Dedalus
DEDALUS TRANSLATOR
Parse command line inputs
Parse Dedalus
Build clock relation
Clock relation
Intermediate Representation (IR)
Dedalus rewrite
IR version 2
Provenance rewrite
IR version 3
SOLVER WORKFLOW
Build proof trees
Convert to CNF
Solver
New clock relation
Explanation
Visualizations
Process fault hypothesis
YES
NO
Bug?
Evaluate Datalog program
Full Datalog program
Datalog translator

Neo4J

Populate, update, and query neo4j dbs here.



1. Grab provTreeComplete reference.
2. Convert to neo4j graph here.
3. If not initial run, integrate new graph with existing graph.
4. Simplify.
5. Output simplified graph as RGG.

```
else :
  # ---------------------------------------------- #
  # 4. get provenance trees

  if PROV_TREES_ON :
    provTreeComplete = self.buildProvTree( parsedResults, self.argDict[ "EOT" ], fault_id, self.cursor )

# ---------------------------------------------- #
# 5. generate CNF formula

provTree_fmla    = self.tree_to_CNF( provTreeComplete )
return_array[1] = provTree_fmla  # update cnf_formula part of returns

# ---------------------------------------------- #
# 6. solve CNF formula

finalSolnList = self.solveCNF( provTree_fmla )
finalSolnList = self.removeSelfComms( finalSolnList )
if DEBUG :
  finalSolnList = self.removeCrashes( finalSolnList ) # debugging only
return_array[2] = finalSolnList  # update solutions part of returns
```

When Neo4J?

When Neo4J?

Anytime is an optimal time, really.

PyLDFI Development

TODOs:

1. Solver correctness.
2. Expand syntax support.
3. Testing (unit tests + user tests).

PyLDFI Development

TODOs:

1. Solver correctness.
2. Expand syntax support.
3. Testing (unit tests + user tests).

Prereqs for New Developers :

1. Fork repo.
2. Understand LDFI workflow.
3. Understand PyLDFI design.
4. Write and run some toy programs.
5. Play with debugging the programs/PyLDFI and get the programs to run/break.
6. Keep me posted! 😄

Reporting Bugs

Method 1 : ***Isolate buggy examples***

1. Create a dev_test/
2. Screen shot/save error output.

Method 2 : ***Map example bugs to specific PyLDFI modules***

1. Create a unit test in qa/
2. Screen shot/save error output.

Links

- PyLDFI: https://github.com/KDahlgren/pyLDFI