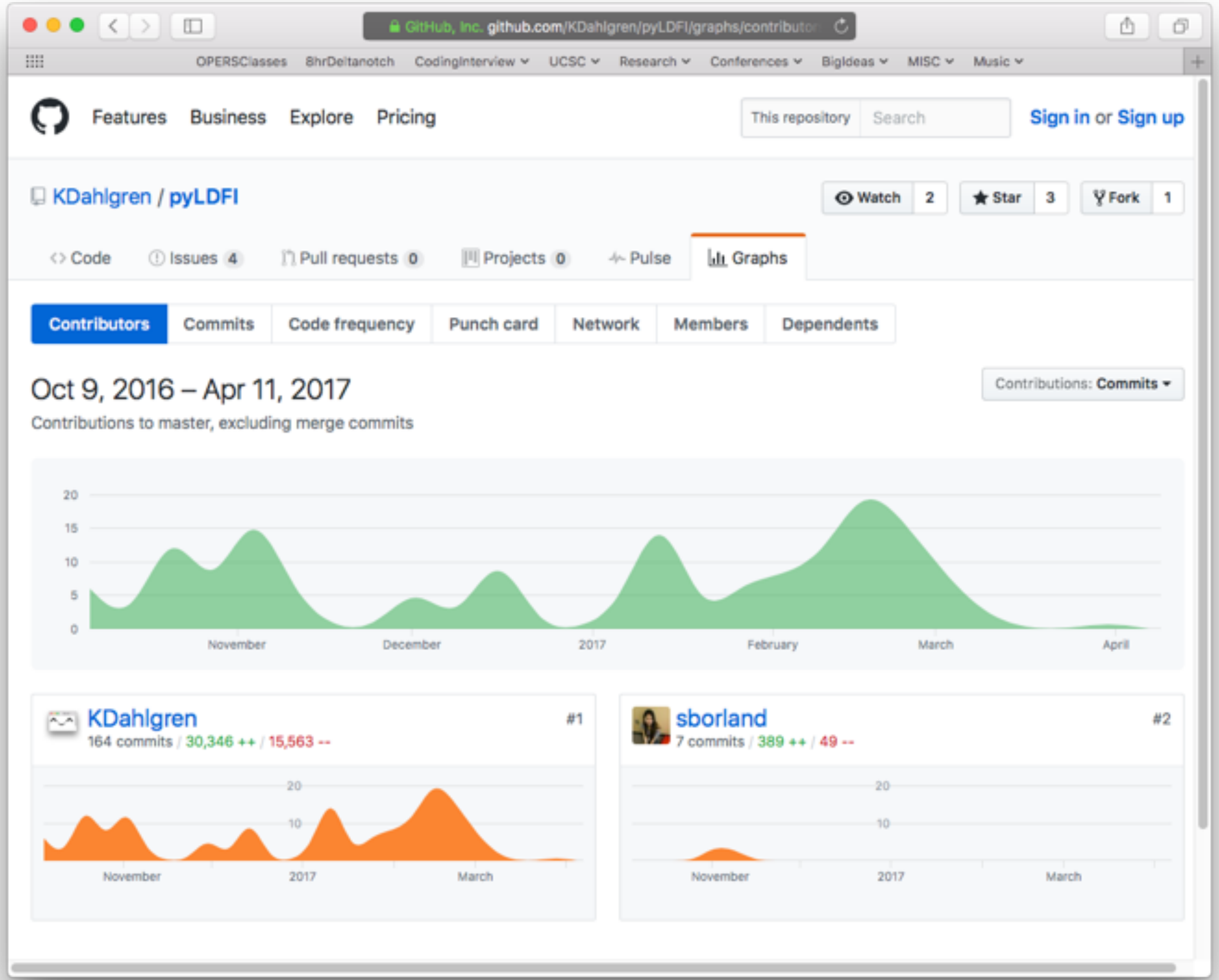
PyLDFI Code Review: Episode 1

Kathryn Dahlgren
Disorderly Labs
UCSC
April 11, 2017

PyLDFI

- A Python Implementation of LDFI.
- Project started Fall 2016.
- PyLDFI Development Group (established Fall 2016)
    - Kathryn, Sarah, Kamala, Asha
    - (Kaos Kadets?)



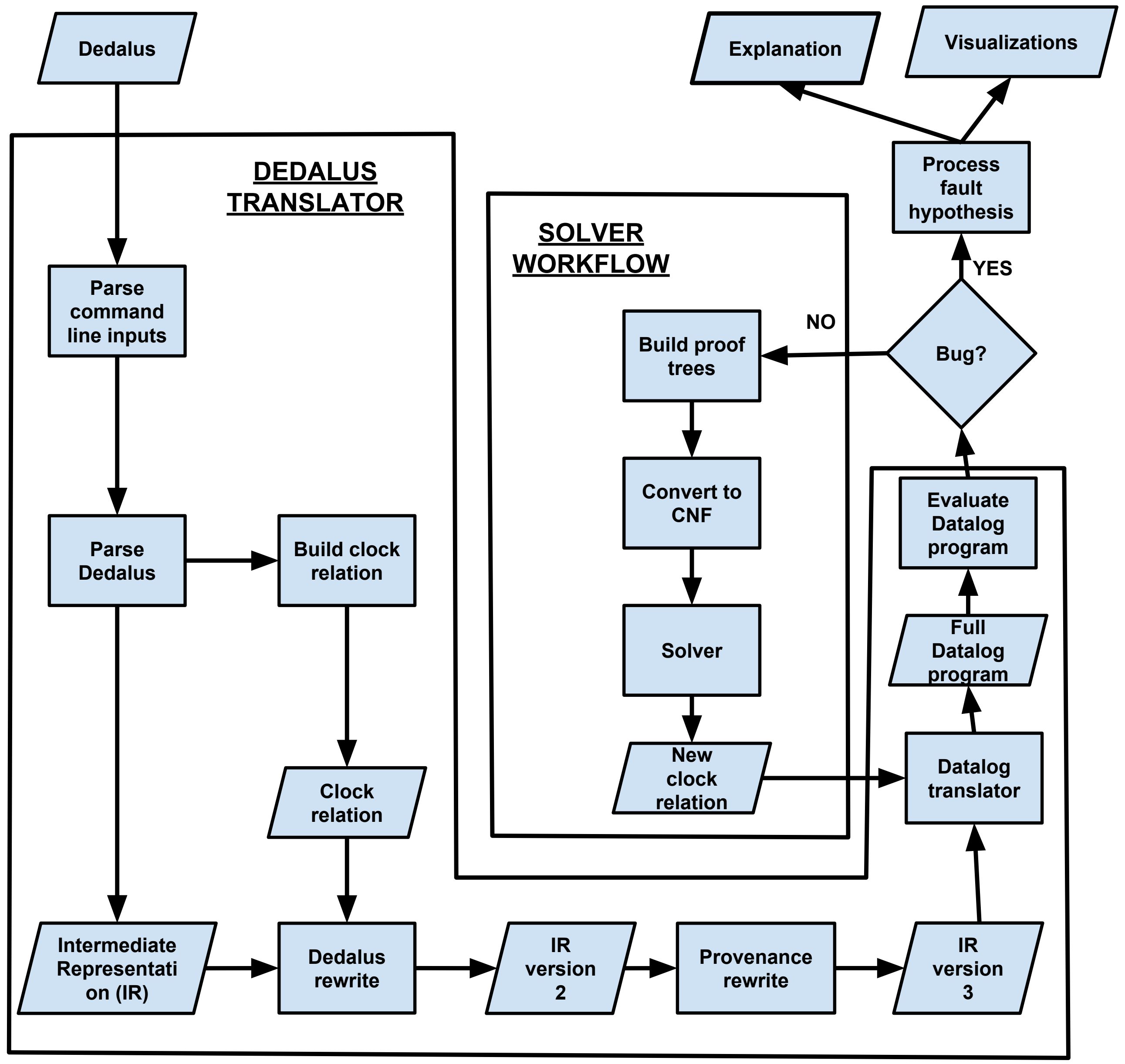
Oct 9, 2016 – Apr 11, 2017

Contributions to master, excluding merge commits

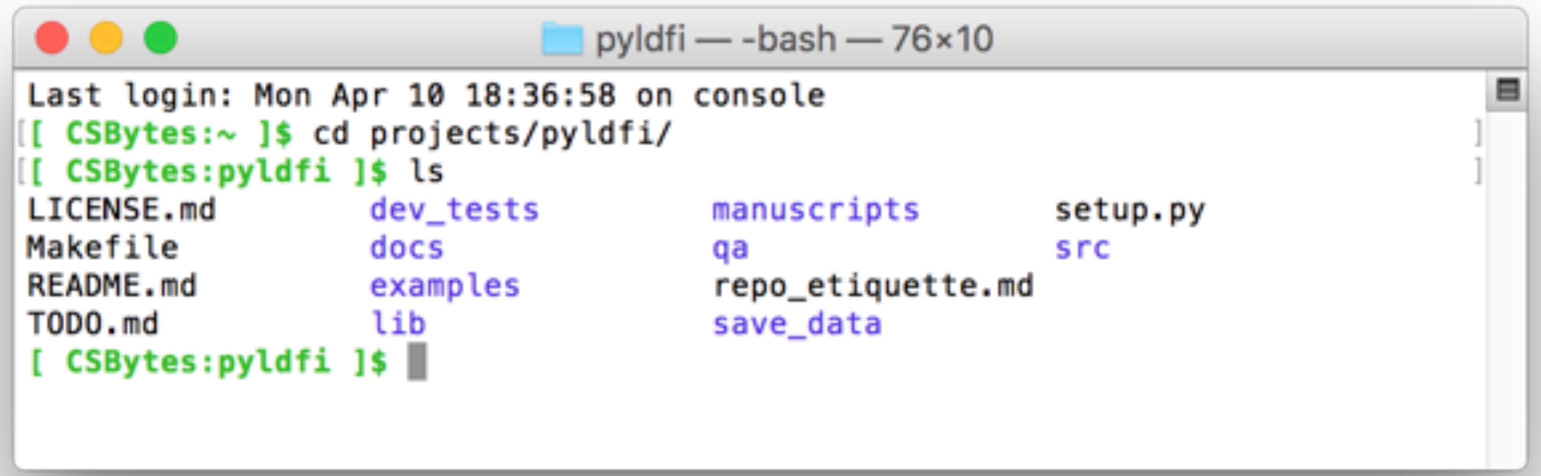
KDahlgren #1
164 commits / 30,346 ++ / 15,563 --

sborland #2
7 commits / 389 ++ / 49 --

Architecture

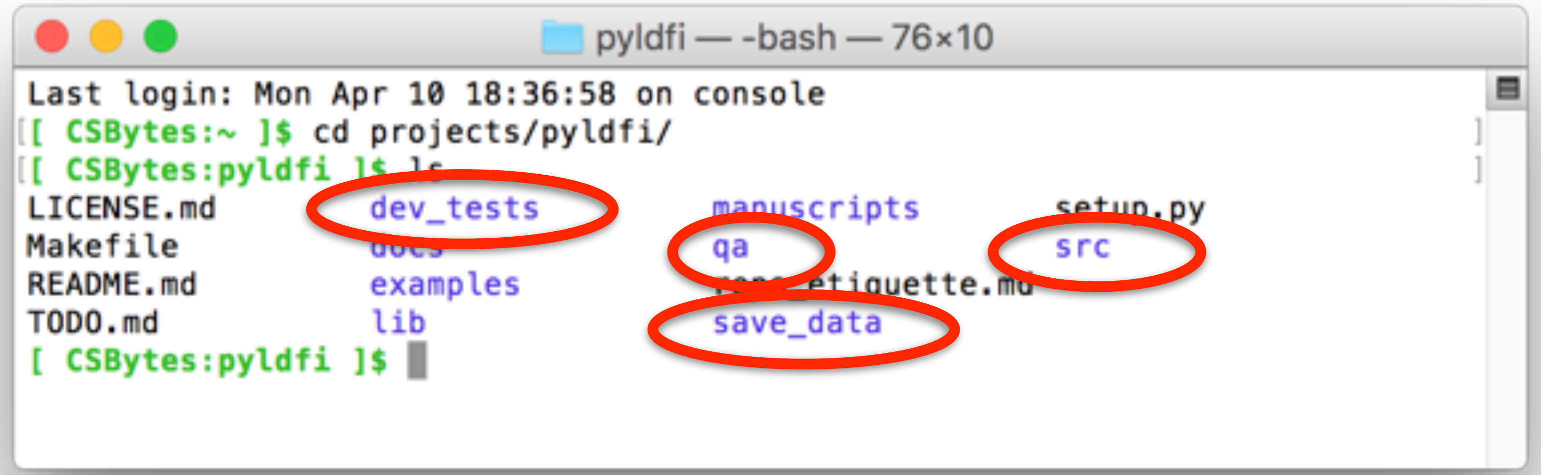


Directory Structure



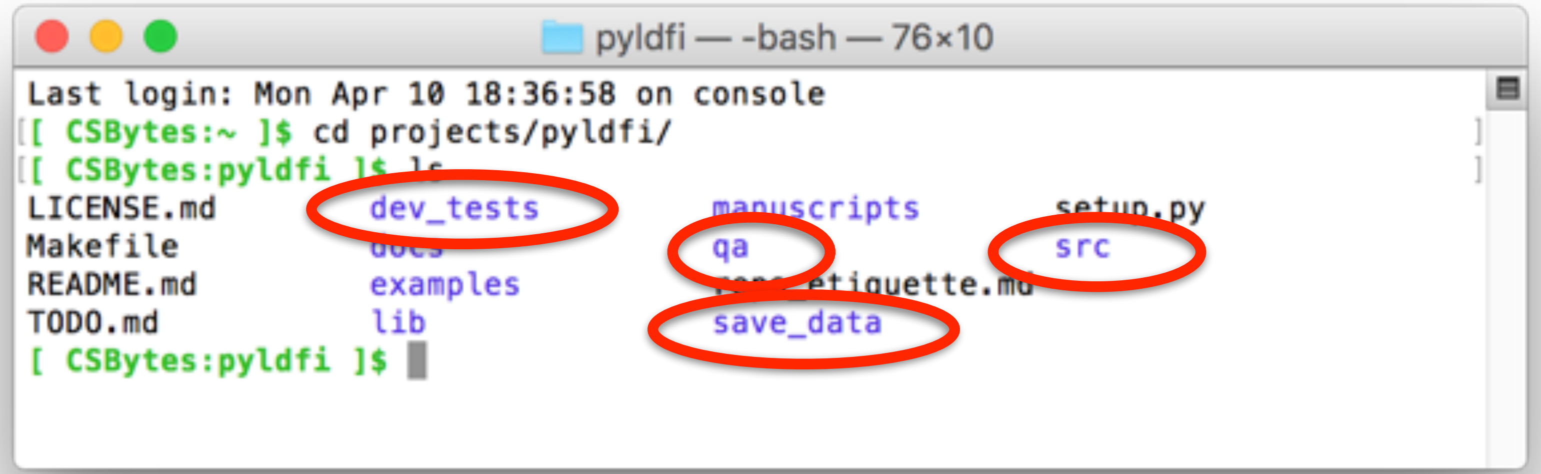
```
Last login: Mon Apr 10 18:36:58 on console
[ CSBytes:~ ]$ cd projects/pyldfi/
[ CSBytes:pyldfi ]$ ls
LICENSE.md          dev_tests           manuscripts         setup.py
Makefile            docs                qa                  src
README.md           examples            repo_etiquette.md
TODO.md             lib                 save_data
[ CSBytes:pyldfi ]$ 
```

Directory Structure


pyldfi — -bash — 76×10

```
Last login: Mon Apr 10 18:36:58 on console
[ CSBytes:~ ]$ cd projects/pyldfi/
[ CSBytes:pyldfi ]$ ls
LICENSE.md        dev_tests         manuscripts       setup.py
Makefile          docs              qa                src
README.md         examples          repo_etiquette.md
TODO.md           lib               save_data
[ CSBytes:pyldfi ]$ 
```

Directory Structure



```
Last login: Mon Apr 10 18:36:58 on console
[ CSBytes:~ ]$ cd projects/pyldfi/
[ CSBytes:pyldfi ]$ ls
LICENSE.md          dev_tests          manuscripts        setup.py
Makefile            docs               qa                 src
README.md           examples           repo_etiquette.md
TODO.md             lib                save_data
[ CSBytes:pyldfi ]$
```

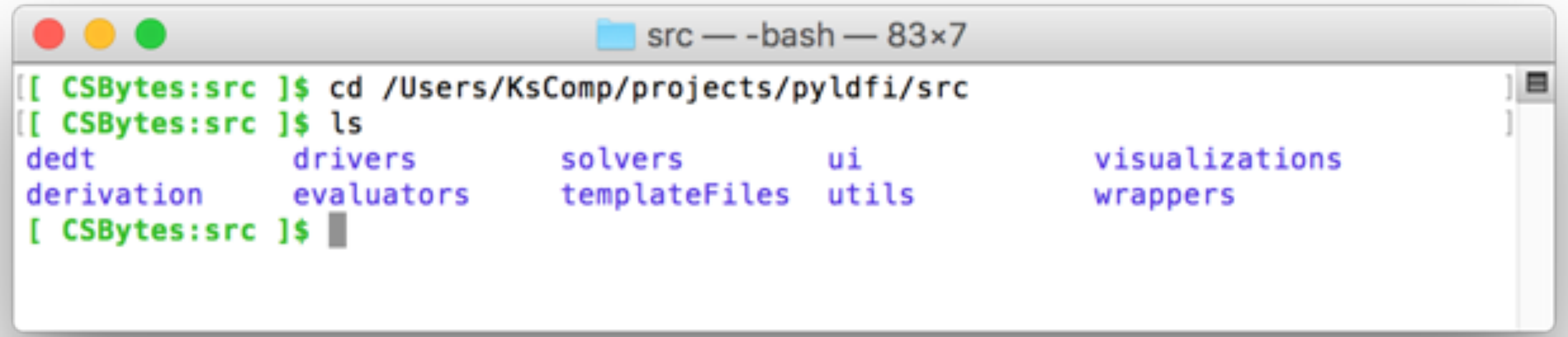
src/ == all source code.

save_data/ == results & graphs.

qa/ == unit tests

dev_tests/ == sketch pad

src/



```
[ CSBytes:src ]$ cd /Users/KsComp/projects/pyldfi/src
[ CSBytes:src ]$ ls
dedt            drivers         solvers         ui              visualizations
derivation      evaluators      templateFiles   utils           wrappers
[ CSBytes:src ]$ 
```

drivers/ == mains

dedt/ == all code* for translating dedalus into datalog

evaluators/ == all code* for evaluating datalog

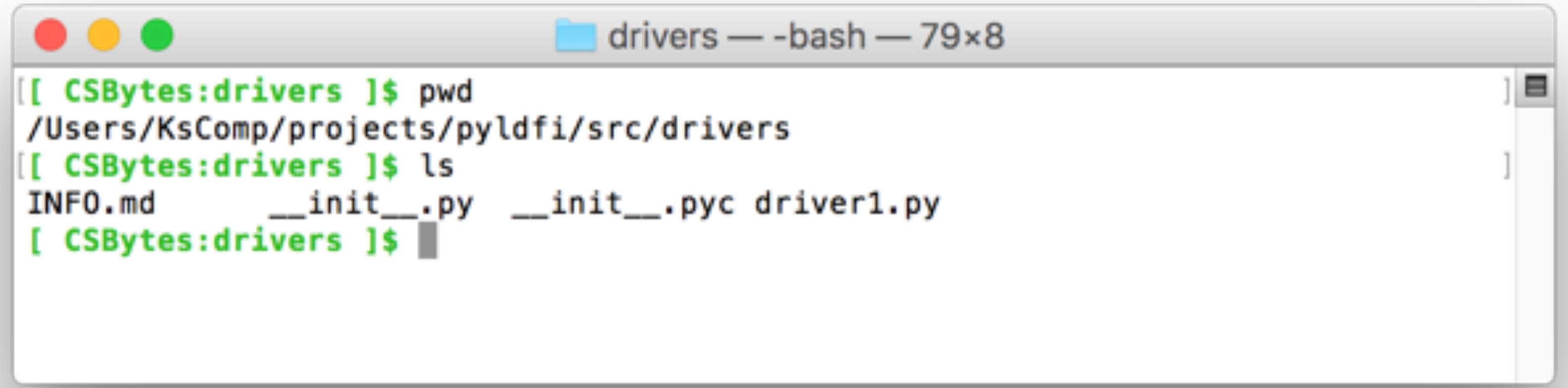
derivation/ == all code* for building provenance trees

utils/ == swiss army knife

templateFiles/ == essential file for installing c4

*kind of

src/drivers/

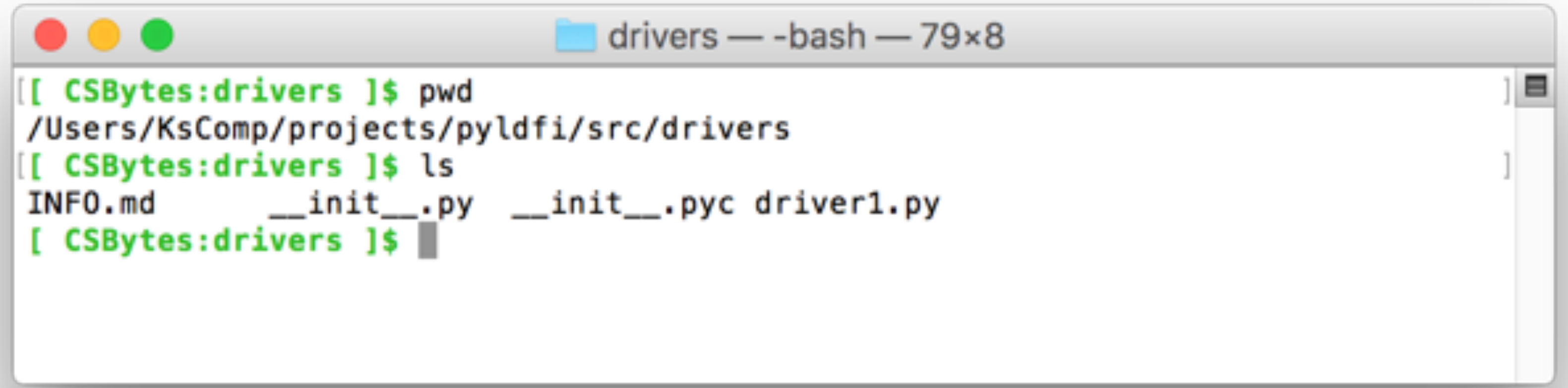


drivers — -bash — 79×8

```
[ CSBytes:drivers ]$ pwd
/Users/KsComp/projects/pyldfi/src/drivers
[ CSBytes:drivers ]$ ls
INFO.md         __init__.py   __init__.pyc driver1.py
[ CSBytes:drivers ]$
```

> manages LDFI Core

src/drivers/

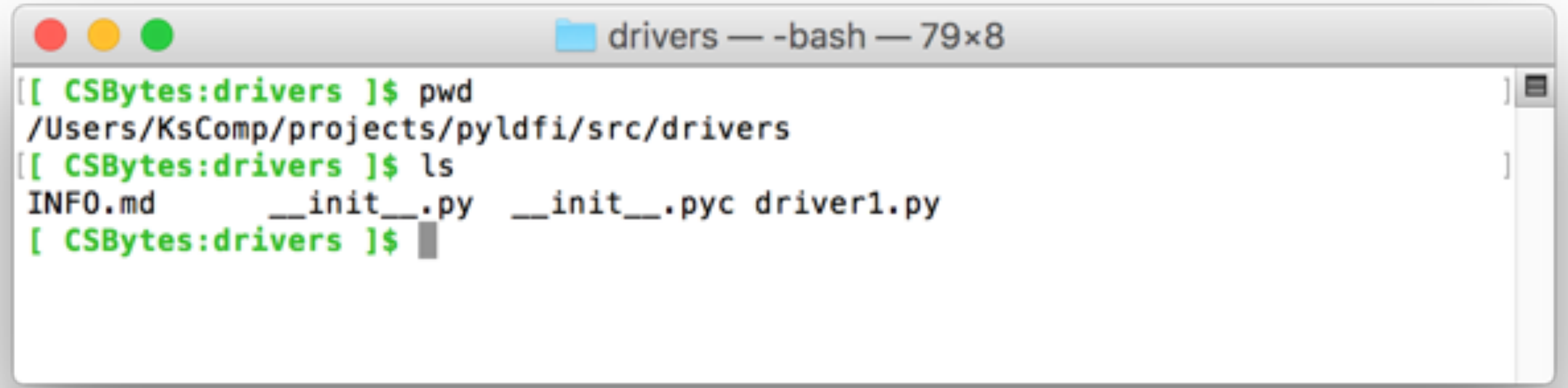

drivers — -bash — 79×8

```
[ CSBytes:drivers ]$ pwd
/Users/KsComp/projects/pyldfi/src/drivers
[ CSBytes:drivers ]$ ls
INFO.md         __init__.py   __init__.pyc driver1.py
[ CSBytes:drivers ]$ 
```

> manages LDFI Core

> checks for bugs

src/drivers/



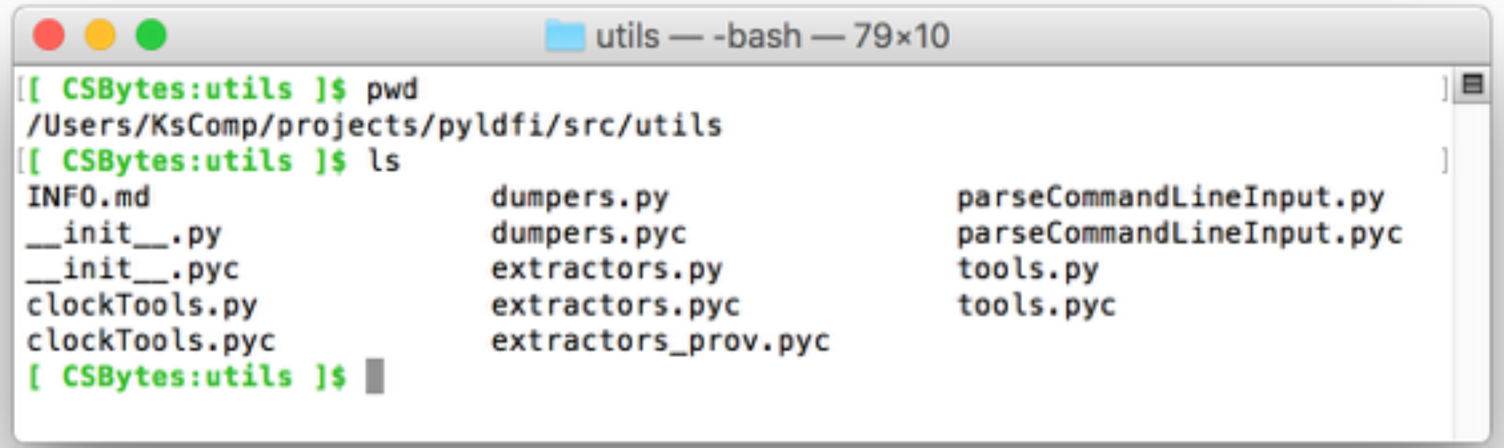
```
[[ CSBytes:drivers ]$ pwd
/Users/KsComp/projects/pyldfi/src/drivers
[[ CSBytes:drivers ]$ ls
INFO.md         __init__.py   __init__.pyc driver1.py
[ CSBytes:drivers ]$
```

> manages LDFI Core

> checks for bugs

> triggers visualizations

src/utils/

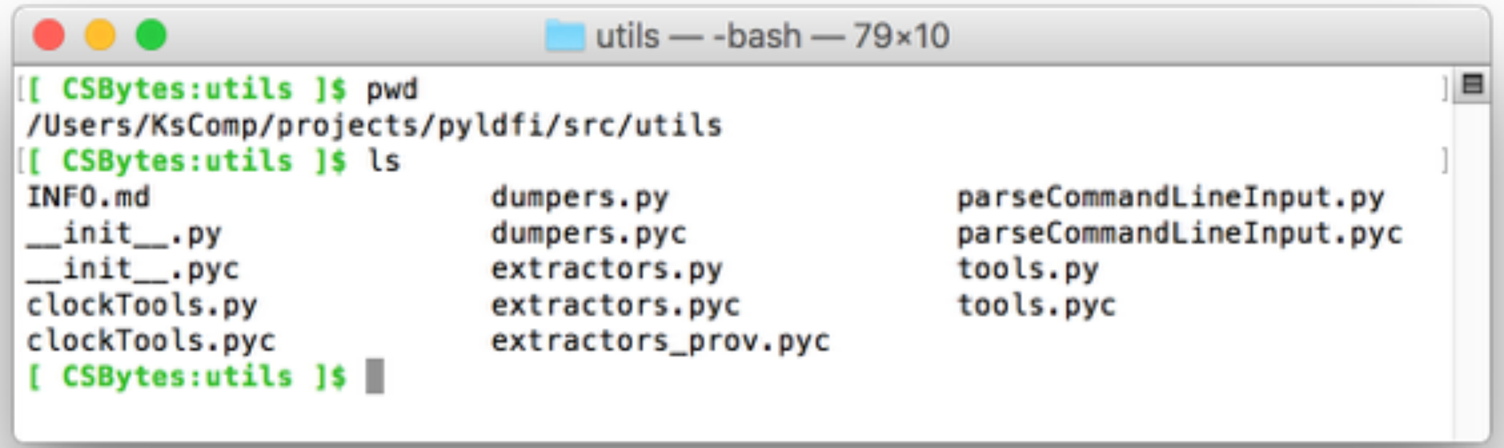


```
[ CSBytes:utils ]$ pwd
/Users/KsComp/projects/pyldfi/src/utils
[ CSBytes:utils ]$ ls
INFO.md                    dumpers.py                 parseCommandLineInput.py
__init__.py                dumpers.pyc                parseCommandLineInput.pyc
__init__.pyc               extractors.py              tools.py
clockTools.py              extractors.pyc             tools.pyc
clockTools.pyc             extractors_prov.pyc
[ CSBytes:utils ]$ 
```

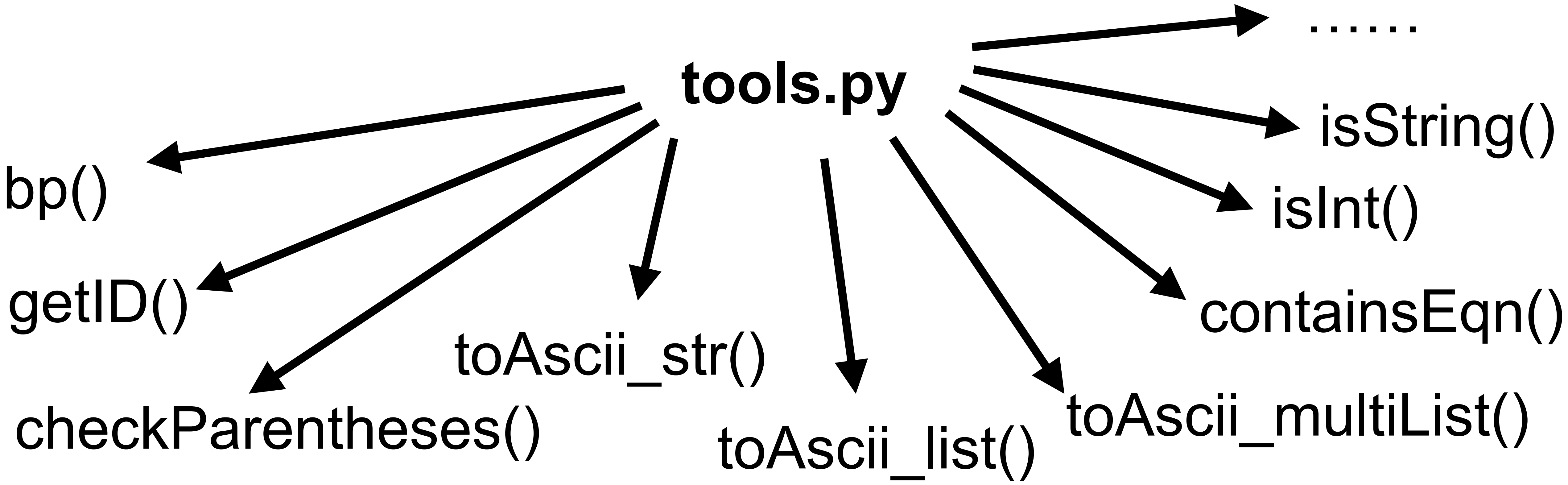
- **Convention*:**
  "tools" files are laundry lists of modules/methods/functions generally useful at different points in the workflow.

*there exist exceptions…

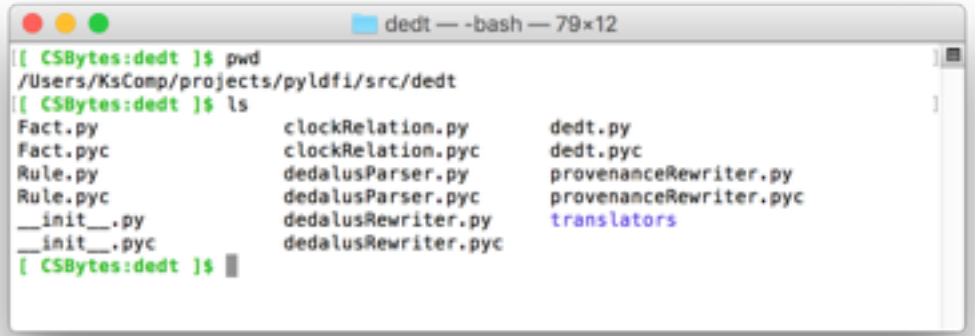
src/utils/



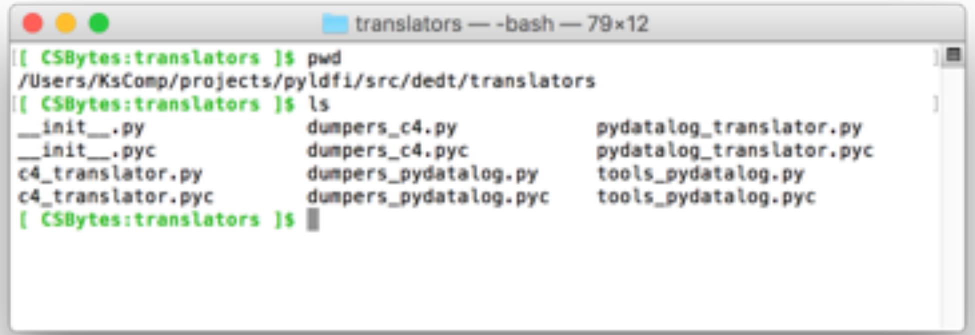
```
[ CSBytes:utils ]$ pwd
/Users/KsComp/projects/pyldfi/src/utils
[ CSBytes:utils ]$ ls
INFO.md                    dumpers.py                 parseCommandLineInput.py
__init__.py                dumpers.pyc                parseCommandLineInput.pyc
__init__.pyc               extractors.py              tools.py
clockTools.py              extractors.pyc             tools.pyc
clockTools.pyc             extractors_prov.pyc
[ CSBytes:utils ]$ 
```



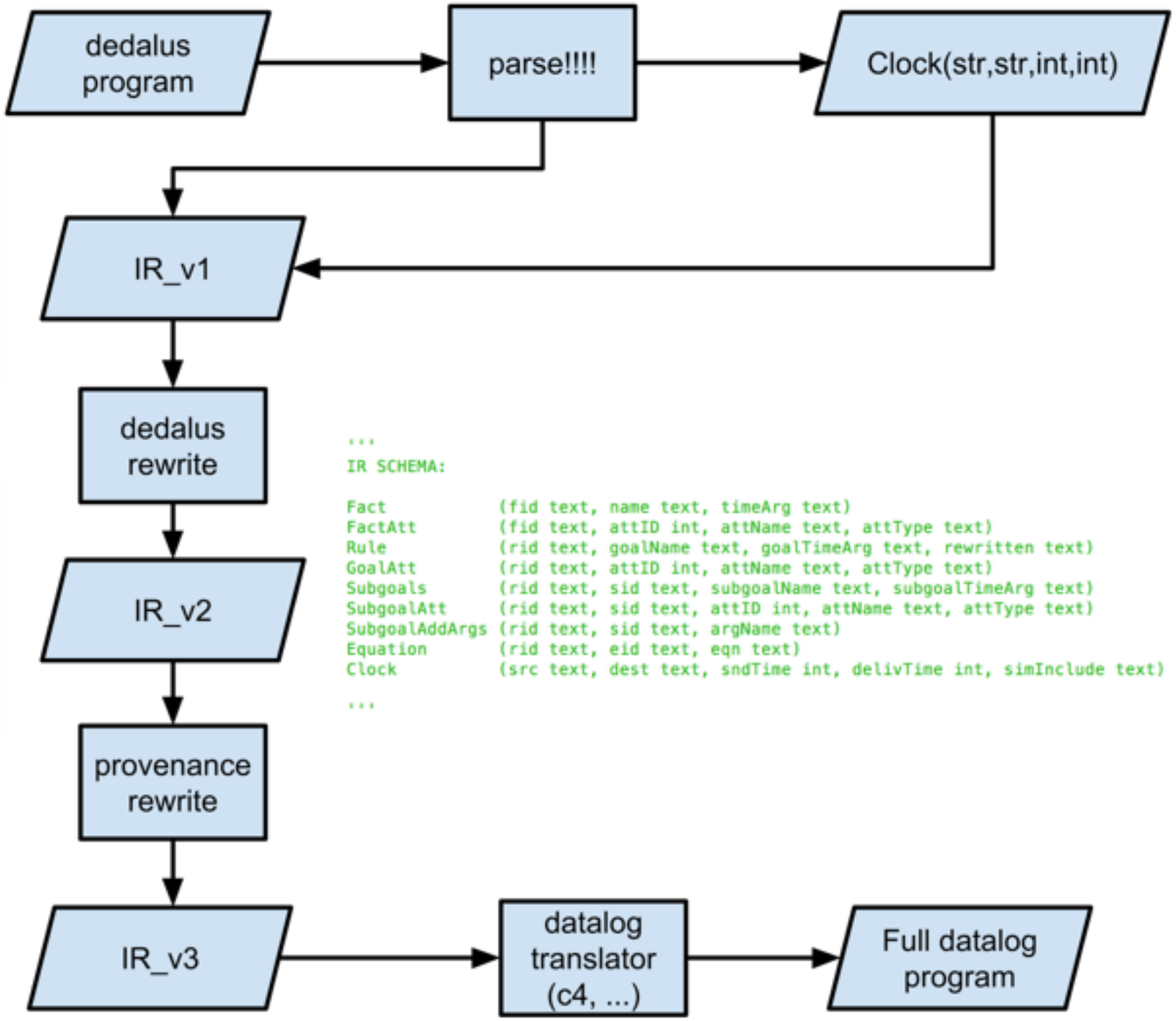
src/dedt/



```
[ CSBytes:dedt ]$ pwd
/Users/KsComp/projects/pyldfi/src/dedt
[ CSBytes:dedt ]$ ls
Fact.py                 clockRelation.py        dedt.py
Fact.pyc                clockRelation.pyc       dedt.pyc
Rule.py                 dedalusParser.py        provenanceRewriter.py
Rule.pyc                dedalusParser.pyc       provenanceRewriter.pyc
__init__.py             dedalusRewriter.py      translators
__init__.pyc            dedalusRewriter.pyc
[ CSBytes:dedt ]$
```

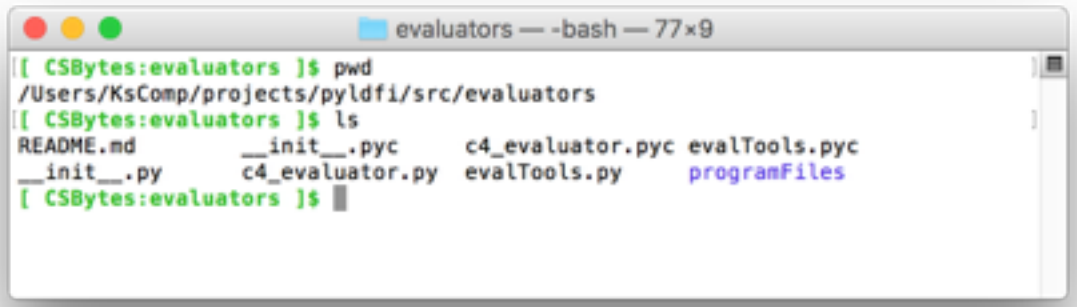


```
[ CSBytes:translators ]$ pwd
/Users/KsComp/projects/pyldfi/src/dedt/translators
[ CSBytes:translators ]$ ls
__init__.py             dumpers_c4.py           pydatalog_translator.py
__init__.pyc            dumpers_c4.pyc          pydatalog_translator.pyc
c4_translator.py        dumpers_pydatalog.py    tools_pydatalog.py
c4_translator.pyc       dumpers_pydatalog.pyc   tools_pydatalog.pyc
[ CSBytes:translators ]$
```



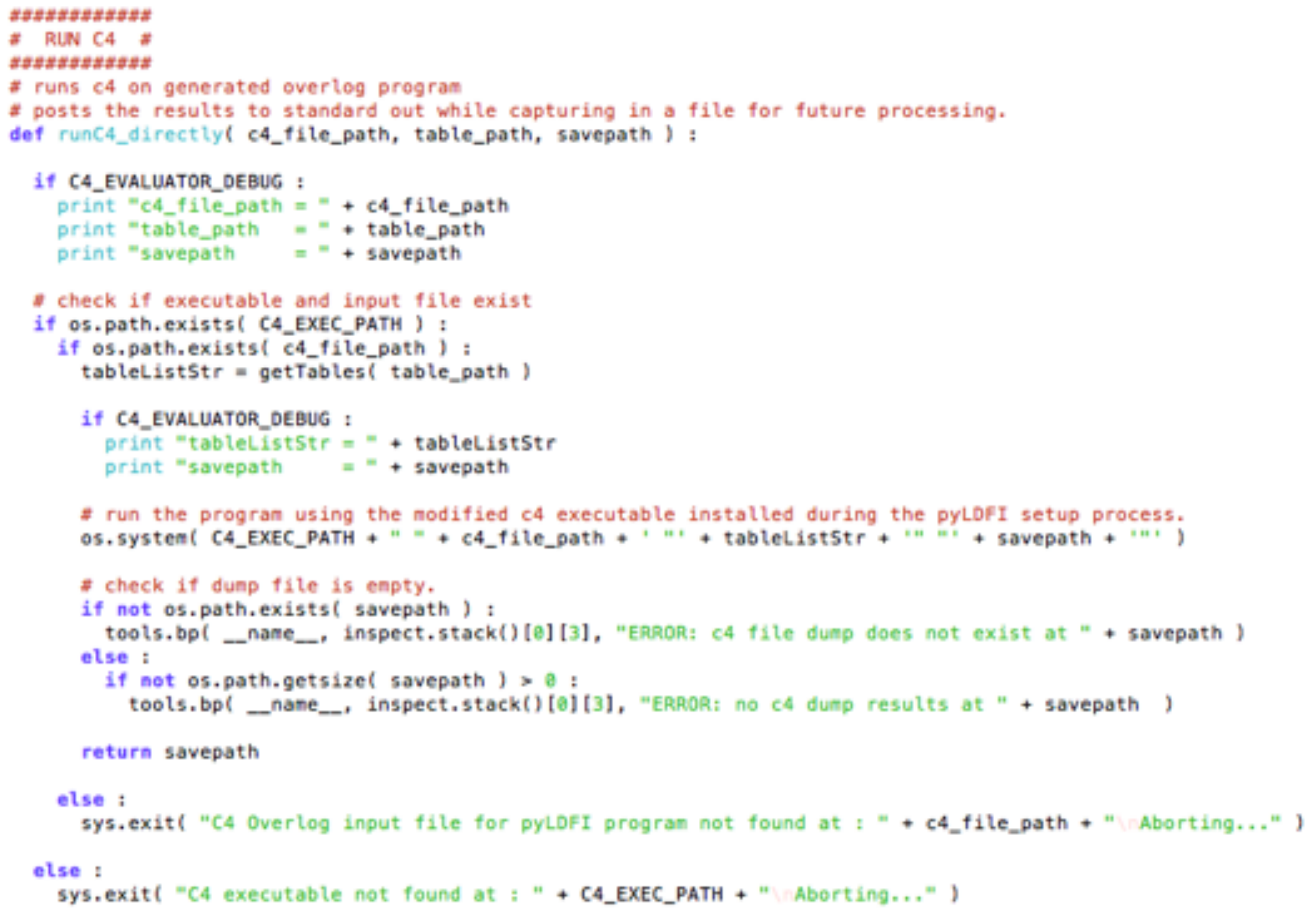
```
...
IR SCHEMA:

Fact           (fid text, name text, timeArg text)
FactAtt        (fid text, attID int, attName text, attType text)
Rule           (rid text, goalName text, goalTimeArg text, rewritten text)
GoalAtt        (rid text, attID int, attName text, attType text)
Subgoals       (rid text, sid text, subgoalName text, subgoalTimeArg text)
SubgoalAtt     (rid text, sid text, attID int, attName text, attType text)
SubgoalAddArgs (rid text, sid text, argName text)
Equation       (rid text, eid text, eqn text)
Clock          (src text, dest text, sndTime int, delivTime int, simInclude text)

...
```

src/evaluators/



evaluators — -bash — 77×9

```
[ CSBytes:evaluators ]$ pwd
/Users/KsComp/projects/pyldfi/src/evaluators
[ CSBytes:evaluators ]$ ls
README.md        __init__.pyc     c4_evaluator.pyc evalTools.pyc
__init__.py      c4_evaluator.py  evalTools.py     programFiles
[ CSBytes:evaluators ]$
```

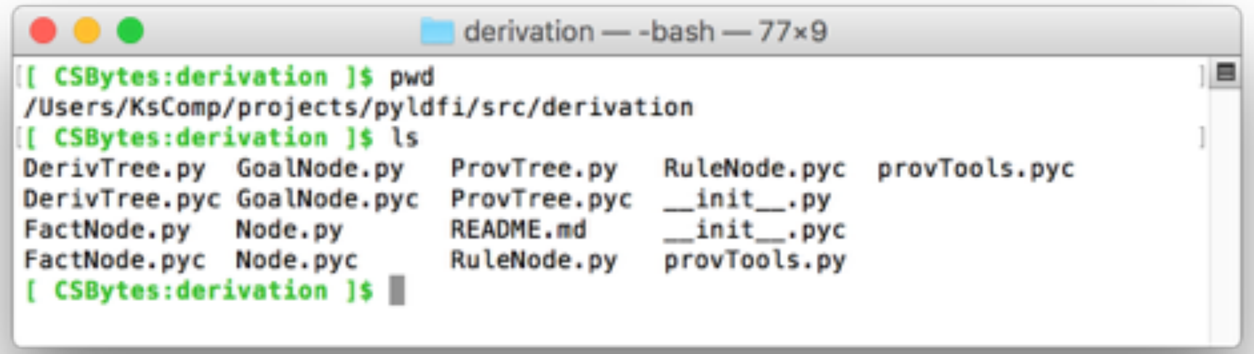


```
############
#  RUN C4  #
############
# runs c4 on generated overlog program
# posts the results to standard out while capturing in a file for future processing.
def runC4_directly( c4_file_path, table_path, savepath ) :

  if C4_EVALUATOR_DEBUG :
    print "c4_file_path = " + c4_file_path
    print "table_path   = " + table_path
    print "savepath     = " + savepath

  # check if executable and input file exist
  if os.path.exists( C4_EXEC_PATH ) :
    if os.path.exists( c4_file_path ) :
      tableListStr = getTables( table_path )

      if C4_EVALUATOR_DEBUG :
        print "tableListStr = " + tableListStr
        print "savepath     = " + savepath

      # run the program using the modified c4 executable installed during the pyLDFI setup process.
      os.system( C4_EXEC_PATH + " " + c4_file_path + ' "' + tableListStr + '" "' + savepath + '"' )

      # check if dump file is empty.
      if not os.path.exists( savepath ) :
        tools.bp( __name__, inspect.stack()[0][3], "ERROR: c4 file dump does not exist at " + savepath )
      else :
        if not os.path.getsize( savepath ) > 0 :
          tools.bp( __name__, inspect.stack()[0][3], "ERROR: no c4 dump results at " + savepath  )

      return savepath

    else :
      sys.exit( "C4 Overlog input file for pyLDFI program not found at : " + c4_file_path + "\nAborting..." )

  else :
    sys.exit( "C4 executable not found at : " + C4_EXEC_PATH + "\nAborting..." )
```

> c4 *programs* save to "pyldfi/src/evaluators/programFiles"

> Evaluation *results* save to "pyldfi/save_data/c4Output/c4dump.txt"

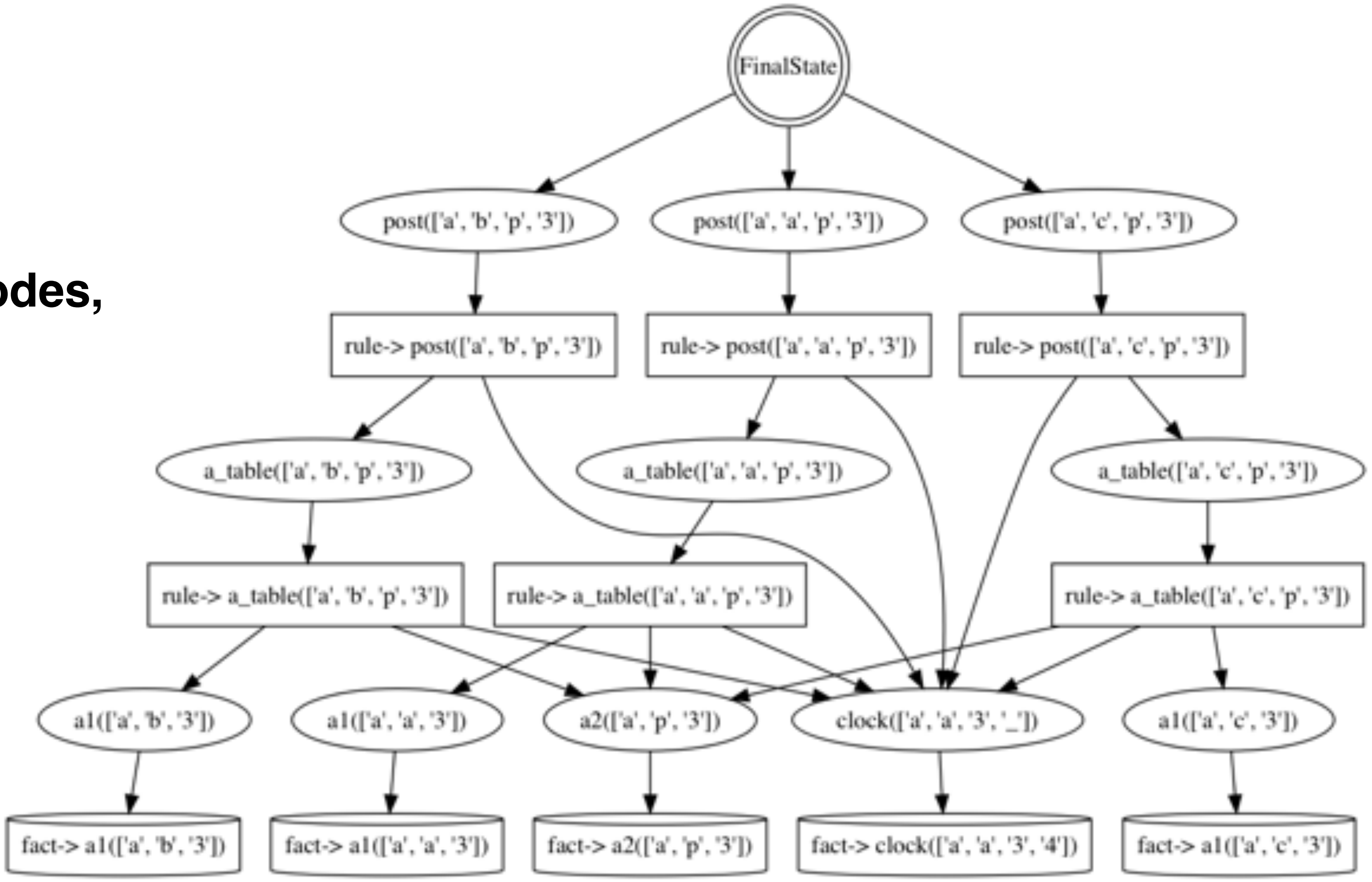
src/derivation/



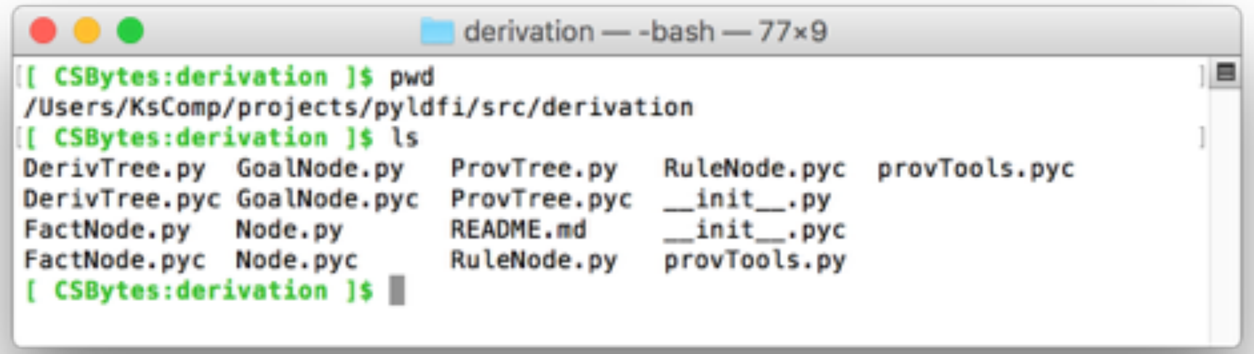
```
[ CSBytes:derivation ]$ pwd
/Users/KsComp/projects/pyldfi/src/derivation
[ CSBytes:derivation ]$ ls
DerivTree.py  GoalNode.py  ProvTree.py  RuleNode.pyc  provTools.pyc
DerivTree.pyc GoalNode.pyc ProvTree.pyc __init__.py
FactNode.py   Node.py      README.md    __init__.pyc
FactNode.pyc  Node.pyc     RuleNode.py  provTools.py
[ CSBytes:derivation ]$
```

> ProvTree is a list of DerivTrees.

> DerivTrees are rooted at GoalNodes,
  RuleNodes, or FactNodes.


FinalState
post(['a', 'b', 'p', '3'])
post(['a', 'a', 'p', '3'])
post(['a', 'c', 'p', '3'])
rule-> post(['a', 'b', 'p', '3'])
rule-> post(['a', 'a', 'p', '3'])
rule-> post(['a', 'c', 'p', '3'])
a_table(['a', 'b', 'p', '3'])
a_table(['a', 'a', 'p', '3'])
a_table(['a', 'c', 'p', '3'])
rule-> a_table(['a', 'b', 'p', '3'])
rule-> a_table(['a', 'a', 'p', '3'])
rule-> a_table(['a', 'c', 'p', '3'])
a1(['a', 'b', '3'])
a1(['a', 'a', '3'])
a2(['a', 'p', '3'])
clock(['a', 'a', '3', '_'])
a1(['a', 'c', '3'])
fact-> a1(['a', 'b', '3'])
fact-> a1(['a', 'a', '3'])
fact-> a2(['a', 'p', '3'])
fact-> clock(['a', 'a', '3', '4'])
fact-> a1(['a', 'c', '3'])

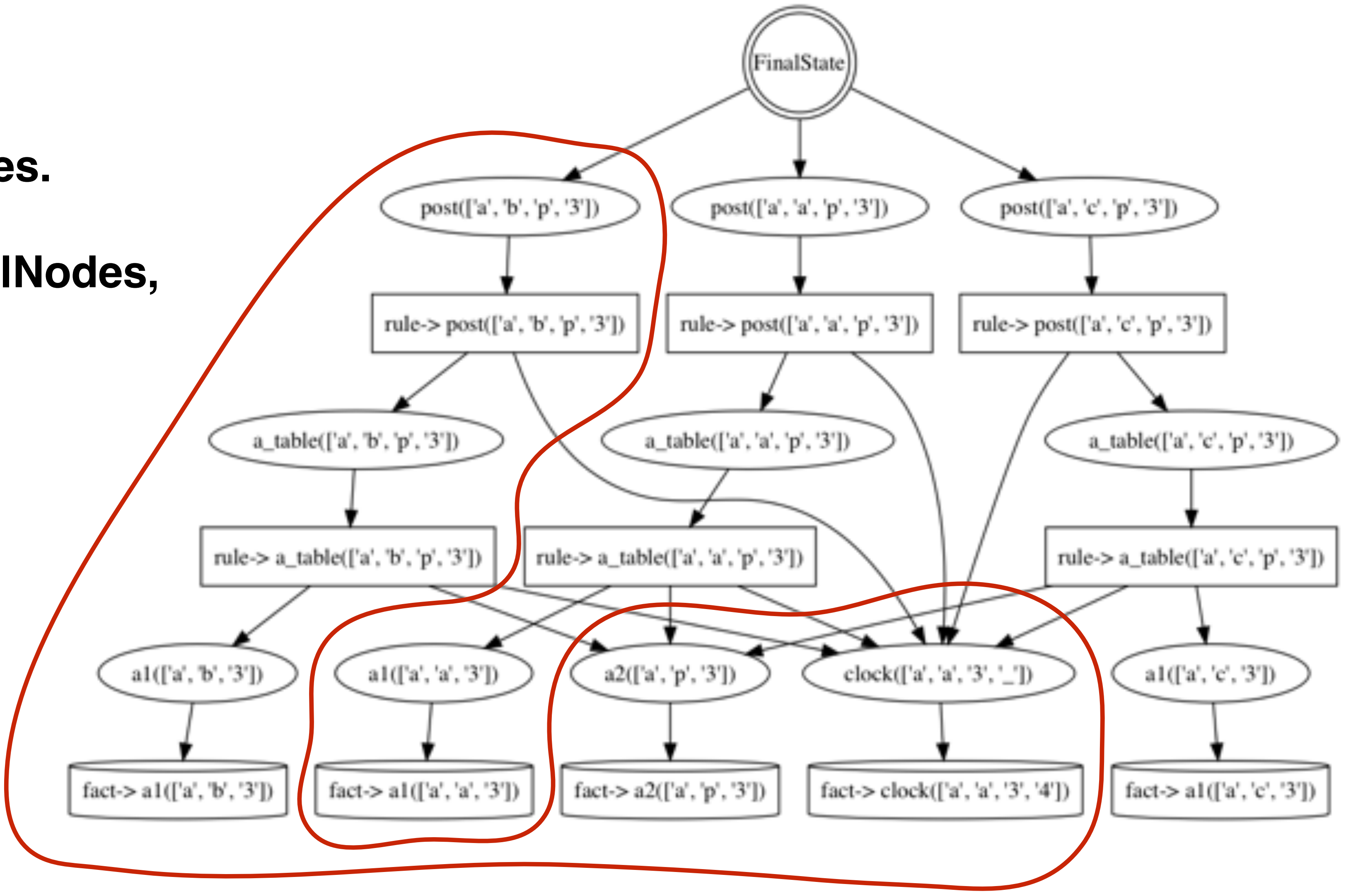
src/derivation/



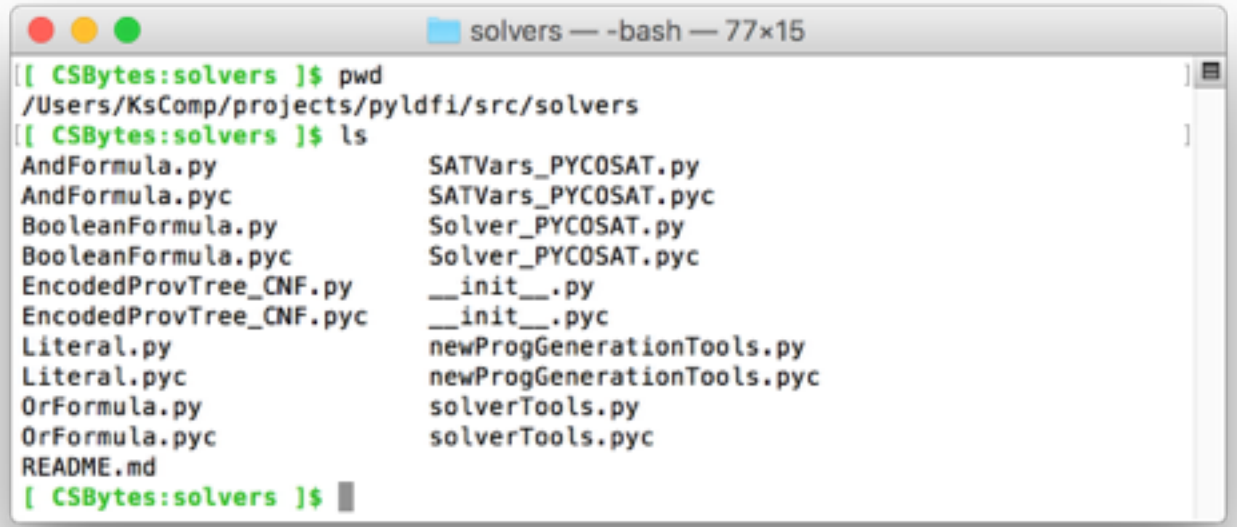
```
derivation — -bash — 77×9
[ CSBytes:derivation ]$ pwd
/Users/KsComp/projects/pyldfi/src/derivation
[ CSBytes:derivation ]$ ls
DerivTree.py  GoalNode.py   ProvTree.py   RuleNode.pyc  provTools.pyc
DerivTree.pyc GoalNode.pyc  ProvTree.pyc  __init__.py
FactNode.py   Node.py       README.md     __init__.pyc
FactNode.pyc  Node.pyc      RuleNode.py   provTools.py
[ CSBytes:derivation ]$
```

> ProvTree is a list of DerivTrees.

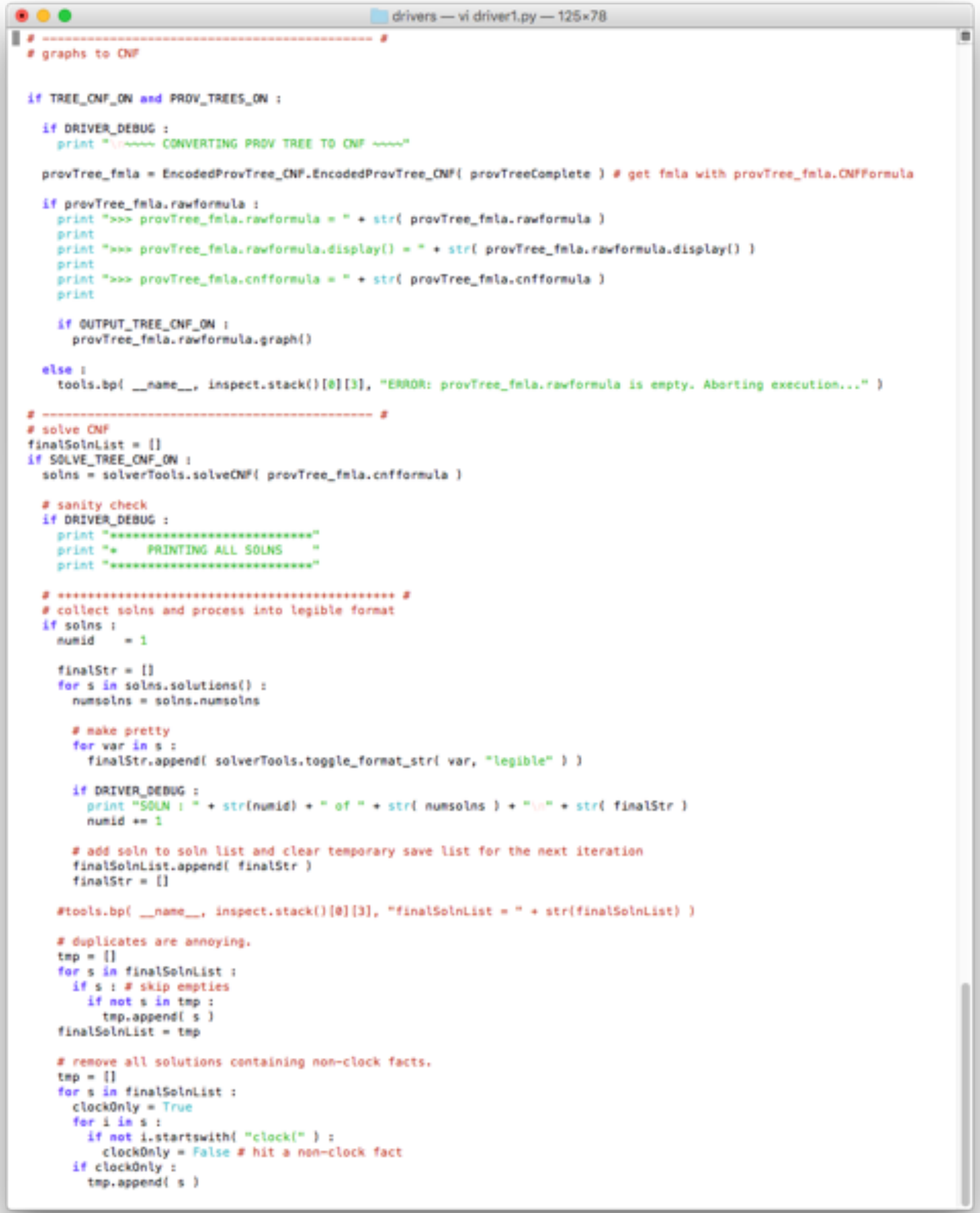
> DerivTrees are rooted at GoalNodes, RuleNodes, or FactNodes.


FinalState
post(['a', 'b', 'p', '3'])
post(['a', 'a', 'p', '3'])
post(['a', 'c', 'p', '3'])
rule-> post(['a', 'b', 'p', '3'])
rule-> post(['a', 'a', 'p', '3'])
rule-> post(['a', 'c', 'p', '3'])
a_table(['a', 'b', 'p', '3'])
a_table(['a', 'a', 'p', '3'])
a_table(['a', 'c', 'p', '3'])
rule-> a_table(['a', 'b', 'p', '3'])
rule-> a_table(['a', 'a', 'p', '3'])
rule-> a_table(['a', 'c', 'p', '3'])
a1(['a', 'b', '3'])
a1(['a', 'a', '3'])
a2(['a', 'p', '3'])
clock(['a', 'a', '3', '_'])
a1(['a', 'c', '3'])
fact-> a1(['a', 'b', '3'])
fact-> a1(['a', 'a', '3'])
fact-> a2(['a', 'p', '3'])
fact-> clock(['a', 'a', '3', '4'])
fact-> a1(['a', 'c', '3'])

src/solvers/



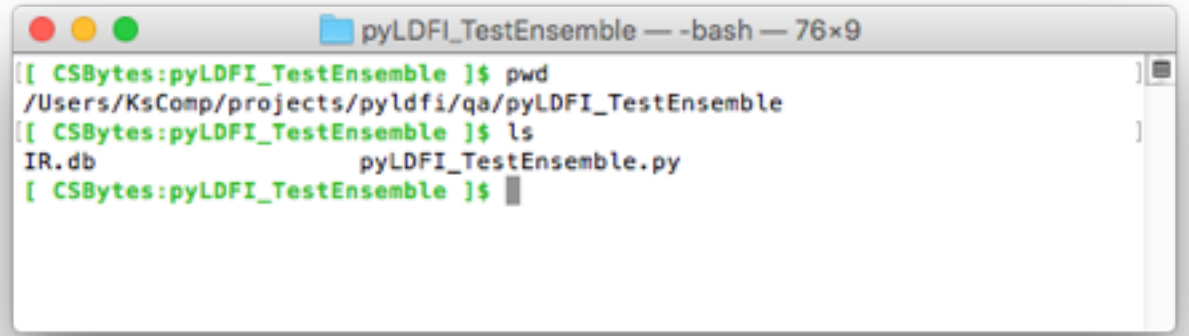
```
[ CSBytes:solvers ]$ pwd
/Users/KsComp/projects/pyldfi/src/solvers
[ CSBytes:solvers ]$ ls
AndFormula.py              SATVars_PYCOSAT.py
AndFormula.pyc             SATVars_PYCOSAT.pyc
BooleanFormula.py          Solver_PYCOSAT.py
BooleanFormula.pyc         Solver_PYCOSAT.pyc
EncodedProvTree_CNF.py     __init__.py
EncodedProvTree_CNF.pyc    __init__.pyc
Literal.py                 newProgGenerationTools.py
Literal.pyc                newProgGenerationTools.pyc
OrFormula.py               solverTools.py
OrFormula.pyc              solverTools.pyc
README.md
[ CSBytes:solvers ]$ 
```



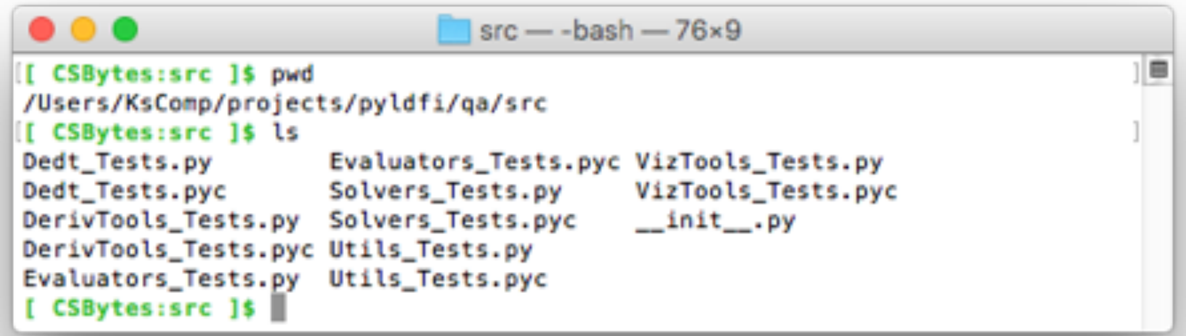
```
# ---------------------------------------------- #
# graphs to CNF

if TREE_CNF_ON and PROV_TREES_ON :

  if DRIVER_DEBUG :
    print "\n~~~~ CONVERTING PROV TREE TO CNF ~~~~"

  provTree_fmla = EncodedProvTree_CNF.EncodedProvTree_CNF( provTreeComplete ) # get fmla with provTree_fmla.CNFFormula

  if provTree_fmla.rawformula :
    print ">>> provTree_fmla.rawformula = " + str( provTree_fmla.rawformula )
    print
    print ">>> provTree_fmla.rawformula.display() = " + str( provTree_fmla.rawformula.display() )
    print
    print ">>> provTree_fmla.cnfformula = " + str( provTree_fmla.cnfformula )
    print

    if OUTPUT_TREE_CNF_ON :
      provTree_fmla.rawformula.graph()

  else :
    tools.bp( __name__, inspect.stack()[0][3], "ERROR: provTree_fmla.rawformula is empty. Aborting execution..." )

# ---------------------------------------------- #
# solve CNF
finalSolnList = []
if SOLVE_TREE_CNF_ON :
  solns = solverTools.solveCNF( provTree_fmla.cnfformula )

  # sanity check
  if DRIVER_DEBUG :
    print "****************************"
    print "*    PRINTING ALL SOLNS    "
    print "****************************"

  # ********************************************** #
  # collect solns and process into legible format
  if solns :
    numid    = 1

    finalStr = []
    for s in solns.solutions() :
      numsolns = solns.numsolns

      # make pretty
      for var in s :
        finalStr.append( solverTools.toggle_format_str( var, "legible" ) )

      if DRIVER_DEBUG :
        print "SOLN : " + str(numid) + " of " + str( numsolns ) + "\n" + str( finalStr )
        numid += 1

      # add soln to soln list and clear temporary save list for the next iteration
      finalSolnList.append( finalStr )
      finalStr = []

    #tools.bp( __name__, inspect.stack()[0][3], "finalSolnList = " + str(finalSolnList) )

    # duplicates are annoying.
    tmp = []
    for s in finalSolnList :
      if s : # skip empties
        if not s in tmp :
          tmp.append( s )
    finalSolnList = tmp

    # remove all solutions containing non-clock facts.
    tmp = []
    for s in finalSolnList :
      clockOnly = True
      for i in s :
        if not i.startswith( "clock(" ) :
          clockOnly = False # hit a non-clock fact
      if clockOnly :
        tmp.append( s )
```

*pyILP is not portable between Python 2.6 and 2.7

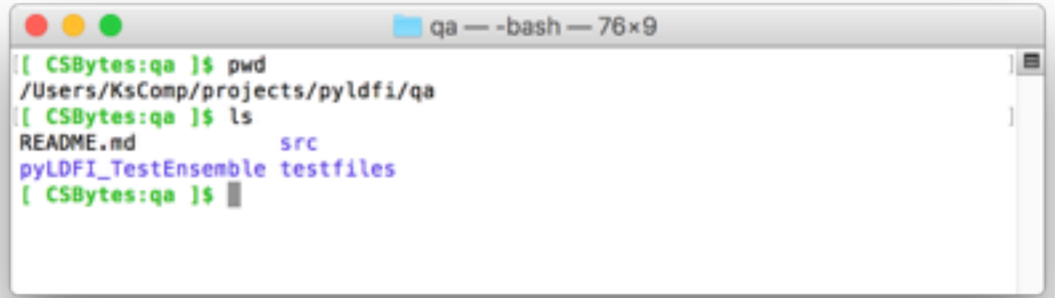
qa/



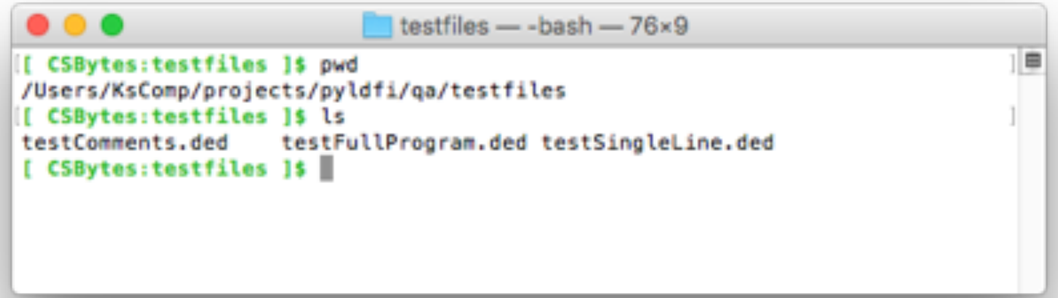
```
[ CSBytes:pyLDFI_TestEnsemble ]$ pwd
/Users/KsComp/projects/pyldfi/qa/pyLDFI_TestEnsemble
[ CSBytes:pyLDFI_TestEnsemble ]$ ls
IR.db                          pyLDFI_TestEnsemble.py
[ CSBytes:pyLDFI_TestEnsemble ]$ 
```



```
[ CSBytes:src ]$ pwd
/Users/KsComp/projects/pyldfi/qa/src
[ CSBytes:src ]$ ls
Dedt_Tests.py           Evaluators_Tests.pyc VizTools_Tests.py
Dedt_Tests.pyc          Solvers_Tests.py     VizTools_Tests.pyc
DerivTools_Tests.py     Solvers_Tests.pyc    __init__.py
DerivTools_Tests.pyc    Utils_Tests.py
Evaluators_Tests.py     Utils_Tests.pyc
[ CSBytes:src ]$ 
```

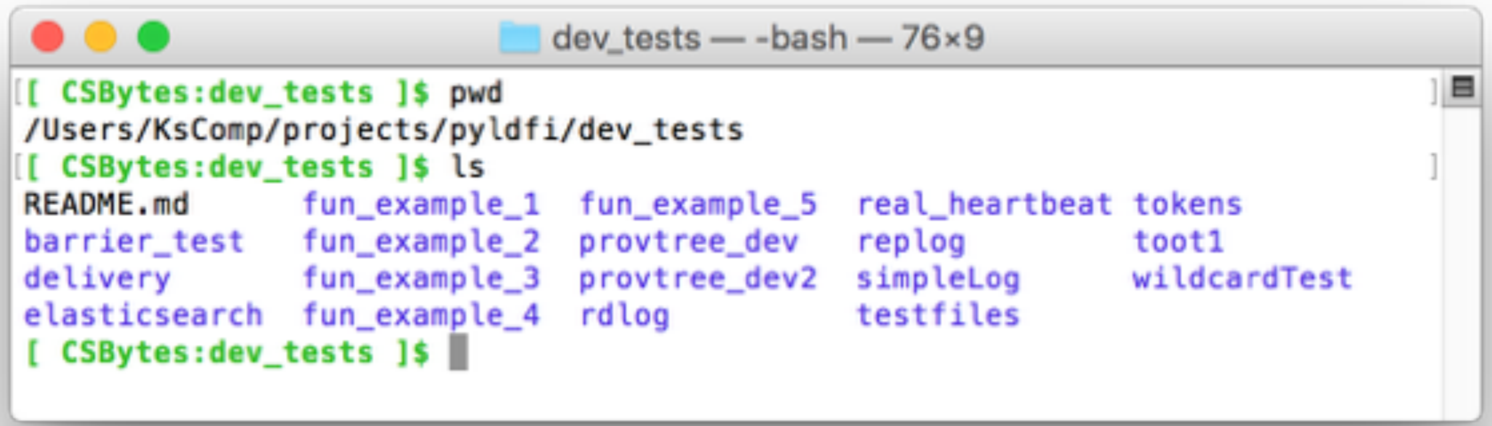


```
[ CSBytes:qa ]$ pwd
/Users/KsComp/projects/pyldfi/qa
[ CSBytes:qa ]$ ls
README.md           src
pyLDFI_TestEnsemble testfiles
[ CSBytes:qa ]$ 
```



```
[ CSBytes:testfiles ]$ pwd
/Users/KsComp/projects/pyldfi/qa/testfiles
[ CSBytes:testfiles ]$ ls
testComments.ded    testFullProgram.ded testSingleLine.ded
[ CSBytes:testfiles ]$ 
```

dev_tests/



```
[ CSBytes:dev_tests ]$ pwd
/Users/KsComp/projects/pyldfi/dev_tests
[ CSBytes:dev_tests ]$ ls
README.md       fun_example_1  fun_example_5  real_heartbeat tokens
barrier_test    fun_example_2  provtree_dev   replog         toot1
delivery        fun_example_3  provtree_dev2  simpleLog      wildcardTest
elasticsearch   fun_example_4  rdlog          testfiles
[ CSBytes:dev_tests ]$ 
```

Live Demo(s)

Reporting Bugs

Option 1:

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

Option 2:

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

Links

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