École Polytechnique - Bachelor Thesis

Towards verifying data science software

Gurupremrana Shabadi

Advised by **Caterina Urban**, INRIA Paris and worked with **Luca Negrini** and **Pietro Ferrara** (Università Ca' Foscari Venezia)

Motivation

- **Data science:** Analysing and transforming raw and potentially dirty *tables of data*
- **Tedious process:** Written as one-off code; rarely tested
- Errors in processing data propagate into unsound analyses and inaccurate ML models

```
In [ ]: import pandas as pd

In [ ]: from fugue_notebook import setup
        setup()

In [ ]: df = pd.DataFrame({'a':[1,2,3,4], 'b':[1,2,3,4]})
        df.to_csv('df.csv', index=False)

In [ ]: %%fsql
        -- This SQL cell sees the dataframe defined in the previous cell
        SELECT *
          FROM df
         WHERE a > 2
         PRINT

In [ ]: %%fsql
        df2 = LOAD "/Users/kevinkho/Work/fugue/df.csv" (header=TRUE, infer_schema=TRUE)
        SELECT *
          FROM df2
         WHERE b < 2
         PRINT
          SAVE OVERWRITE "/Users/kevinkho/Work/fugue/df.csv"

In [ ]:
```


Jupyter Notebook

Goal

Statically analyse Jupyter Notebooks

(focussing on the pandas library)

Problem

Are there any **erroneous** data transforms?

```
df = pd.read('covid.csv')

df.drop('spo2')

⋮

print(df['spo2'].mean())
```

Problem

Is there any **unused** data?
Was any **sensitive** data used?

```
# risk 'D' unused
df['risk'].map(
  {'A': 5, 'B': 4, 'C': 3})
⋮
# sensitive data used
train(df['gender', 'risk'])
```

Problem

Do any transforms introduce **bias/skew**?

```
df.filter(‘age’ > 40)

⋮

# possible dependence
between ‘age’ and ‘risk’

train(df[‘age’, ‘risk’])
```

Challenges

- These problems do not always throw errors
- Hard to debug manually
- Notebook semantics add a level of complexity
- Structure of the data required to reason about data usage, bias/skew

Infer structure and shape of data from the code!

Coder places assumptions on shape of data

```
df1 = pd.read(‘covid1.csv’)
df1.filter(‘spo2’ < 95)
df1[‘risk’].map(
  {‘A’: 5, ‘B’: 4, ‘C’: 3})

df2 = pd.read(‘covid2.csv’)
df2.select_rows(500, 1000)

df3 = pd.concat([df1, df2])
```

Input data shape inference

- Constraints are placed by each dataframe transformation
- We use abstract interpretation to gather all constraints placed by the code
- Column and row info, types of data in columns, bounds on values…

Dataframe graph abstract domain

Helps track all the transformations applied to dataframes

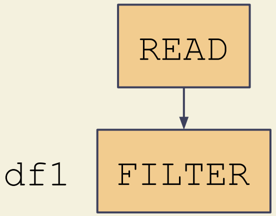
Graph construction


```
df1 = pd.read(‘covid1.csv’)
df1.filter(‘spo2’ < 95)
df1[‘risk’].map(
   {‘A’: 5, ‘B’: 4, ‘C’: 3})


df2 = pd.read(‘covid2.csv’)
df2.select_rows(500, 1000)


df3 = pd.concat([df1, df2])
```

Graph construction

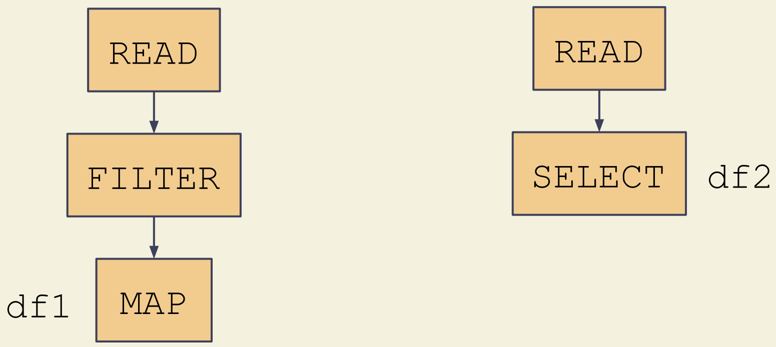



```
df1 = pd.read('covid1.csv')
df1.filter('spo2' < 95)
df1['risk'].map(
   {'A': 5, 'B': 4, 'C': 3})

df2 = pd.read('covid2.csv')
df2.select_rows(500, 1000)

df3 = pd.concat([df1, df2])
```

Graph construction

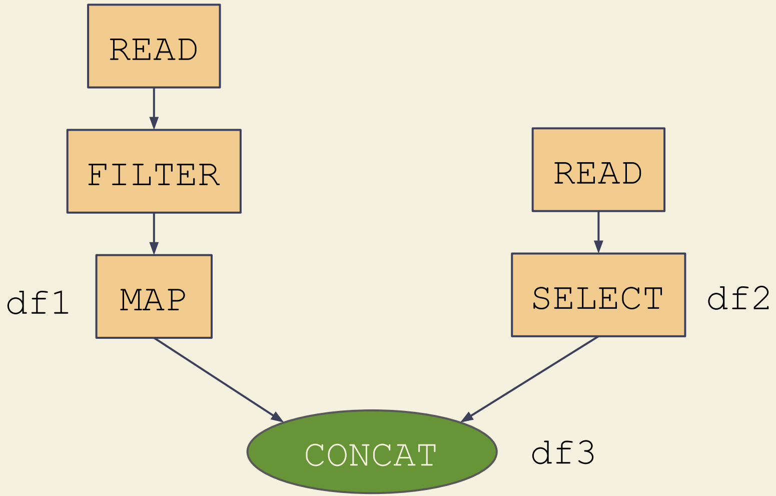



```
df1 = pd.read('covid1.csv')
df1.filter('spo2' < 95)
df1['risk'].map(
   {'A': 5, 'B': 4, 'C': 3})

df2 = pd.read('covid2.csv')
df2.select_rows(500, 1000)

df3 = pd.concat([df1, df2])
```

Graph construction

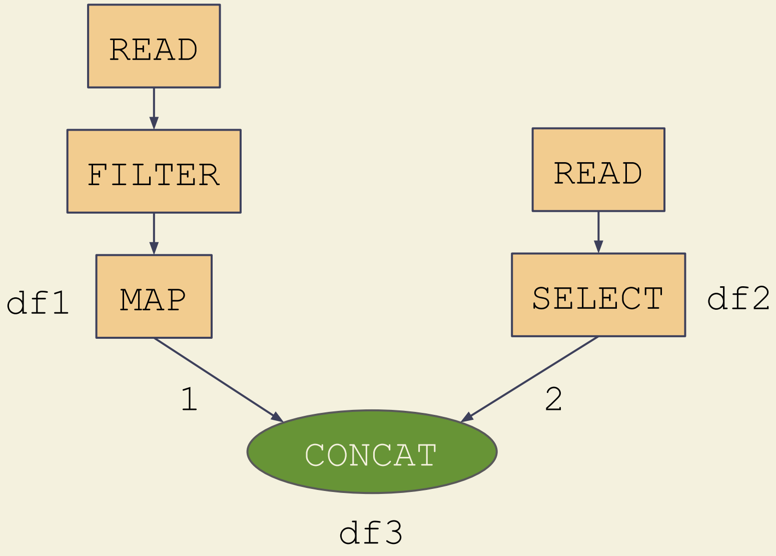



```
df1 = pd.read('covid1.csv')
df1.filter('spo2' < 95)
df1['risk'].map(
  {'A': 5, 'B': 4, 'C': 3})


df2 = pd.read('covid2.csv')
df2.select_rows(500, 1000)


df3 = pd.concat([df1, df2])
```

Graph construction



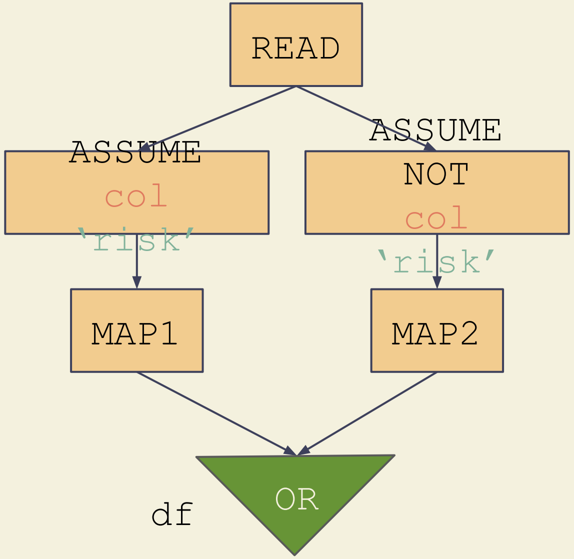

```
df1 = pd.read('covid1.csv')
df1.filter('spo2' < 95)
df1['risk'].map(
   {'A': 5, 'B': 4, 'C': 3})


df2 = pd.read('covid2.csv')
df2.select_rows(500, 1000)


df3 = pd.concat([df1, df2])
```

Conditionals

(least upper bound)



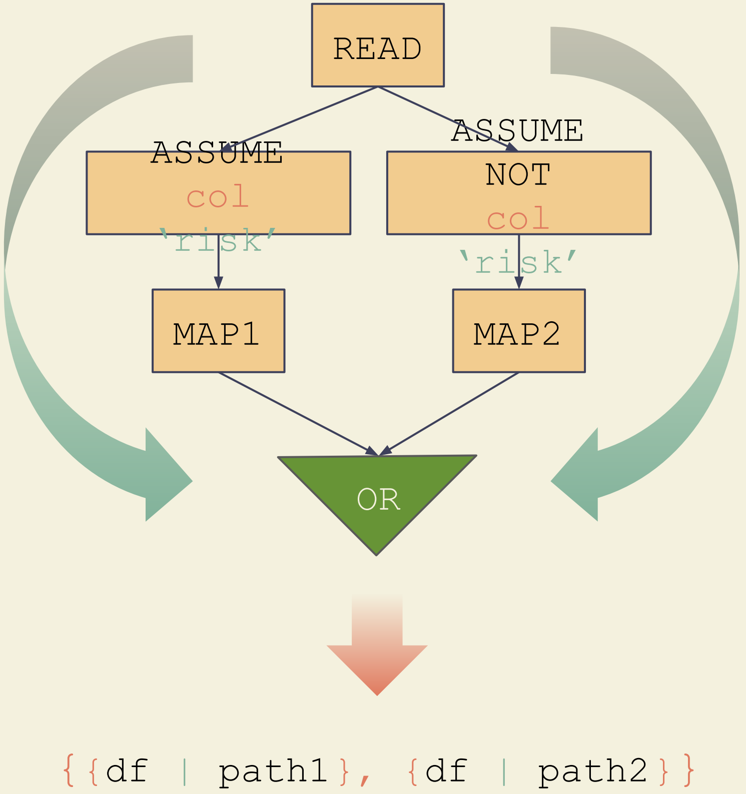

```
df = pd.read('covid1.csv')
if 'risk' in df:
   df['risk'].map(map1)
else:
   df['risk'].map(map2)
```

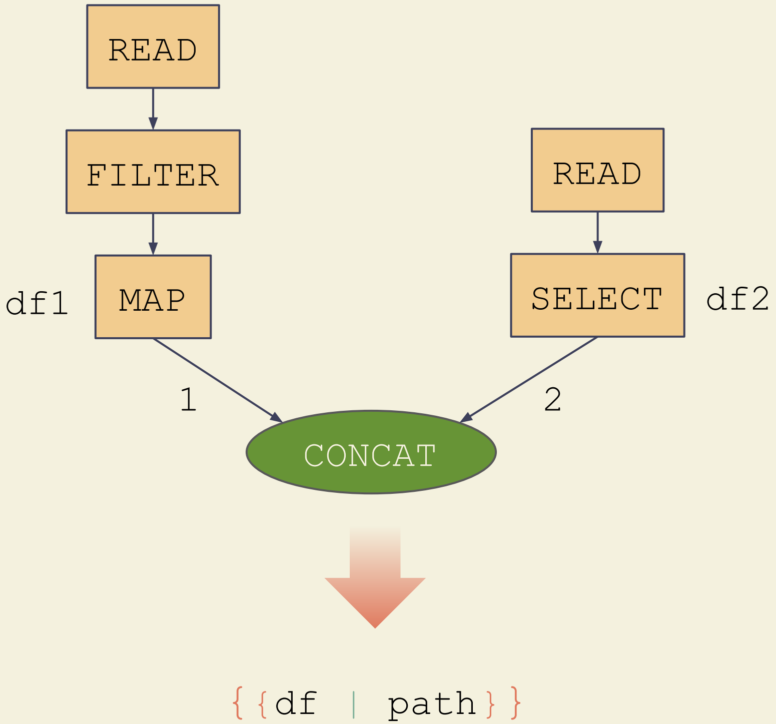
Dataframe graphs

Graph node = **Constraint**

Graph = **Set of set of Dataframes**

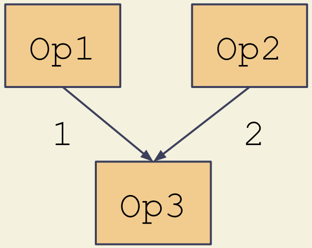
One set corresponding to each *OR-path* of the graph



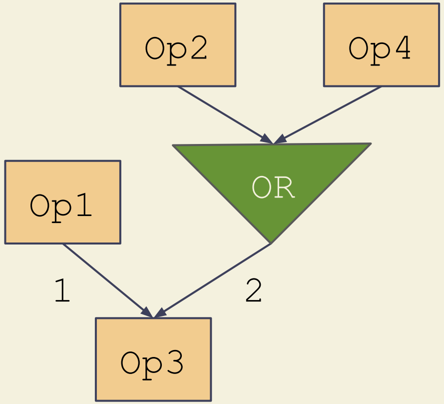


Without OR-nodes, we only have one set of dataframes

Partial order on dataframe graphs



$\sqsubseteq$



{{df | path1}} $\subseteq$ {{df | path1}, {df | path2}}

The set of dataframe graphs forms a lattice!

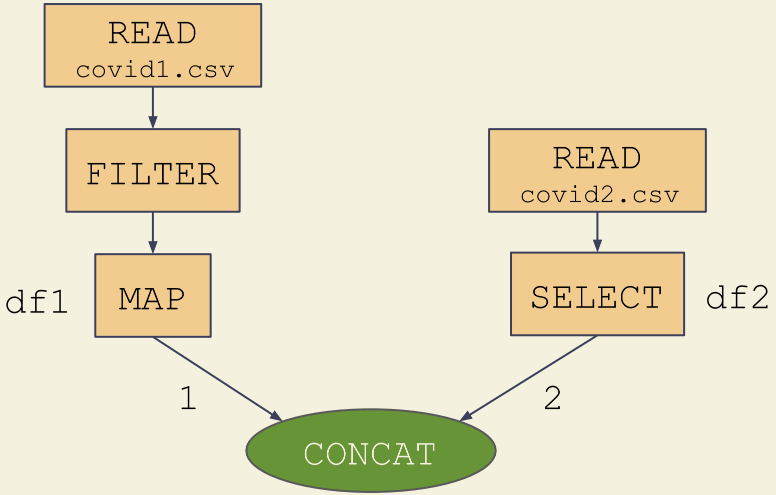
Dealing with loops using widening

- Loops would add unbounded nodes to graphs
- We use **naive widening** to avoid this, i.e., jump to $\top$
- Alternative: introduce **cycles!**

```
# luckily this is uncommon
while bool_cond:
  Op1(df)
```

Using dataframe graphs

1. Restrict to condensed set of dataframe transformations
2. Understand the constraints placed by each transform
3. Collect all constraints on input data files from all the graphs



Implementation

- The dataframe graph abstract domain was implemented as a part of PyLiSA
- Built on LiSA - a modular multi-language static analyser
- We can produce graphs from notebooks for a subset of pandas
- Able to specify order of execution of cells within notebooks

Next step

Use the generated graphs to reason about input data usage, data provenance, bias inducing transforms

Thank you!