

# **Building Theseus Plans**

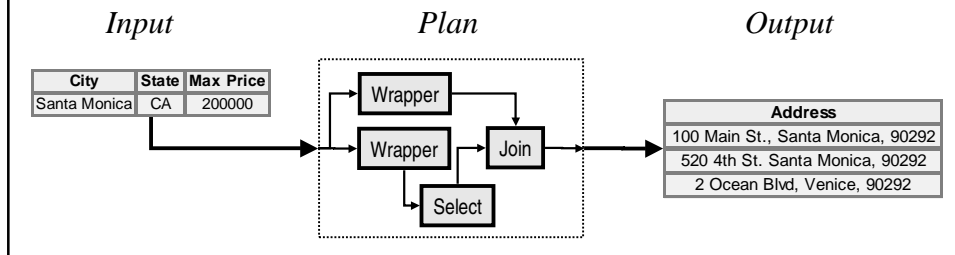
**Greg Barish  
Ph.D. Student  
Advised by Prof. Craig Knoblock**

## **Outline of talk**

- **Review of Theseus**
- **Developing and running a Theseus plan**
- **Theseus operators**
- **Debugging**
- **Questions**

## Review: Information gathering plans

- **Fetch, manipulate, and combine data**
  - Often using multiple sources (data integration)
- **Plans consist of a network of operators**
  - Each operator like a function
    - Example: Wrapper, Select, etc.
  - Data routed between operators are relations
    - Zero or more tuples with one or more attributes



## Review: Efficient plan execution

- **Standard techniques**
  - **Dataflow (horizontal parallelism)**
    - Decentralized, independent operator execution
    - Enables "maximally parallel" operator execution
      - Also known as the "dataflow limit"
  - **Streaming/pipelining (vertical parallelism)**
    - **Producer emits tuples to consumer ASAP**
      - Producer & consumer can process same relation simultaneously
    - **Effective because information gathering latencies can be high – even at the tuple level**
      - Data often "trickles" out of I/O-bound operators

## Building and running Theseus plans

### Theseus

- **Composed of**
  - An information gathering plan language
  - Execution system
- **To use Theseus, you need to know**
  - How you can use the language to write plans...
  - ...and how to execute the plans that you write
- **Theseus requirements**
  - Windows NT/2000
  - Java runtime environment (on all campus PCs)
  - Java development kit (maybe)

## Downloading & installing Theseus

- Later on today you will be able to download
- What to do:
  - Download theseus.zip
  - Unzip it into any local directory
    - Public PCs have limited write permission
    - You can download it to your own machine also
  - Make sure that you can run Theseus
    - `cd c:\downloads\theseus`
    - `trcli.bat` should return

```
::: trcli - Theseus 'relational' client
USAGE: java theseus.tools.tcli.TRCli <.plan file> <relational data file>
```

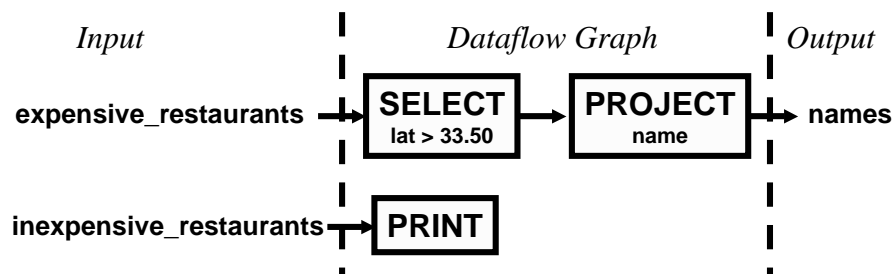
## How to use Theseus components

- After downloading Theseus, you will notice
  - A lot of JAR files. Yes, so there are...
  - One batch file (`trcli.bat`)
  - One Java file (`Functions.java`)
  - Input data for hw4a (`hw4a.data`)
  - A sample plan (`hw4a.sample.plan`)
- To run a Theseus plan, you need to

```
C> trcli.bat hw4a.sample hw4a.data
```

## Designing plans

- To design a plan, you need to
  - Name the plan
  - Identify INPUT and OUTPUT
  - Design dataflow graph for how INPUT → OUTPUT
- Example: hw4a.sample.plan



## Writing plans

- To write a plan, you will need to
  - Name the plan
  - Specify INPUT and OUTPUT
  - Translate your dataflow graph (use operators)
  - Create an input file (e.g., hw4a.data)
- Example:
  - hw4a.sample.plan, hw4a.data
- Editing plan and input files
  - Use NOTEPAD, WORDPAD, whatever

## **Theseus Operators**

### **Theseus extension: functions**

- **Performs computation and return a value**
  - Input comes from one or more tuples
  - Output is the result of computation
- **Functions take parameters**
  - Example: ROUND(price)
  - Example: AVG(salary)

## Single and multi-row functions

- **Single row functions**
  - Compute a value for each tuple in a relation
  - Example
    - ROUND rounds values based on specified precision
    - **SQL:** SELECT ROUND (salary) FROM employee
- **Multi-row (aggregate) functions**
  - Compute a value based on a group of tuples
  - Example
    - MAX finds the maximum value of a column
    - **SQL:** SELECT MAX(salary) FROM employee

## Writing a Theseus function

- Decide if it is single-row (APPLY) or multi-row (AGGREGATE)
- Edit Functions.java and add your function as appropriate
- Notice
  - Single row functions return an ArrayList!
    - Apply(... SPLIT(msg) ...) → "Hi" "there"
  - Multi-row functions return a Java String/Integer/Double
    - Aggregate(... MAX(price) ...) → 12.99

## **Debugging and questions**

### **Debugging is not always easy**

- **Dataflow programs are generally hard to debug**
  - Everything runs in parallel
  - Sometimes the system does not terminate
  - One missing EOS can ruin your whole day
- **Debugging strategies**
  - Simplify plan and/or input
  - Use the Print operator
  - Use the Eoscheck operator

## Example: hw4a.bad.plan

```
/*
 * Sample Theseus plan
 */
PLAN hw4a
{
  INPUT: stream inexpensive_restaurants,
         stream expensive_restaurants
  OUTPUT: stream expensive_names

  BODY
  {
    /* Print the inexpensive restaurants */
    print(inexpensive_restaurants : )

    /* Output the expensive names */
    select(expensive_restaurants, "lat > 33.40" : selected)
    project(selected, foo : expensive_names)
  }
}
```

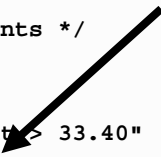
## Example: hw4a.bad.plan

```
/*
 * Sample Theseus plan
 */
PLAN hw4a
{
  INPUT: stream inexpensive_restaurants,
         stream expensive_restaurants
  OUTPUT: stream expensive_names

  BODY
  {
    /* Print the inexpensive restaurants */
    print(inexpensive_restaurants : )

    /* Output the expensive names */
    select(expensive_restaurants, "lat > 33.40" : selected)
    eoscheck(selected, "selected" : )
    project(selected, foo : expensive_names)
  }
}
```

Use eoscheck



## Example: hw4a.bad.plan

```
name char, lat number, lon number
McDonalds, 34.01, -118.41
Edies, 34.51, -117.11
Subway, 34.31, -117.28
Taco Bell, 33.98, -117.99
Berris, 33.90, -119.31
Panda Express, 33.44, -116.35
Baja Fresh, 32.32, -119.45
Natalee, 34.32, -119.12
Chin Chin, 33.67, -118.10
California Pizza Kitchen, 33.48, -117.55
McCormick and Schmicks, 33.90, -118.78
Burger King, 33.92, -118.72
Islands, 34.50, -117.65
*** Got EOS for selected
-----
RELATION: hw4a.sample_0_expensive_names
  attrs:
```

## Last resort

- **Contacting me**
  
- **Before you do so...**
  - **Make sure that you can reproduce**
  - **Send me one e-mail at a time**
    - **barish@isi.edu**
  - **Send me**
    - **Message describing the problem**
    - **Attachments**
      - **Plan**
      - **Input file**