Professor Stephen A. Edwards
COMS W4115
Programming Languages and Translators
Fall 2015

DAVE

James HyunSeung Hong (hh2473)
Min Woo Kim (mk3351)
Fan Yang (fy2207)
Chen Yu (cy2415)

Etymology

BCE (Before Compiler Era)

Data

Analytical

Visualization with

Ease

CE (Compiler Era)

Data

Analytics Made

Very

Easy

Development Overview

manager
James HyunSeung Hong

language guru
Min Woo Kim

system architect
Fan Yang

tester
Chen Yu

Date and Time	Milestone
Sep. 30th	Language Proposal Completed
Oct. 26th	Language Reference Manual Completed
Oct. 28th - Nov. 9th	Scanner and Parser Completed
Nov. 10th - Nov. 15th	Minimally Working Build of DAVE Compiler Completed
Nov. 16th	Hello_World Presentation
Nov. 17th - Dec. 14th	Expanding Features
Dec. 15th - Dec. 20th	Review and General Testing
Dec. 22nd	Final Report Completed

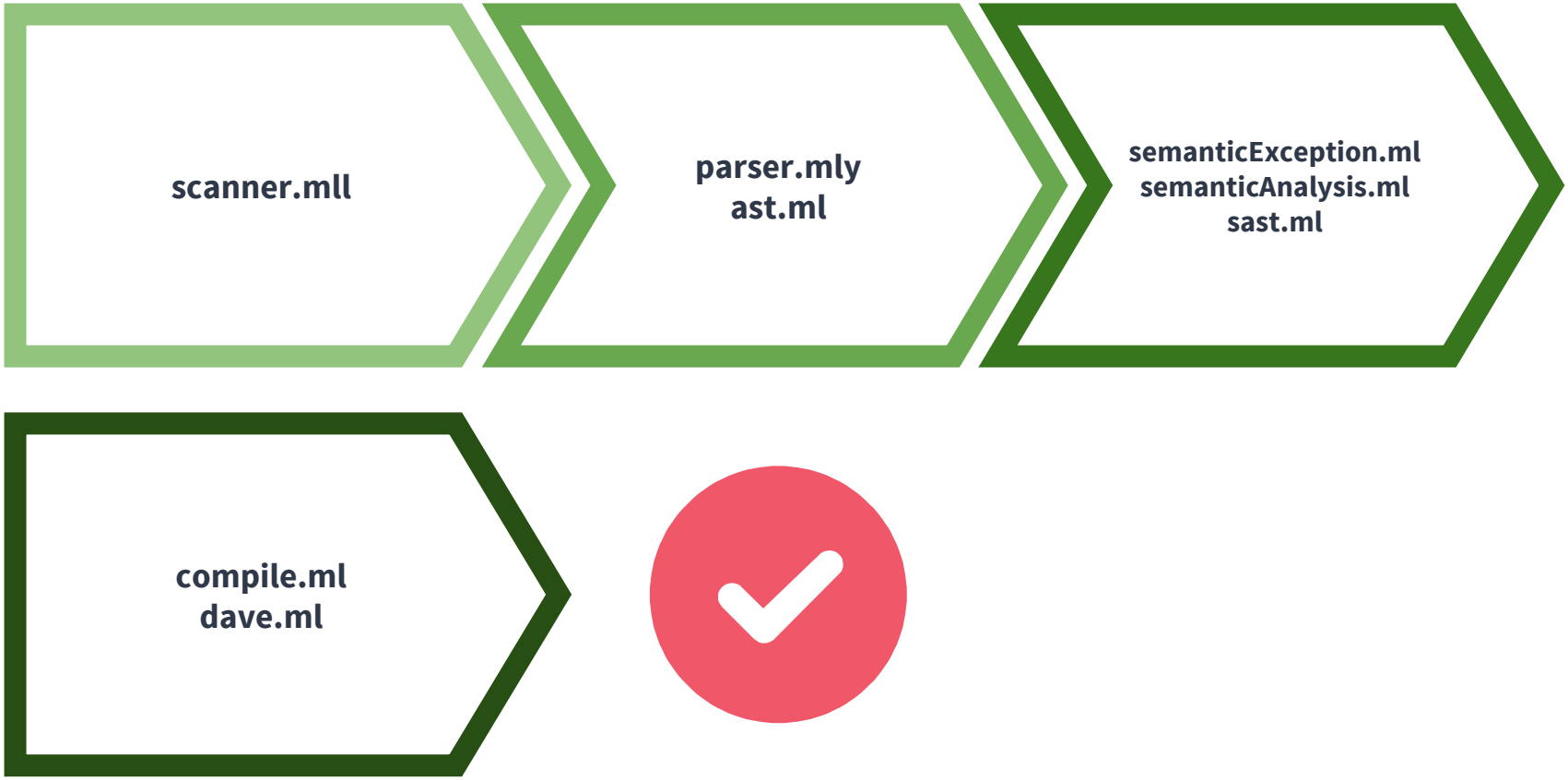
Philosophy

Easy to Manipulate

Useful to Analyze

Quick to Learn

Compiler Architecture



DAVE Data Structure

tbl roster	fld names	fld ages	fld scores
rec student1	**Michael**	**22**	**95**
rec student2	**James**	**23**	**98**
rec student3	**Stephen**	**41**	**100**

fld

Is a DAVE data structure consisting of a sequential collection of homogeneous variables

rec

Is a DAVE data structure consisting of a sequential collection of heterogeneous variables

tbl

Is a DAVA data structure consisting of a collection of flds and recs

Data Construction

fld

```
> fld names =    new fld(["Michael", "James", “Stephen”], "name");
> fld ages = new fld([22, 23, 41], "age");
```

rec

```
> rec student0 = new rec(name: "Michael", age: 22, score: 95);
> rec student1 = new rec(name: "James", age: 23, score: 98);
> rec student2 = new rec(name: "Stephen", age: 41, score: 100);
> rec[] students = [student0, student1, student2];
```

tbl

```
> tbl roster = new tbl(students);
```

Statistical Functionalities

```
> rec student0 = new rec(name: "Michael", age: 22, score: 95);
> rec student1 = new rec(name: "James", age: 23, score: 98);
> rec student2 = new rec(name: "Stephen", age: 41, score: 100);
> rec[] students = [student0, student1, student2];
> tbl roster = new tbl(students[0:2]);
> roster = append(roster, student2);
> roster = modify(roster, 1, 1, 24);

> print(“The number of students is”);
> print(roster.row_length); /* print(students.length); */

> print(“The average score is”);
> print(mean(access(roster, “score”));
```

Showcase Program: Merge Sort

MergeSort.dave

```
int main() {
  int[] A=[6,9,3,10];
  int[] B=[0,0,0,0];
  …
  mergeSort(A,B,4);
  …
}

void mergeSort(int[] A, int[] B, int n) {
  int w = 1;
  int i = 0;
  for (w = 1; w < n; w = 2 * w) {
      for (i = 0; i < n; i = i + 2 * w) {
          bupMerge(A, i, min(i+w, n), min(i+2*w, n), B);
      }
      cpyArr(B,A,n);
    }
}

void bupMerge(int[] A, int left, int right, int end, int[] B) {
  …
}

void cpyArr (int[] B, int[] A, int n) {
  …
}
```

MergeSort.cc

```
#include <iostream>
#include <string>

void cpyArr(int B[], int A[], int n) {
  …
}

void bottomUpMerge(int A[], int left, int right, int
end, int B[]) {
  …
}

void mergeSort (int A[], int B[], int n) { {
  …
}

int main () {
  …
}
```

Output

```
3,6,9,10
```

Showcase Program: Simple Linear Regression

linearRegression (Table-Based).dave

```
int main() {
   …
   tbl sample = new tbl(f);
   …
   print(sample);
   tbl newsample = simpleLinearRegression(sample,a,b);
   …
   print(newsample);
}

tbl simpleLinearRegression(tbl sample, str ind, str dep) {
   …
}
```

linearRegression (Table-Based).cc

```
int max_value(fld a) { … }

int min_value(fld a) { … }

tbl simpleLinearRegression(tbl sample, string ind, string dep) {
   …
}

int main() { … }
```

Output

```
Original Dataset:
ind dep
1 7
2 2
3 9
4 0
5 5
6 4
7 10
8 900
9 1
10  23

Updated Dataset
ind dep est
1 7 -29.9273
2 2 -1.92121
3 9 26.0848
4 0 54.0909
5 5 82.097
6 4 110.103
7 10  138.109
8 900 166.115
9 1 194.121
10  23  222.127

The beta value is
28.0061
The alpha value is
-57.9333
The R-Square value is
(%)
9.00658
```

Test-Driven Development

50

Fifty Test Cases Implemented

- Functional Testing
- Expect-to-Pass + Expect-to-Fail Cases
- Enhanced Coverage

Pair Programming + Code Review

Thanks!

Any questions?

James HyunSeung Hong (hh2473)
Min Woo Kim (mk3351)
Fan Yang (fy2207)
Chen Yu (cy2415)