PA Introduction

Brandeis University
Ryan Marcus

February 24, 2015

Introduction

- Generating DFA from whitelist
- Minimization
- Representing graphs
- Questions

Introduction

Warning! Some text may be kind of small.
Get the slides here: `http://rmarcus.info/dfa.pdf`

Whitelist

Serena Lily Vanessa Jenny Blaire

Start

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

S̲erena Lily Vanessa Jenny Blaire
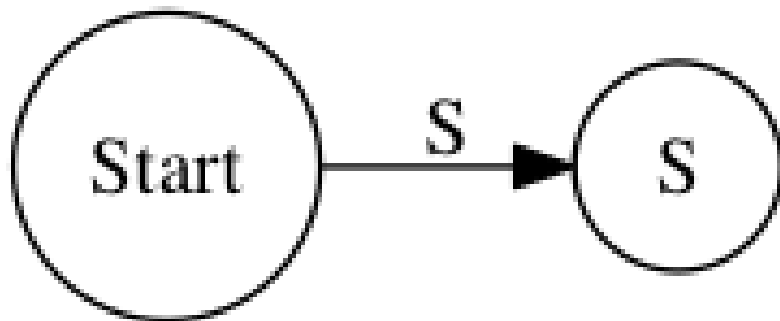
Whitelist

Se<u>r</u>ena Lily Vanessa Jenny Blaire
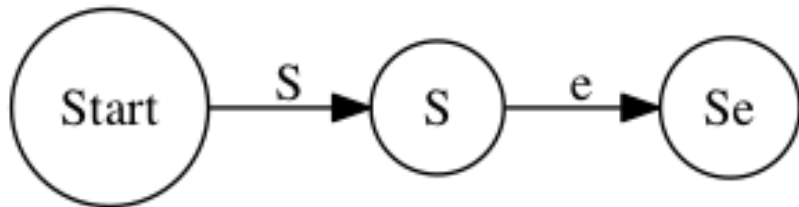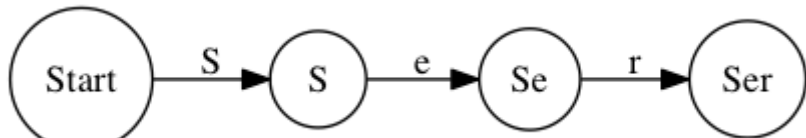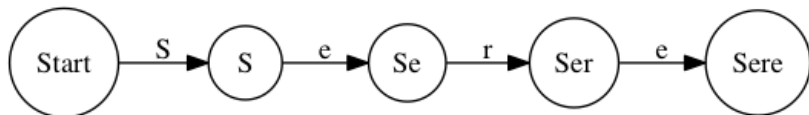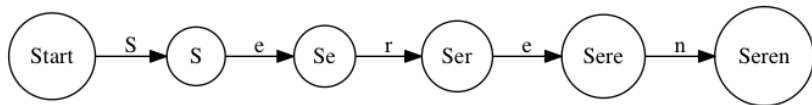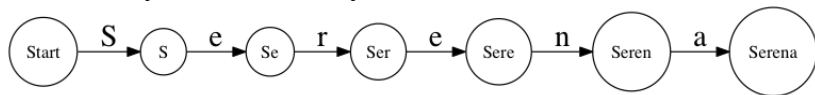
Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

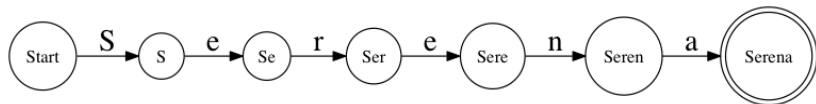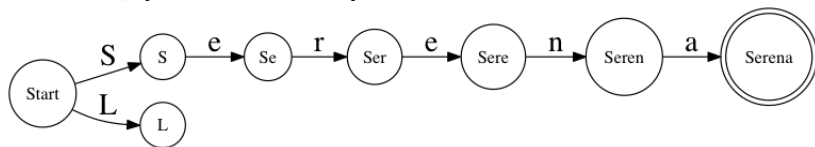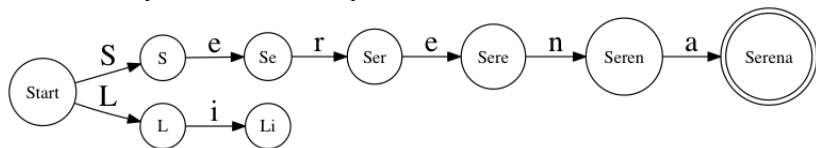Serena L̲ily Vanessa Jenny Blaire
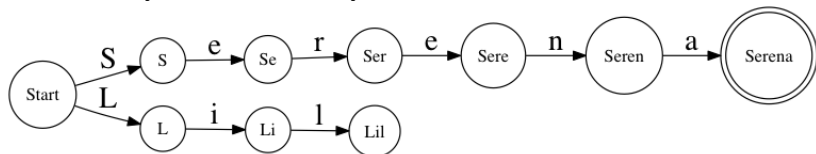
Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire
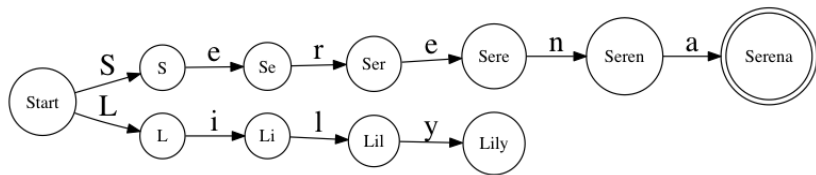
Whitelist

Serena Lily <u>V</u>anessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena   Lily   Vanessa   Jenny   Blaire

Whitelist

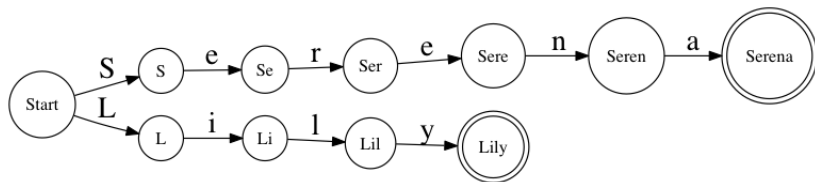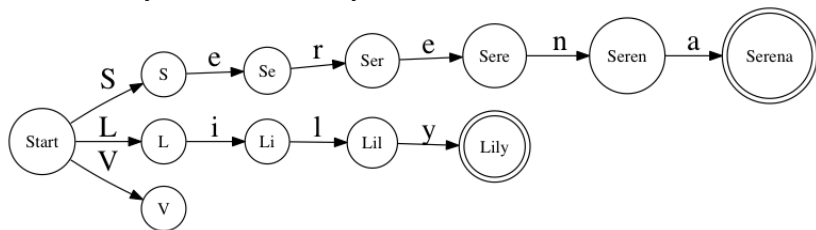Serena Lily Van**e**ssa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily Vanessa Jenny Blaire

Whitelist

Serena Lily <u>Vanessa</u> Jenny Blaire

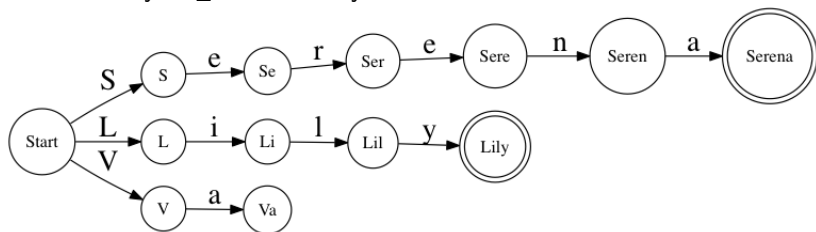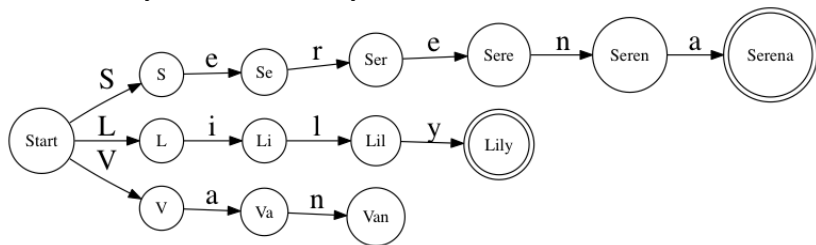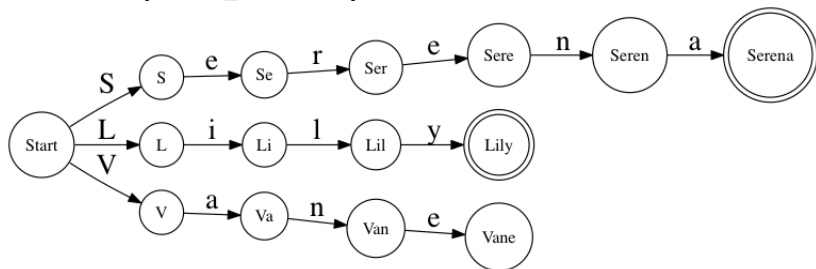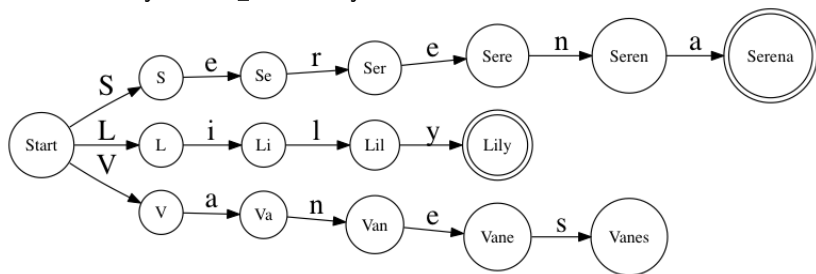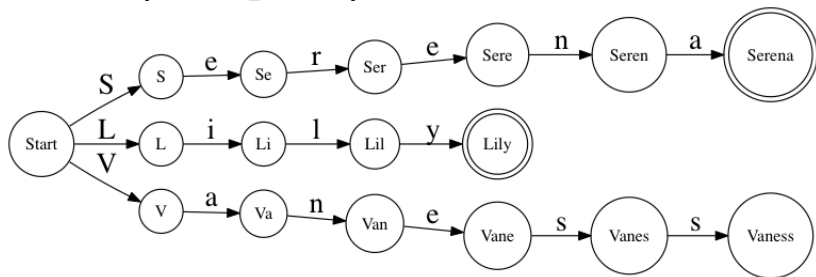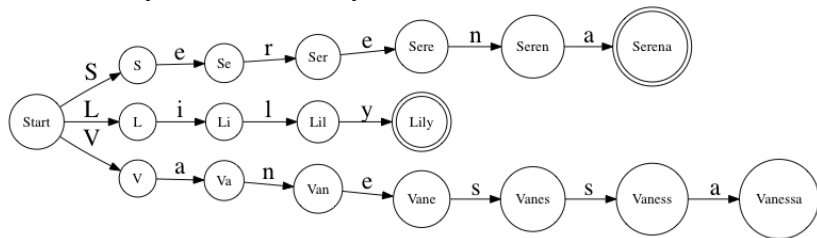# Whitelist

Serena Lily Vanessa <u>Jenny</u> Blaire
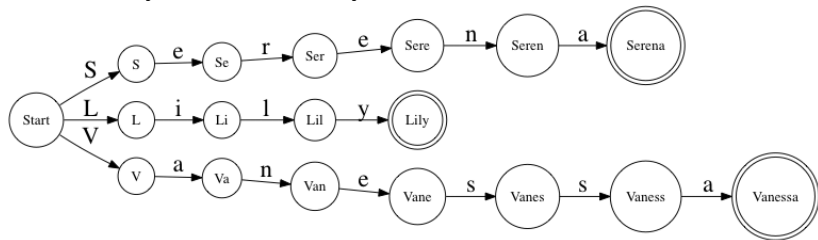
Whitelist

Serena   Lily   Vanessa   Jenny   Blaire

Whitelist

Serena   Lily   Vanessa   Jenny   Blaire

Minimization

Not enough to give the DFA...
Must give a minimal DFA!

Minimization
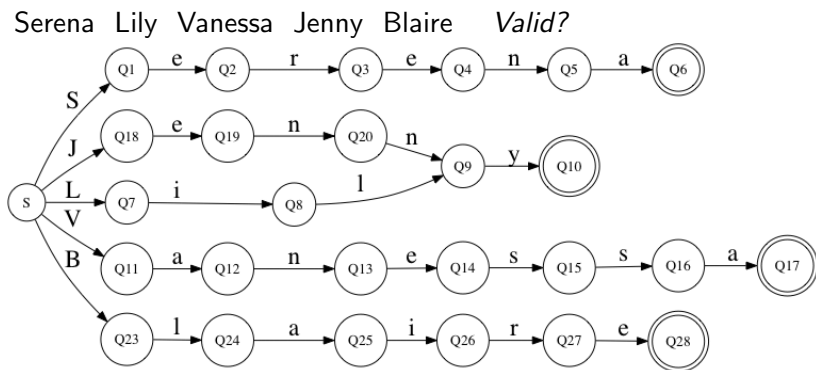
Serena Lily Vanessa Jenny Blaire *Valid?*

Minimization

Serena Lily Vanessa Jenny Blaire *Valid?*
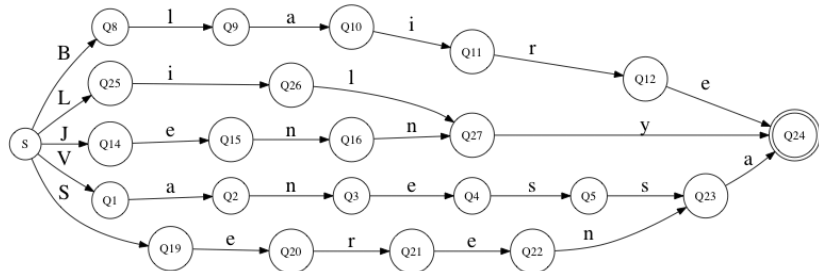
Minimization

Serena Lily Vanessa Jenny Blaire *Valid?*

Minimization
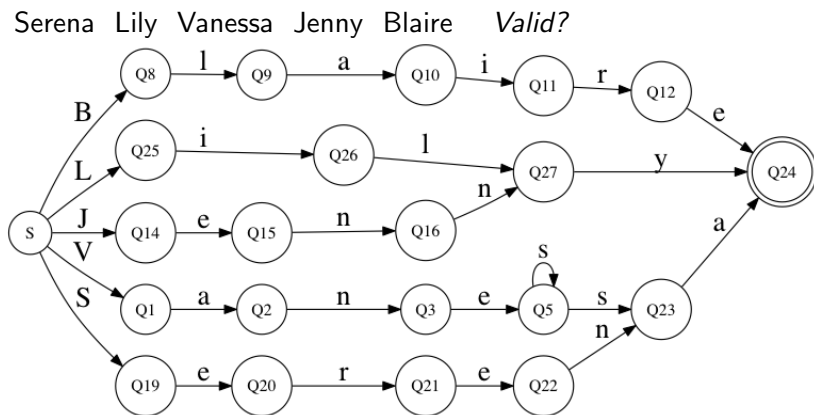
Representation

How can we represent a DFA in Java?

Well, a DFA is a graph, so...

Representation

How can we represent a DFA in Java?

Well, a DFA is a graph, so...

Node class

Adjacency list

Adjacency matrix

Representation: Node Class

```
class Node {
  Map < String , Node > transitions ;
  String nodeLabel ;

  public Node ( String label ) {
    transitions = new HashMap < String , Node >();
    this . nodeLabel = label ;
  }

  public void addTrans ( String on , Node to ) {
   transitions . put ( on , to );
  }
}

Node s = new Node ( "S" );
Node q1 = new Node ( "Q1" );
s . addTrans ( "b" , q1 );
```
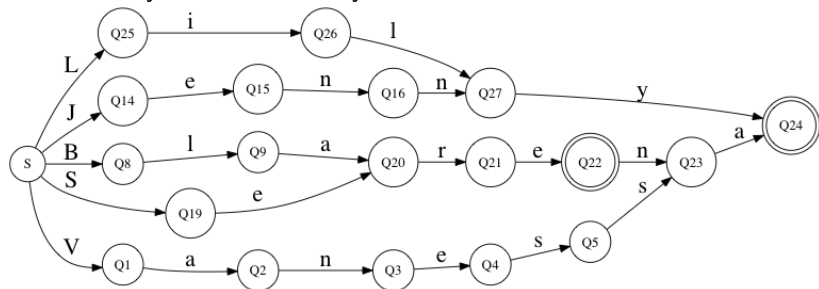
Representation: Adjacency List

```java
Map<String, Map<String, String>> adjList =
    new HashMap<String, Map<String, String>>();

Map<String, String> edgesOfS =
    new HashMap<String, String>();

edgesOfS.put("b", "Q1");

adjList.put("S", edgesOfS);
```

Representation: Adjacency Matrix

```
int c = getNumberOfNodesNeeded();
String[][] m = new String[c][c];

m[0][1] = "b";
```

Representation

Method	Insert	Lookup	Space
Node	$O(1)$	$O(1)$	$O(n)$
List	$O(1)$	$O(1)$	$O(n)$
Matrix	$O(1)$	$O(1)$	$O(n^2)$

Questions?