Handwriting Recognition

Yu-Hong Wang

Senior Developer, GUI

Maplesoft

http://www.maplesoft.com/

TS-3690

Goal

Learn how to apply handwriting recognition to your Java™ product

Agenda

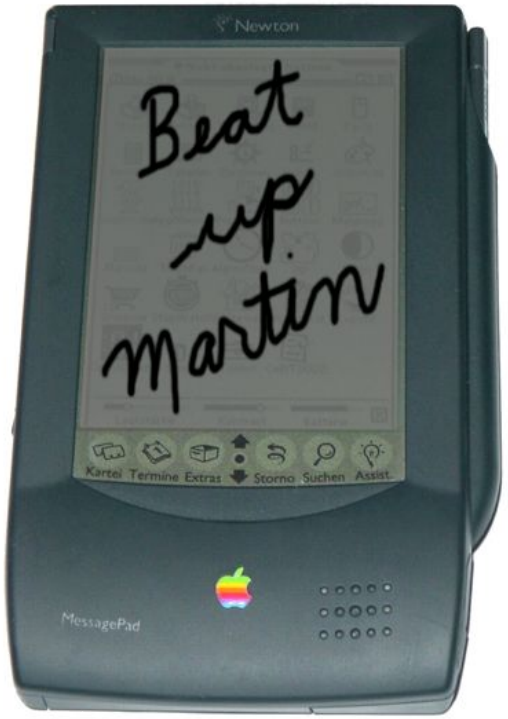
Application

Implementation

Integration

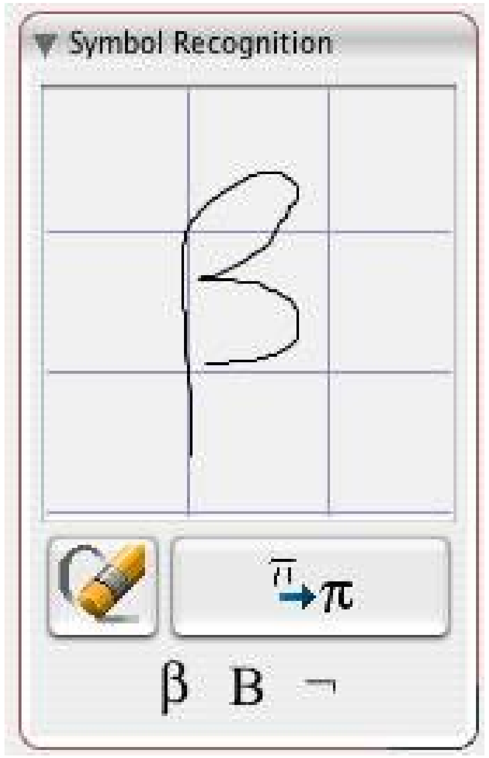
Application

visor


Newton
Beat up Martin
Kartei
Termine
Extras
Storno
Suchen
Assist.
MessagePad

2D

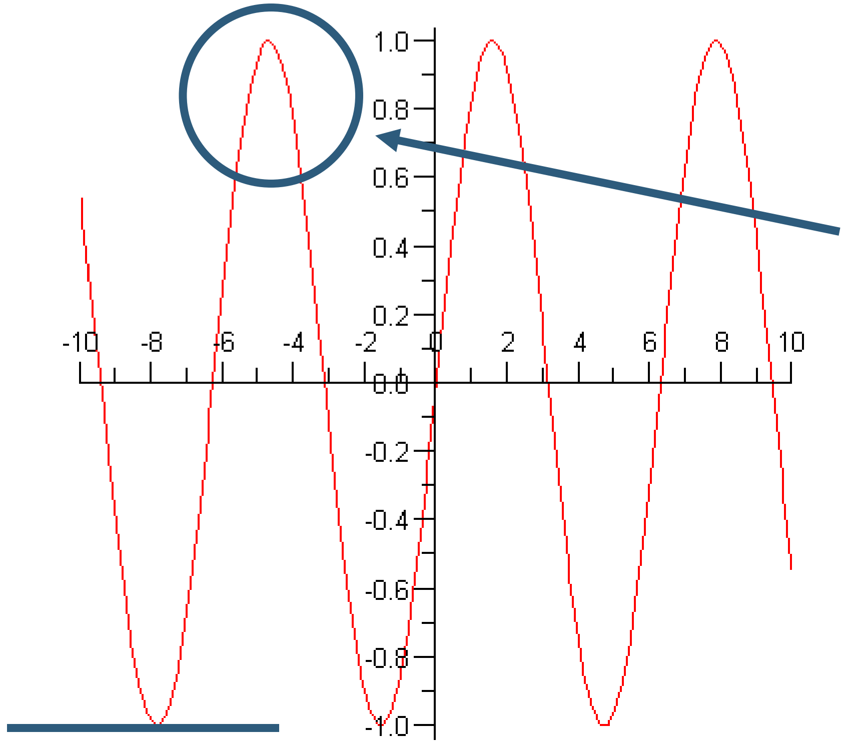
Kanji



Math symbols

$$\sum_{i\geq 0}\binom{n}{i}x^i y^{n-i}$$

Math

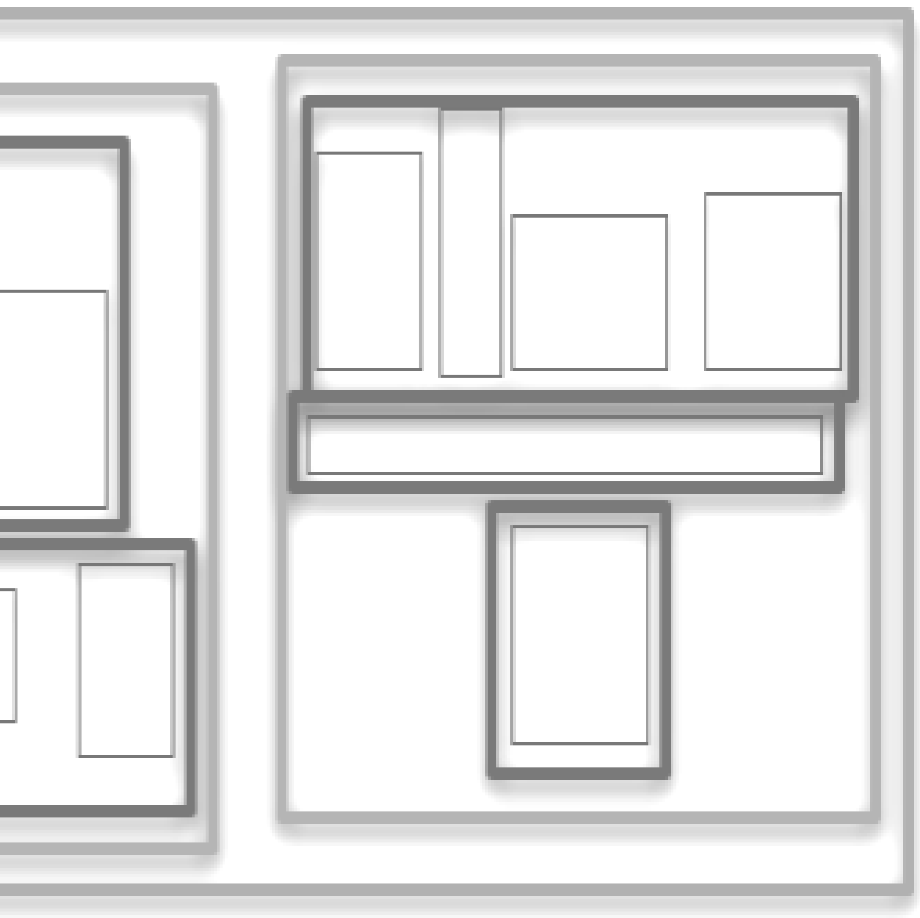


Shapes

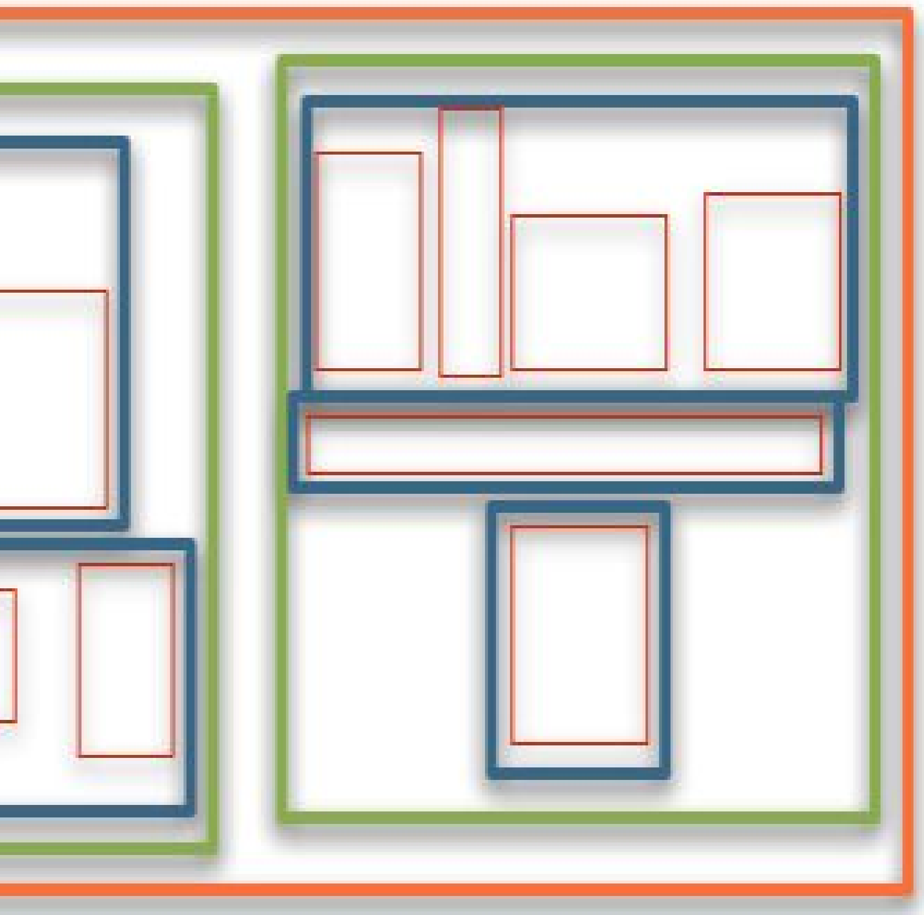
Other Applications

- UML diagrams
- Signature verification
- Accessibility
- Kiosks
- Car navigation systems

Implementation



Structural Character



Structural

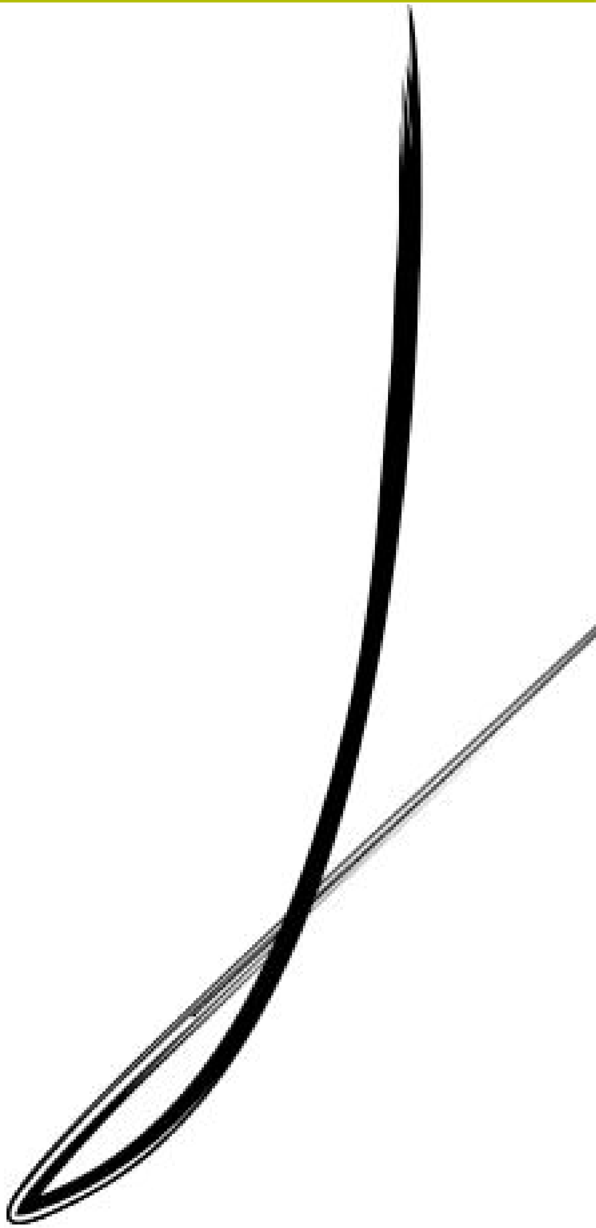
Character

Structural

Character

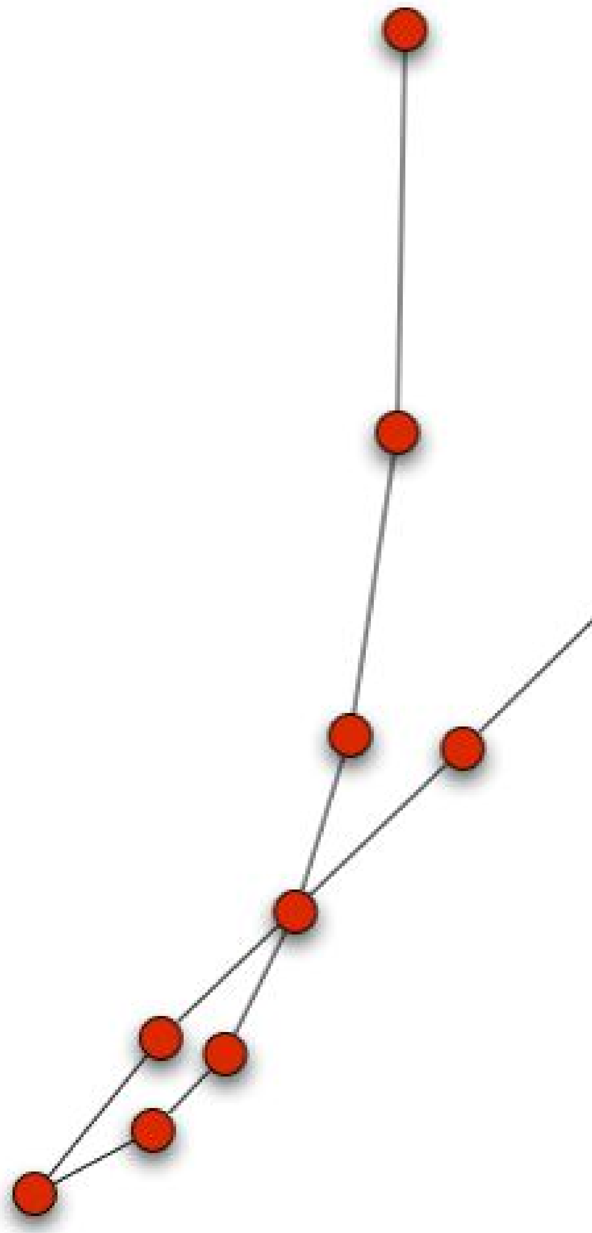
Stroke

Continuous run of input data



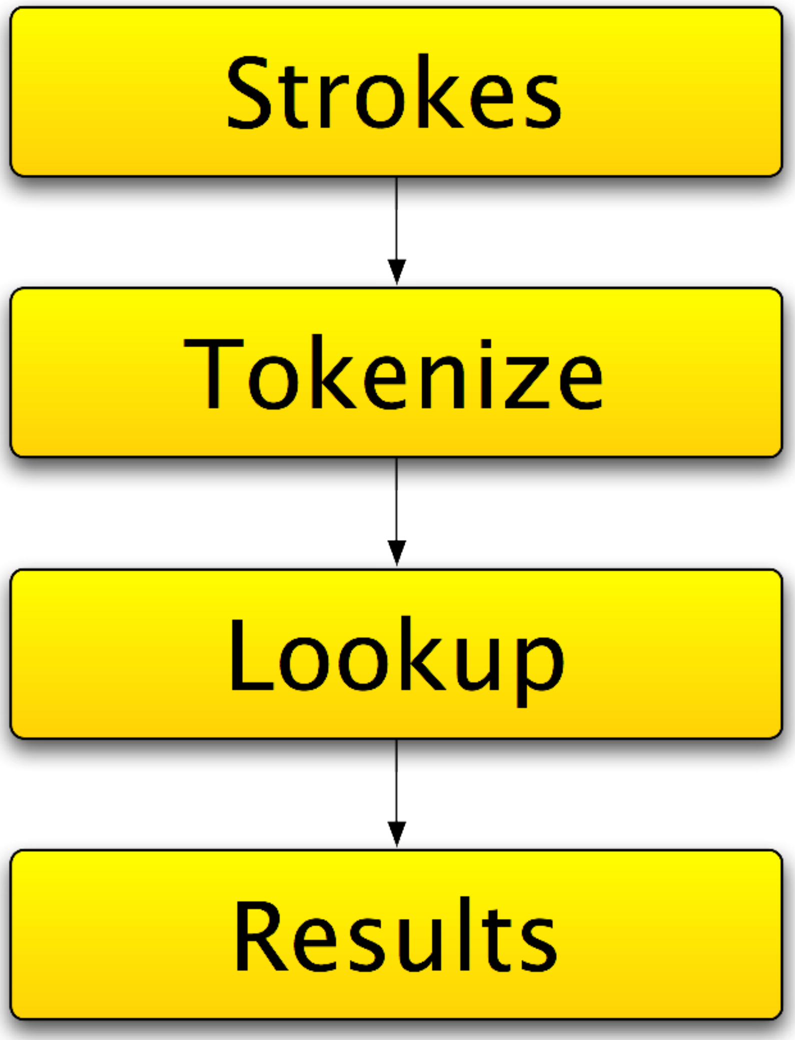
Packet

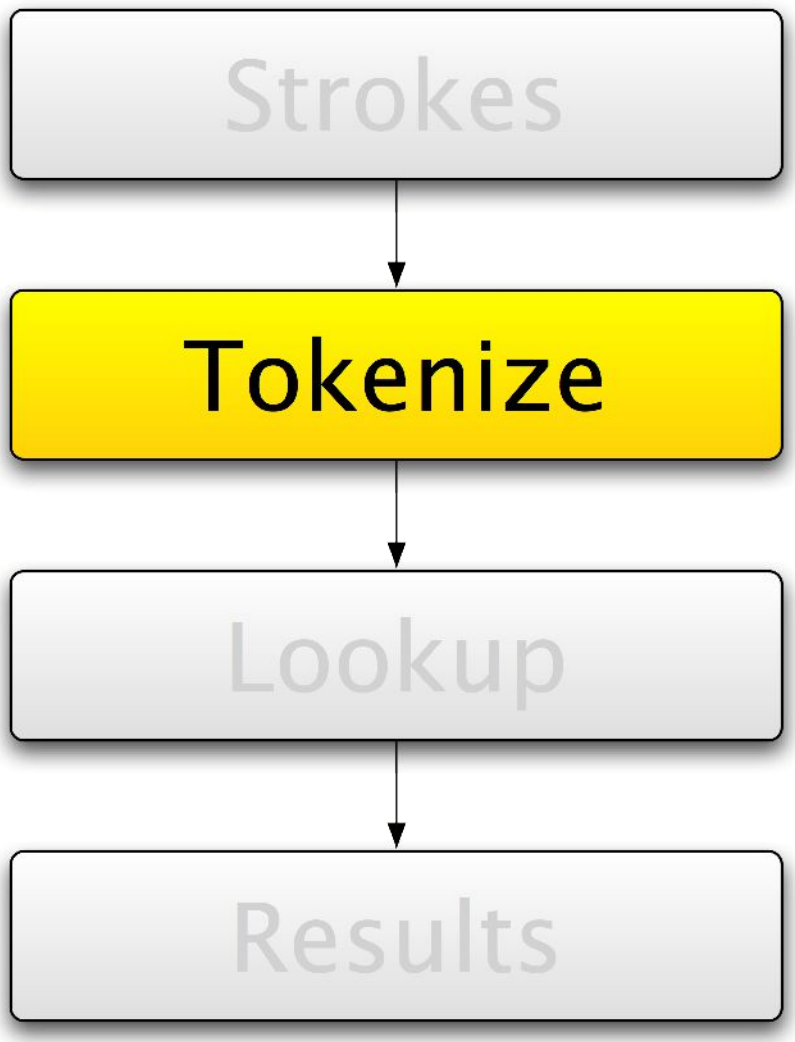
Each individual (x, y) pair



Neural Nets

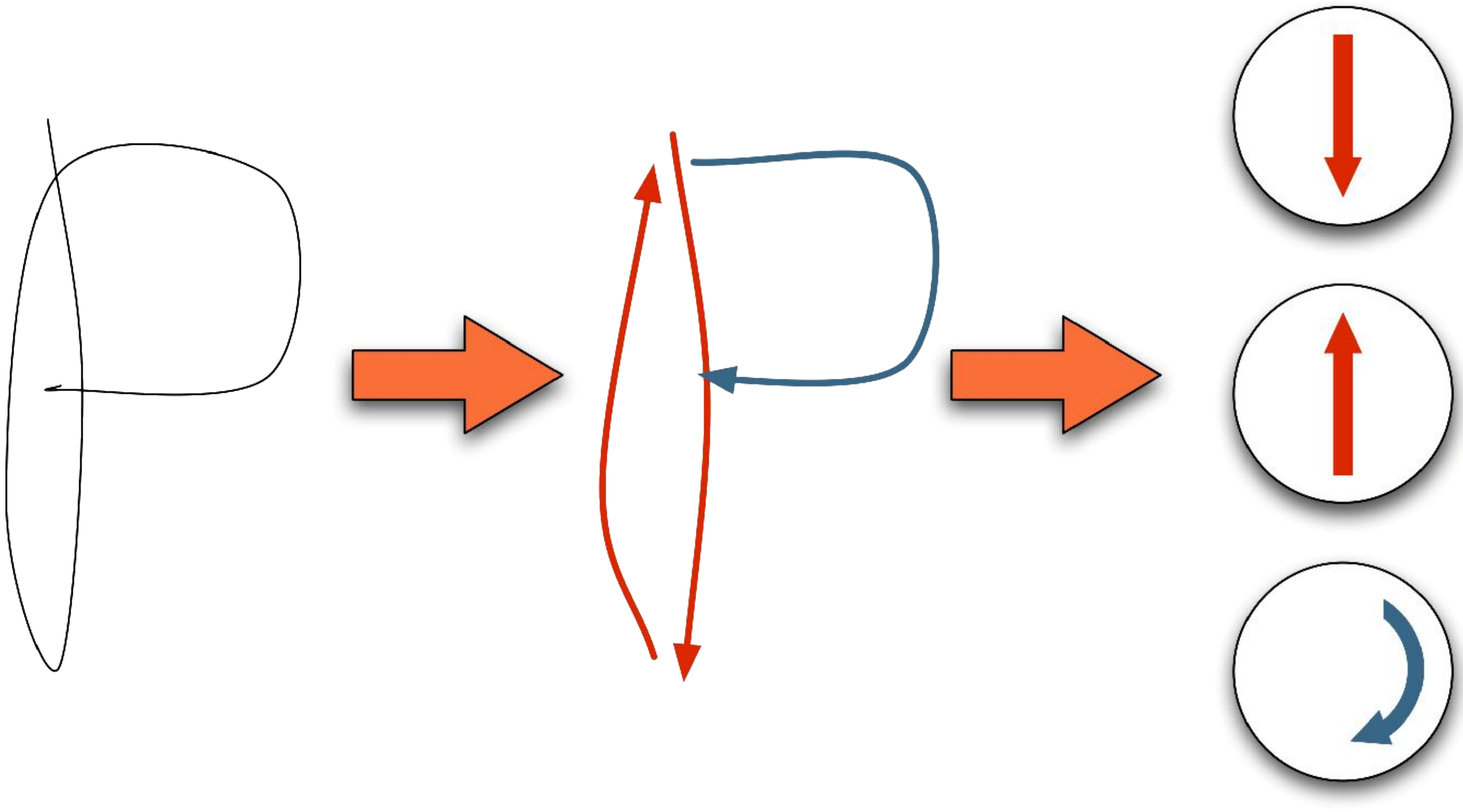
Hidden Markov Models

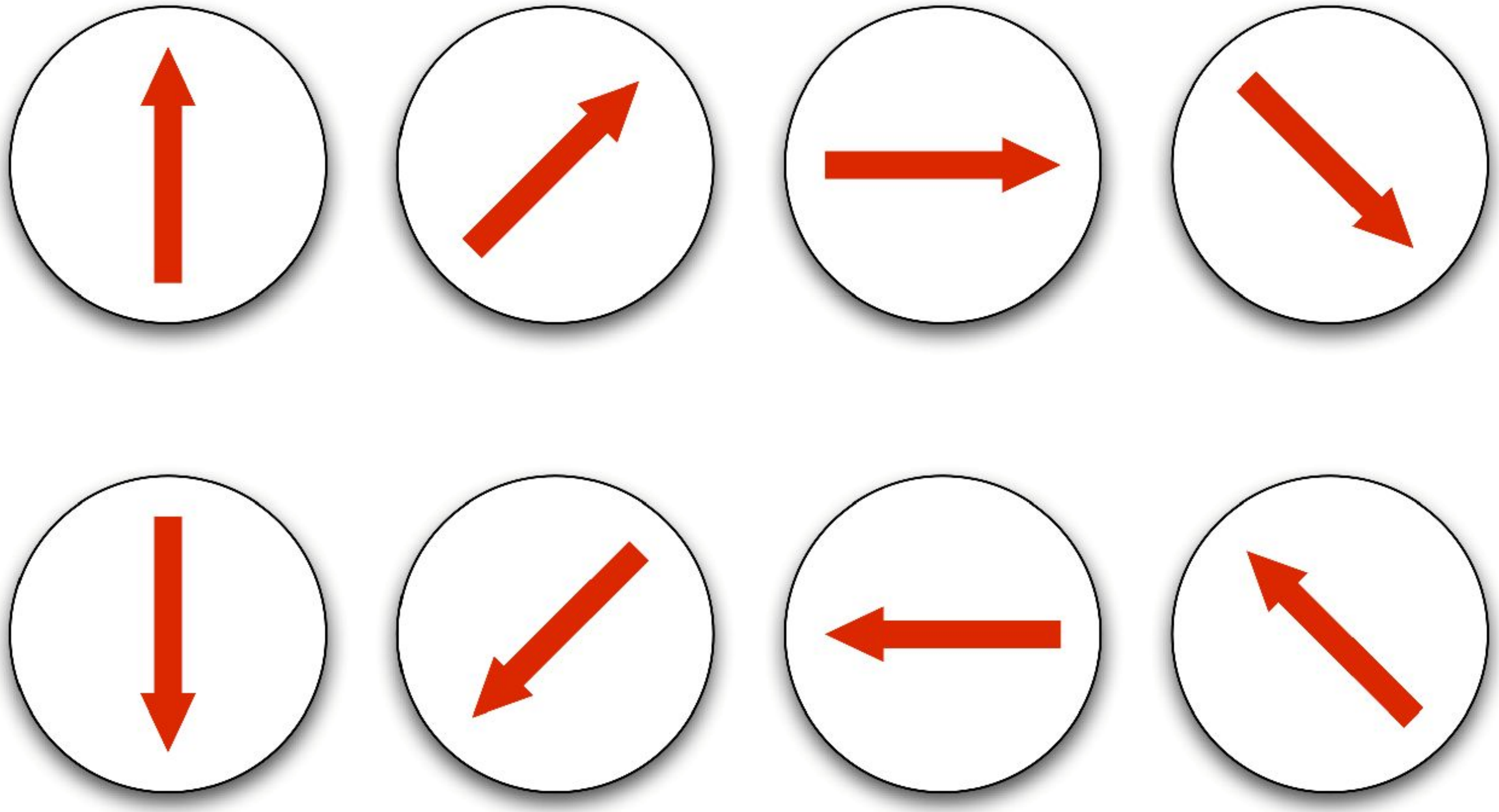

Strokes
Tokenize
Lookup
Results

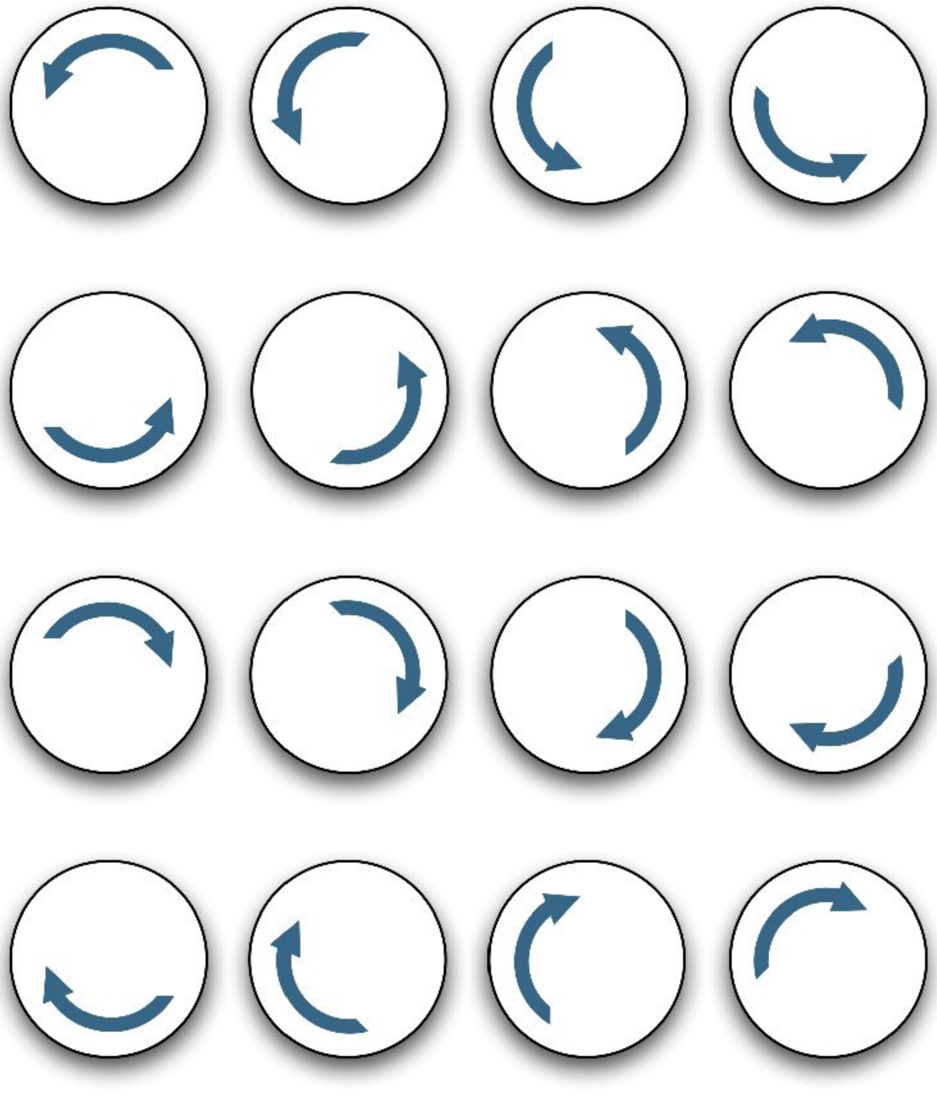

Strokes
Tokenize
Lookup
Results

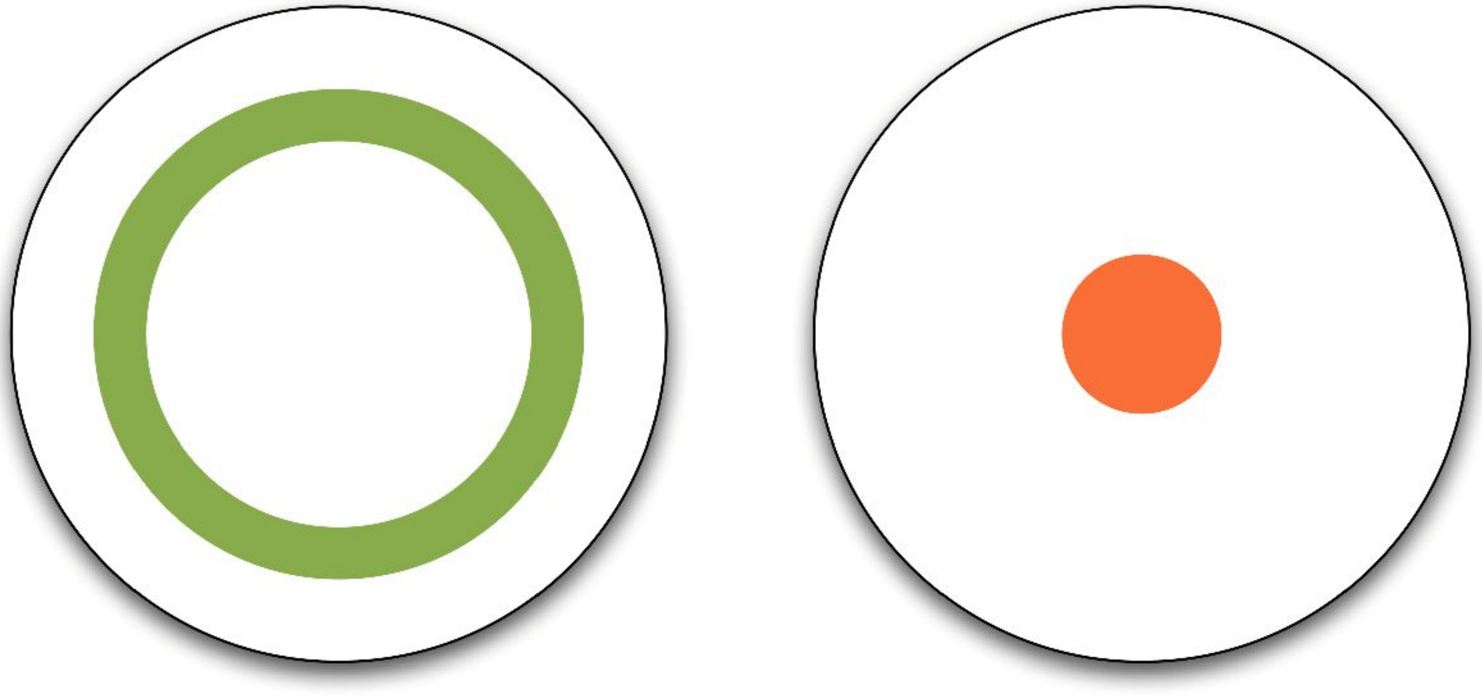
Tokenize

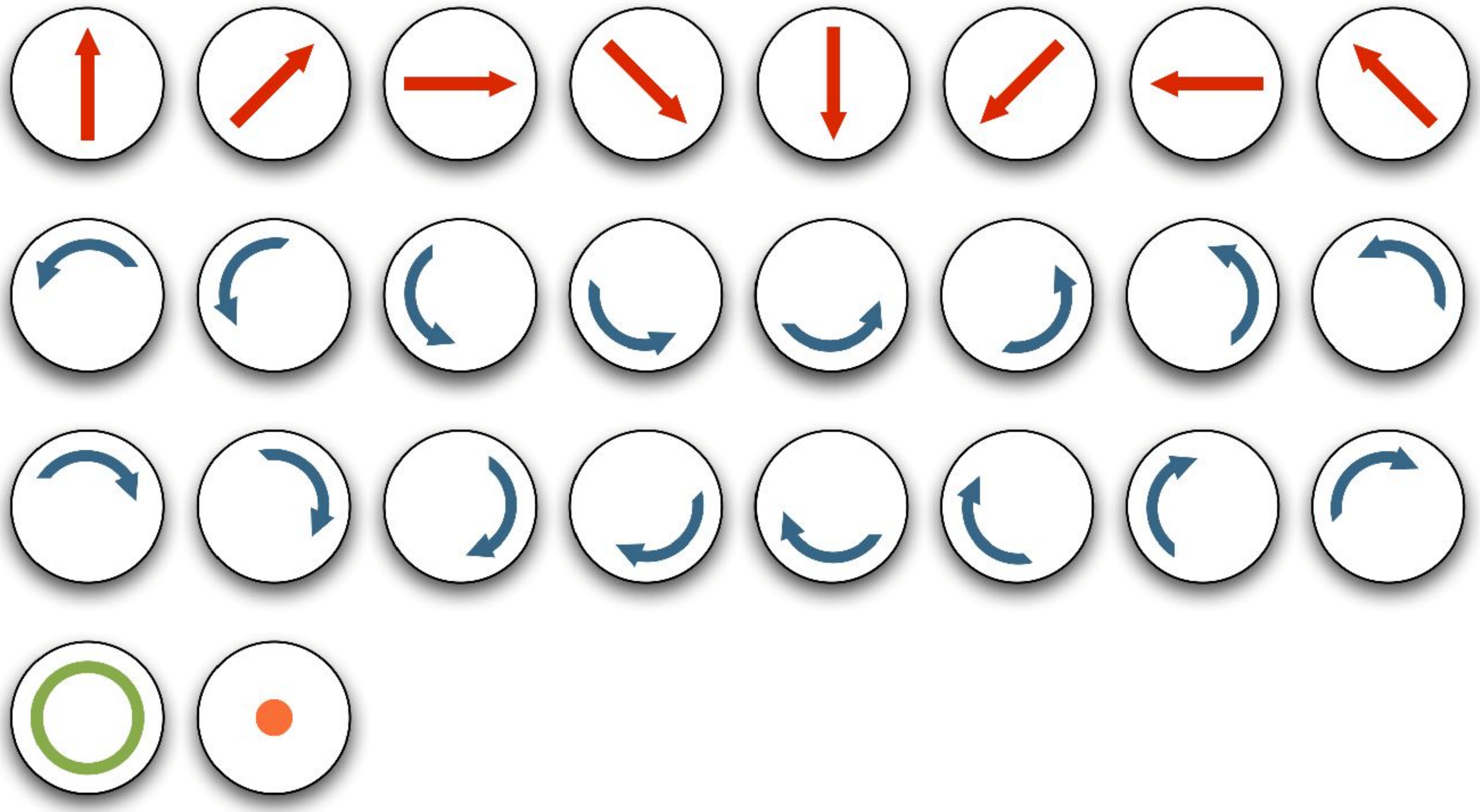
Turn strokes into bite-size chunks

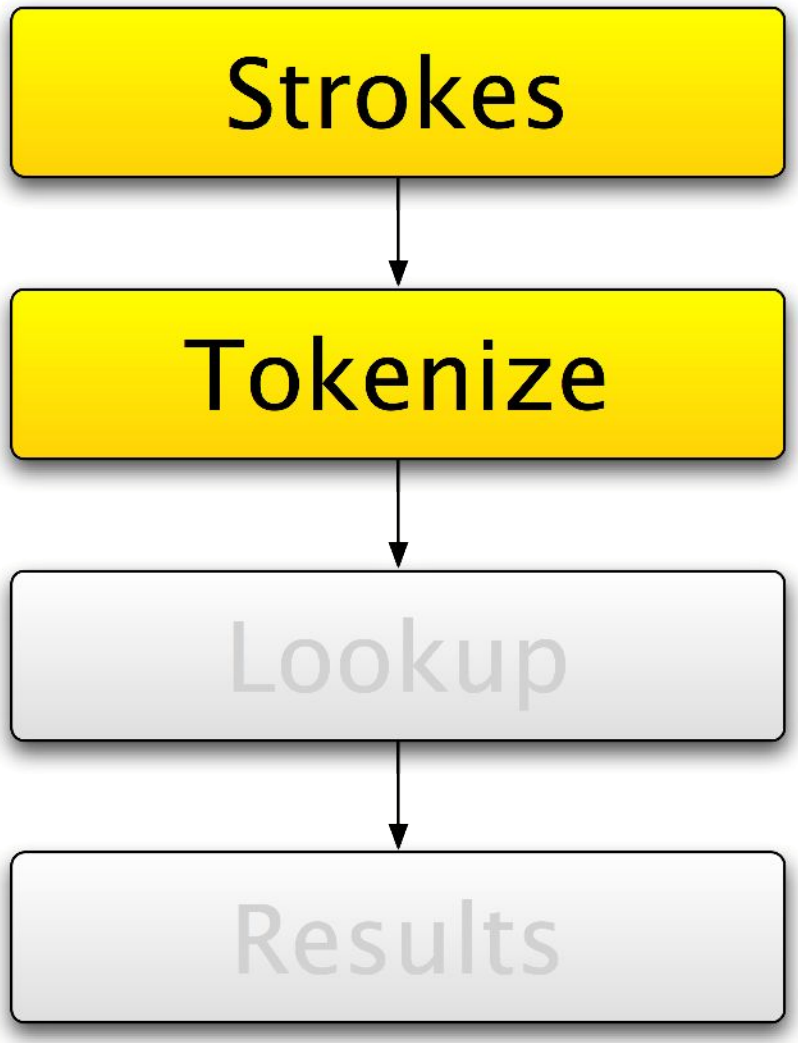






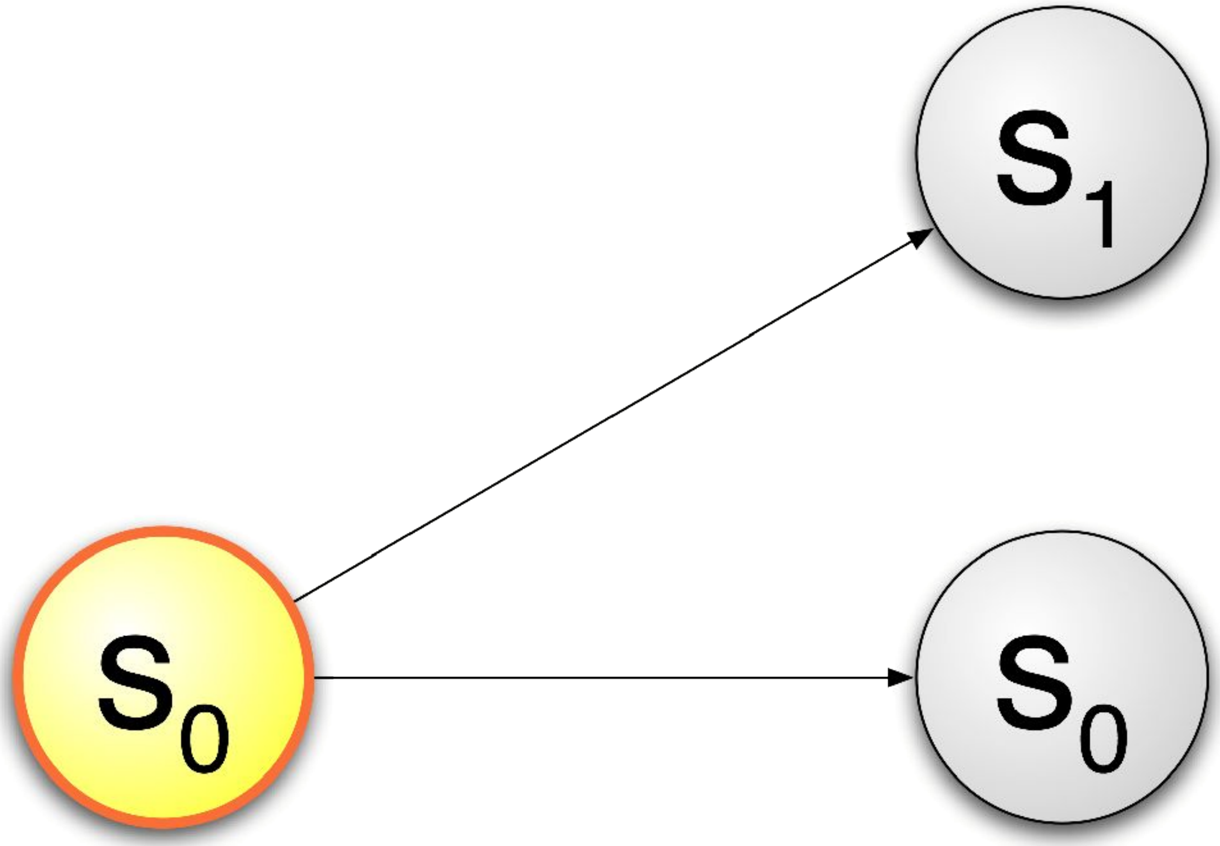


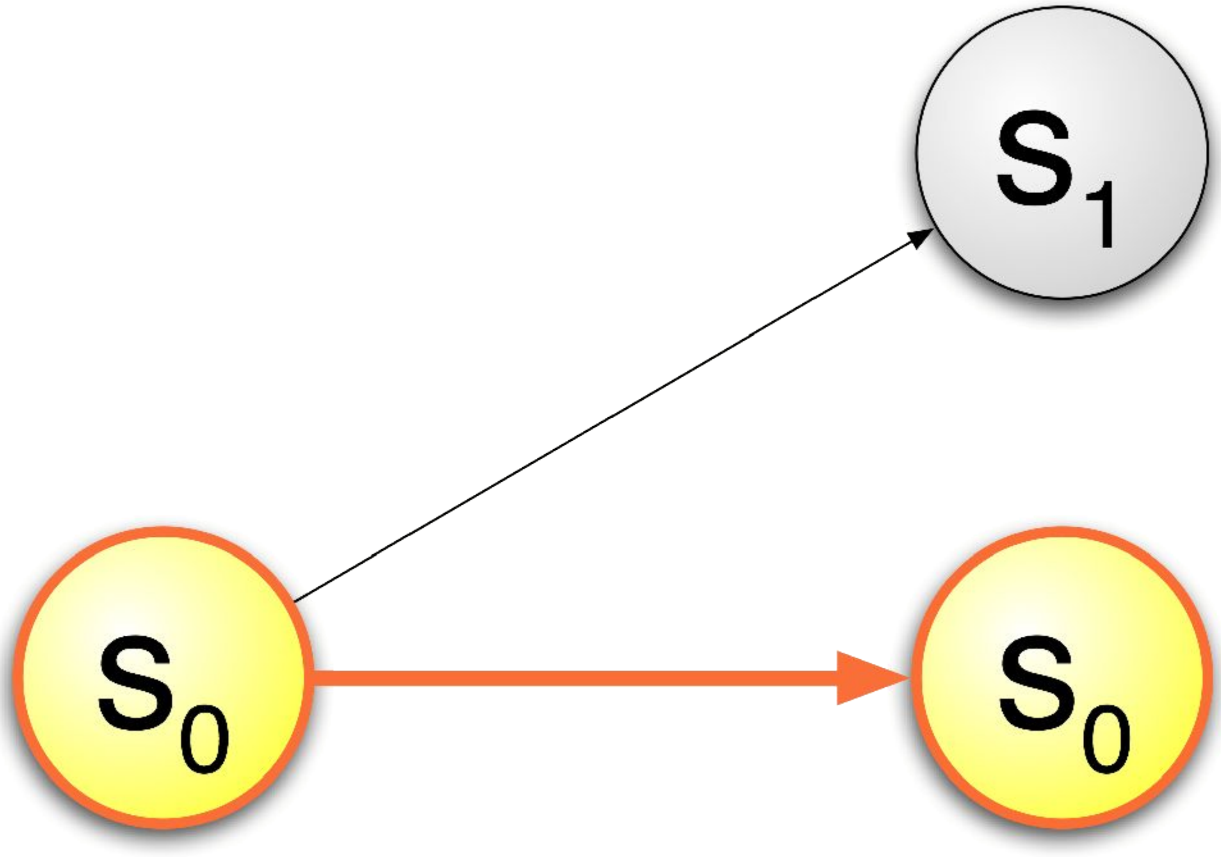



Strokes
Tokenize
Lookup
Results

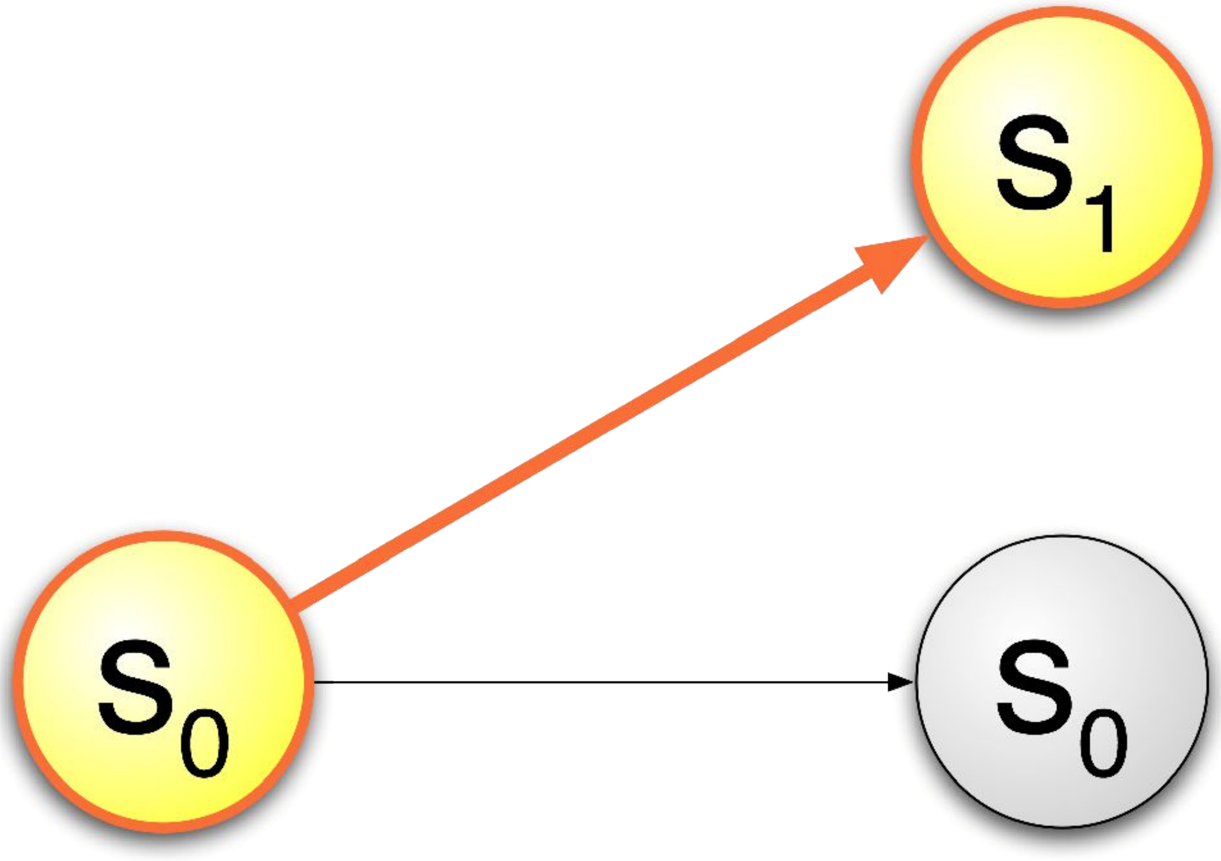
Hidden Markov Model

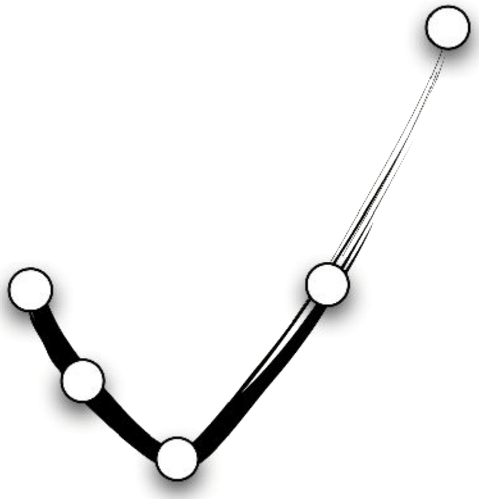
States and transitions

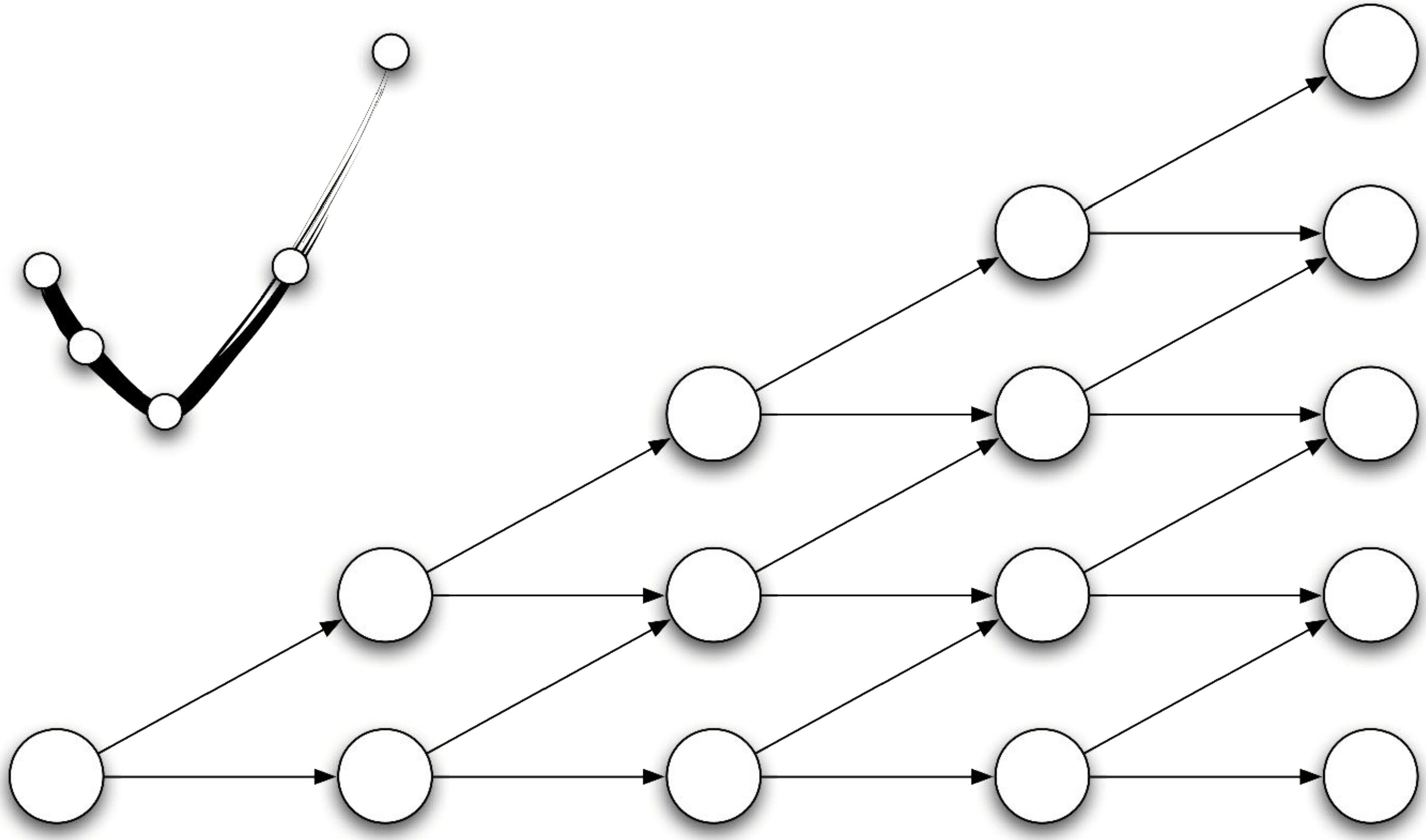

S_1
S_0
S_0

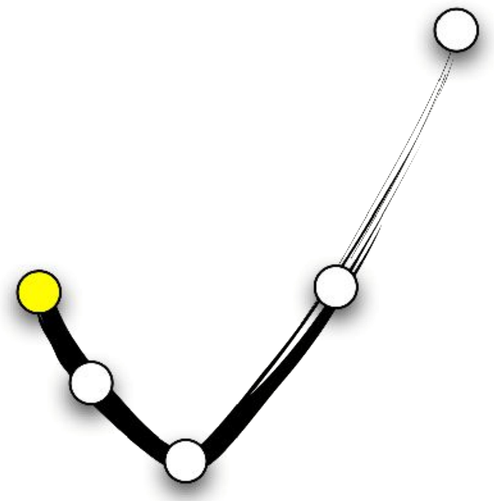
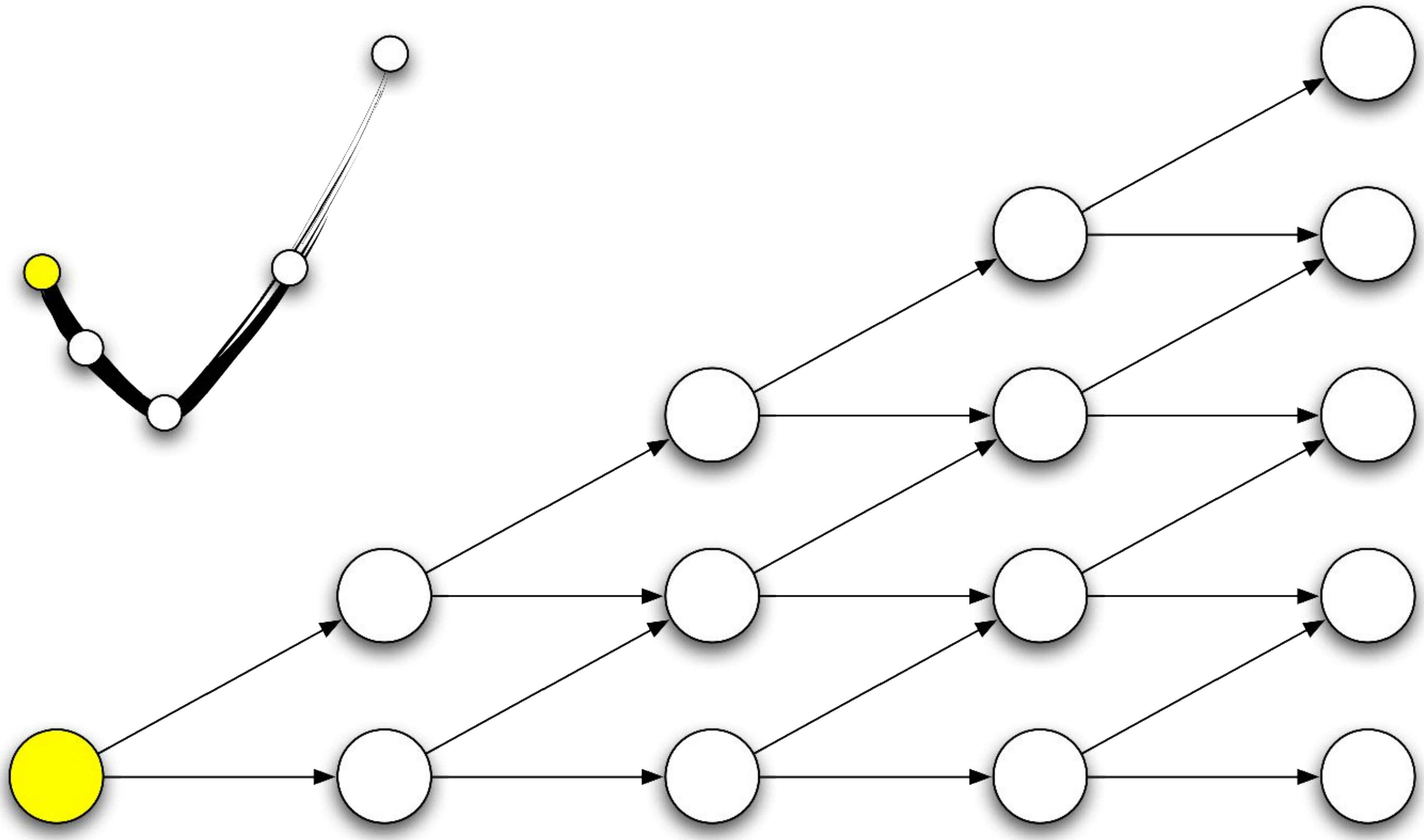


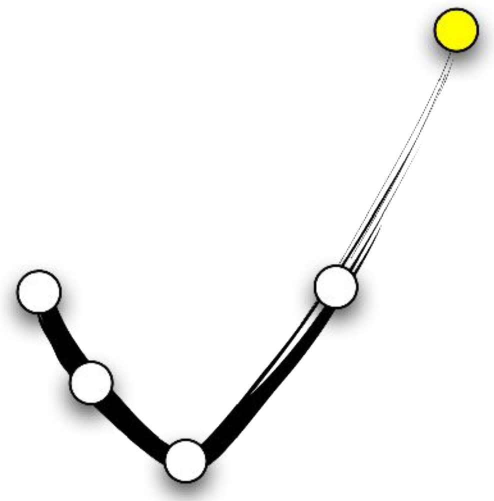
S_1
S_0
S_0

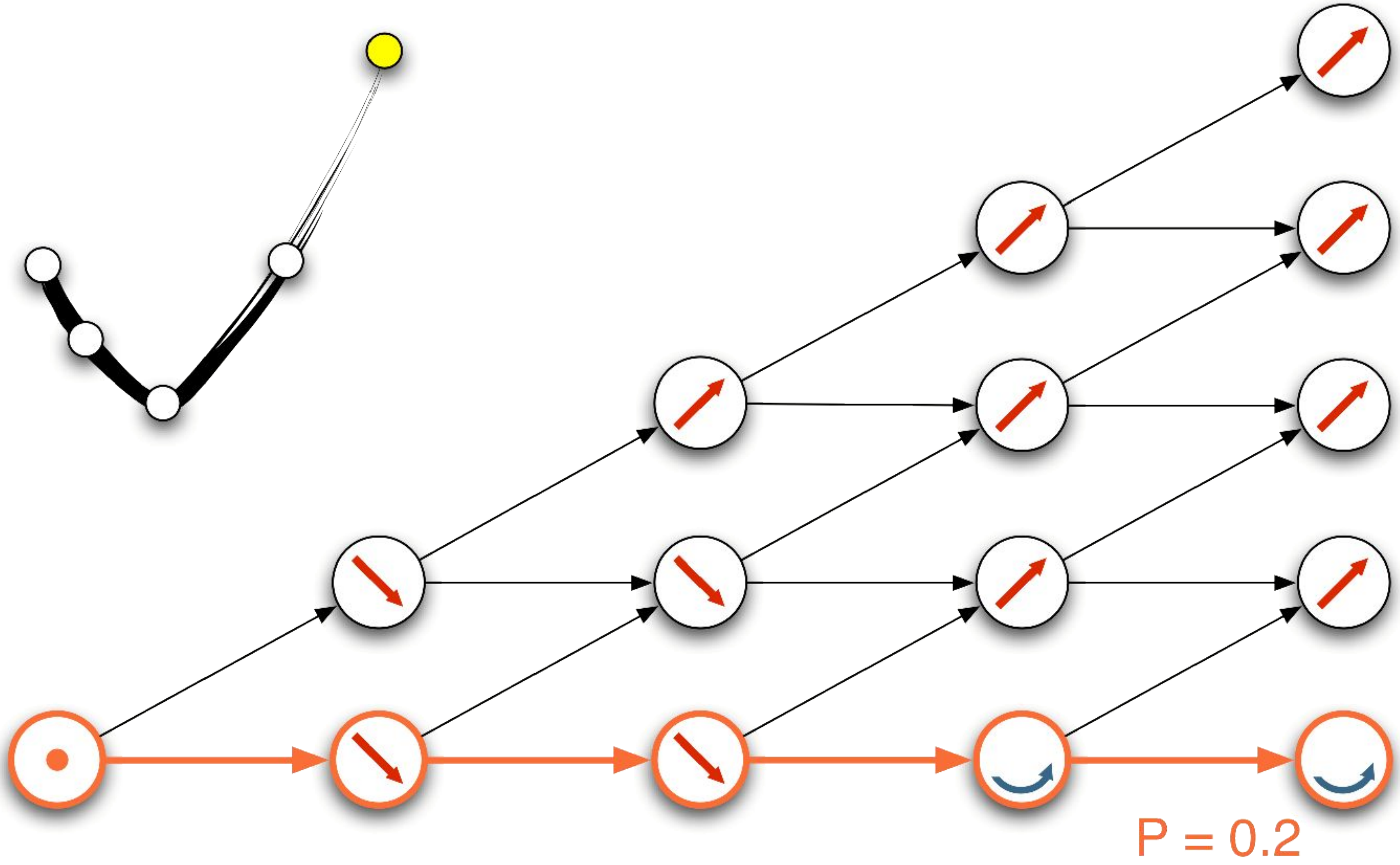

S_1
S_0
S_0

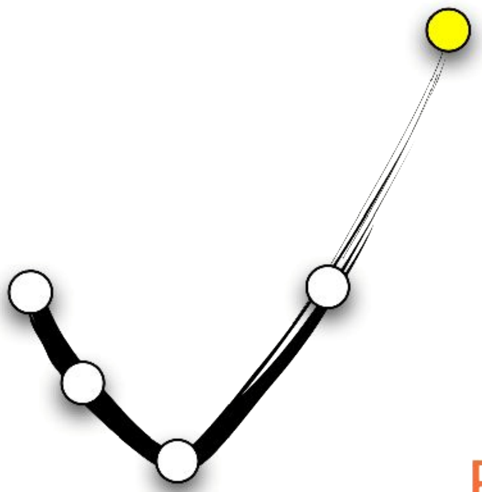


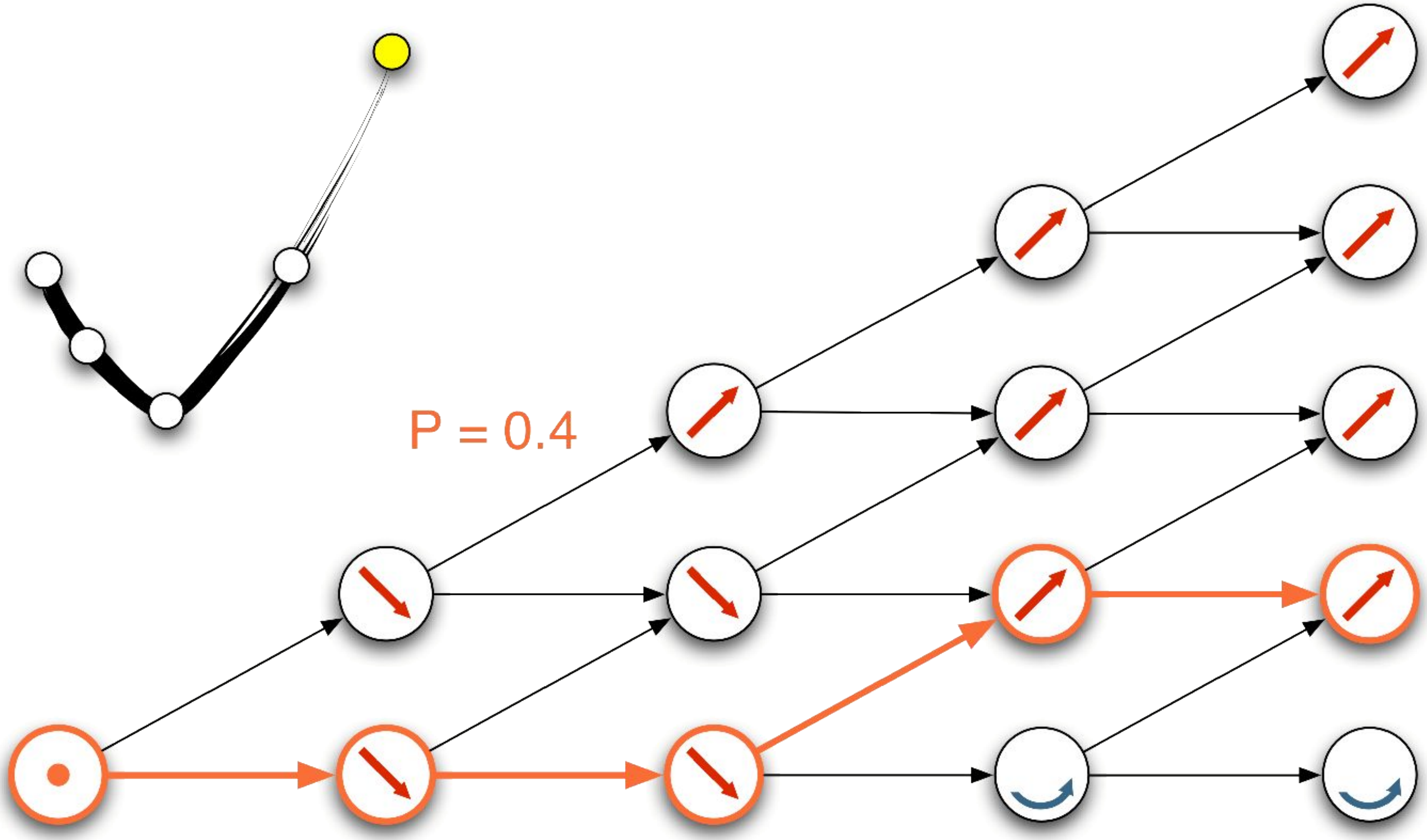


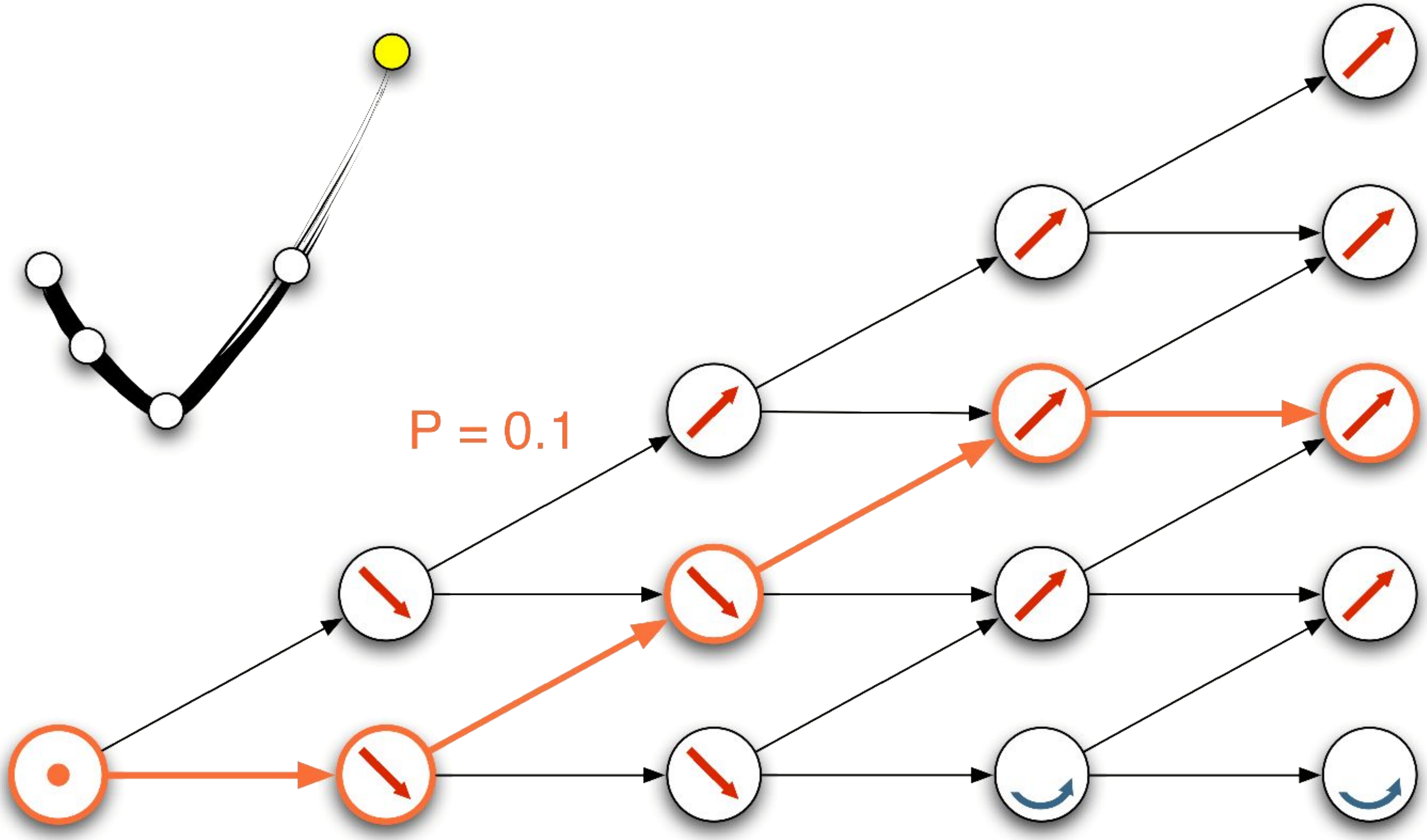



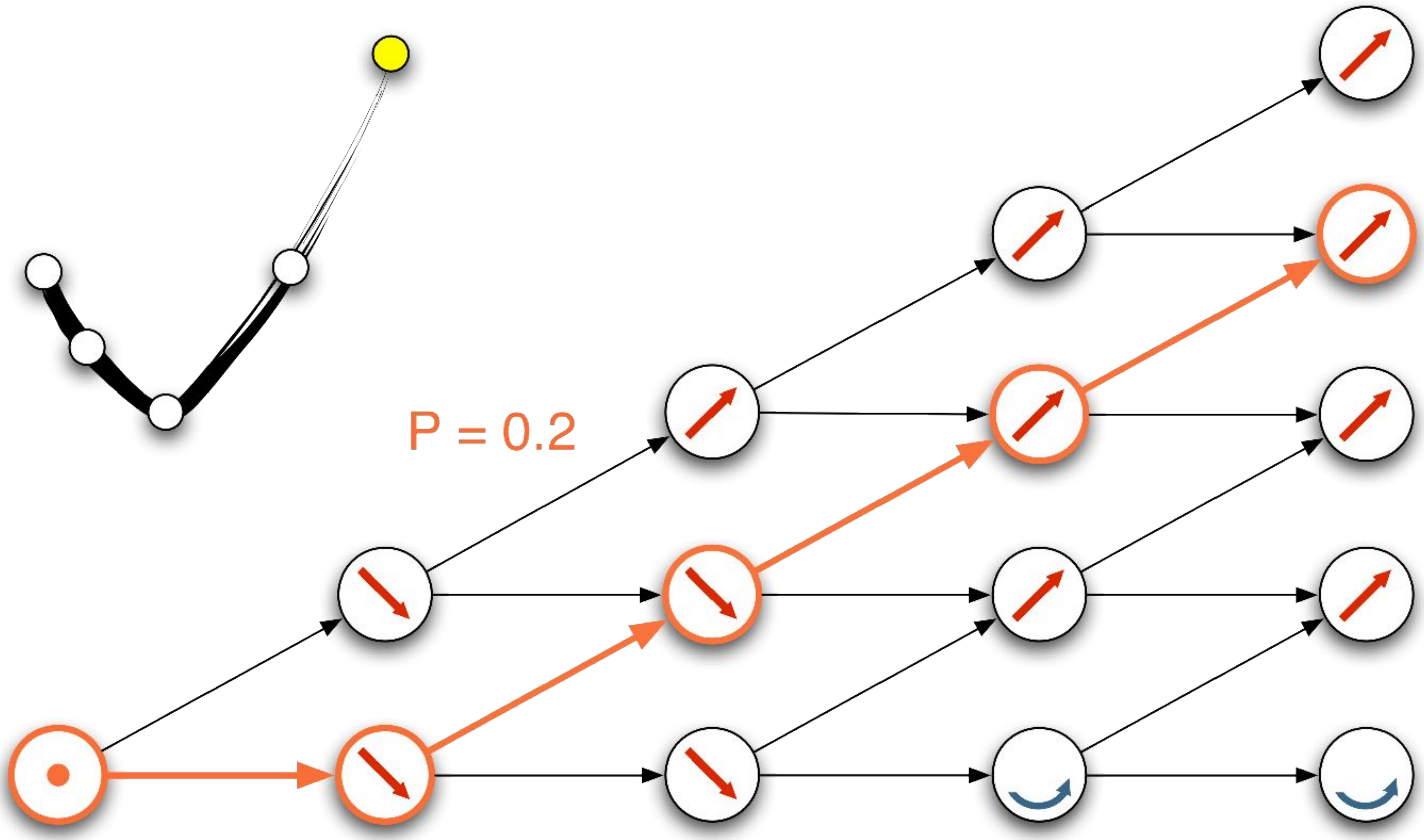


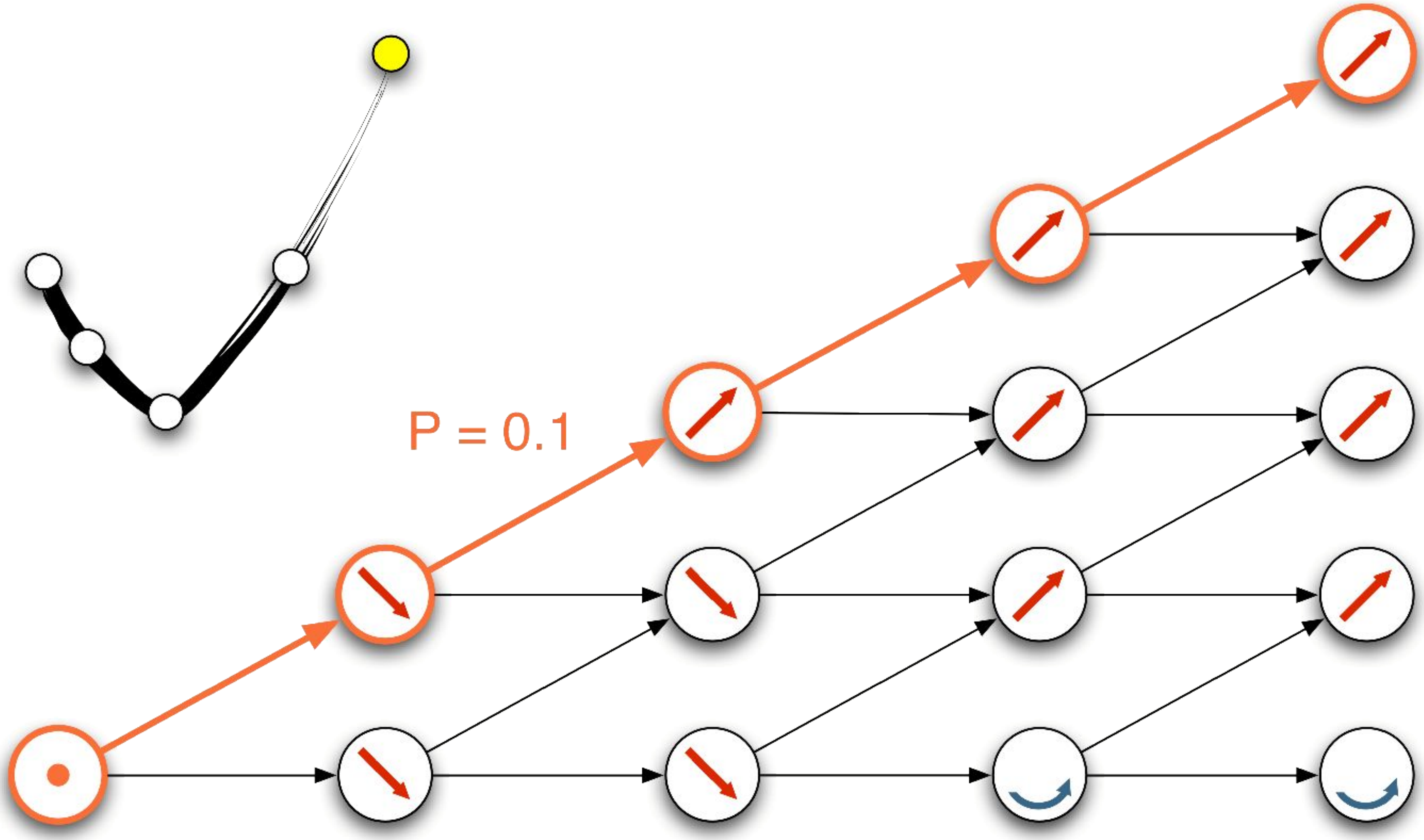

P = 0.2

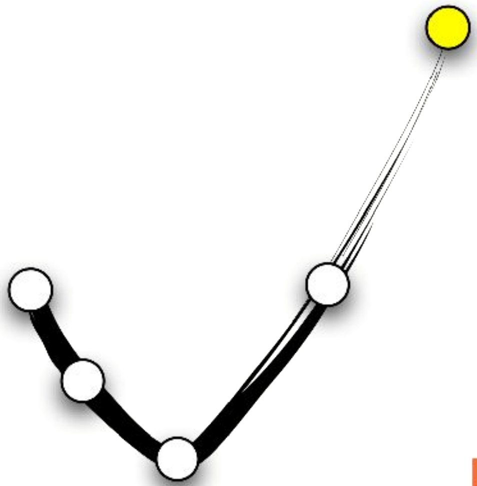


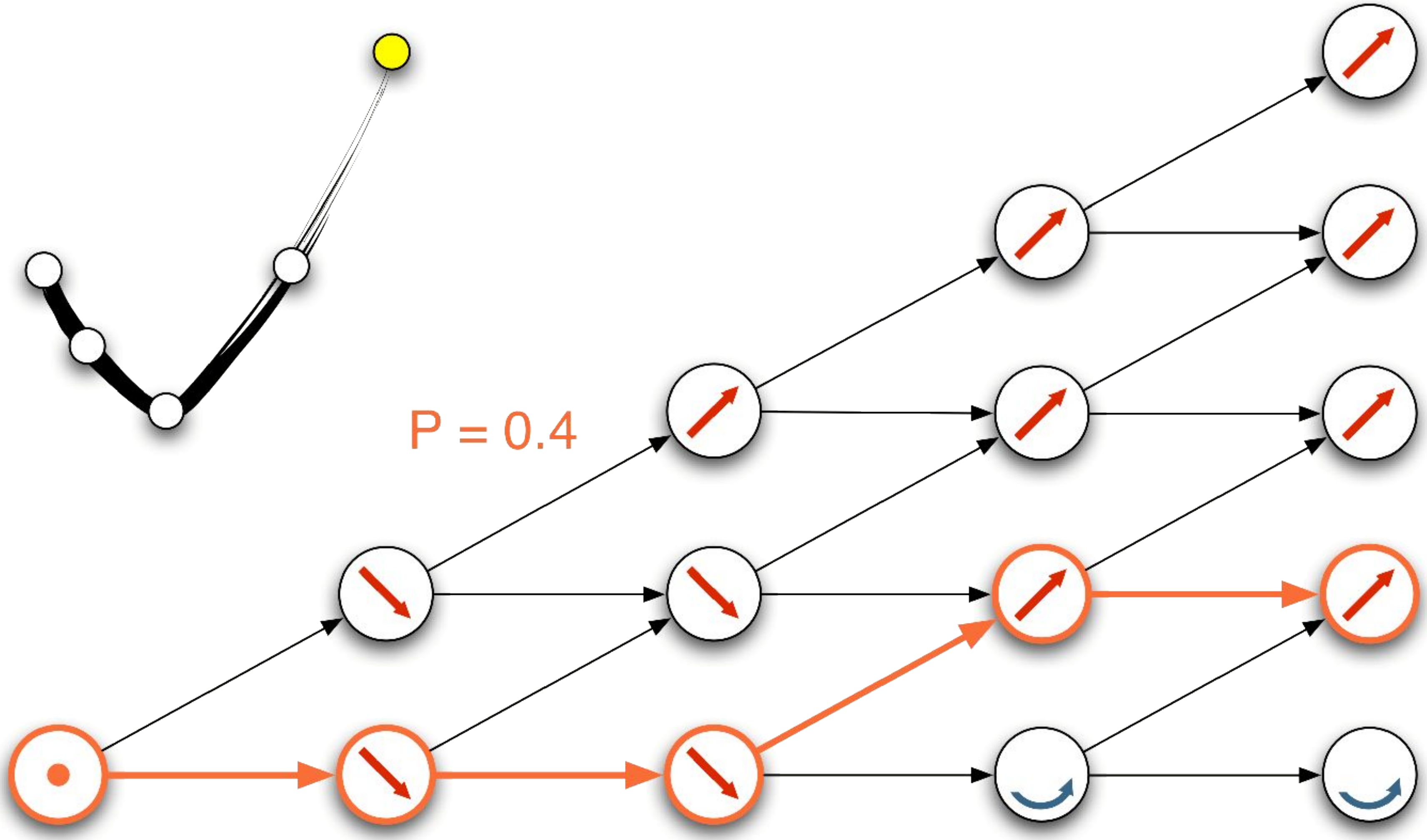

P = 0.4

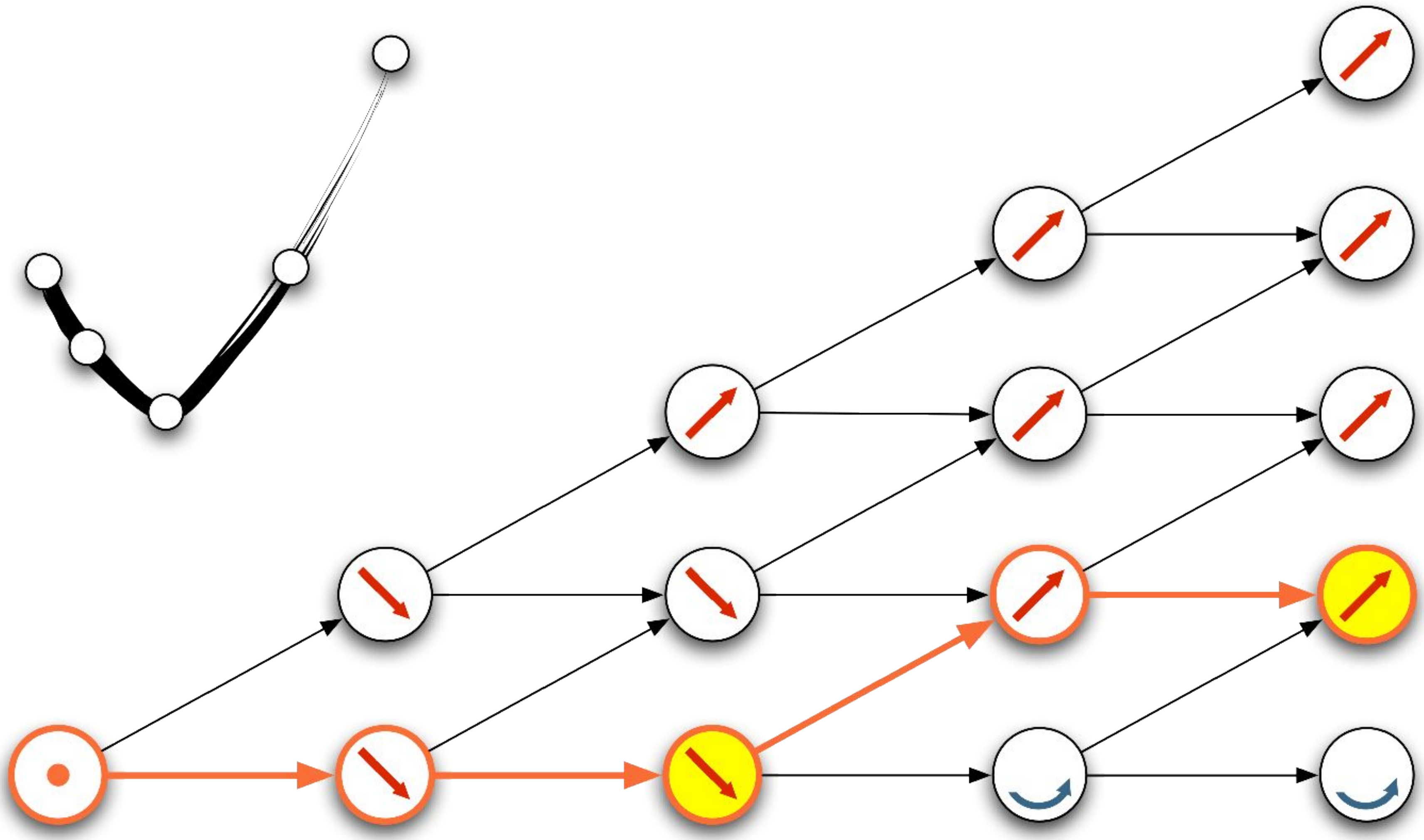

P = 0.1

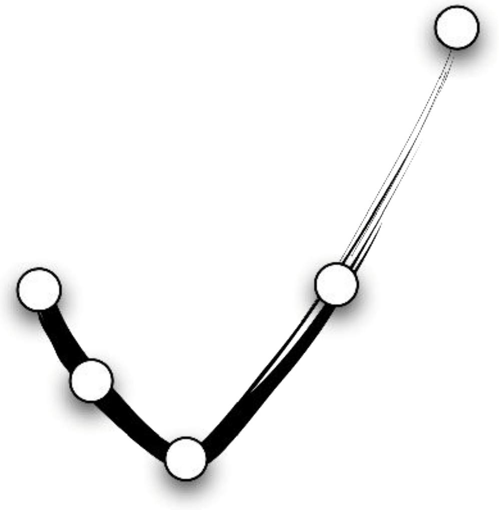

P = 0.2

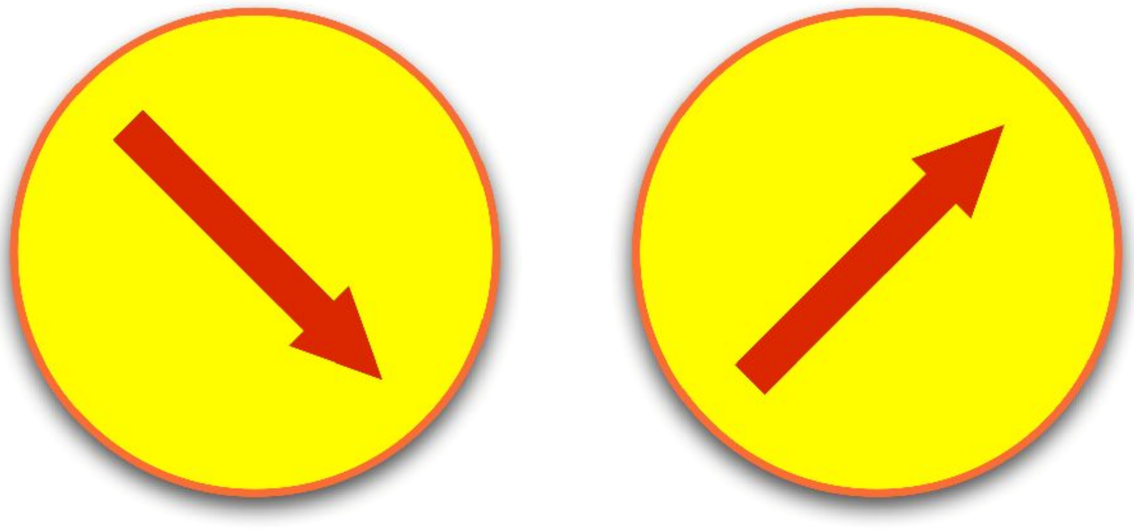

P = 0.1




P = 0.4







Viterbi Algorithm

Best path

Probabilities

$$a_{ij} = P(q_{t+1} = j \mid q_t = i), 1 \le i, j \le N$$

$$b_j(k) = P(o_t = \nu_k \mid q_t = j), 1 \le j \le N, 1 \le k \le M$$

$$P = \prod_{t=1}^{N} a_{ij} b_j$$

$$a_{ij} = P(q_{t+1} = j \mid q_t = i)$$

Probability of State Transition

$$a_{ij} = P(q_{t+1} = j \,|\, q_t = i)$$
$$b_j(k) = P(o_t = \nu_k \,|\, q_t = j)$$

Probability of Seeing Input in State

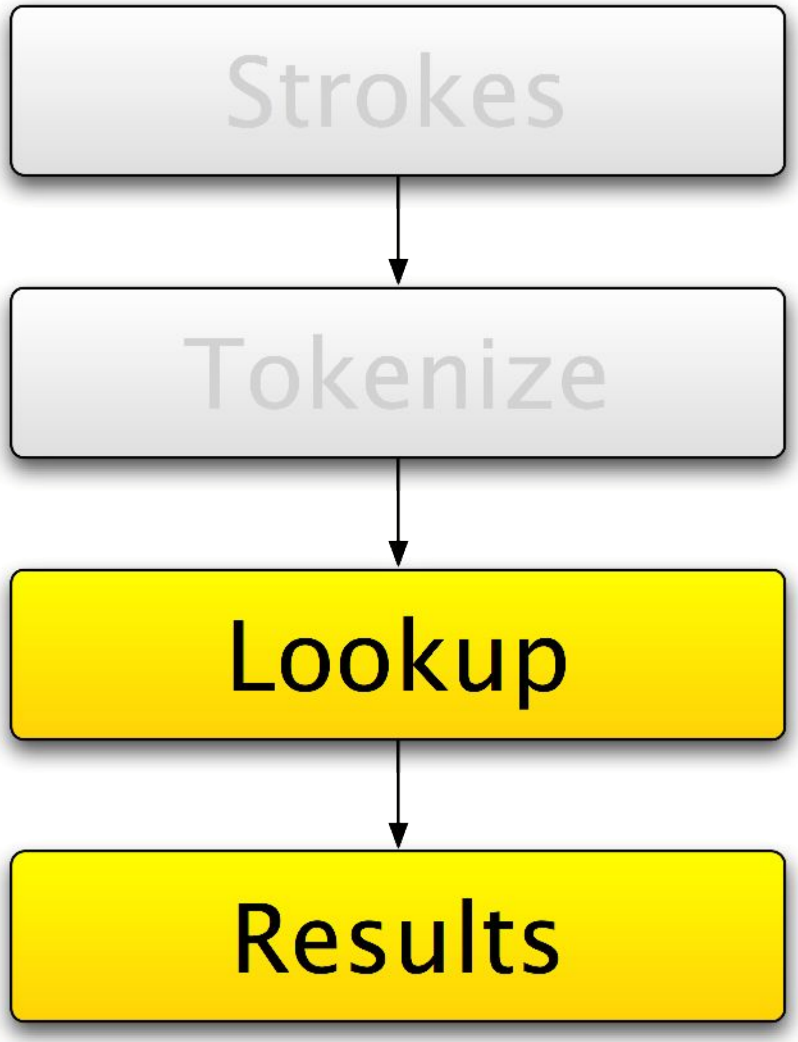
$$P = \prod_{t=1}^{N} a_{ij} b_j$$

Total Probability Along Path

$$\prod_{i=1}^{n} P(i) \rightarrow \sum_{i=1}^{n} \log P(i)$$

Log Probabilities

Avoid tiny floating point numbers

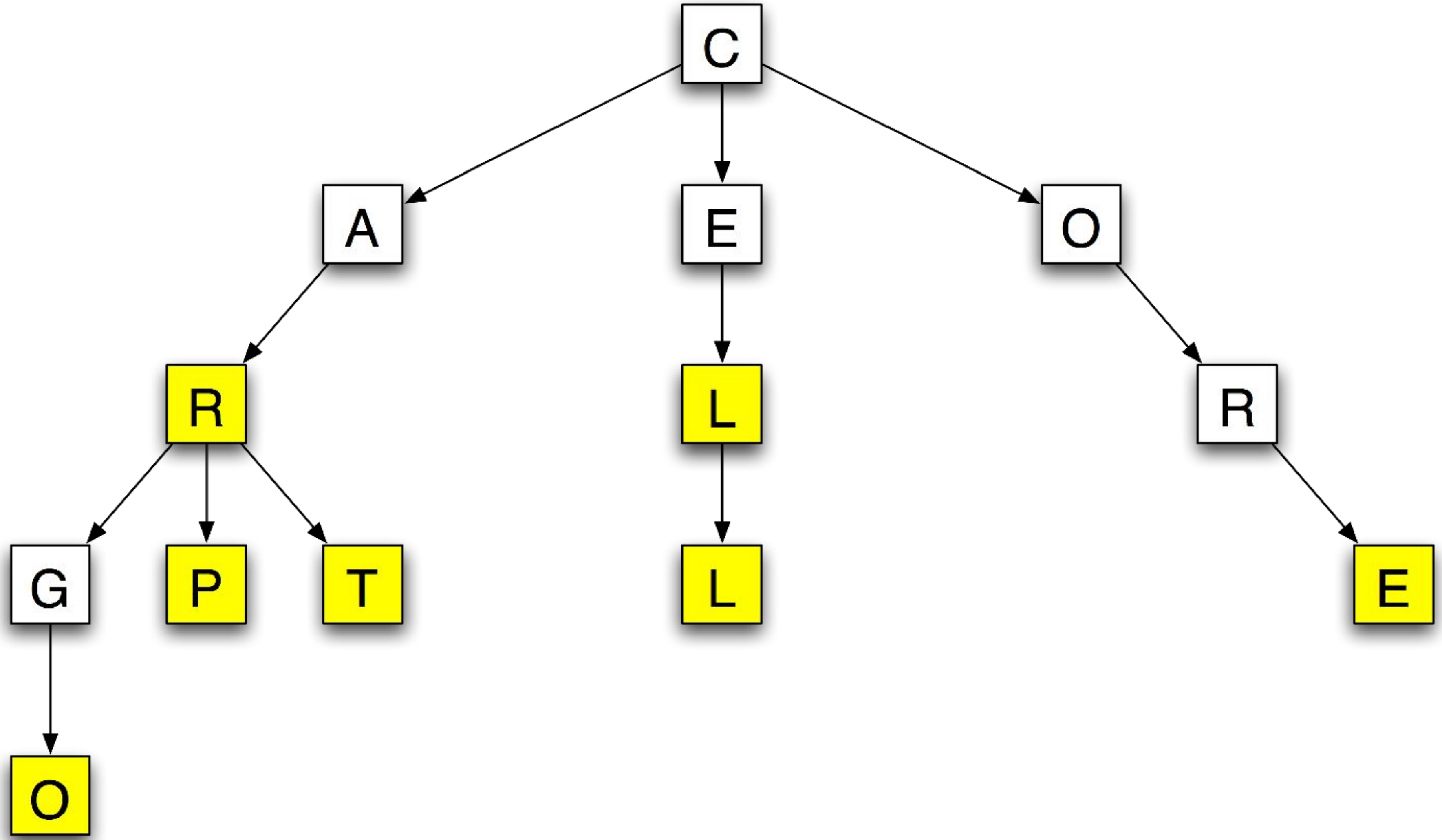

Strokes
Tokenize
Lookup
Results

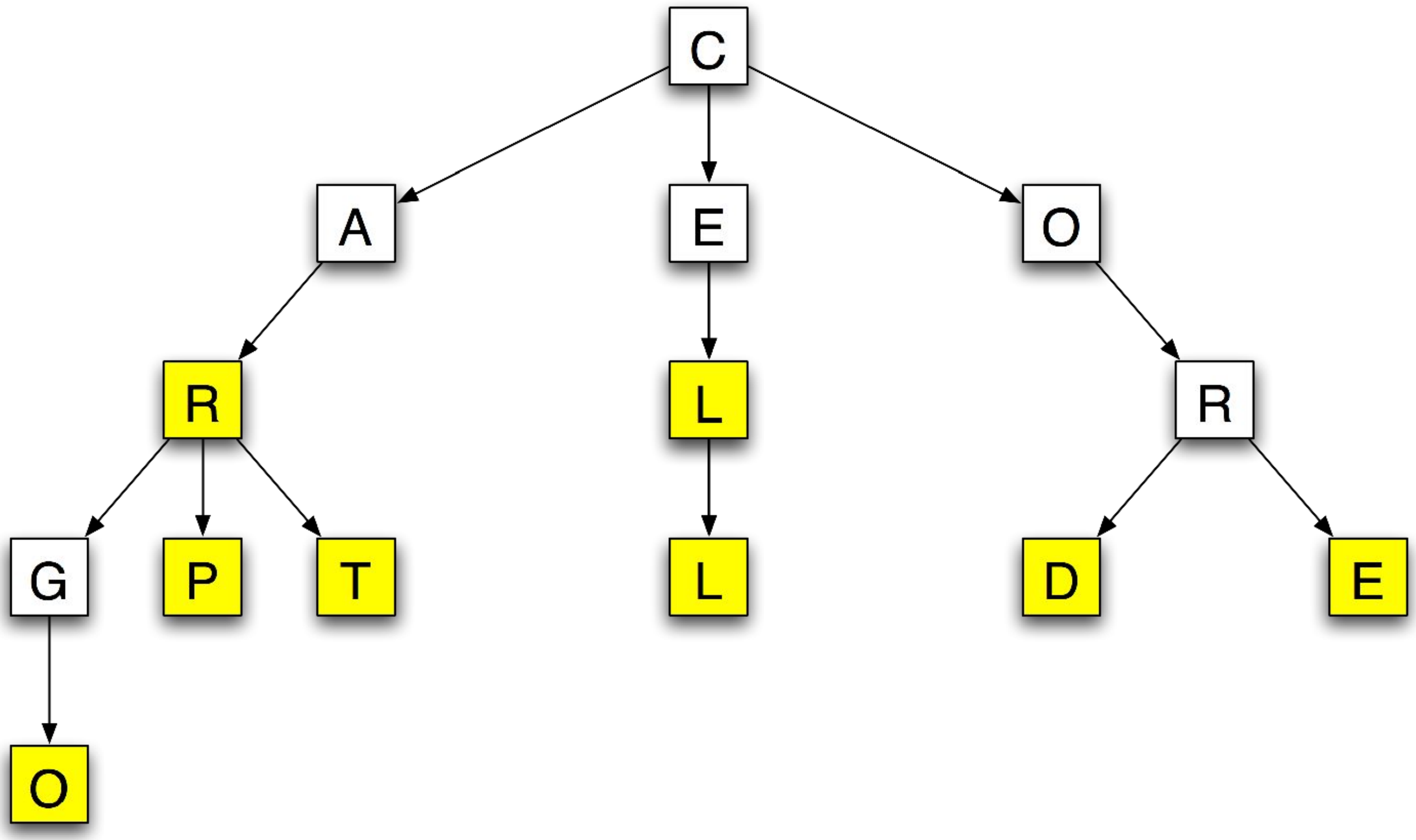
Database

Requirements

- String of tokens
- Hundreds of thousands of entries
- Fast lookup
- Near matches
- **Spell checker?**

Trie


C
A
E
O
R
L
R
G
P
T
L
E
O


C
A
E
O
R
L
R
G
P
T
L
D
E
O

Exact Lookup

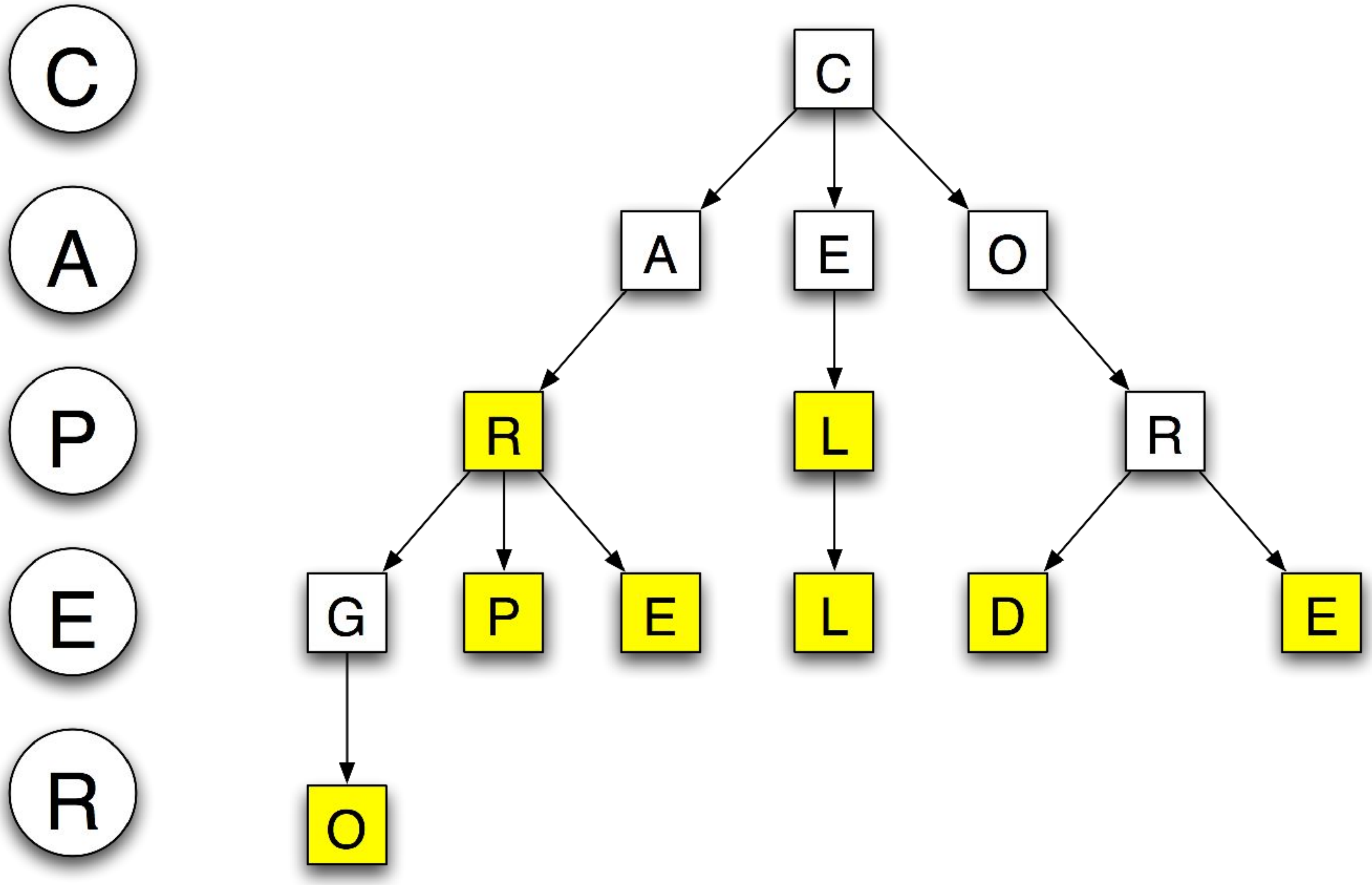
Linear time

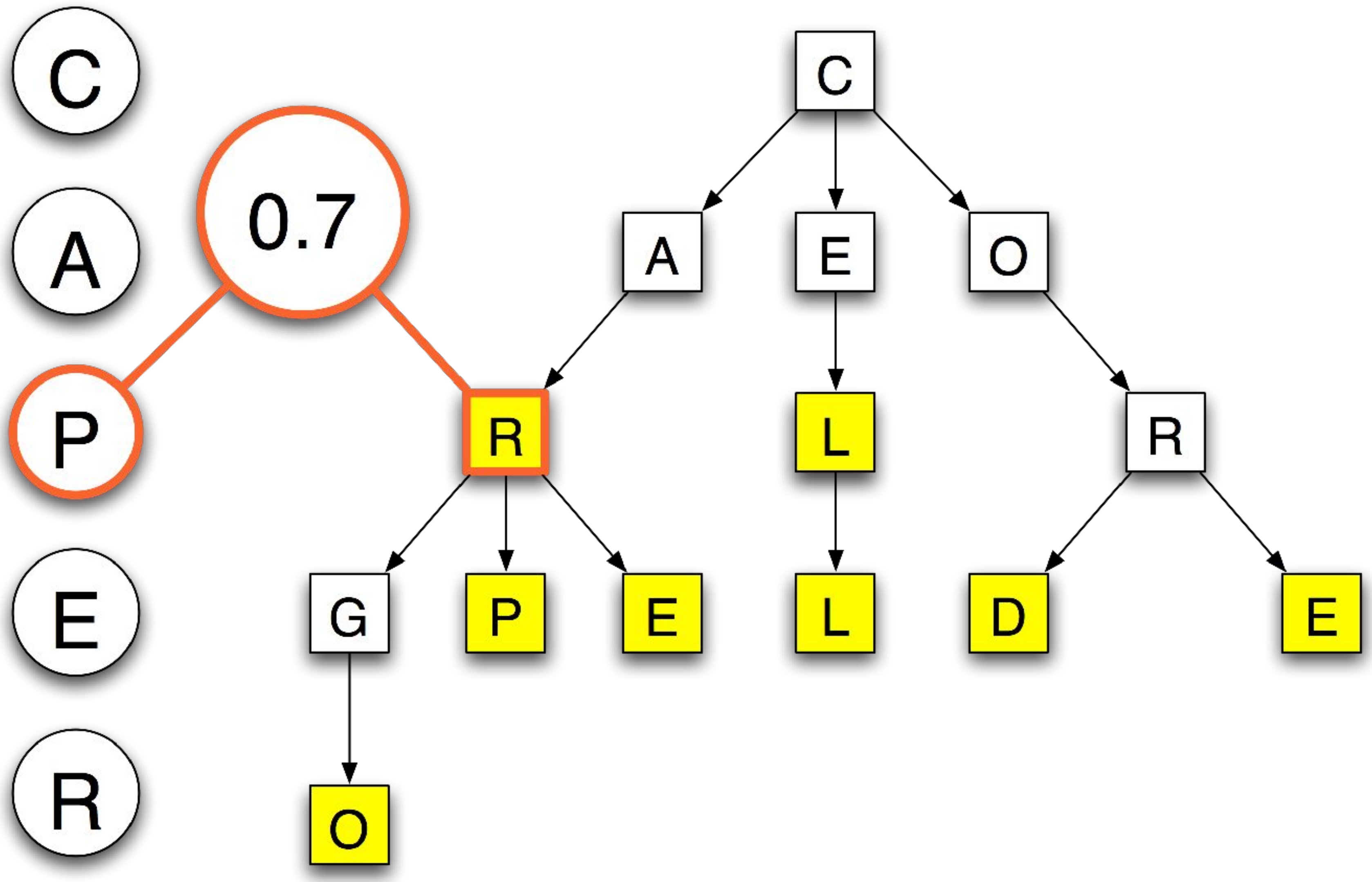
Near Lookup?

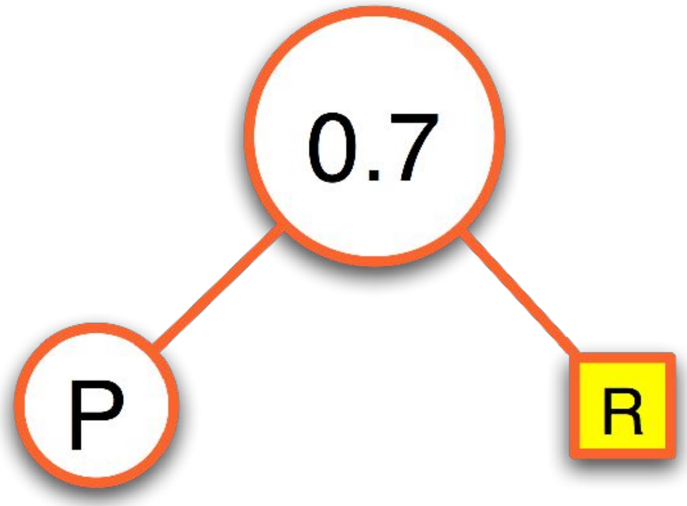
Linear time (almost)

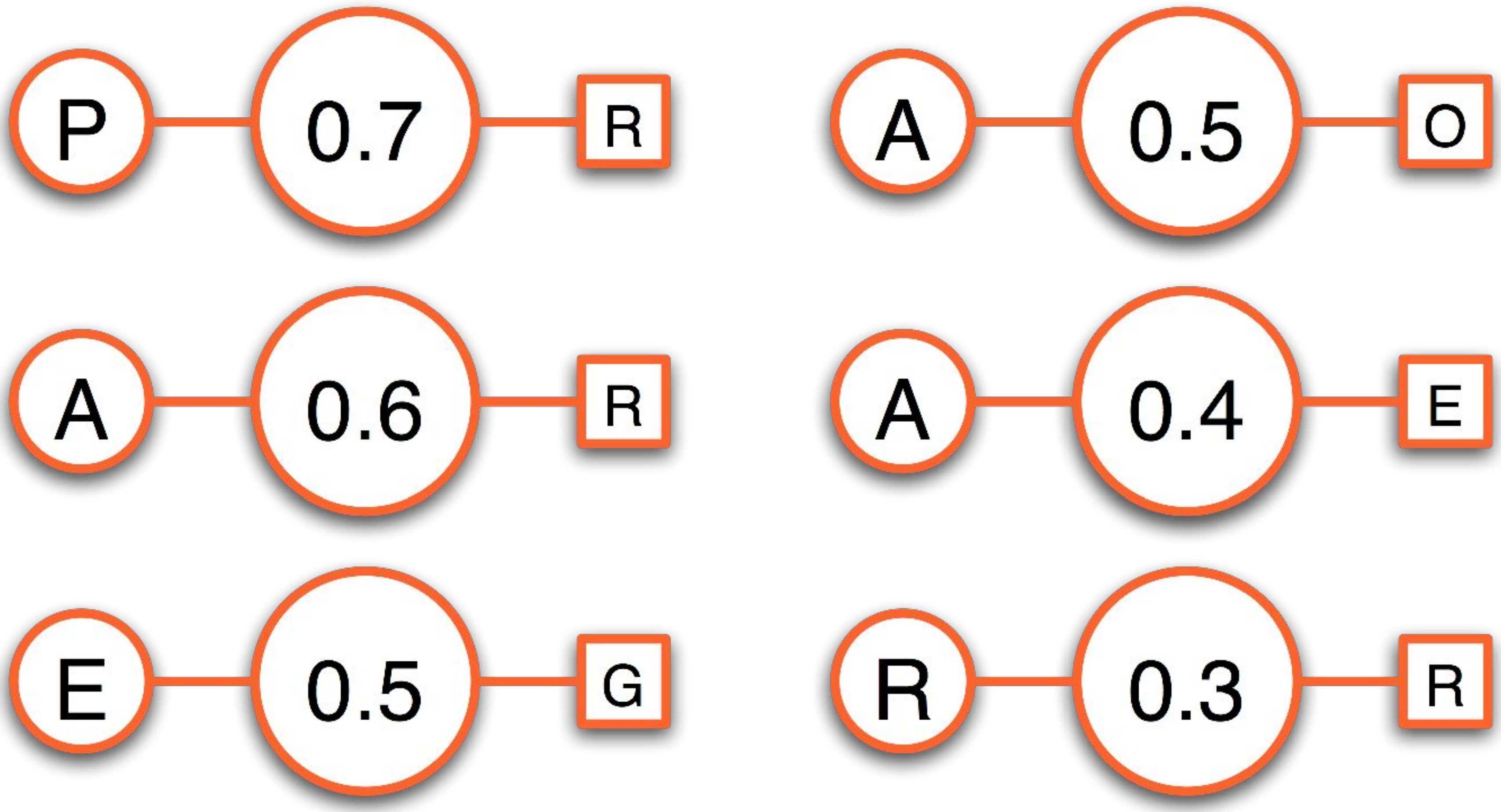
Priority Queue

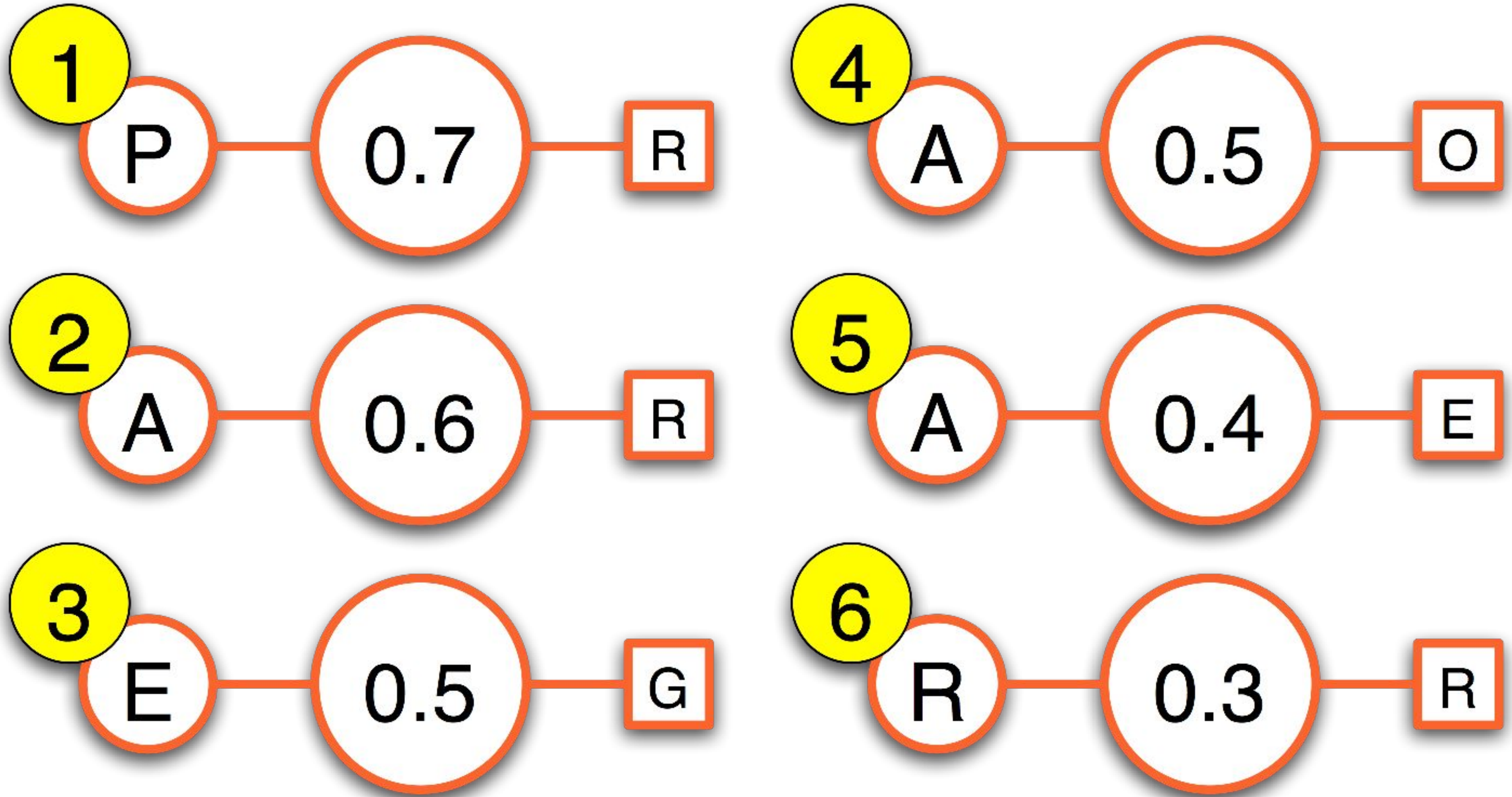
Store search state

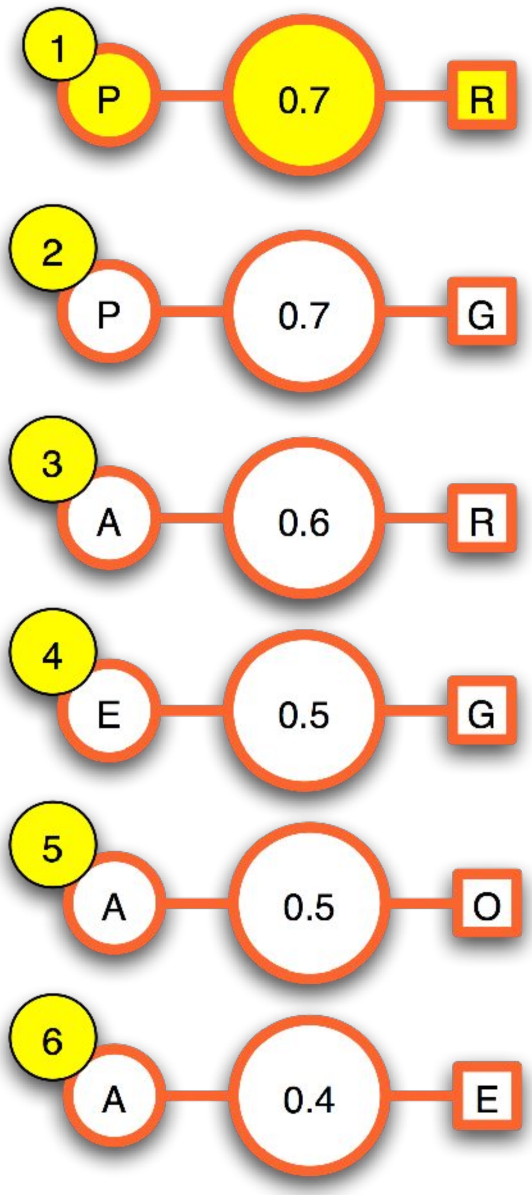

C
A
P
E
R
C
A
E
O
R
L
R
G
P
E
L
D
E
O

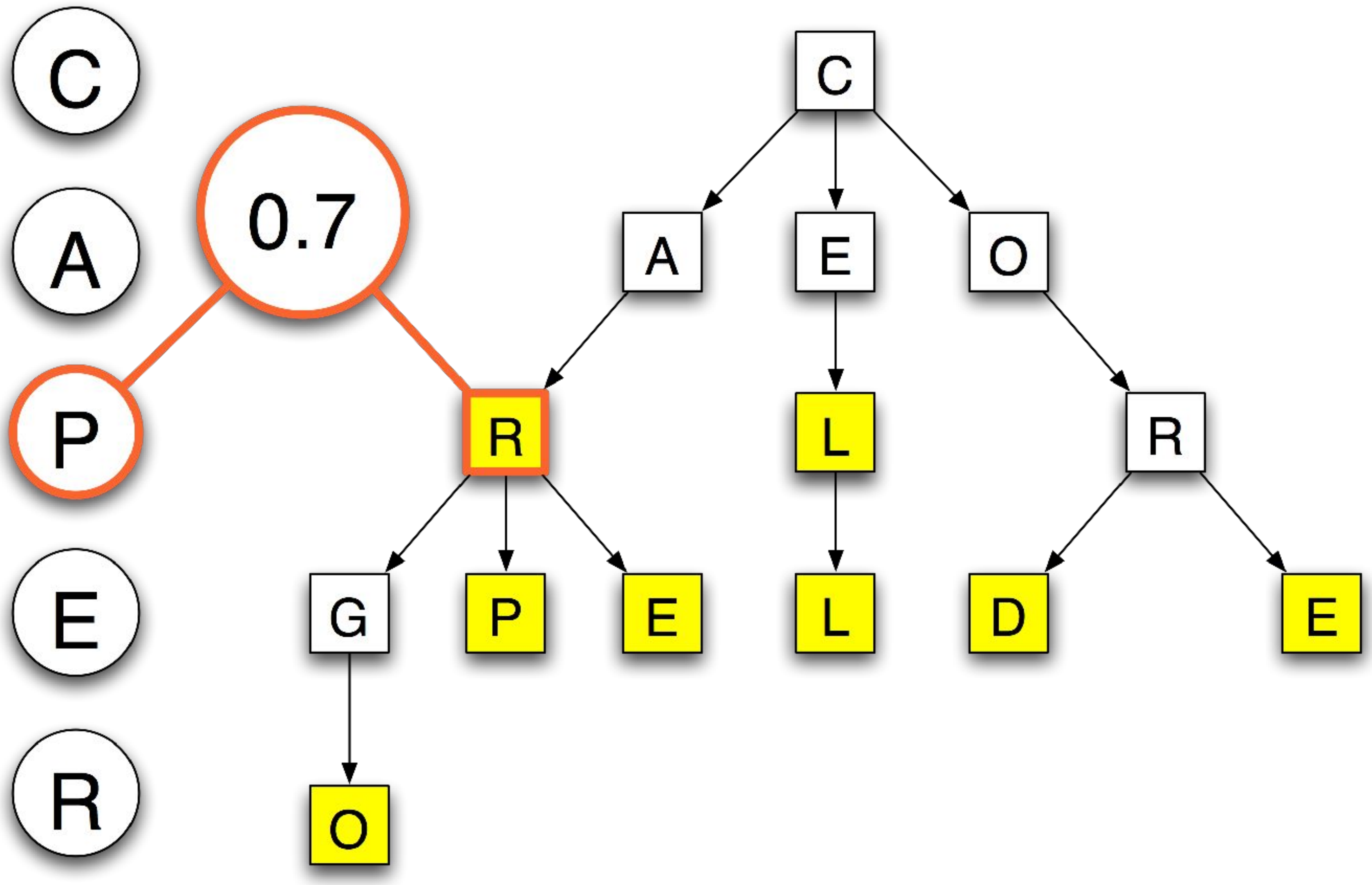

C
A
P
E
R
0.7
C
A
E
O
R
L
R
G
P
E
L
D
E
O

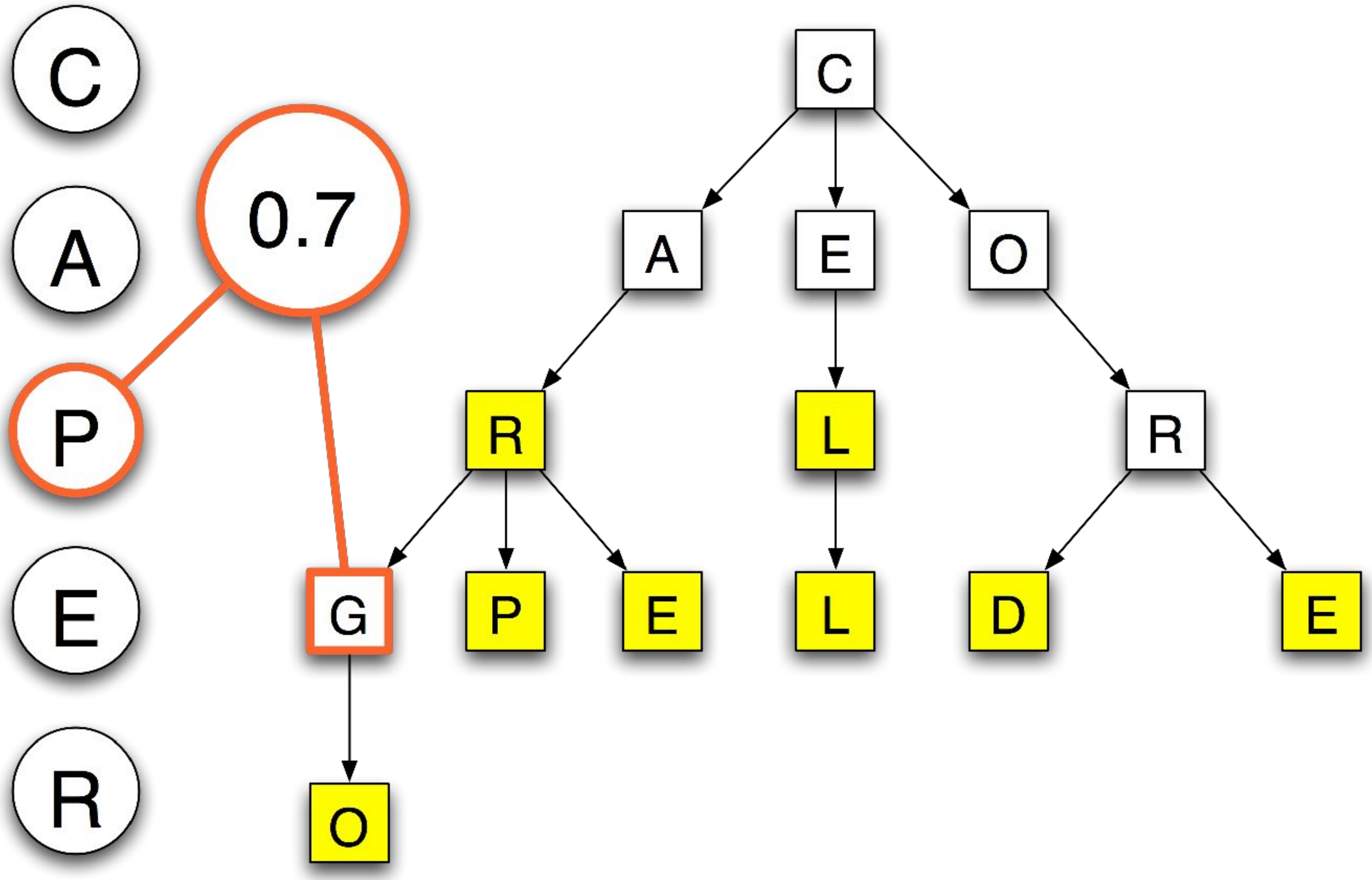

0.7
P
R

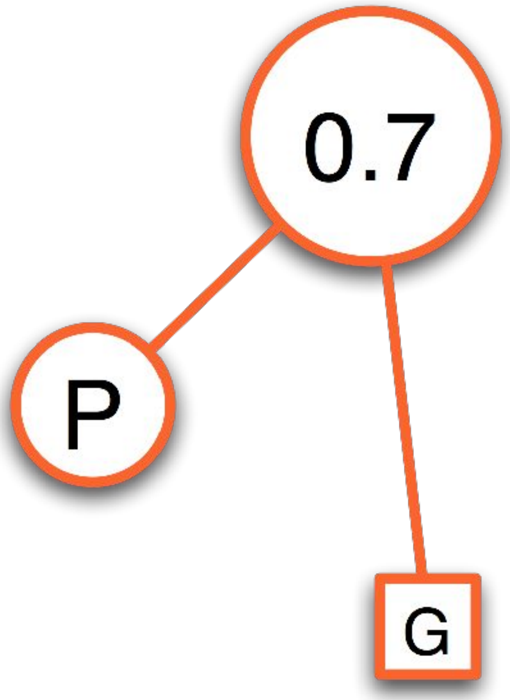

P
0.7
R
A
0.5
O
A
0.6
R
A
0.4
E
E
0.5
G
R
0.3
R

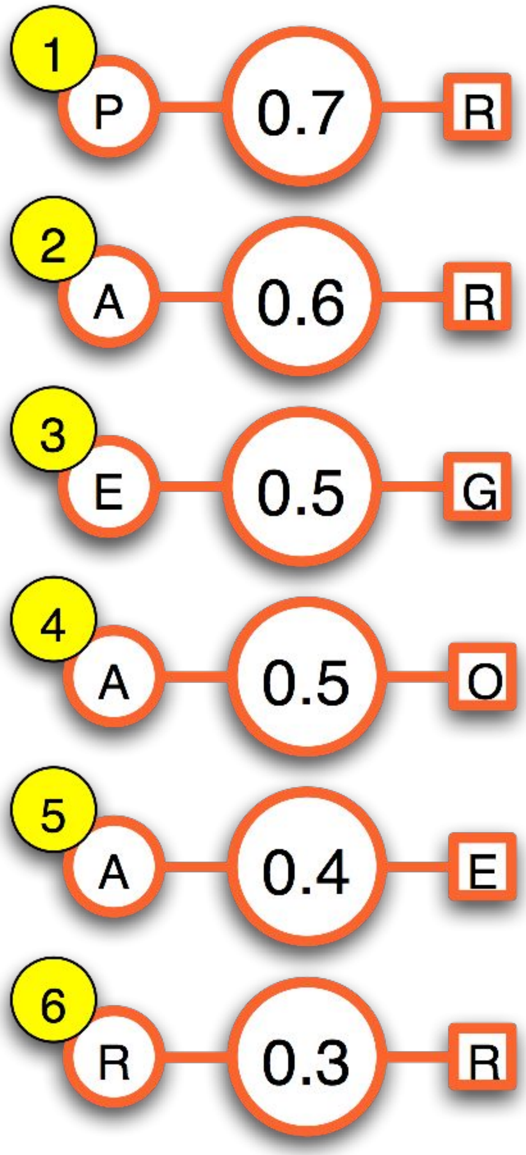

1
P
0.7
R
4
A
0.5
O
2
A
0.6
R
5
A
0.4
E
3
E
0.5
G
6
R
0.3
R

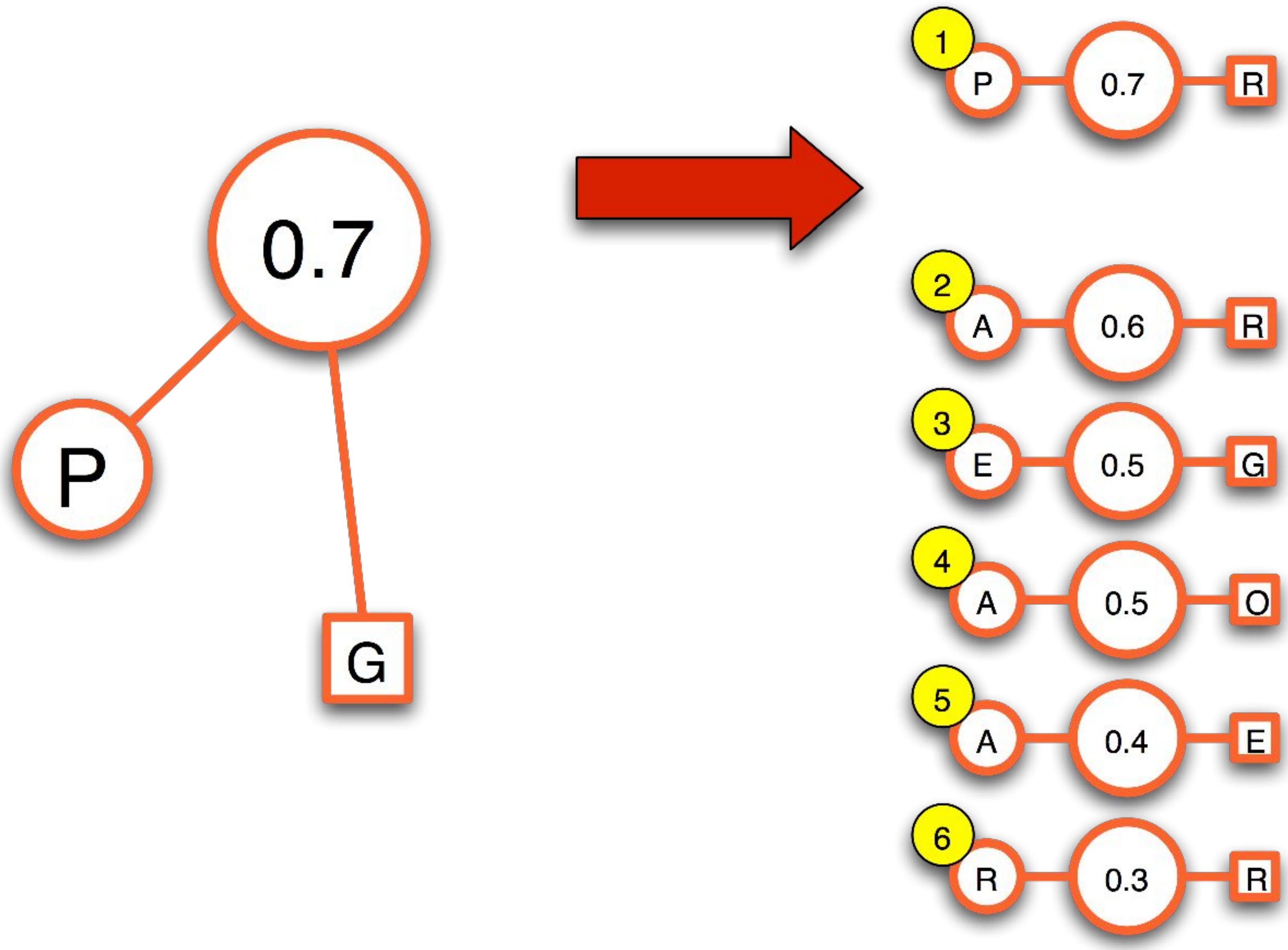

1
P
0.7
R
2
P
0.7
G
3
A
0.6
R
4
E
0.5
G
5
A
0.5
O
6
A
0.4
E

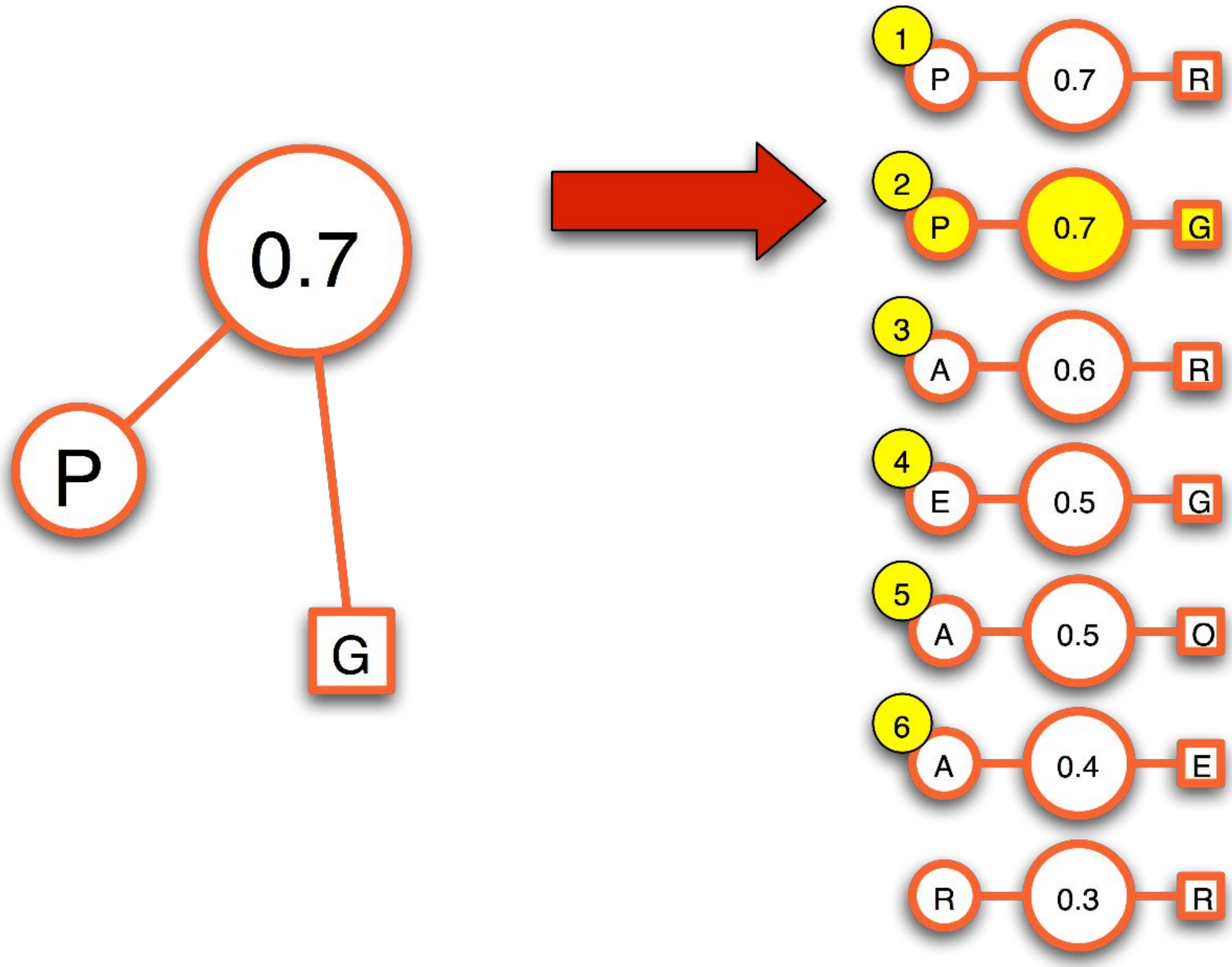

C
A
P
E
R
0.7
C
A
E
O
R
L
R
G
P
E
L
D
E
O

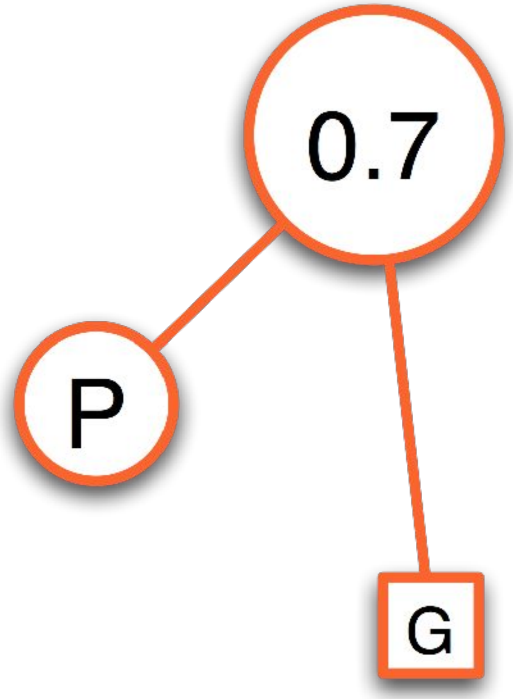

C
A
P
E
R
0.7
C
A
E
O
R
L
R
G
P
E
L
D
E
O


0.7
P
G

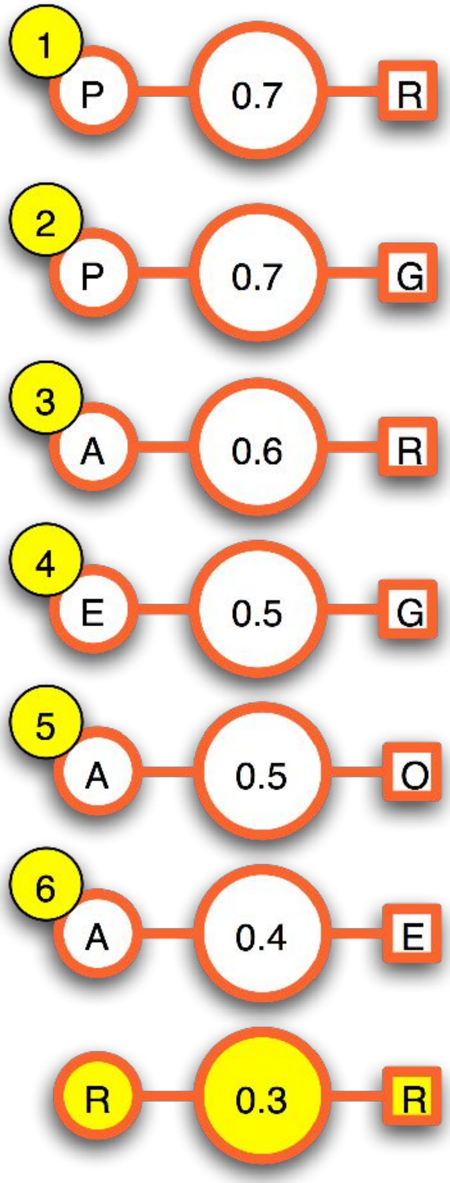

1
P
0.7
R
2
A
0.6
R
3
E
0.5
G
4
A
0.5
O
5
A
0.4
E
6
R
0.3
R


0.7
P
G
1
P
0.7
R
2
A
0.6
R
3
E
0.5
G
4
A
0.5
O
5
A
0.4
E
6
R
0.3
R

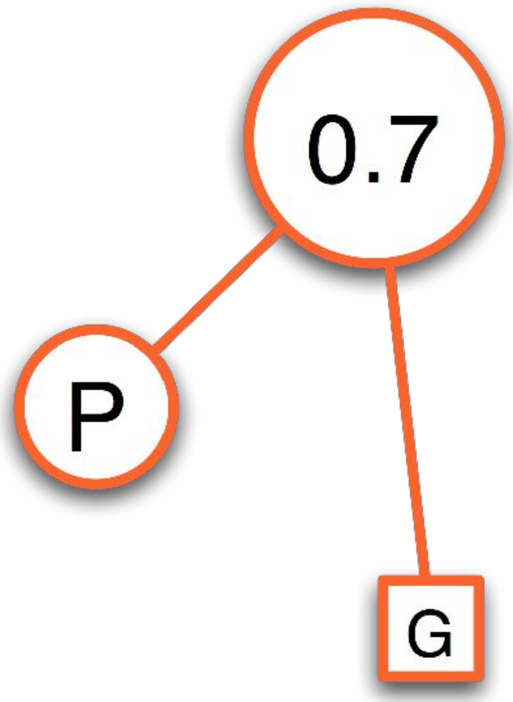

0.7
P
G
1
P
0.7
R
2
P
0.7
G
3
A
0.6
R
4
E
0.5
G
5
A
0.5
O
6
A
0.4
E
R
0.3
R

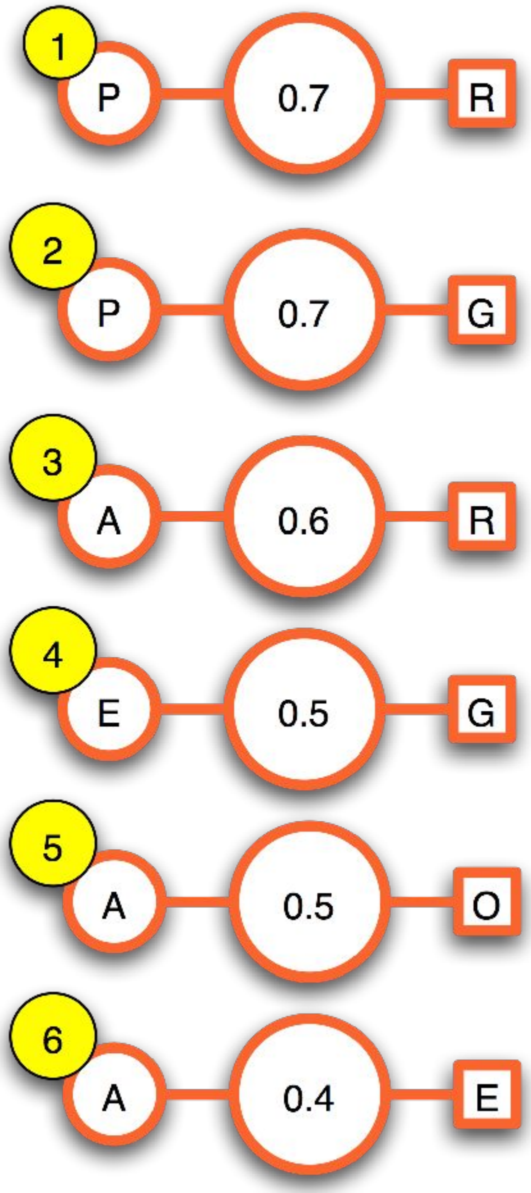


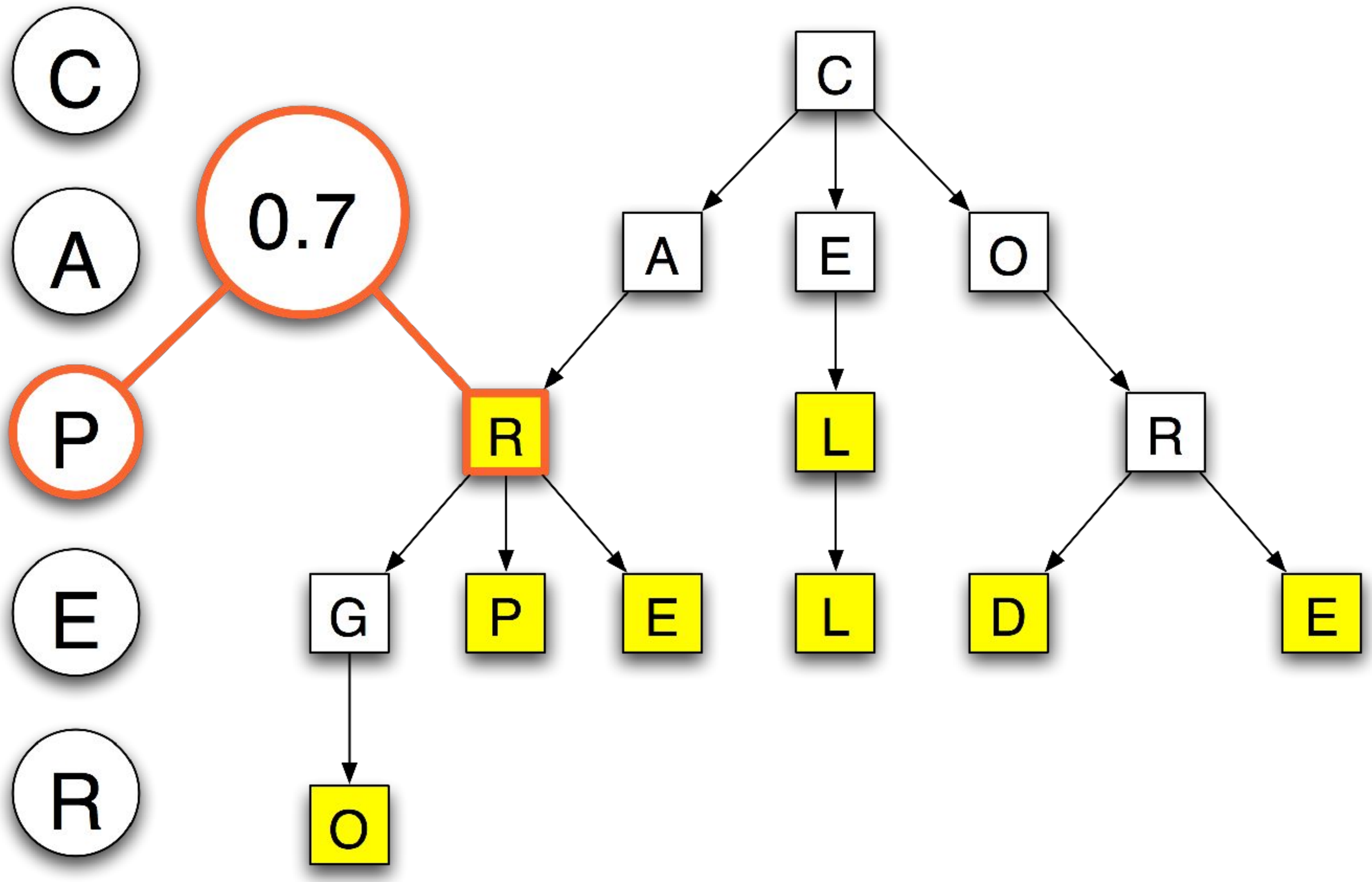
0.7
P
G

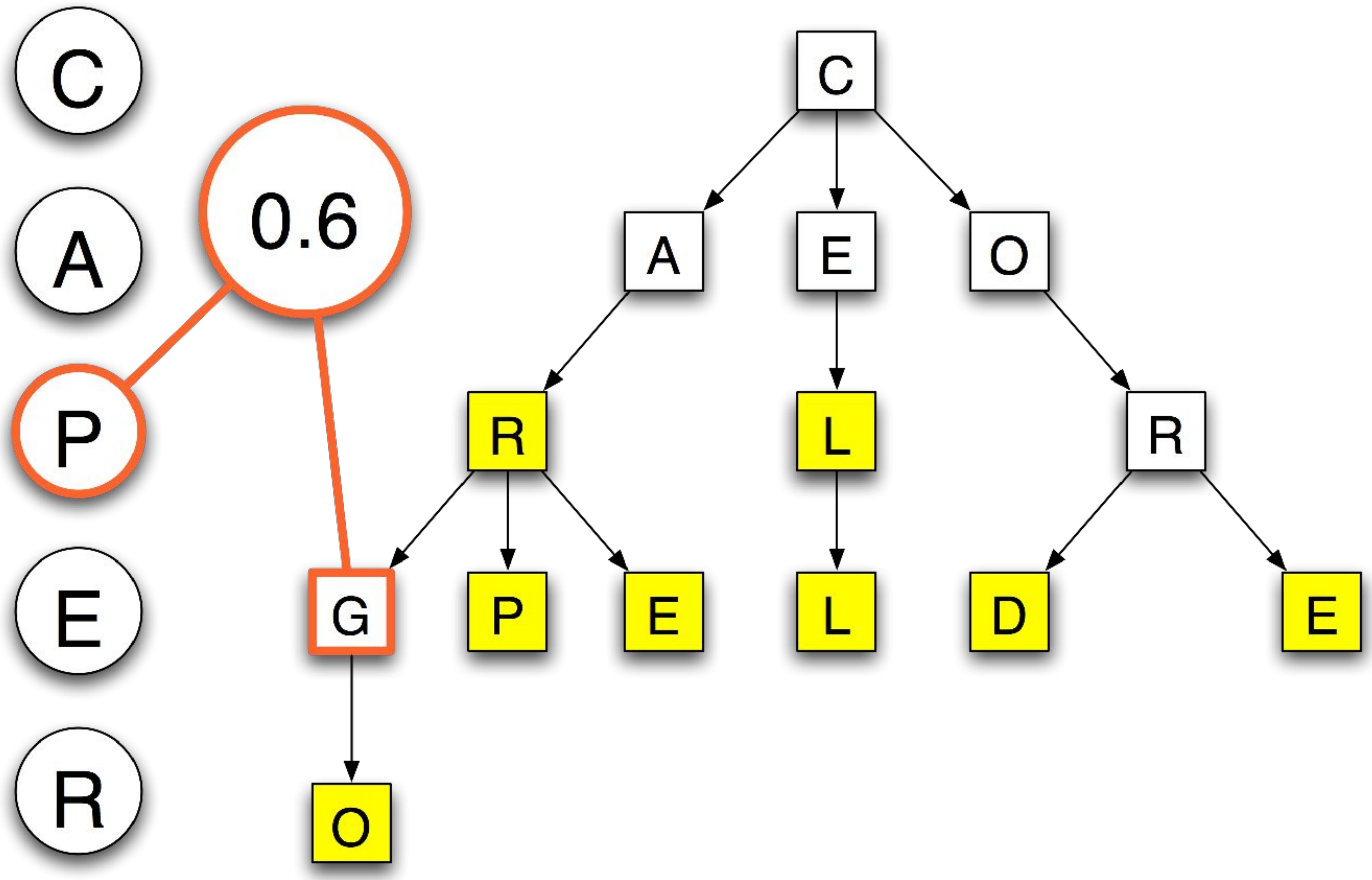


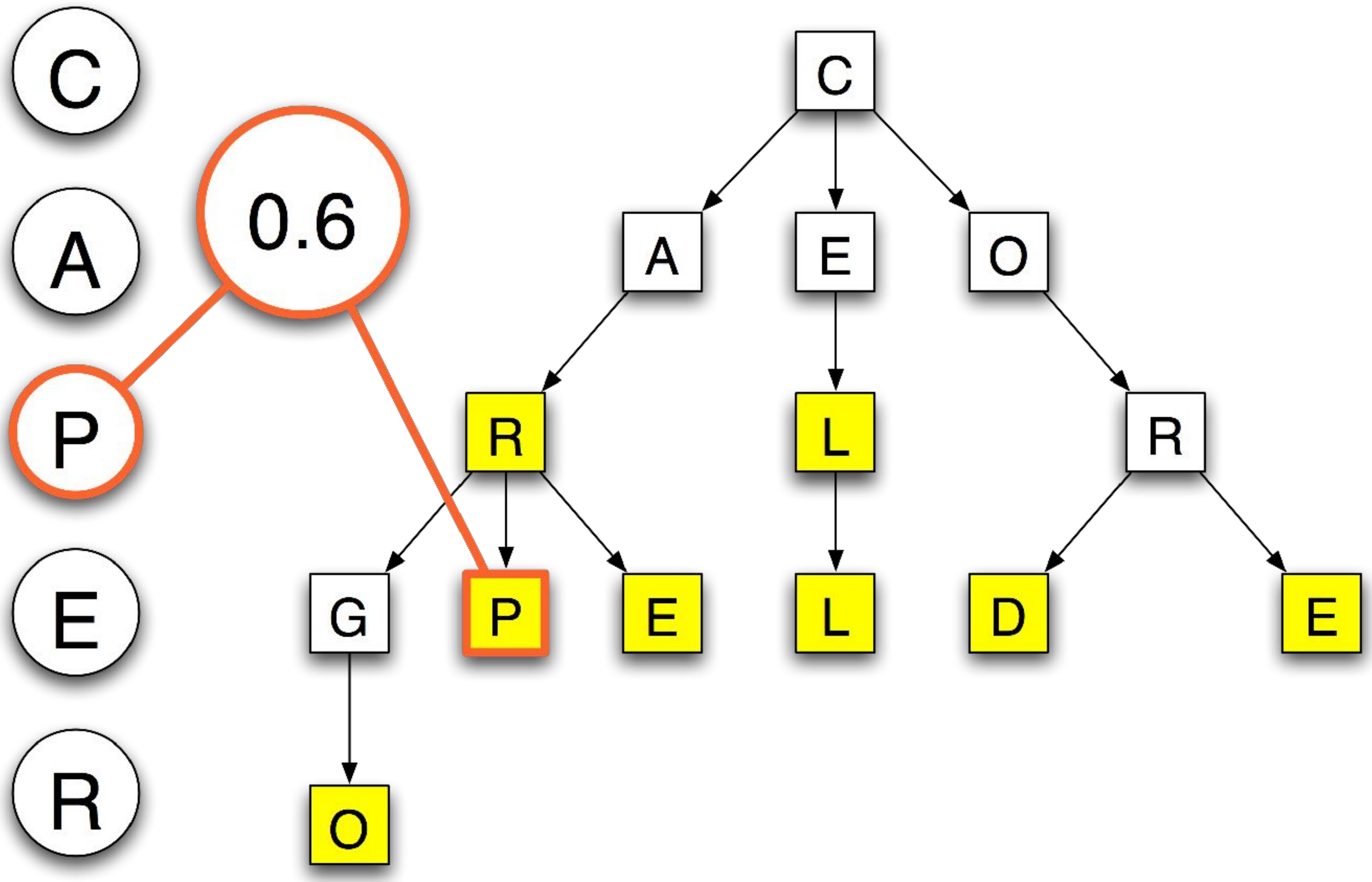
1
P
0.7
R
2
P
0.7
G
3
A
0.6
R
4
E
0.5
G
5
A
0.5
O
6
A
0.4
E
R
0.3
R

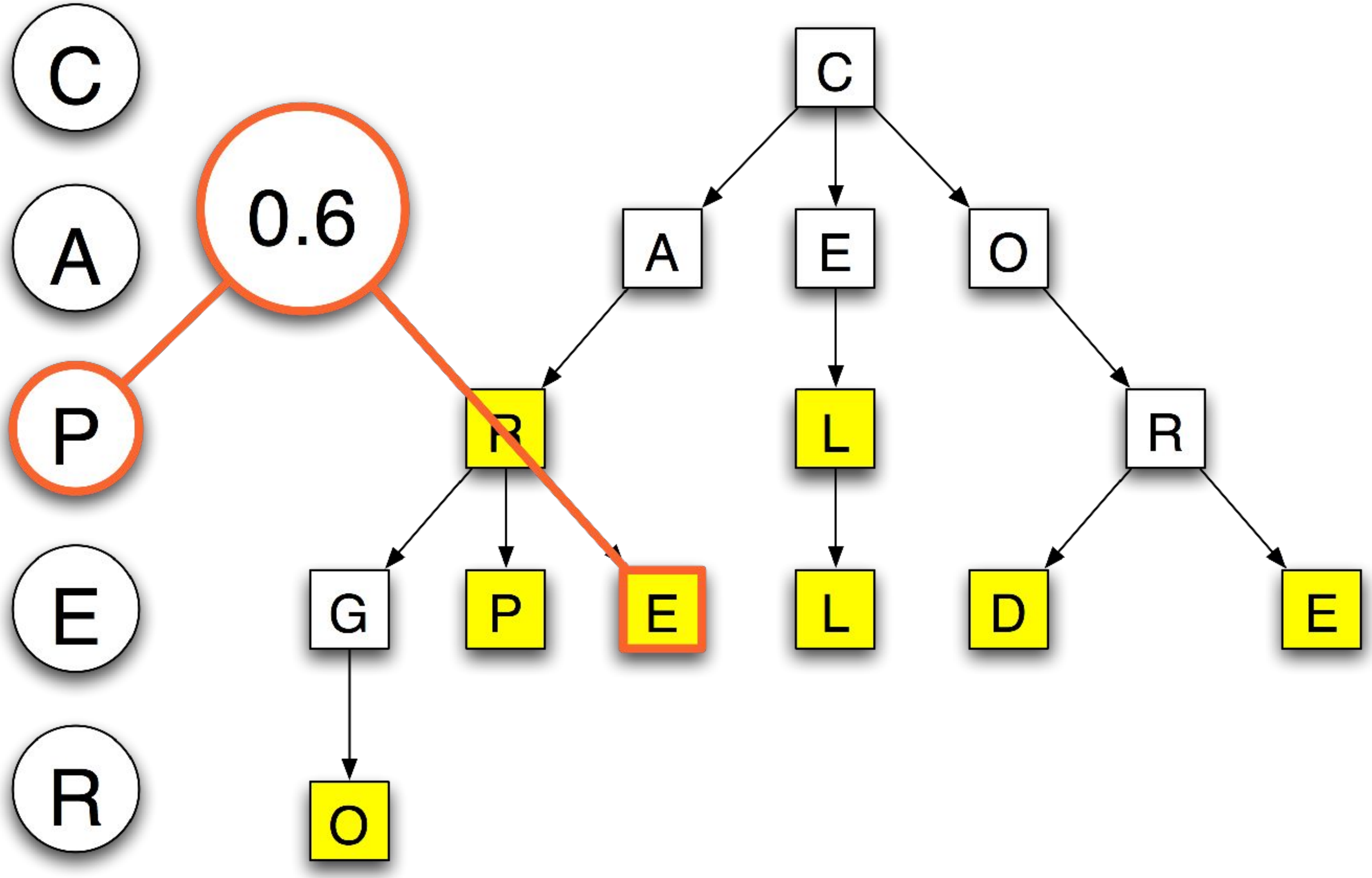

0.7
P
G

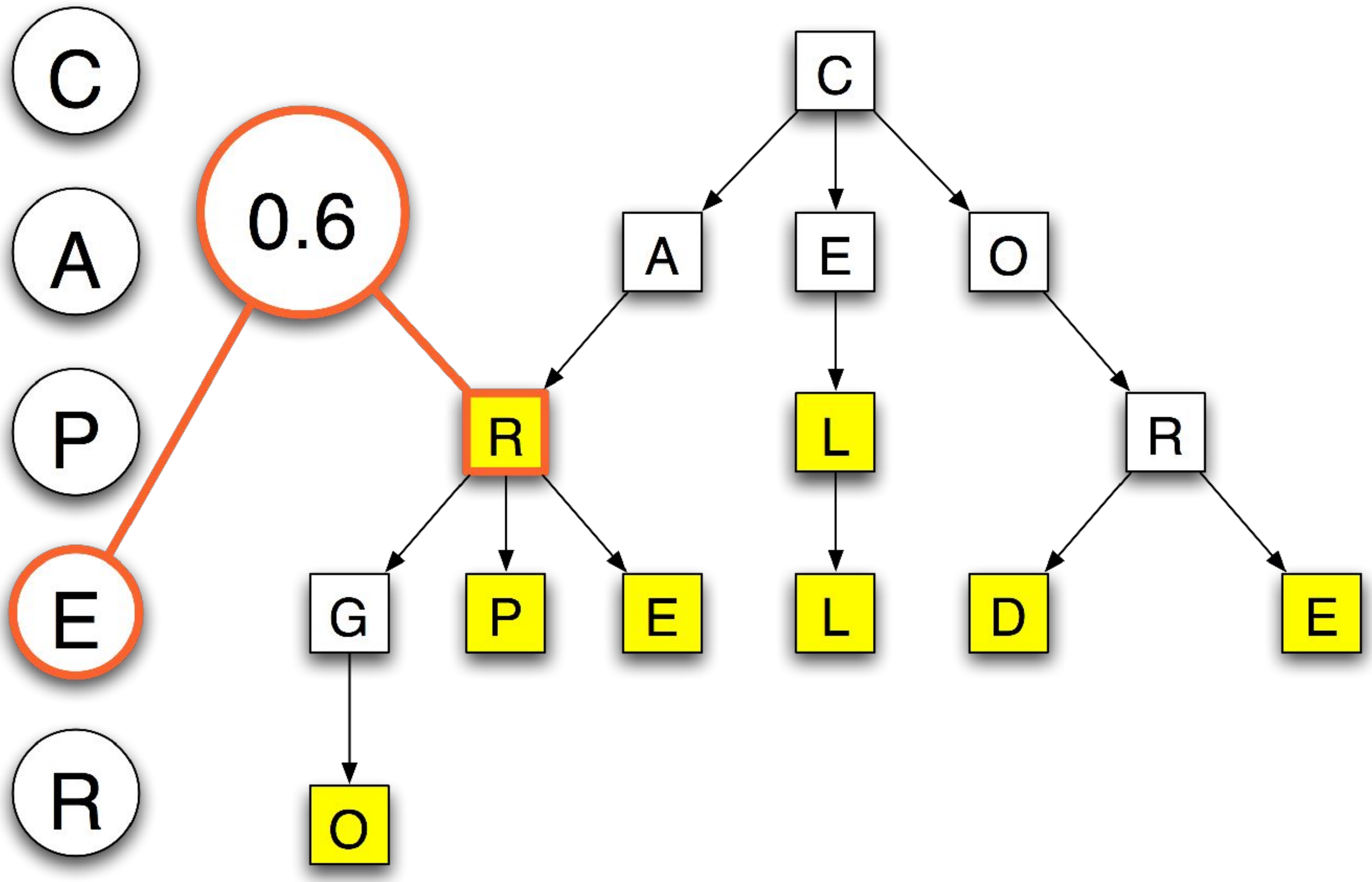

1
P
0.7
R
2
P
0.7
G
3
A
0.6
R
4
E
0.5
G
5
A
0.5
O
6
A
0.4
E

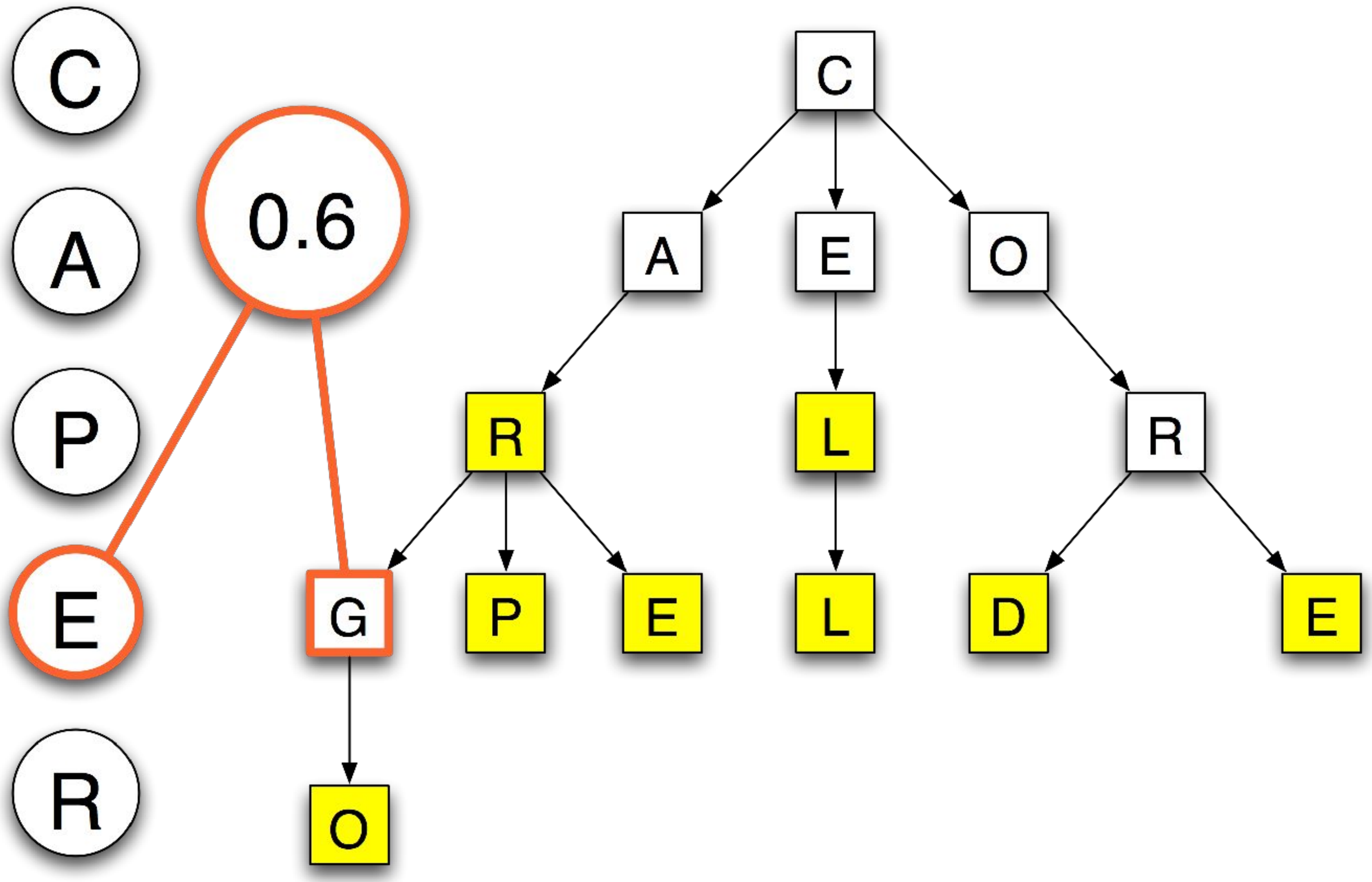

C
A
P
E
R
0.7
C
A
E
O
R
L
R
G
P
E
L
D
E
O

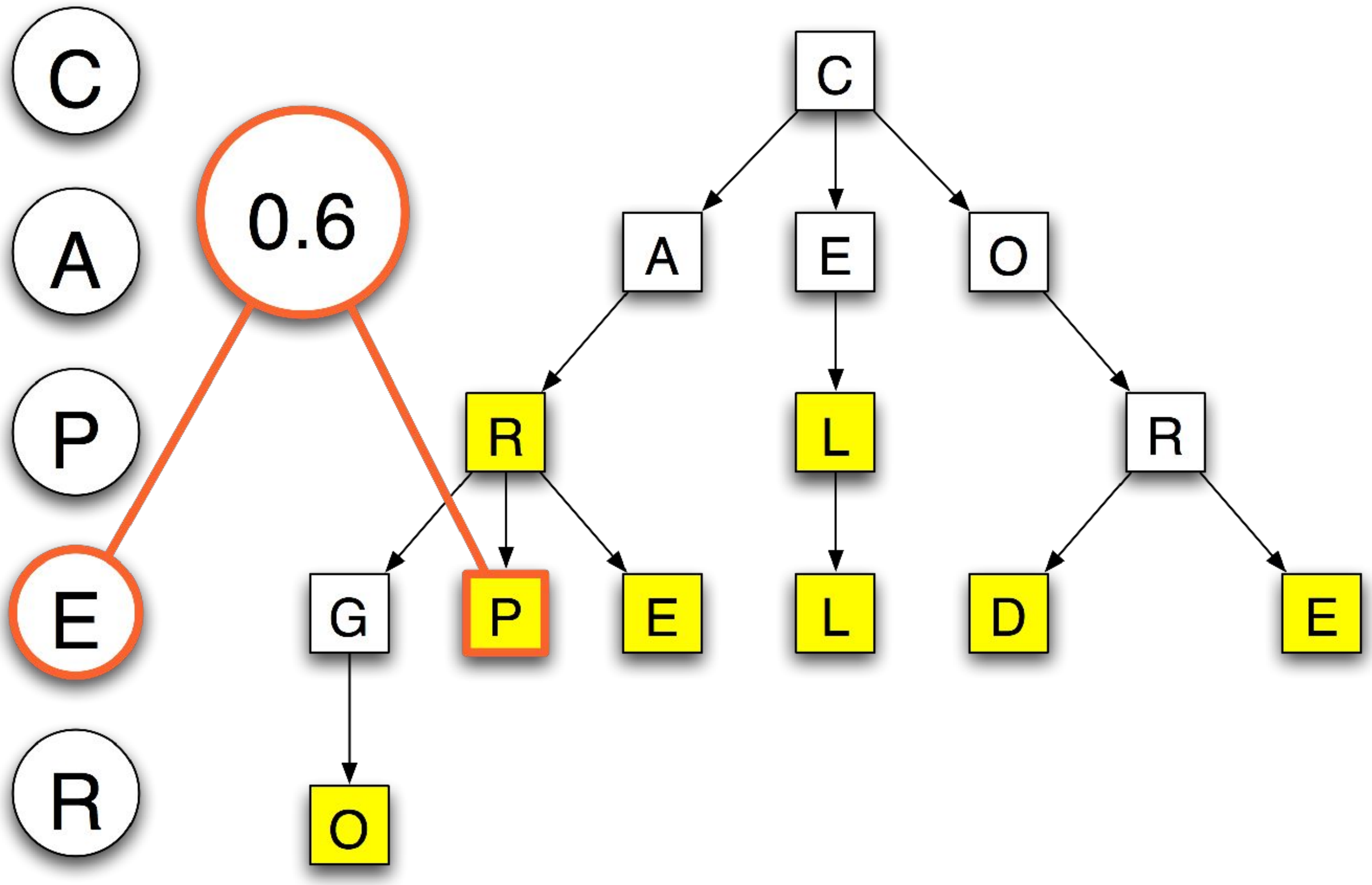

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

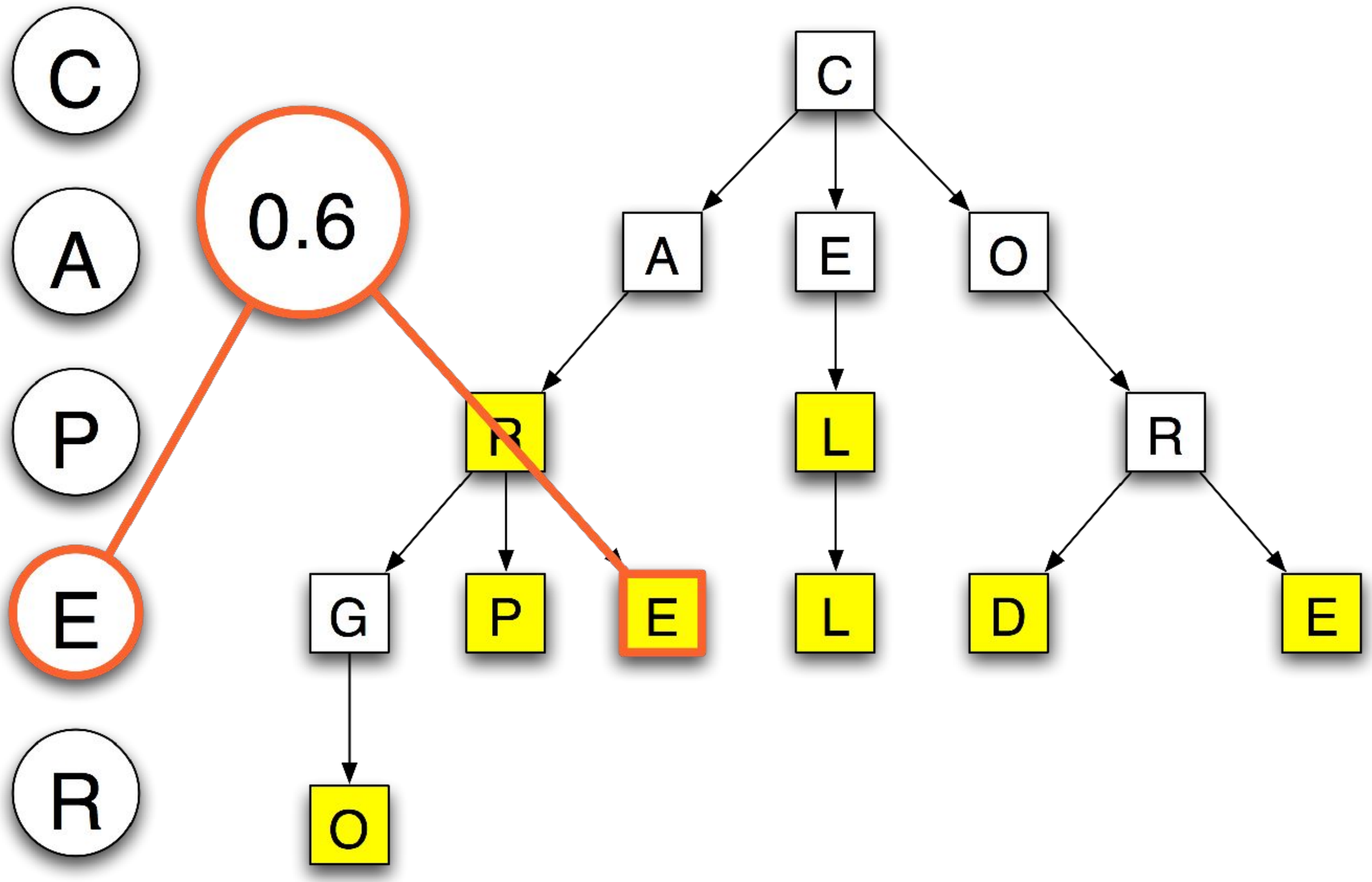

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

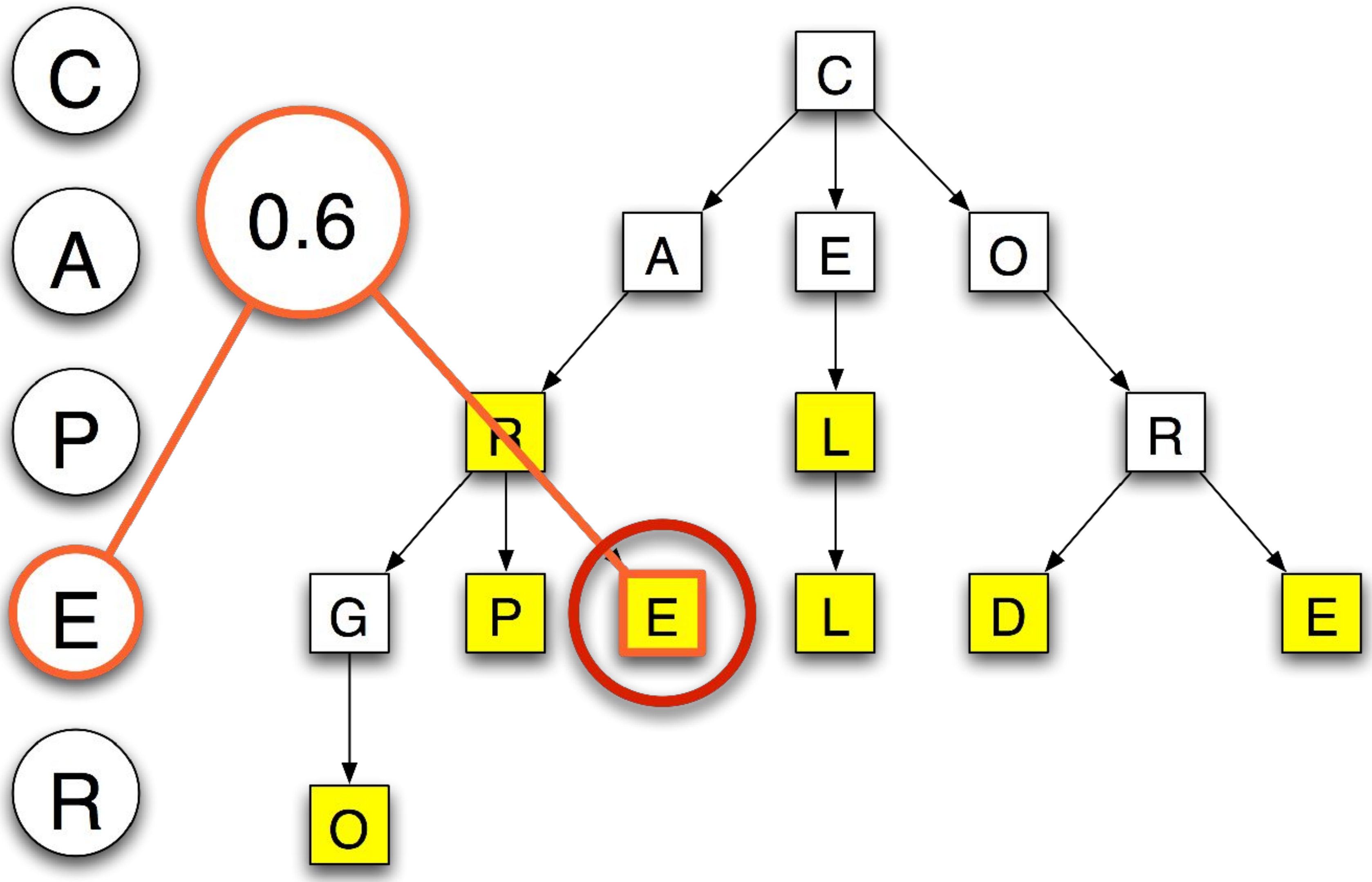

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

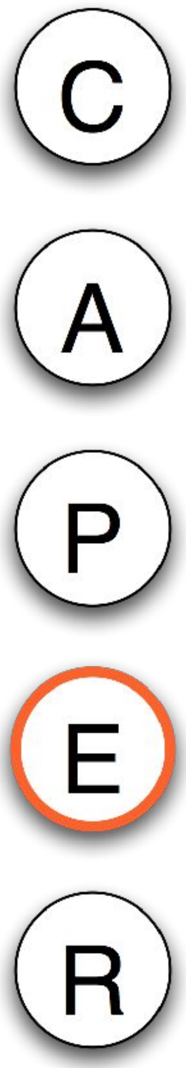

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

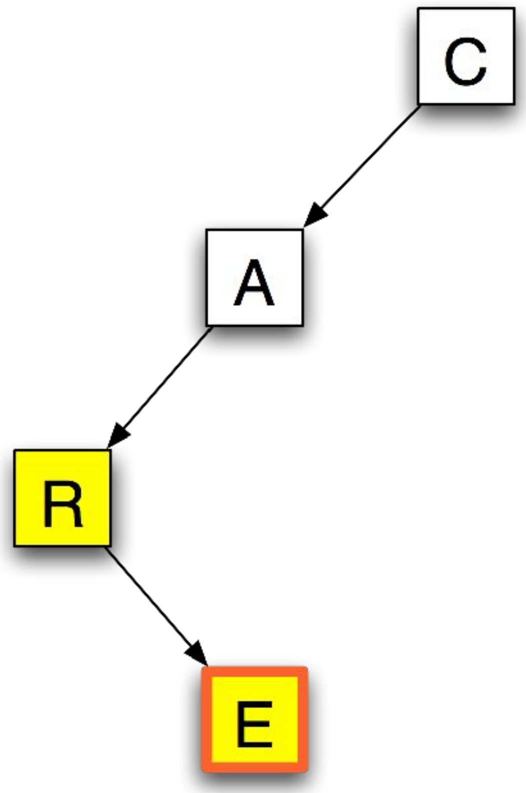

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

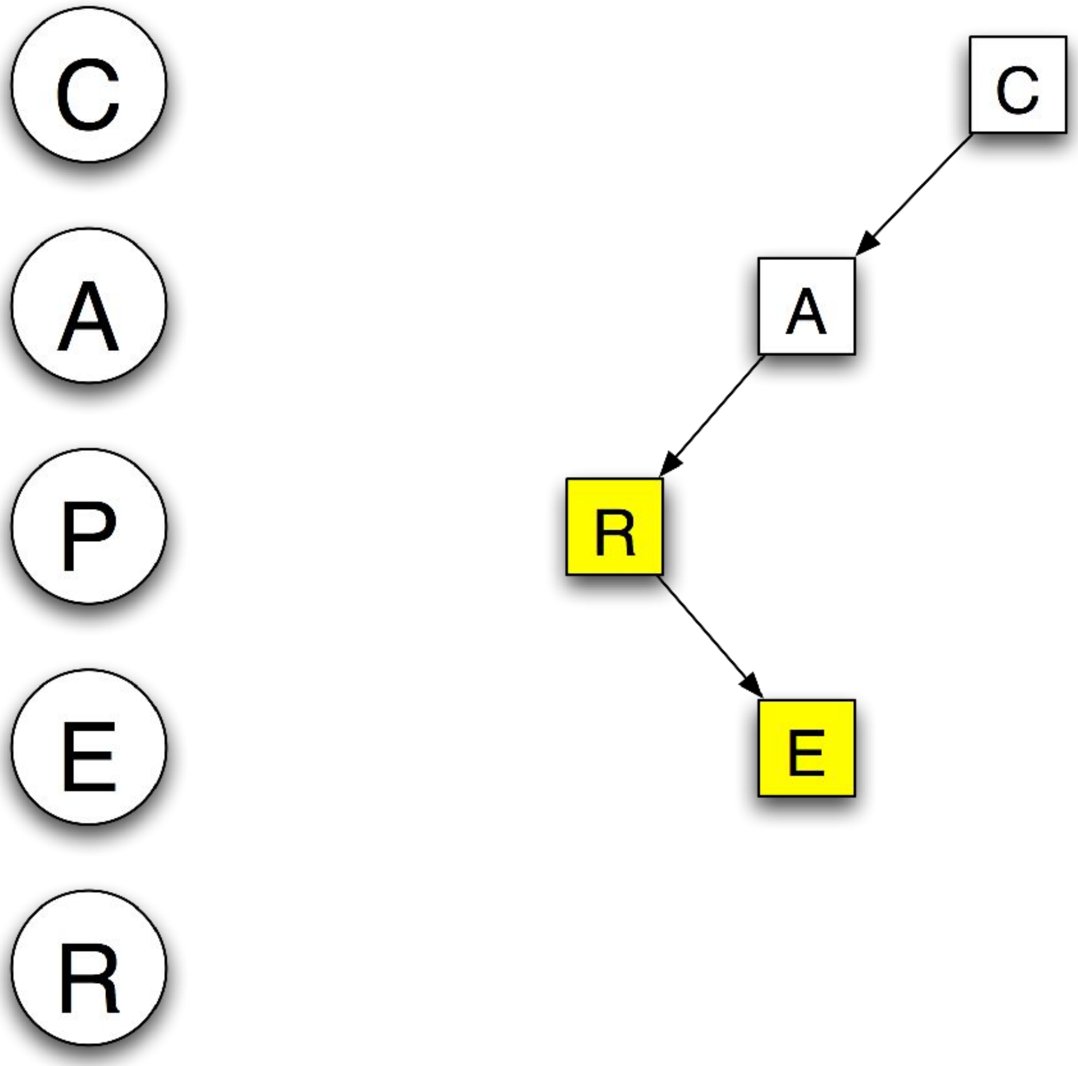

C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O


C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O


C
A
P
E
R
0.6
C
A
E
O
R
L
R
G
P
E
L
D
E
O

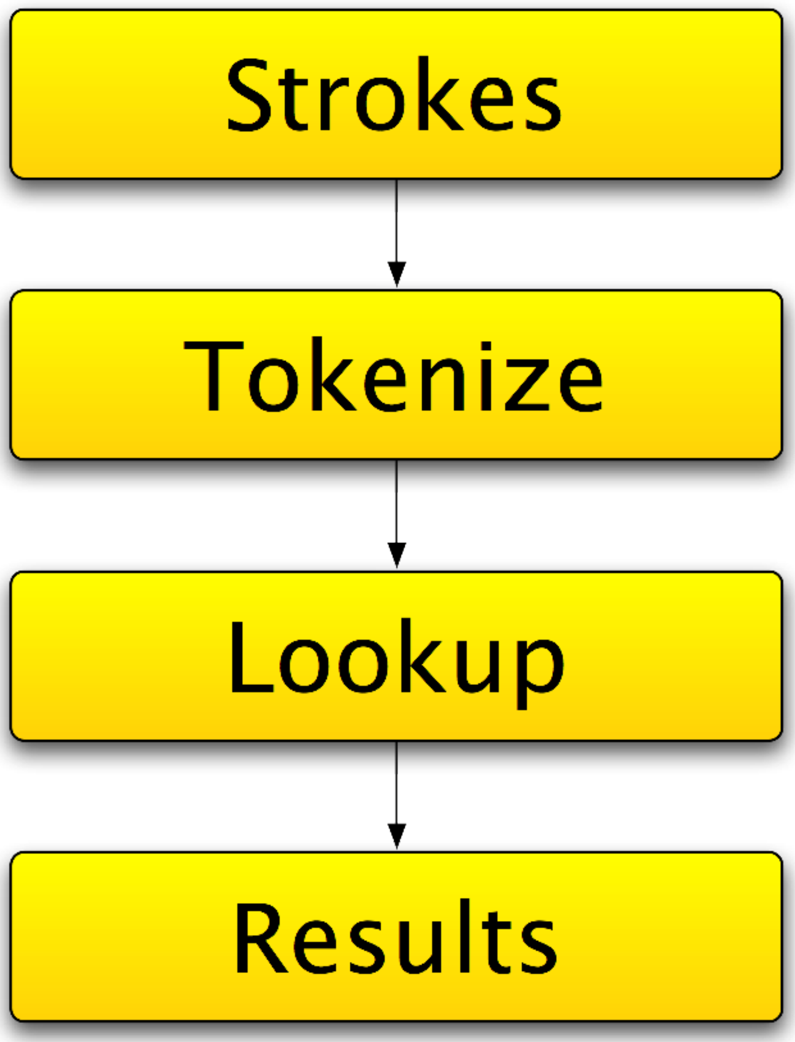

C
A
P
E
R


C
A
R
E



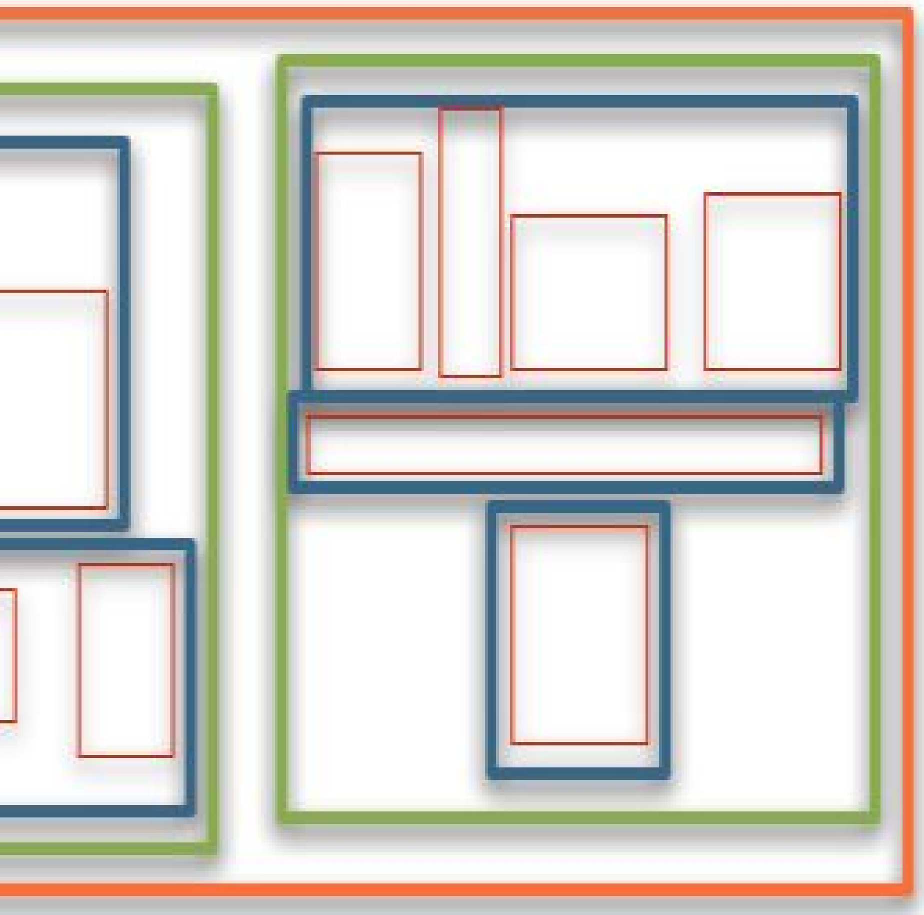
Match probability

0.6


Strokes
Tokenize
Lookup
Results

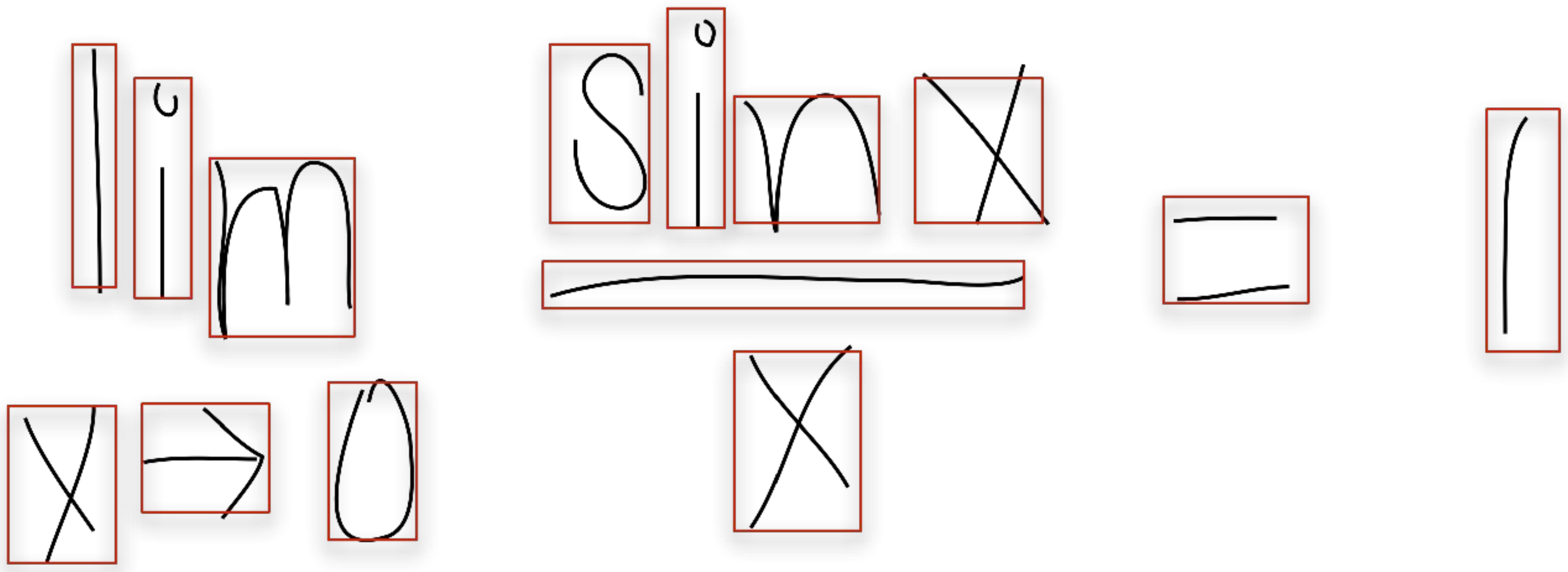
Structural

Character

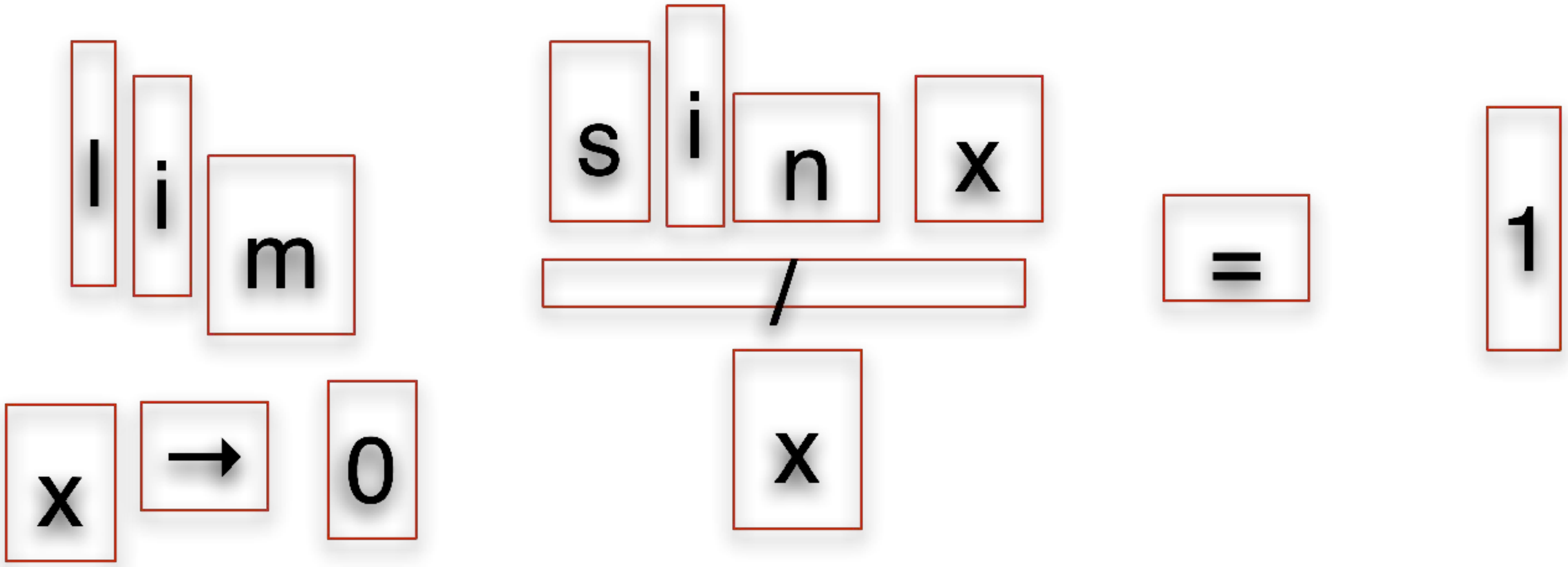


Structural Character

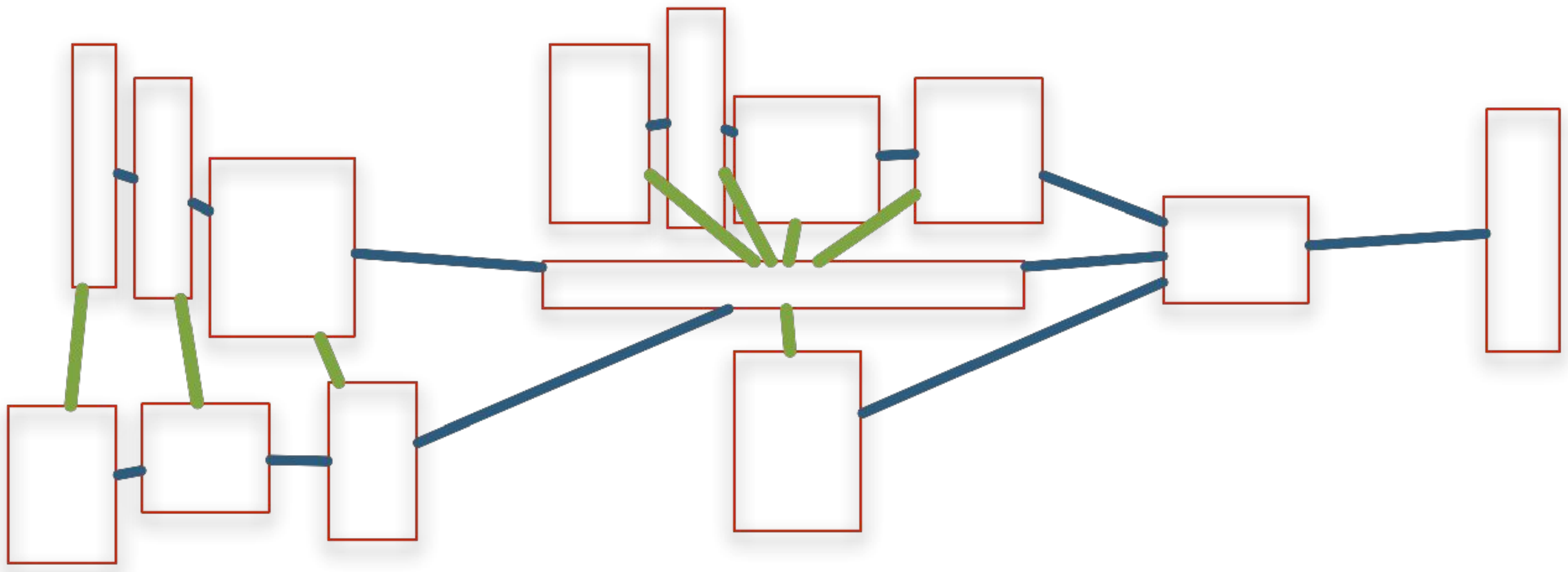
$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$



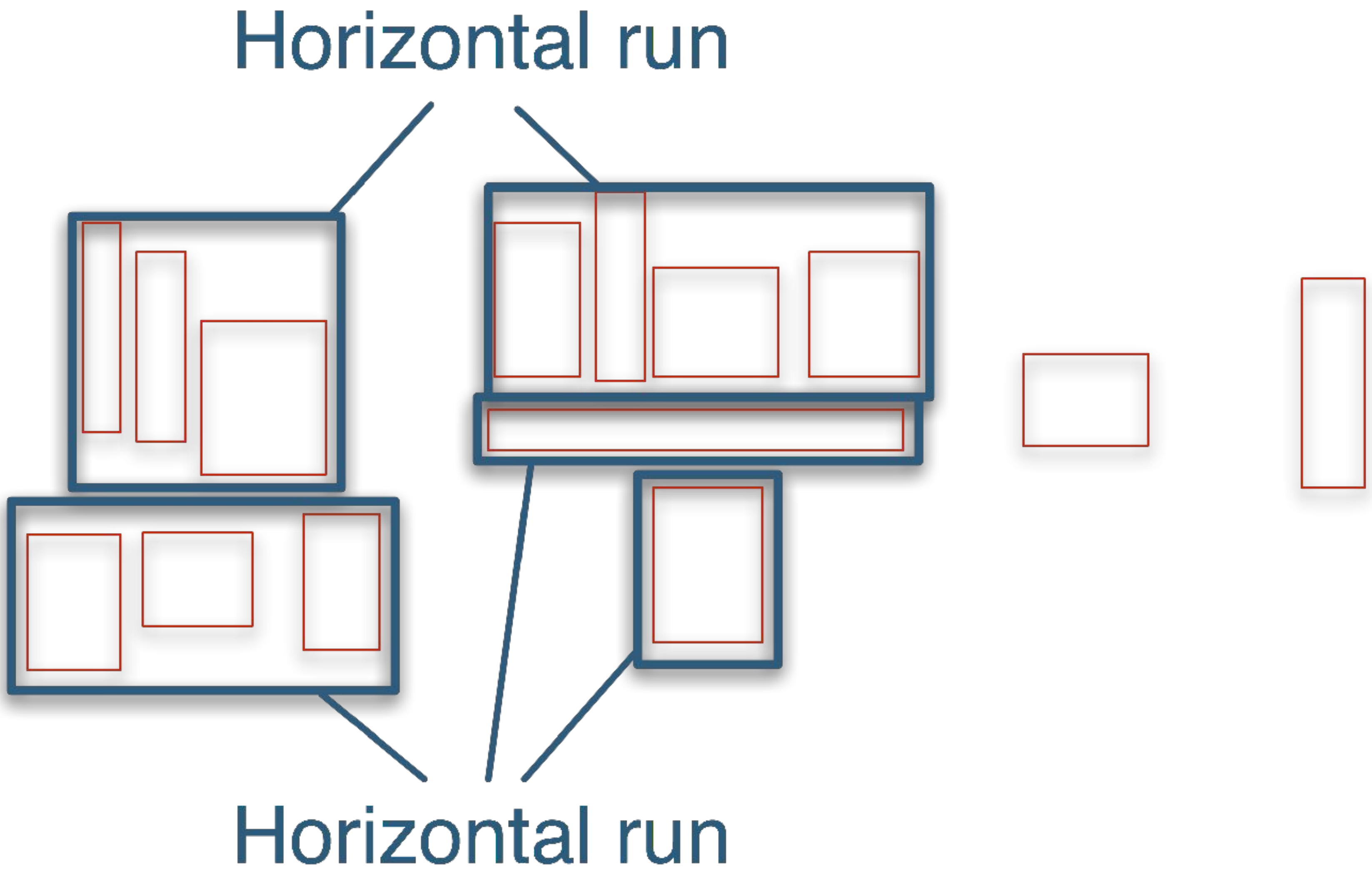
Bounding Boxes

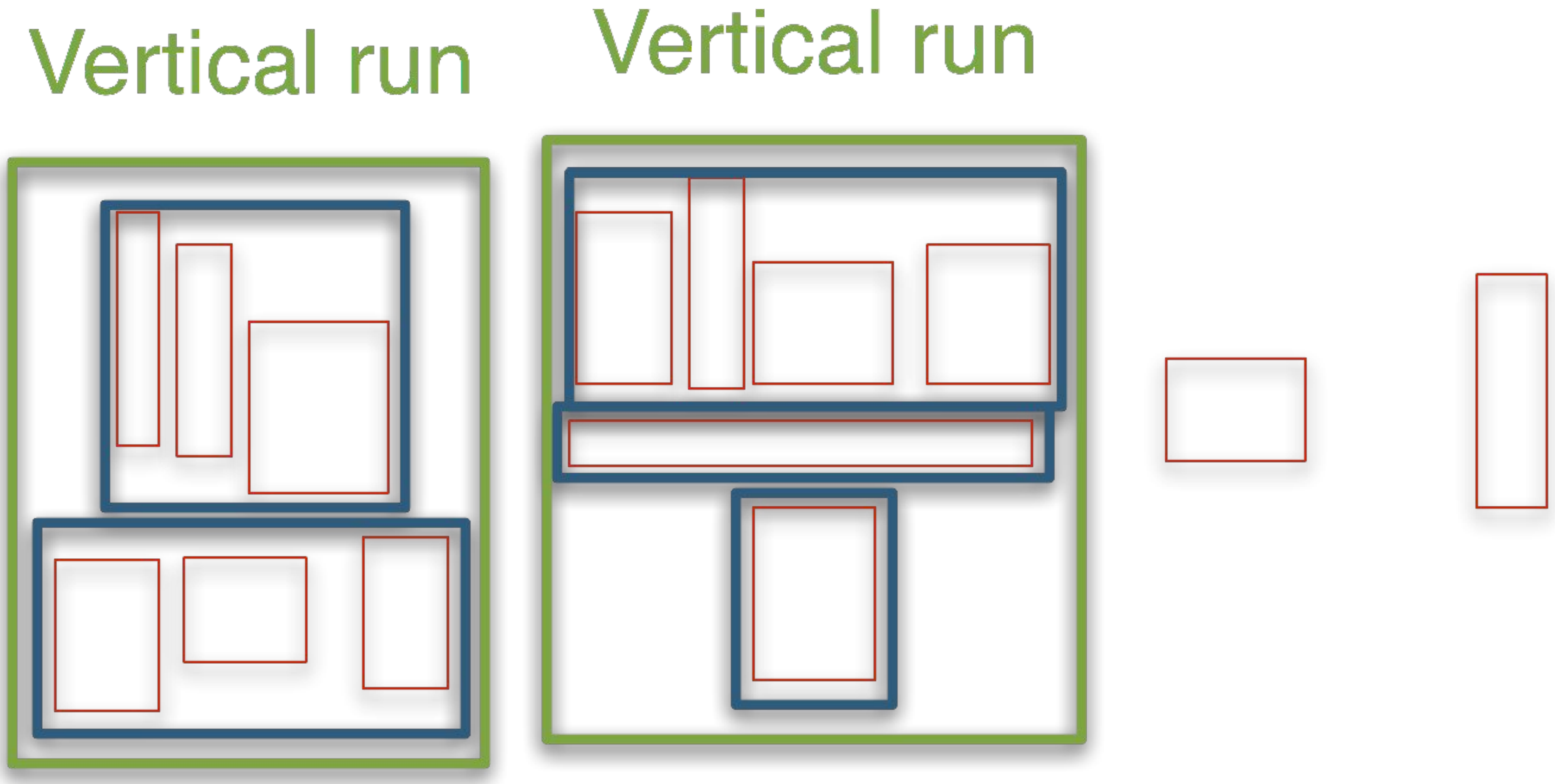


Character Recognition

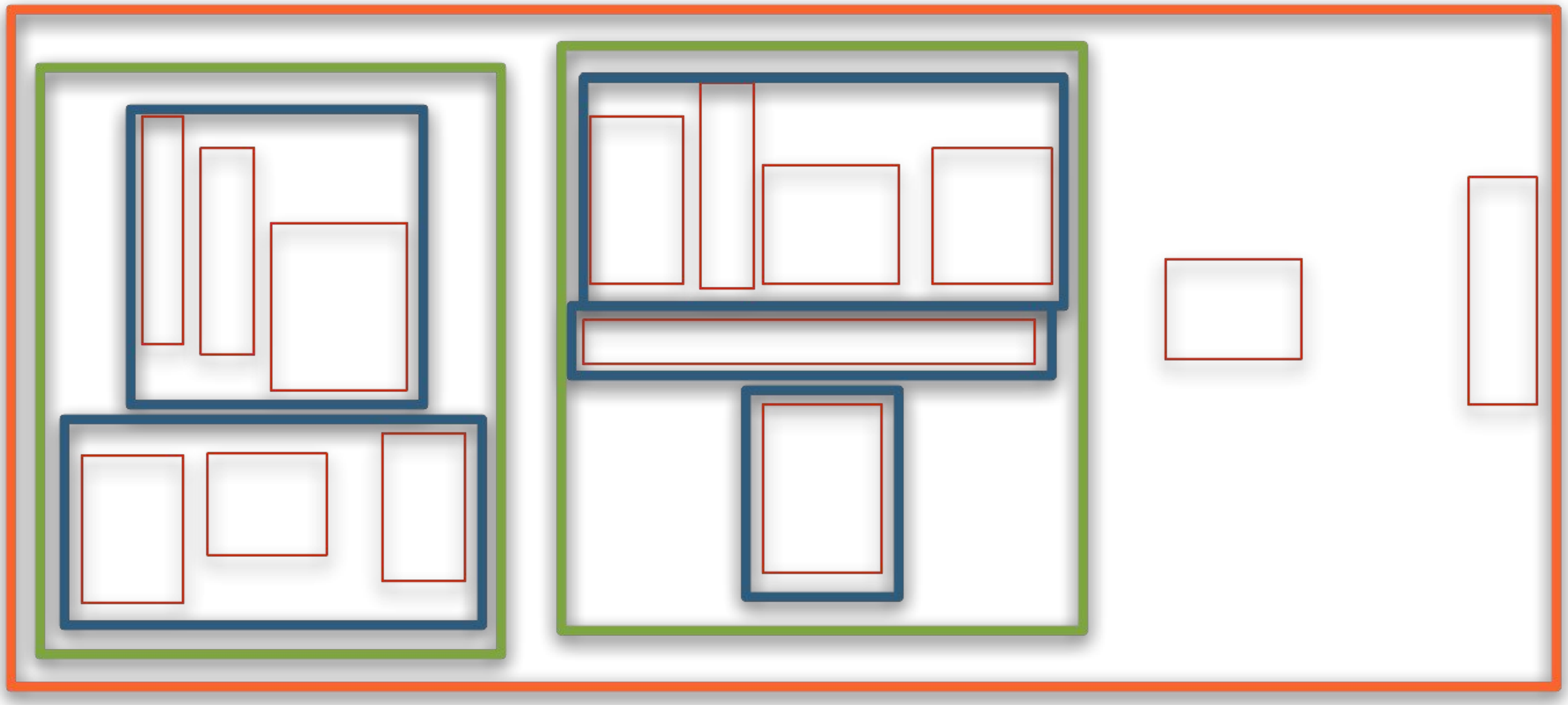


Anchors


Horizontal run
Horizontal run


Vertical run
Vertical run

Expression



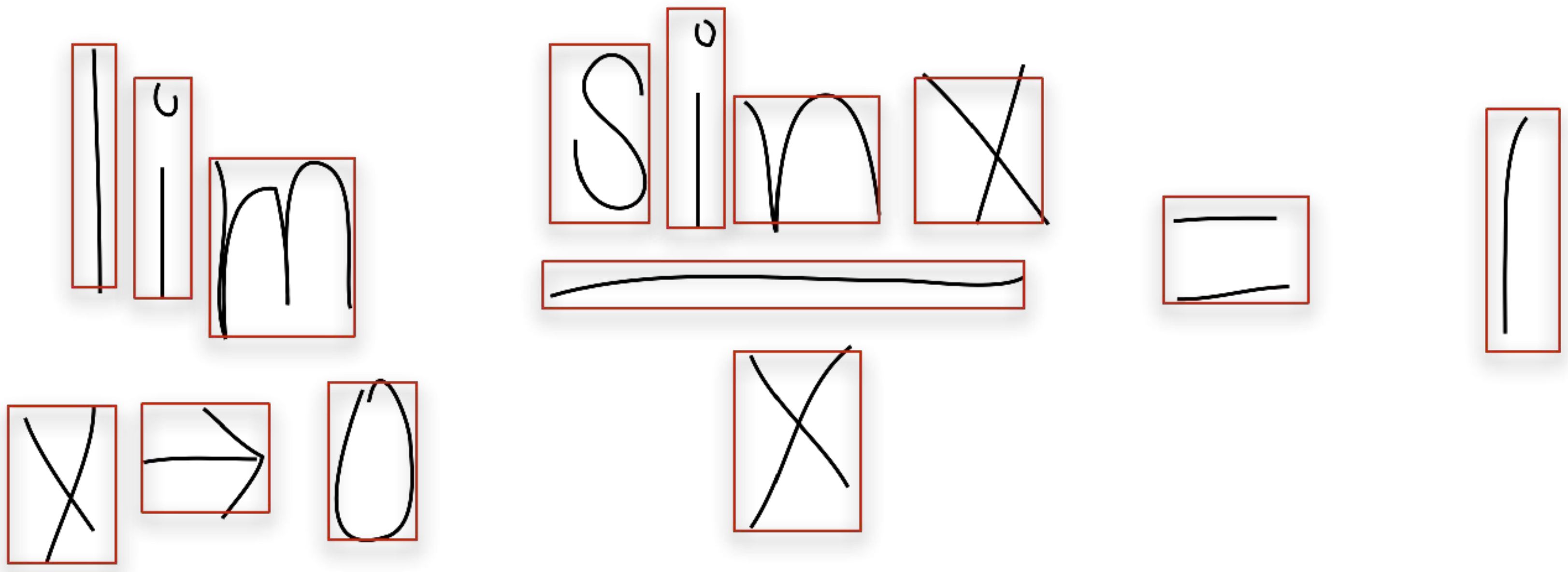
Result

```
<Horizontal-run depth="0" x="93" y="48" width="485" height="144">
       <Vertical-run depth="1" x="93" y="61" width="40" height="85">
           <Horizontal-run depth="2" x="101" y="61" width="27" height="33">
               <Character depth="3" x="101" y="61" width="27" height="33"/>
           </Horizontal-run>
           <Horizontal-run depth="2" x="93" y="102" width="40" height="1">
               <Character depth="3" x="93" y="102" width="40" height="1""/>
           </Horizontal-run>
       </Vertical-run>
       <Character depth="1" x="152" y="68" width="19" height="18"/>
       <Vertical-run depth="1" x="194" y="48" width="62" height="89">
           <Horizontal-run depth="2" x="205" y="48" width="43" height="43">
               <Character depth="3" x="205" y="48" width="21" height="43"/>
               <Character depth="3" x="235" y="49" width="13" height="16"/>
           </Horizontal-run>
           <Horizontal-run depth="2" x="194" y="94" width="62" height="4">
               <Character depth="3" x="194" y="94" width="62" height="4"/>
           </Horizontal-run>
```

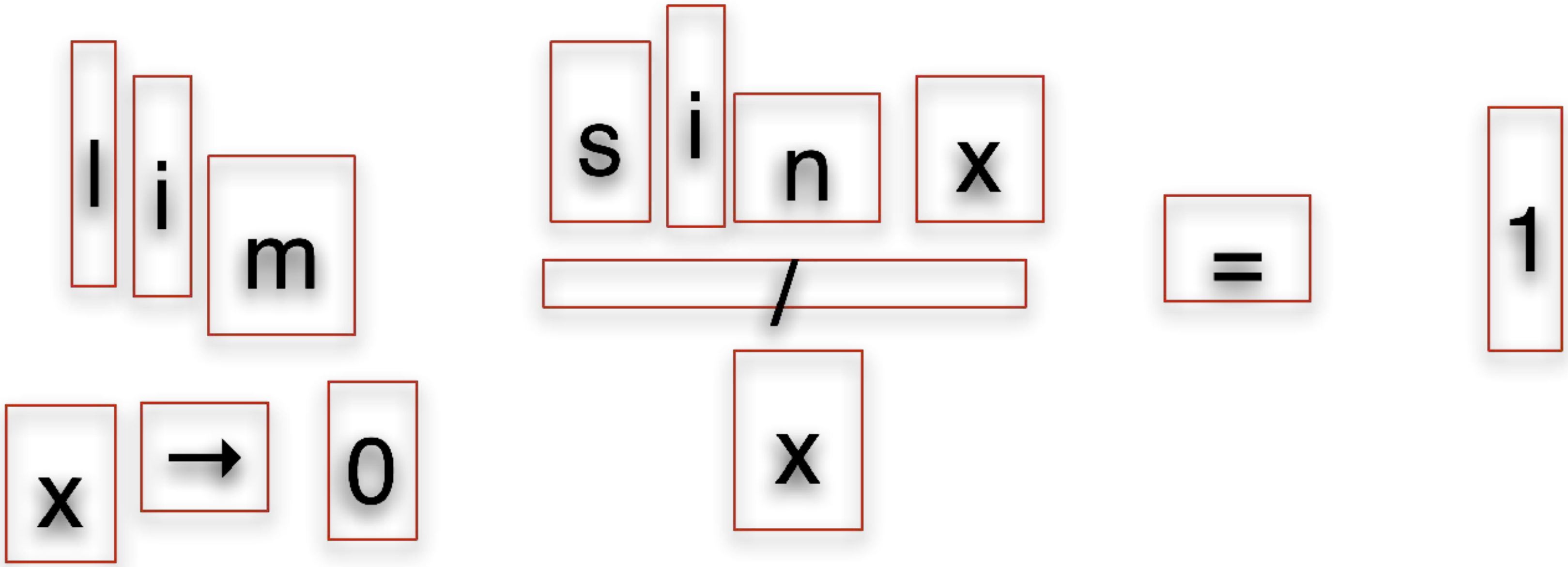
Putting It All Together

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

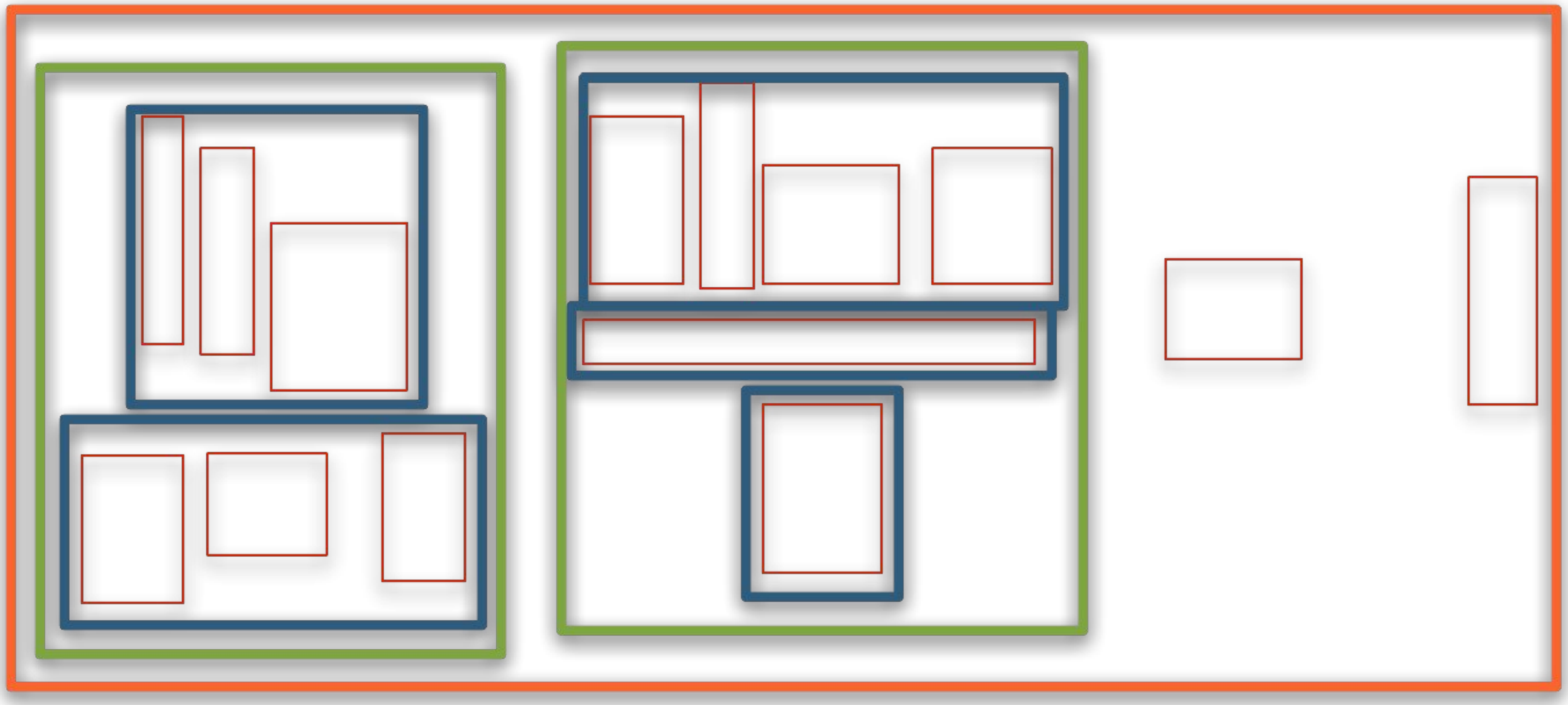
Handwritten Math



Bounding Boxes



Character Recognition



Structural Recognition

```
<Horizontal-run depth="0" x="93" y="48" width="485" height="144">
       <Vertical-run depth="1" x="93" y="61" width="40" height="85">
           <Horizontal-run depth="2" x="101" y="61" width="27" height="33">
               <Character depth="3" x="101" y="61" width="27" height="33"/>
           </Horizontal-run>
           <Horizontal-run depth="2" x="93" y="102" width="40" height="1">
               <Character depth="3" x="93" y="102" width="40" height="1""/>
           </Horizontal-run>
       </Vertical-run>
       <Character depth="1" x="152" y="68" width="19" height="18"/>
       <Vertical-run depth="1" x="194" y="48" width="62" height="89">
           <Horizontal-run depth="2" x="205" y="48" width="43" height="43">
               <Character depth="3" x="205" y="48" width="21" height="43"/>
               <Character depth="3" x="235" y="49" width="13" height="16"/>
           </Horizontal-run>
           <Horizontal-run depth="2" x="194" y="94" width="62" height="4">
               <Character depth="3" x="194" y="94" width="62" height="4"/>
           </Horizontal-run>
```

Parsing

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

Math

Integration

MouseListener

Record incoming packets

```
class StrokeListener implements MouseListener,
   MouseMotionListener {
    private Stroke stroke;
    public void mousePressed( MouseEvent e ) {
        stroke = new Stroke();
        stroke.addPacket( e.getX(), e.getY() );
    }
    public void mouseDragged( MouseEvent e ) {
        stroke.addPacket( e.getX(), e.getY() );
    }
    public void mouseReleased( MouseEvent e ) {
        stroke.addPacket( e.getX(), e.getY() );
        context.addStroke( stroke );
        stroke = null;
    }
// etc...
}
```

Shape and PathIterator

Draw strokes

```
class StrokeShape implements
  java.awt.Shape {
      Stroke stroke;
      public PathIterator getPathIterator(
           AffineTransform transform )
      {
           return new StrokePathIterator(
                stroke, transform );
      }
}
```

```
class StrokePathIterator implements PathIterator {
public int currentSegment( float[] coords ) {
    int pathType;
    Packet packet = packets[ iterCount ];
    if( iterCount == 0 ) {
        pathType = PathIterator.SEG_MOVETO;
    } else {
        pathType = PathIterator.SEG_LINETO;
    }
    coords[0] = packet.x; coords[1] = packet.y;
    coords[0] *= transform.getScaleX();
    coords[1] *= transform.getScaleY();
    coords[0] += transform.getTranslateX();
    coords[1] += transform.getTranslateY();
    return pathType;
}
```

BasicStroke

Control rendering of strokes

Microsoft

Tablet PC handwriting recognizer

Java Native Interface

Use ***Microsoft***'s recognizer from Java!

```
private native String recognize(
     int[][] data );
```

```
data = {
   { x1, y1, x2, y2, x3, y3, … },
   { x1, y1, x2, y2, x3, y3, … },
}
```

Interleaved

MicrosoftCharacterRecognizer.h

javah

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class MicrosoftCharacterRecognizer */
#ifndef _Included_MicrosoftCharacterRecognizer
#define _Included_MicrosoftCharacterRecognizer
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     MicrosoftCharacterRecognizer
 * Method:    recognize
 * Signature: ([[I)Ljava/lang/String;
 */
JNIEXPORT jobject JNICALL
Java_MicrosoftCharacterRecognizer_recognize
  (JNIEnv *, jobject, jobjectArray);
#ifdef __cplusplus
}
#endif
#endif
```

Microsoft Visual Studio

Create a DLL

```
JNIEXPORT jobject JNICALL
Java_MicrosoftCharacterRecognizer_recognize
( JNIEnv *env, jobject object, jobjectArray strokeArray )
{

   CoInitialize( NULL );

   // Create the dummy InkCollector object
   // so that we can obtain its Ink object.

   CoCreateInstance( CLSID_InkCollector,
       NULL, CLSCTX_INPROC_SERVER,
       IID_IInkCollector,
       (void**)&pIInkCollector );
   pIInkCollector->get_Ink( &pIInk );
   pIInk->CreateStrokes( emptyVar, &pIInkStrokes );
```

```
JNIEXPORT jobject JNICALL
Java_MicrosoftCharacterRecognizer_recognize
( JNIEnv *env, jobject object, jobjectArray strokeArray )
{
   // Obtain stroke info from JNI side
   int numStrokes = env->GetArrayLength( strokeArray );
   for( int j = 0; j < numStrokes; j++ ) {
       jintArray packets = (jintArray)
           env->GetObjectArrayElement( strokeArray, j );
       int numPackets = env->GetArrayLength( packets );
       psa = SafeArrayCreateVector( VT_I4, 0, numPackets );
       SafeArrayAccessData( psa, (VOID**)&plongArray );
       env->GetIntArrayRegion( packets, 0, numPackets, plongArray );
       SafeArrayUnaccessData( psa );
       var.vt     = VT_ARRAY | VT_I4;
       var.parray = psa;
       pIInk->CreateStroke( var, varPK, &pIInkStroke );
       pIInkStrokes->Add( pIInkStroke );
   }
```

```
JNIEXPORT jobject JNICALL
Java_MicrosoftCharacterRecognizer_recognize
( JNIEnv *env, jobject object, jobjectArray strokeArray )
{
   // Recognize the Strokes

   CoCreateInstance(

       CLSID_InkRecognizerContext,

       NULL, CLSCTX_INPROC_SERVER,

       IID_IInkRecognizerContext,

       (void **) &pIInkRecoContext );

   pIInkRecoContext->putref_Strokes( pIInkStrokes );

   IInkRecognitionResult* pIInkRecoResult = NULL;

   InkRecognitionStatus RecognitionStatus;

   pIInkRecoContext->Recognize(

       &RecognitionStatus, &pIInkRecoResult );
```

```
JNIEXPORT jobject JNICALL
Java_MicrosoftCharacterRecognizer_recognize
( JNIEnv *env, jobject object, jobjectArray strokeArray )
{

   // Return the best string
   BSTR bstrBestResult = NULL;
   pIInkRecoResult->get_TopString( &bstrBestResult );
   pIInkRecoResult->Release();
   pIInkRecoResult = NULL;
   jstring res = BstrToJstring( env, bstrBestResult );
   SysFreeString( bstrBestResult );
   pIInkRecoContext->putref_Strokes( NULL );
   CoUninitialize();
   return res;

}
```

```
static jstring BstrToJstring( JNIEnv *env, BSTR bstr ) {
    long len = SysStringLen( bstr ) + 1;
    char* a = new char[ len ];
    jstring ret;
    WideCharToMultiByte( CP_ACP, 0, bstr,
        len, (char*)a, len, NULL, FALSE );
    a[ len - 1 ] = 0;
    ret = env->NewStringUTF( a );
    delete[] a;
    return ret;
}
```

DEMO

Microsoft Handwriting Recognizer

From Java

Summary

Handwriting has many novel applications

Use it in Java programs

Get it for free from Microsoft

For More Information

Structural Analysis for
Pen-Based Math Input Systems

Ian Rutherford

http://www.cs.uwaterloo.ca/~ijruther/thesis.pdf

For More Information

Demo code

http://www.desktopjava.com

For More Information

High Performance GUI

TS-1305

Q&A

Handwriting Recognition

Yu-Hong Wang

Senior Developer, GUI

Maplesoft

http://www.maplesoft.com/

TS-3690