Lecture 6

August 9, 2016

Lambda Functions.
Probability Distributions.

Reminder: start the recording

Announcements

- Assignment 2
  - Due now
- Project 2
  - Due next Tuesday 11:45pm
- Assignment 3
  - due Thursday August 18th at 4:30pm
- Project 3
  - Due Tuesday August 23 at 11:45 pm
  - Today's lecture will provide information
- Questions?

Project 1

- Parentheses matching was the biggest challenge—it's a task that regular expressions are not well suited for
- One solution: pre-process each file to replace '(' and ')' with matching codes
- Better solution: use recursive function to track nesting level

Track nesting with recursion

```
static void ParseLevel(ref int c_np, ref int c_constit)
{
    c_constit++;        // count one constituent for the parent
    bool is_vp = s_in.Substring(ix, 3) == "VP ";
    if (is_vp)
        vp++;
    else if (s_in.Substring(ix, 2) == "S ")
        s++;
    else if (s_in.Substring(ix, 3) == "NP ")
        { np++; c_np++; }

    int child_nps = 0, child_constit = 0;       // local counters for immediate children
    bool f_any = false;
    while (ix < s_in.Length)
    {
        Char ch = s_in[ix++];
        if (ch == '(')
        {
            f_any = true;
            ParseLevel(ref child_nps, ref child_constit);   // count constituents in lower level
        }
        else if (ch == ')')
        {
            // end of a constituent. for VPs, adjust global counts based on immediate child counts
            if (is_vp)
            {
                if (!f_any)
                    ivp++;
                if (child_nps == 2 && child_constit == 2)
                    dvp++;
            }
            return;
        }
    }
}
```

Assignment 2

1. Using the following sets, we run a trial which selects exactly one word from each set. Within each set, all words are equally likely.

A = { monkey, donkey, yak, kangaroo, aardvark, antelope, puma, cheetah }

B = { whale, shark, dolphin, eel }

$$|A| = 8$$
$$|B| = 4$$

There are 32 tuples

E = { either of the words contain a ‘y’ }

(monkey,whale) (monkey,shark) (monkey,dolphin) (monkey,eel) (donkey,whale) (donkey,shark) (donkey,dolphin) (donkey,eel) (yak,whale) (yak,shark) (yak,dolphin) (yak,eel)

F = { both words contain an ‘e’ }

(monkey,whale) (monkey,eel) (donkey,whale) (donkey,eel) (antelope,whale) (antelope,eel) (cheetah,whale) (cheetah,eel)

G = { both words contain the same number of letters }

(yak,eel) (cheetah,dolphin)

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

(kangaroo,whale) (kangaroo,shark) (kangaroo,dolphin) (kangaroo,eel) (aardvark,whale) (aardvark,shark) (aardvark,dolphin) (aardvark,eel) (antelope,whale) (antelope,shark) (antelope,dolphin) (antelope,eel) (cheetah,whale) (cheetah,shark) (cheetah,dolphin) (cheetah,eel)

At this point the tuples are fixed for this problem so you could assign each tuple a unique number

E = { either of the words contain a ‘y’ }

0 1 2 3 4 5 6 7 8 9 10 11 $\frac{|EE|}{|\Omega|}$ =.375

F = { both words contain an ‘e’ }

0 3 4 7 20 23 28 31 $\frac{|EE|}{|\Omega|} = .25$

G = { both words contain the same number of letters }

11 30 $\frac{|^{EE}|}{|\Omega|} = .0625$

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31 $\frac{|EE|}{|\Omega|} = .5$

E = { either of the words contain a ‘y’ }

0 1 2 3 4 5 6 7 8 9 10 11

F = { both words contain an ‘e’ }

0 3 4 7 20 23 28 31

G = { both words contain the same number of letters }

11 30

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31

$$P(E \cup H) = \frac{28}{32} = .875$$

E = { either of the words contain a 'y' }

0 1 2 3 4 5 6 7 8 9 10 11

F = { both words contain an 'e' }

0 3 4 7 20 23 28 31

G = { both words contain the same number of letters }

11 30

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31

$$P(F \cap H) = \frac{4}{32} = .125$$

E = { either of the words contain a 'y' }

0 1 2 3 4 5 6 7 8 9 10 11

F = { both words contain an 'e' }

0 3 4 7 20 23 28 31

G = { both words contain the same number of letters }

11 30

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31

$$E \cap F \cap G = \emptyset$$

$$P(E \cap F \cap G) = 0$$

E = { either of the words contain a 'y' }

0 1 2 3 4 5 6 7 8 9 10 11

F = { both words contain an 'e' }

0 3 4 7 20 23 28 31

G = { both words contain the same number of letters }

11 30

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31

$$P(G \cup H) = \frac{17}{32} = .53125$$

F = { both words contain an ‘e’ }

0 3 4 7 20 23 28 31

$\bar{F}$: 1 2 5 6 8 9 10 11 12 13 14 15 16 17 18 19 21 22 24 25 26 27 29 30

H = { either (or both) of the words contains *more than two* vowels { a e i o u }. This count includes repeated uses of the same vowel. }

12 13 14 15 16 17 18 19 20 21 22 23 28 29 30 31

$$P(H \cap \bar{F}) = \frac{12}{32} = .375$$

	E	F	G
H	X	I	I
G		X	
F			

(E and G) are dependent
(E and F) are dependent

2. Working in Yunnan, a field linguist has discovered an extinct version of the Dongba pictographic script. So far, his team has found 32 distinct glyphs in this script, and the linguist has deciphered 22 of them. He just received news that another researcher has discovered a new inscription that consists of 8 glyphs. These 8 have all previously been encountered, but he doesn’t yet know if the new inscription has repeated glyphs, or not.

a. What is the probability that the linguist will fully understand the newly discovered inscription?

"Inscription" implies that a glyph can appear more than once in the new discovery, so we assume trials that select with replacement:

Got this correct

$$\left(\frac{22}{32}\right)^8 = \frac{214{,}358{,}881}{4{,}294{,}967{,}296} \approx 0.05$$

b. What is the probability that the linguist will understand at least half of the glyphs in the newly discovered inscription?

look into this!

$$\sum_{kk=4}^{8} \left(\frac{22}{32}\right)^{kk} \left(\frac{10}{32}\right)^{8-kk} \binom{8}{kk} \approx .9318$$

Explanation:

http://courses.washington.edu/ling473/A2Q2.pdf extra credit: If each of the 8 glyphs in the newly discovered inscription **are distinct from each other** (but still in the set of 32 known glyphs), what is the probability that the linguist will understand at least half of them?

number of sets containing kk fully-understood glyphs

number of sets of $8 - kk$ not-understood glyphs

$$\sum_{kk=4}^{8} \frac{\binom{22}{k}\binom{10}{8-kk}}{\binom{32}{8}} = \frac{134387}{140244} \approx 0.9582$$

number of possible subsets

Greedy RegEx

- The previous example used the Kleene star (*) which is a unary operator for matching zero or more elements
- A RegEx processor is a state machine
- To keep a RegEx FSM deterministic, operations such as Kleene star are usually implemented as greedy
- Example:

```
Ipsum lorem <title>My web page</title> Ipsum lorem
Regex:  <.*>
Result:    <title>My web page</title>
```

Greedy algorithm

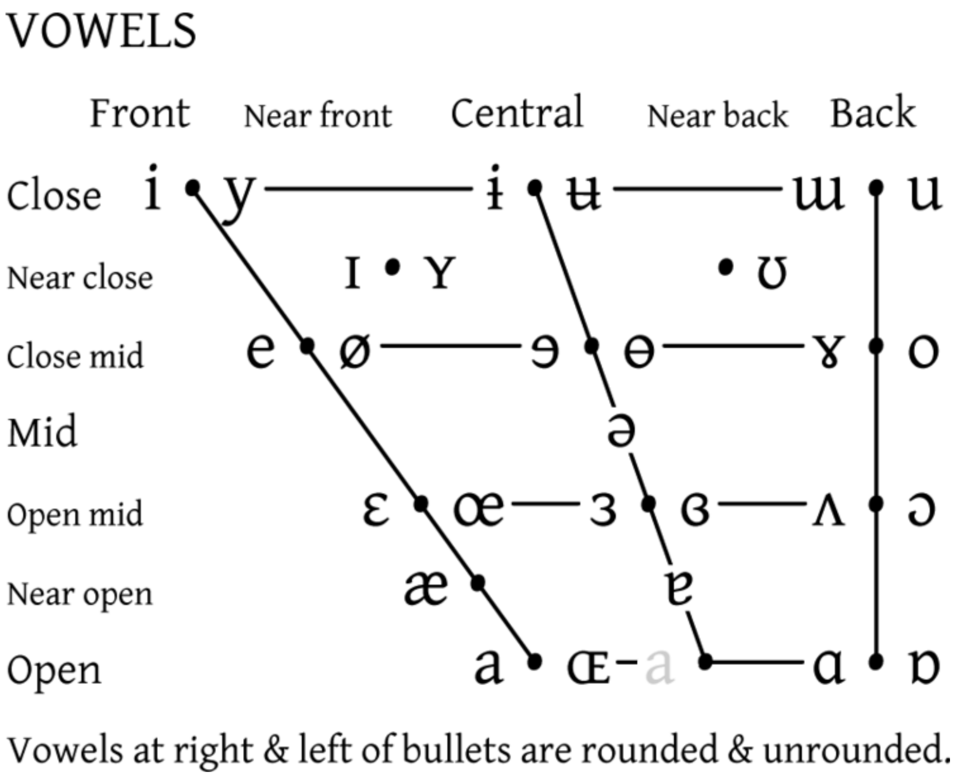
- General term for any algorithm that moves irrevocably forward based on best available information or pre-determined policy
- Choices cannot be reconsidered later
- Example: Dijkstra's algorithm for finding the shortest path in a graph
- Contrast with: Dynamic programming (later lecture)

Assignment 3

- http://courses.washington.edu/ling473/Assignment3.pdf
- Due next Tuesday, August 20th before class
- 4 probability/statistics problems that will draw together what we've been studying:
  - Conditional probability
  - Random variables
  - Bayes theorem

Open/Closed Vowels

Quick intro to vowel phonetics, since this is mentioned in Assignment 3



POS tag probabilities

The Red Badge of Courage, by Stephen Crane

DT	NN	VBD	RB	IN	DT	NN	,	CC	DT	VBG	NNS	VBD	DT	NN	VBD	IN	IN	DT
the	cold	passed	reluctantly	from	the	earth	,	and	the	retiring	fogs	revealed	an	army	stretched	out	on	the

NNS	,	VBG	.	IN	DT	NN	VBN	IN	JJ	TO	VB	,	DT	NN	VBN	,	CC	VBD	TO	VB
hills	,	resting	.	as	the	landscape	changed	from	brown	to	green	,	the	army	awakened	,	and	began	to	tremble

IN	NN	IN	DT	NN	IN	NNS	.	PRP	NN	PRP$	NNS	IN	DT	NNS	,	WDT	VBD	VBG	IN	JJ
with	eagerness	at	the	noise	of	rumors	.	it	cast	its	eyes	upon	the	roads	,	which	were	growing	from	long

NNS	IN	NN	NN	TO	JJ	NNS	.	DT	NN	,	JJ	IN	DT	NN	IN	PRP$	NNS	,
troughs	of	liquid	mud	to	proper	thoroughfares	.	a	river	,	amber-tinted	in	the	shadow	of	its	banks	,

VBD	IN	DT	NN	POS	NNS	:	CC	IN	NN	,	WRB	DT	NN	VBD	VBN	IN	DT	JJ	NN	,	NN
purled	at	the	army	's	feet	;	and	at	night	,	when	the	stream	had	become	of	a	sorrowful	blackness	,	one

MD	VB	IN	PRP	DT	JJ	,	JJ	NN	IN	JJ	NNS	VBN	IN	DT	JJ	NNS	IN	JJ	NNS	.
could	see	across	it	the	red	,	eyelike	gleam	of	hostile	camp-fires	set	in	the	low	brows	of	distant	hills	.

individual words

unigram count: 123
bigram count: 122

$$P(NN) = \frac{18}{123} \quad P(IN) = \frac{20}{123} \quad P(NN|IN) = \frac{3}{20} \quad P(DT\ NN) = \frac{10}{122}$$

Unigram Bigram Bigram

Properties of probability distributions

- Let’s look at some of the important probability distributions
- Summarizing parameters:
  - Expected Value (Mean)
  - Variance
  - Standard Deviation

Expected Value

- Notation: $E[X]$
- Discrete: $E[X] = \sum x P_X(x)$
  - This should not be confused with "most probable value."
  - The expected value may be avalue that is not in the domain
  - The expected value is only meaningful if the random variable's values are chosen meaningfully
- Continuous: $E[X] = \int x f(x) dx$
  - A weighted sum of all the possible values

Expected value as average

$$EE\ XX = \text{?}\ xxPP_{XX}(xx)$$

$$EE\ XX = \text{?}_{ii}^{nn} \frac{1}{nn}\ xx_{ii}$$

"uniform" distribution

$$EE\ XX = \frac{1}{nn} \text{?}_{ii}^{nn}\ xx_{ii}$$

$u = x\bar{x}$

nn

In uniform distribution

Think about the values of a dice role

[ ]

[ ] —

[ ] —

[ ] —

Measuring "spread"

$$E[X] = 50$$

$$X = \{50, 50, 50, 50, 50, 50\}$$
$$X = \{ 47, 48, 49, 51, 52, 53 \}$$
$$X = \{ 0, 0, 0, 100, 100, 100 \}$$

$$E[X - \mu] = ?$$

mean

Variance

- Discrete

$$Var(X) = \sum_{i}^{n} P(x_i)(x_i - \mu)^2$$

- Continuous

$$Var(X) = \int (x_i - \mu)^2 f(x_i)\, dx$$

Variance

Derivation of Variance

$$\sigma_X^2 = \mathrm{Var}(X) = E[(X-\mu)^2]$$

$$= \sum (x-\mu)^2 P(x)$$

$$= \sum (x^2 - 2x\mu + \mu^2) P(x)$$

$$= \sum x^2 P(x) - 2P(x)x\mu + \mu^2 P(x)$$

$$= \sum x^2 P(x) - \sum 2P(x)x\mu + \sum \mu^2 P(x)$$

$$= E[X^2] - 2\mu \sum P(x)x + \mu^2 \cancel{\sum P(x)}$$

$$= E[X^2] - 2\mu^2 + \mu^2$$

$$= E[X^2] - \mu^2$$

$$= E[X^2] - E[X]^2$$

Q: why do we get to cancel this term?

Standard deviation

Defined as the square root of the variance

$$\sigma\sigma_{XX} = \sqrt{\mathrm{Var}(XX)}$$

Covariance

- How much does X vary with regard to Y

random variable

$$
\begin{aligned}
\mathrm{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\
&= E[XY] - E[XE[Y]] - E[YE[X]] + E[E[X]E[Y]] \\
&= E[XY] - E[X]E[Y] - E[X] - E[Y] + E[X]E[Y] \\
&= E[XY] - E[X]E[Y]
\end{aligned}
$$

Probability distributions

- Discrete distributions
  - Uniform
  - Bernoulli
  - Binomial
  - Geometric
  - Poisson
- Continuous distributions
  - Uniform
  - Normal

Discrete probability distributions

Uniform distribution (discrete)

- Every discrete value is equally likely to occur

range

$$VV, bb \in \mathbb{Z}, a \leq b$$

$$nn = bb - VV + 1$$

mean

$$\mu\mu = \frac{VV + bb}{2}$$

variance

$$\sigma\sigma^2 = \frac{nn^2 - 1}{12}$$

Only two outcomes?

- Remember random variables
- Events alone were not convenient for correlating probabilities with stochastic trials because they
  - only partition sample spaces into two subsets
  - each imply independent, well-formed probability spaces without regard to other outcomes that we might be interested in.

Doesn't allow to compound

Having said this, what if there *are* only two outcomes in our experiment?

Bernoulli Trial

- A Bernoulli trial is an experiment with only two outcomes

$$\Omega = \{\, yyyyyy, nnnn \,\}$$

- If the outcome is modeled by a random variable

$$XX = \begin{cases} 1, iiff\ tt\boldsymbol{t}yy\ Wyyyyrrrrtt\ iiyy\ yyyyyy, \\ 0, iiff\ tt\boldsymbol{t}yy\ Wyyyyrrrrtt\ iiyy\ nnnn. \end{cases}$$

then random variable X has a Bernoulli distribution

- This discrete probability distribution can be described with a single parameter

$$pp = PP(XX = 1)$$

Bernoulli distribution

- Two outcomes: { success, failure }
- Parameter: $0 \leq p \leq 1, p \in \mathbb{R}$

$$P(X=x) = \begin{cases} p, & \text{if x=success} \\ 1-p, & \text{if } x = \text{failure} \\ 0, & \text{otherwise} \end{cases}$$

$$\mu = p$$

$$\sigma^2 = p(1-p)$$

Binomial distribution

- Model the number of successes in n Bernoulli trials
- Parameters: p, n

$$P(X = x) = \binom{n}{x} p^{x}(1 - p)^{n-x}$$

$$\mu = np$$

$$\sigma^2 = np(1 - p)$$

- Binomial$(p, 1)$ = Bernoulli distribution

Binomial distribution

Q: A corpus contains 4,000 newswire articles, covering every day of the week. An article is selected at random. Let EE be the event that the article is from a Sunday. What is the probability distribution for EE?

A: Binomial distribution with:

$$pp = \frac{1}{7}$$

$$\mu\mu = \frac{1}{7}$$

$$\sigma\sigma^2 = \frac{1}{7}\left(1 - \frac{1}{7}\right)$$

Geometric distribution

$X = \{$ number of Bernoulli trials until obtaining success $\}$

Parameter: p from Bernoulli trial

$$(1-p)(1-p)(1-p)\ \dots\ (1-p)p$$

$$P(X = x) = (1-p)^{x-1}p$$

$$P(X > x) = (1-p)^{x}$$

$$\mu = \frac{1}{p}$$

$$\sigma^2 = \frac{1-p}{p^2}$$

Geometric distribution

Q: A fair coin is flipped T times until it comes up heads. Characterize $PP(DD)$.

A: Geometric distribution with

$$pp = .5$$
$$\mu\mu = 2$$
$$\sigma\sigma^2 = 2$$

Poisson distribution

- The number of independent events that will probably occur during a period of time, given the rate of events
- Parameter: λ = expected # of events per interval

$$P(X = x) = \frac{\lambda^{x} e^{-\lambda}}{x!}$$

$$\mu = \lambda$$

$$\sigma^2 = \lambda$$

Poisson Distribution

- The phone rings 5 times per hour on average
- What is the probability of an hour going by without the phone ringing?

$$p(0) = \frac{\lambda^{0} y^{-5}}{0!}$$

$$= .0067$$

Continuous probability distributions

Uniform distribution (continuous)

$$ff(xx) = \begin{cases} \dfrac{1}{bb - VV}, & iiff\ VV \le xx \le bb \\ 0, & iiff\ xx < VV\ nnVV\ xx > bb \end{cases}$$

any real number in Range

$$\mu\mu = \frac{VV + bb}{2}$$

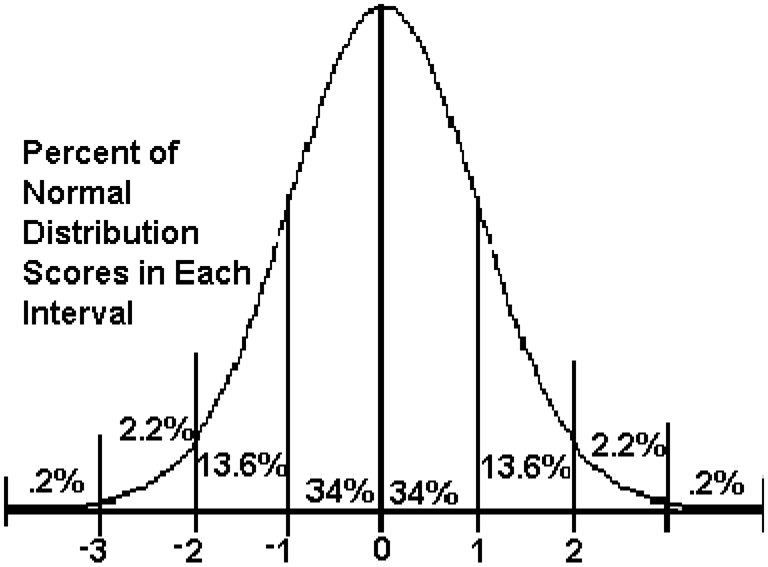
$$\sigma\sigma^2 = \frac{(bb - VV)^2}{12}$$

?

variance

Normal Distribution

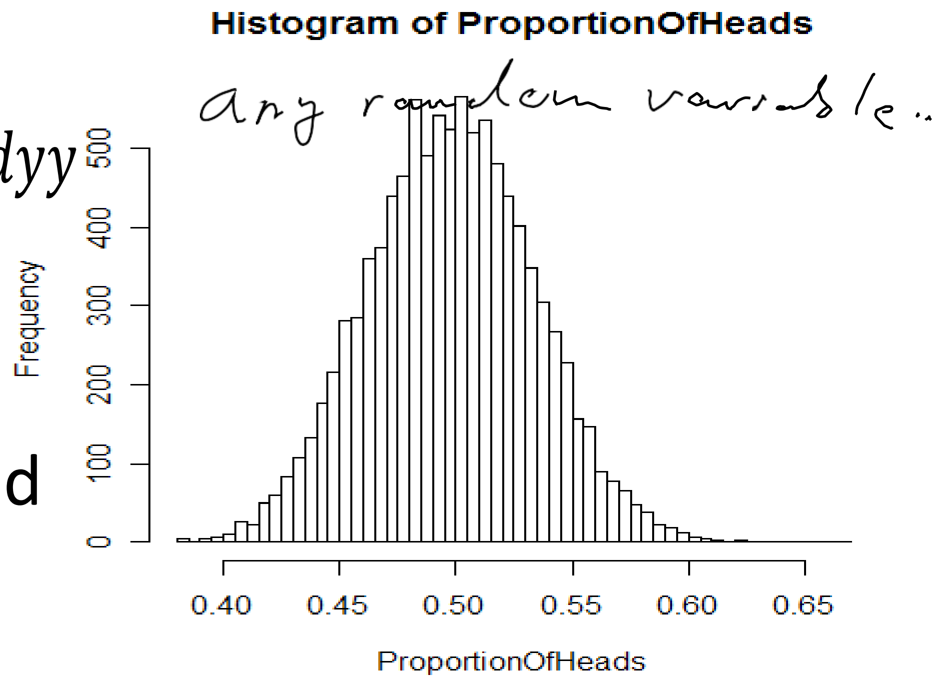
- aka Gaussian distribution
- Parameters:
  - μ — mean
  - $\sigma\sigma^2$ variance
- $ff(xx) = \frac{1}{\sqrt{2\pi\pi\sigma\sigma^2}} yy^{\frac{-(xx-\mu\mu)^2}{2\sigma\sigma^2}}$



Central Limit Theorem

- When a large number of independent random variables is added together, its sum approaches a normal distribution
- Consider a fair coin toss
- $XX = \{\ iitt\ yytnnssyy\ tyyVV \} ddyy$
- $PP(XX = tyyVVddyy) = .5$
- Many trials of this r.v.

will be normally distributed



Finite state machines

or, finite state automata

- Deterministic
- Non-deterministic

{ set of states, transitions, start state, input alphabet, final states }

- Finite state transducers
- Acceptor

Deterministic FSM

$qq \in SS$	States
$\delta\delta: SS \times \Sigma \rightarrow SS$	Transitions
$SS_0 \in SS$	Start state
$xx \in \Sigma$ Run time input	Input alphabet
$FF \in SS$	Final states (or Ø)

Does not include input

Each state/input pair has no more than one transition

Non-deterministic FSM

$qq \in SS$	States
PP_{SS}	Transition probabilities
$\delta\delta: SS \times \Sigma \times PP_{SS} \rightarrow SS$	Transitions
$SS_0 \in SS$	Start state
$xx \in \Sigma$	Input alphabet
$FF \in SS$	Final states (or Ø)

For a given state/input, there may be more than one possible transition

At runtime

- This is sufficient description of the machine. At runtime, an input stream composed of symbols from alphabet Σ is provided
- If $\delta(q,x)$ is incomplete, the FSM is said to reject the input

Example

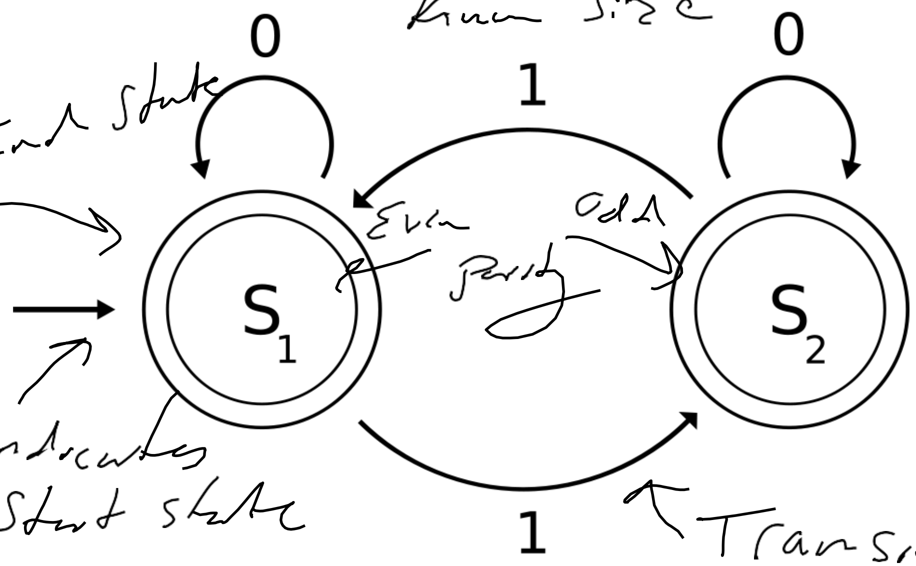
parity: the number of bits in a binary value that are 'set' to one (1)

Final state

Doesn't need to know size

Double-circles are used to indicate accepting states

Determines parity

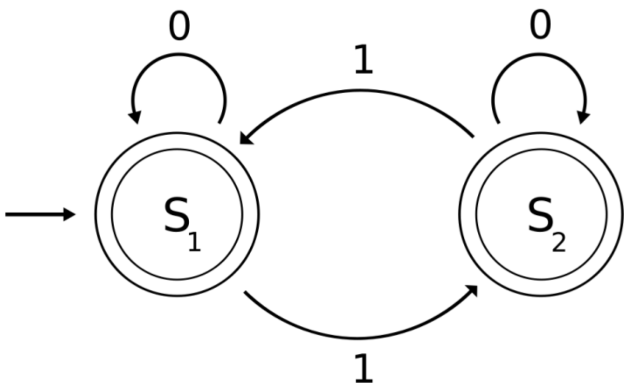


parity of the binary input:
S1 : even
S2 : odd

1 0 1 1 0 0 1 → S1
0 0 0 1 0 0 0 → S2

0,1 alphabet

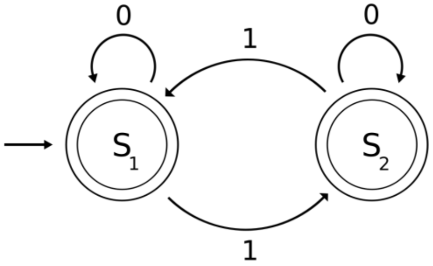
Example



State	Transition
S1	0 → S1, 1 → S2
S2	0 → S2, 1 → S1

Programming FSTs

```
int Parity(String s)    // i.e. "00101010"
{
    int state = 1;
    foreach (Char ch in s)
        switch (state)
        {
            case 1:
                if (state == '1')
                    state = 2;
                break;
            case 2:
                if (  ch  == '1')
                    state = 1;
                break;
        }
    return state;
}
```



Defines a state machine

Example

- Write an FSA for the RegEx:

  a[ab]*b[cd]

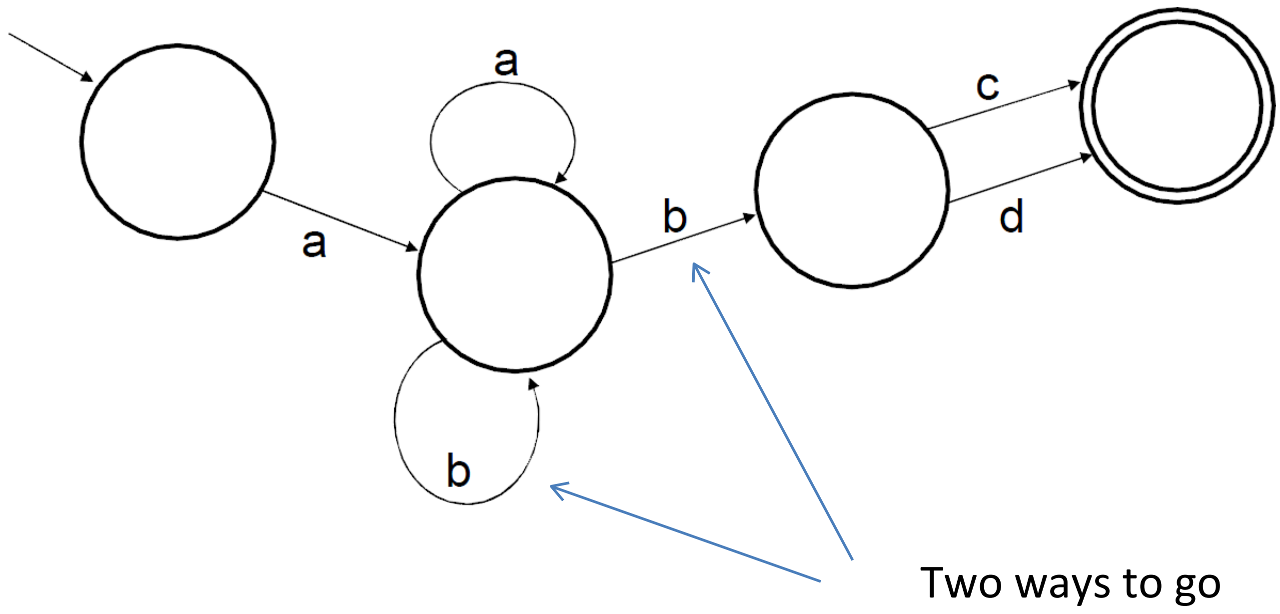
FSM example

Is your FSM deterministic or non-deterministic?

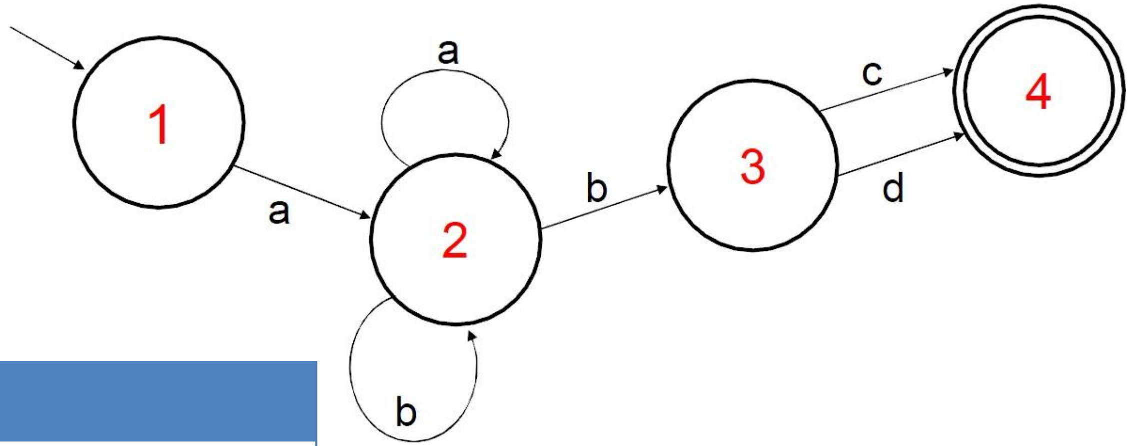
Example

- Non-deterministic

  a[ab]*b[cd]



a[ab]*b[cd]



State	Transition
1	a → 2
2	a → 2, b → 2, b → 3
3	c → 4, d → 4

How would we implement this state machine? We would need more information on how to proceed out of state 2 when the input is 'b'

Example: abbcd
if we choose state 3 here, we will fail to accept this pattern when we should have

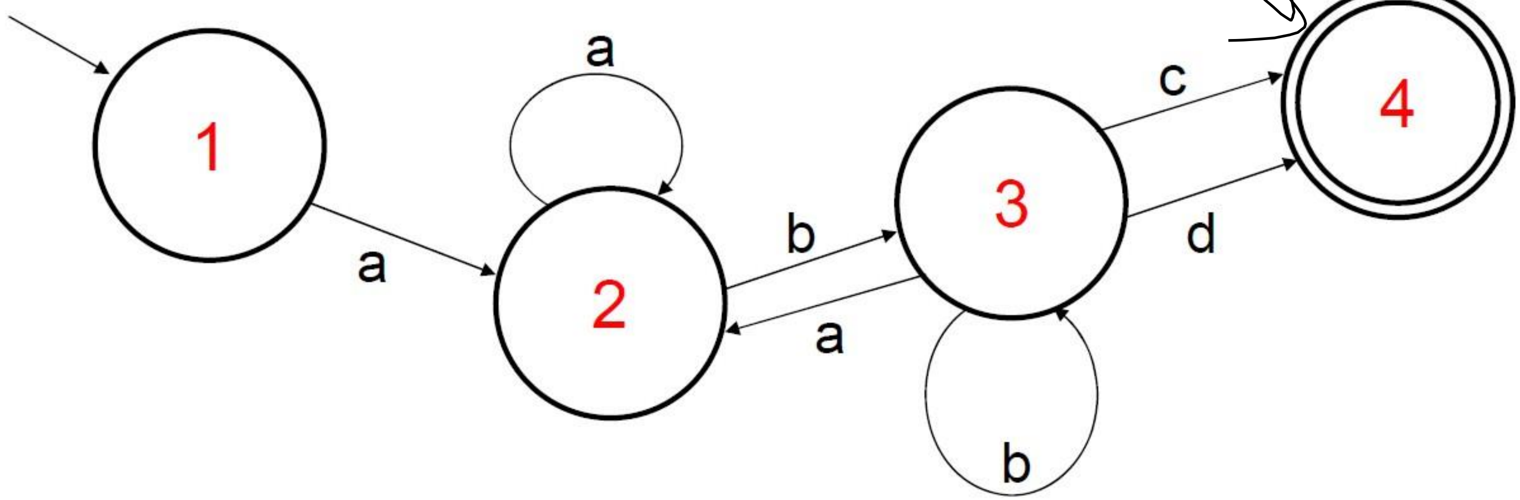
Example

- deterministic

  a[ab]*b[cd]

acceptor FSM
only accepted if
ends up in

one final state



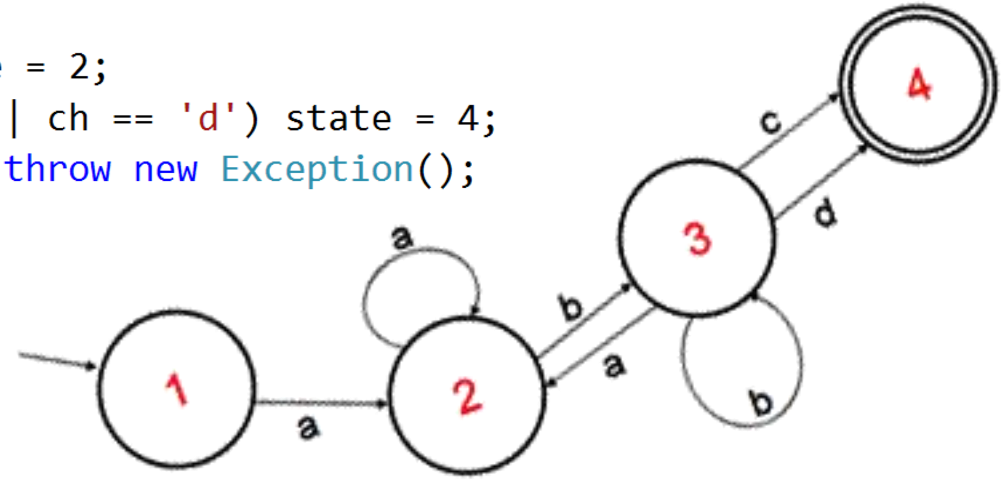
Finite state transducer (FST)

- Add an output function (per state) to an FSM
- The function fires upon arriving at a state
  - Or, when “transitioning”
- Output models:
  - Mealy model: the output depends on both the current input and the state
  - Moore model: the output depends only on the state

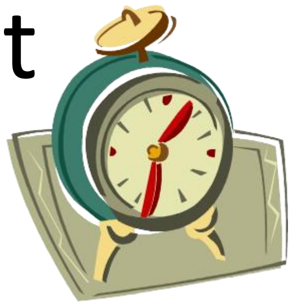
FST

```
IEnumerable<int> FST(String input) {
    int state = 1;
    foreach (Char ch in input) {
        switch (state) {
            case 1:
                if (ch == 'a') state = 2;
                else throw new Exception();
                break;
            case 2:
                if (ch == 'b') state = 3;
                else if (ch != 'a') throw new Exception();
                break;
            case 3:
                if (ch == 'a') state = 2;
                else if (ch == 'c' || ch == 'd') state = 4;
                else if (ch != 'b') throw new Exception();
                break;
            case 4:
                yield break;
        }

    }
}
```



Project 3: Tokenize Thai Text



- This project could be challenging
- 10-minute rule:
  - If you are stuck for 10 minutes, post your question to GoPost before returning to think about it

- http://courses.washington.edu/ling473/Project3.pdf

Project 3

Divide each line of Thai text into words

คู่แข่งขันต่างก็คุมเชิงกัน
เขาเงียบไปครู่หนึ่งแล้วพูดขึ้น
เธอหันมาคุ้ยทรายขึ้นมาใหม่

Process each line, character by character

Change state according to the state machine

Emit spaces for states 7, 8, and 9

Incorrect Thai tokenization

Some operating systems (i.e. Windows) will show a dotted circle for an invalid sequence of Unicode combining characters. This tells you that your result is not correct.

คู่แ ข่งข ันต ่างก ็ค ุมเ ชิงก ัน
เขาเ งียบไ ปค รู่ห นึ่งแ ล้วพ ูดข ึ้น
เธอห ันม าค ุ้ยท รายข ึ้นม าใ หม่
ยินด ีท ี่ไ ด้ร ู้จ ักค ุณ

Using HTML <meta> tag to specify encoding

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

Additional step for reading and writing:

- Save the file using the specified encoding
- Make sure your editor is set to read or write the encoding
- This requires using an editor that is capable of doing so

Additional step for using the page on a web server

- Configure the web server to send a matching HTTP header
- This is an out-of-band mechanism for specifying the content encoding of every single web page

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)  (Red-
Hat/Linux) Last-Modified: Wed, 08 Jan 2003
23:11:55 GMT Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges:
bytes Content-
Length: 438
Connection: close
```

Programming language support for encodings

```
String s = "สวัสดีครับ";        // C# strings are always unicode
int i = s.Length;              // number of code points: 10
Char ch = s[0];                // always a 16-bit character.
                               // In this case the value is 3626 (U+0E2A)

Byte[] bytes = Encoding.GetEncoding("TIS-620").GetBytes(s);
i = bytes.Length;              // number of bytes: 10
byte b = bytes[0];             // a byte. In this case, the value is 202 (\xCA)

bytes = Encoding.UTF8.GetBytes(s);
i = bytes.Length;              // number of bytes: 30

s = Encoding.UTF8.GetString(bytes);     // back to the original string
```

Using C#

With your UW-NetID, you can download and install the full version of Microsoft Visual Studio 2013 Professional for free: http://www.dreamspark.com

To compile a C# program on patas:

```
/home2/joe-student$ gmcs project2-b.cs
```

To run it:

```
/home2/joe-student$ mono project2.exe
```

Closures

- Lambda expressions automatically capture local variables that they reference, which are then passed around as part of the lambda variable
    - Caution: languages do this differently with respect to reference (the lambda expression will modify the original value) versus value (the lambda expression has a snapshot of the value)
- This can lead to interesting scoping issues

Lambda expressions

```
// recall Select(ch => ('a' <= ch && ch <= 'z') || ch == '\'' ? ch : ' ');

String s = "Al's 20 fat-ish oxen.";
Func<Char, Char> myfunc = (ch) => ('a' <= ch && ch <= 'z') || ch == '\'' ? ch : ' ';
IEnumerable<Char> iech = s.Select(myfunc);
// iech is now a deferred enumerator for the characters in: " l's    fat ish oxen "

Func<Char, Char> myfunc = (ch) =>
    {
        if ('a' <= ch && ch <= 'z')
            return ch;
        if (ch == '\'')
            return ch;
        return ' ';
    };

Func<String, int, bool> string_is_longer_than = (s, i) => s.Length > i;

bool b = string_is_longer_than("hello", 3);     // true
```

Closure example

```
int x = 3;
Action a = () =>
    {
        Console.WriteLine(x);
    };
a();            // prints 3
x = 5;
a();            // prints 5
```

State machine example

```
using System;
using System.Collections.Generic;
using System.Linq;

static class Program
{
    static class MainClass
    {
        enum State { Zero, One, Two };

        static Dictionary<State, Func<Char, State>> machine = new Dictionary<State, Func<Char, State>>
        {
            {
                State.Zero, (ch) => { return State.Two; }
            },
            {
                State.One, (ch) => { return State.One; }
            },
        };

        static void Main(String[] args)
        {
            String s = "the string to parse";
            int i = 0;

            State state = State.Zero;
            while (i < s.Length)
                state = machine[state](s[i++]);
        }
    }
}
```

A dictionary of lambda functions

The state machine

LINQ in C#

- Sequences: IEnumerable<T>
- Deferred execution
- Type inference
- Strong typing
  - despite the 'var' keyword
  - (C# 4.0 now has the `'dynamic'` keyword, which allows true runtime typing where desired)

LINQ operators

- Filter/Quantify:
  `Where, ElementAt, First, Last, OfType`
- Aggregate:
  `Count, Any, All, Sum, Min, Max`
- Partition/Concatenate:
  `Take, Skip, Concat`
- Project/Generate:
  `Select, SelectMany, Empty, Range, Repeat`
- Set:
  `Union, Intersect, Except, Distinct`
- Sort/Ordering:
  `OrderBy, ThenBy, OrderByDescending, Reverse`
- Convert/Render:
  `Cast, ToArray, ToList, ToDictionary`

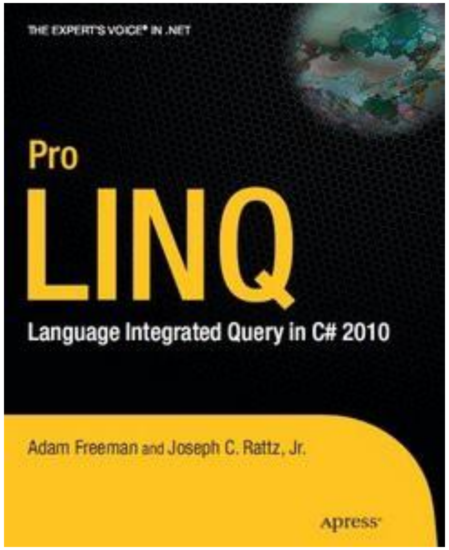
Example

```
Dictionary<String, int> pet_sym_map =
    File.ReadAllLines("/programming/analytical-grammar/erg-funcs/pet-symbol-key.txt")
    .Select(s => s.Split(sc7, StringSplitOptions.RemoveEmptyEntries))
    .Where(rgs => rgs.Length == 3)
    .Select(rgs => new { id = int.Parse(rgs[0]), sym = rgs[1].Trim().ToLower() })
    .GroupBy(a => a.sym)
    .Select(grp => grp.ArgMin(a => a.id))
    .ToDictionary(a => a.sym, a => a.id);
```

C# LINQ (Language Integrated Query)

- Declarative operations on sequences
- Recommendation:

Joseph C. Rattz, Jr. (2007) *Pro LINQ: Language Integrated Query in C# 2008*. Apress.



Programming Paradigms: Procedural

- Procedural (“imperative”) programming
  - FORTRAN (1954) grew out of hardware assembly languages, which are necessarily procedural
  - We explicitly specify the (synchronous) steps for doing something (i.e. an algorithm)

```
int Factorial(int n)
{
        int f = 1;
        for (int i = n; i > 1; i--)
                f *= i;
        return f;
}
```

Functional Programming

- A type of declarative programming
  - Constraint-based syntax formalisms such as unification grammars (LFG, HPSG, …) are also declarative
- Like function definitions in math, the “program” describes asynchronous relationships
- Functions are stateless and should have no side-effects
- Immutable values
  - As a bonus, this really facilitates concurrent programming
- Scheme, Haskell, F#

F# Example

- F# interactive on patas:

```
$ mono /opt/fsharp/bin/fsi.exe --gui-
          (you must use an ANSI terminal)
```

```
> let rec factorial = function
    | 0 -> 1
    | n -> n * factorial(n -
1);; val factorial : int -> int
> factorial 5;;
val it : int =
120
> #quit;;
```