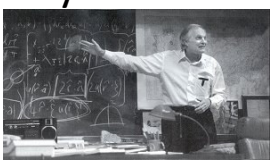
Lecture 9: Mostly Not About Matter, Music, and Mayhem

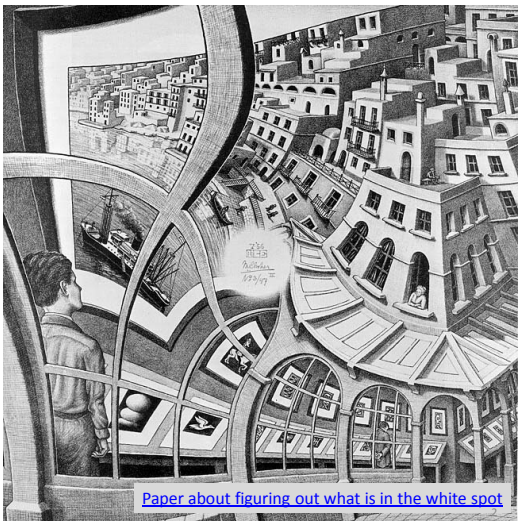
Richard Feynman's van


Alan Alda playing Richard Feynman in *QED*

cs1120 Fall 2009
David Evans
http://www.cs.virginia.edu/evans

Menu

- Finishing **find-closest**
- Mapping on Lists
- GEB Chapter V


Paper about figuring out what is in the white spot

Last class: find-closest-number

```
(define (find-closest-number goal p)
  (if (null? (cdr p))
      (car p)
      (if (< (abs (- goal (car p)))
             (abs (- goal (find-closest-number goal (cdr p)))))
          (car p)
          (find-closest-number goal (cdr p)))))
```


Avoiding Duplicate Work

```
(define (pick-closest goal a b)
  (if (< (abs (- goal a)) (abs (- goal b))) a b))

(define (find-closest-number goal p)
  (if (null? (cdr p))
      (car p)
      (pick-closest goal
        (car p)
        (find-closest-number goal (cdr p)))))
```

Generalizing find-closest-number

- How would we implement find-closest-number-without-going-over?
- What about find-closest-word?
- ...

The "closeness" metric should be a procedure parameter

find-minimizer

```
(define (pick-minimizer f a b)
  (if (< (f a) (f b)) a b))

(define (find-minimizer f p)
  (if (null? (cdr p))
      (car p)
      (pick-minimizer f (car p)
                      (find-minimizer f (cdr p)))))
```

find-minimizer takes two inputs: a procedure and a list. Its output is the element of the input list for which applying the procedure produces the lowest value.

Examples

```
> (find-minimizer (lambda (n) (abs (- 1120 n)))
                  (list 1130 1125 1203 1108))
1125
> (find-minimizer (lambda (n) (- n))
                  (list 1 2 4 8 16))
16
> (find-minimizer
   (lambda (n) (if (<= n 1120) (- 1120 n) 99999))
   (list 1130 1125 1203 1108))
1108
```


find-closest-number

```
(define (find-closest-number goal p)
   (find-minimizer
    (lambda (n) (abs (- goal n)))
    p))
```


Seen Anything Like This?

```
(define (find-best-match sample tiles color-comparator)
  (if (= (length tiles) 1)
     (car tiles)
     (pick-better-match
       sample
       (car tiles)
       (find-best-match
         sample
         (cdr tiles)
         color-comparator)
       color-comparator))))
```

```
(define (pick-better-match
          sample tile1 tile2
          color-comparator)
  (if (color-comparator sample
       (tile-color tile1) (tile-color tile2))
      tile1
      tile2))
```


Better find-best-match

```
(define (find-best-match sample tiles color-differencer)
  (find-minimizer
   (lambda (tile)
     (color-differencer tile sample))
   tiles))
```

```
(define (color-difference colora colorb)
  (+ (square (- (get-red colora) (get-red colorb)))
     (square (- (get-green colora) (get-green colorb)))
     (square (- (get-blue colora) (get-blue colorb)))))
```


GEB Chapter V

You could spend the rest of your life just studying things in this chapter (25 pages)!

- **Music Harmony**
- **Stacks and Recursion**
- Theology
- **Language Structure**
- **Number Sequences**
- Chaos
- **Fractals (PS3 out today)**
- **Quantum Electrodynamics (later lecture)**
- **DNA (later lecture)**
- Sameness-in-differentness
- **Game-playing algorithms (later lecture)**

Fibonacci's Problem

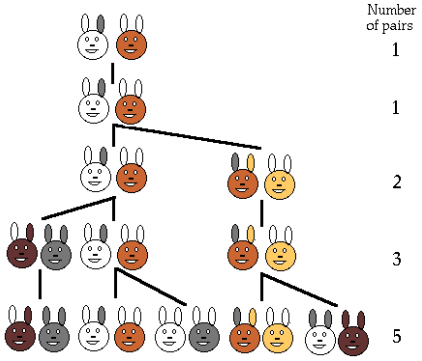
Filius Bonacci, 1202 in Pisa:

Suppose a newly-born pair of rabbits, one male, one female, are put in a field. Rabbits mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits.

Suppose that our rabbits **never die** and that the female **always** produces one new pair (one male, one female) **every month** from the second month on.

How many pairs will there be in one year?

Rabbits



From http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fibnat.html

Fibonacci Numbers

GEB p. 136:

These numbers are best defined recursively by the pair of formulas

$$\text{FIBO}(n) = \text{FIBO}(n-1) + \text{FIBO}(n-2) \quad \text{for } n > 2$$

$$\text{FIBO}(1) = \text{FIBO}(2) = 1$$

Can we turn this into a Scheme procedure?

Defining FIBO

1. Be optimistic - assume you can solve it, if you could, how would you solve a bigger problem.
2. Think of the simplest version of the problem, something you can already solve.
3. Combine them to solve the problem.

These numbers are best defined recursively by the pair of formulas

$$\text{FIBO}(n) = \text{FIBO}(n-1) + \text{FIBO}(n-2) \quad \text{for } n > 2$$

$$\text{FIBO}(1) = \text{FIBO}(2) = 1$$

Defining fibo

```
;;; (fibo n) evaluates to the nth Fibonacci
;;; number
(define (fibo n)
  (if (or (= n 1) (= n 2))
      1 ;;; base case
      (+ (fibo (- n 1))
         (fibo (- n 2)))))
```

$$\text{FIBO}(1) = \text{FIBO}(2) = 1$$

$$\text{FIBO}(n) = \text{FIBO}(n-1) + \text{FIBO}(n-2) \quad \text{for } n > 2$$

Fibo Results

```
> (fibo 2)
1
> (fibo 3)
2
> (fibo 4)
3
> (fibo 10)
55
> (fibo 60)
```

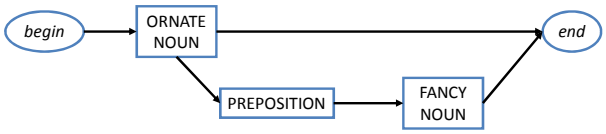
Still working after 4 hours…

Why can't our 4Mx Apollo Guidance Computer figure out how many rabbits there will be in 5 years?

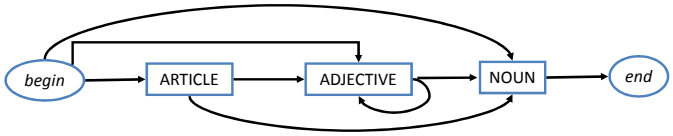
To be continued... (Chapter 7)

Recursive Transition Networks

(not quite as) FANCY NOUN:



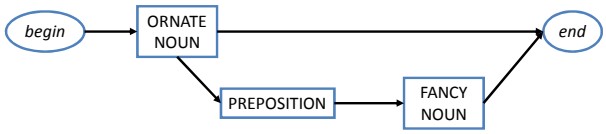
ORNATE NOUN:



"the strange bagels below the purple cow on the cloud"

Can we describe this using a BNF Grammar?

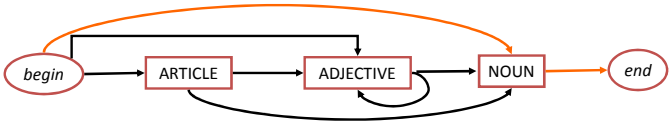
(not quite as) FANCY NOUN:



FANCY-NOUN ::= *ORNATE-NOUN*
FANCY-NOUN ::= *ORNATE-NOUN PREPOSITION FANCY-NOUN*
PREPOSITION ::= **below | on | above | under | …**

Recursive Transition Networks

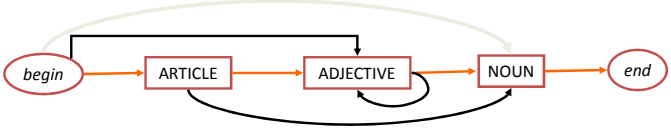
ORNATE NOUN



ORNATE-NOUN ::= *NOUN*

Recursive Transition Networks

ORNATE NOUN

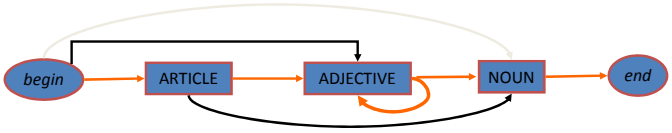


ORNATE-NOUN ::= *NOUN*

ORNATE-NOUN ::= *ARTICLE ADJECTIVE NOUN*

Recursive Transition Networks

ORNATE NOUN



ORNATE-NOUN ::= *ARTICLE ADJECTIVE NOUN*

ORNATE-NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE NOUN*

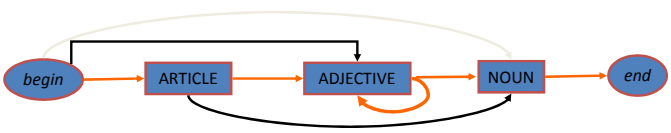
ORNATE-NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

ORNATE-NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

ORNATE-NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

Recursive Transition Networks

ORNATE NOUN



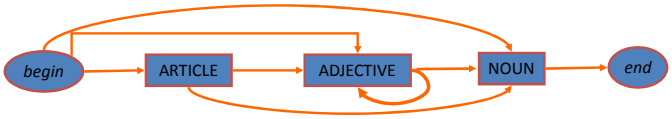
ORNATE-NOUN ::= *ARTICLE ADJECTIVES NOUN*

ADJECTIVES ::= *ADJECTIVE ADJECTIVES*

ADJECTIVES ::=

Recursive Transition Networks

ORNATE NOUN



ORNATE-NOUN ::= *OPTARTICLE ADJECTIVES NOUN*
ADJECTIVES ::= *ADJECTIVE ADJECTIVES*
ADJECTIVES ::= ε
OPTARTICLE ::= *ARTICLE*
OPTARTICLE ::= ε

BNF Grammar

ORNATE-NOUN ::= *OPTARTICLE ADJECTIVES NOUN*
ADJECTIVES ::= *ADJECTIVE ADJECTIVES*
ADJECTIVES ::= ε
OPTARTICLE ::= *ARTICLE*
OPTARTICLE ::= ε
FANCY-NOUN ::= *ORNATE-NOUN*
FANCY-NOUN ::= *ORNATE-NOUN PREPOSITION FANCY-NOUN*
PREPOSITION ::= **below | on | above | under | …**
ARTICLE ::= **the | a | an**
NOUN ::= **bagel | cow | cloud | …**

This demonstrates that one RTN can be converted to a BNF. For a proof that all RTNs can be converted to BNFs, we would need to formalize the transformation as a systematic and general algorithm.

Music Harmony

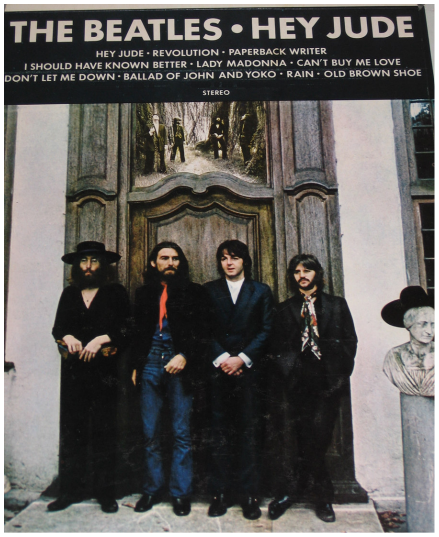
Kleines Harmonisches Labyrinth

(Little Harmonic Labyrinth)

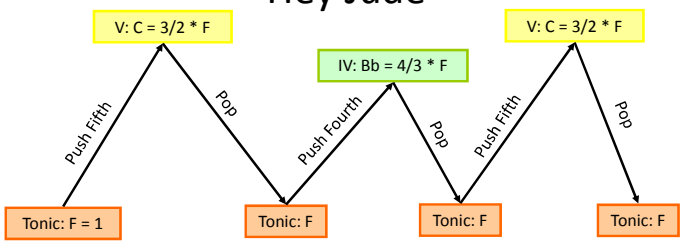
Not actually by Bach
(by Johann David Heinichen)

Hey Jude

John Lennon and
Paul McCartney,
1968



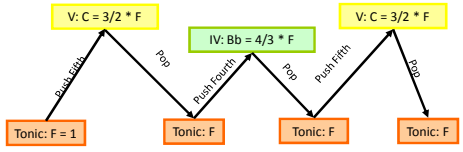
Hey Jude



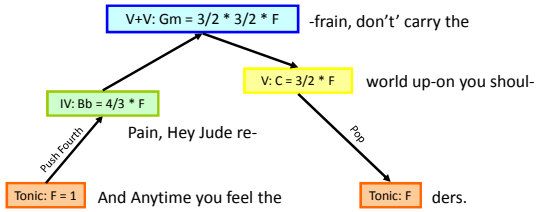
Tonic: Hey Jude, don't make it
V: bad. take a sad song and make it
Tonic: better Re-
IV: member to let her into your
Tonic: heart, then you can
V: start to make it bet-
Tonic: -ter.

Link

Verse ::=



Bridge ::=



HeyJude ::= *Verse VBBD VBBD Verse Verse Better Coda VBBD* ::=
Verse Bridge Bridge Dadada (ends on C)
Coda ::= F Eb Bb F *Coda*

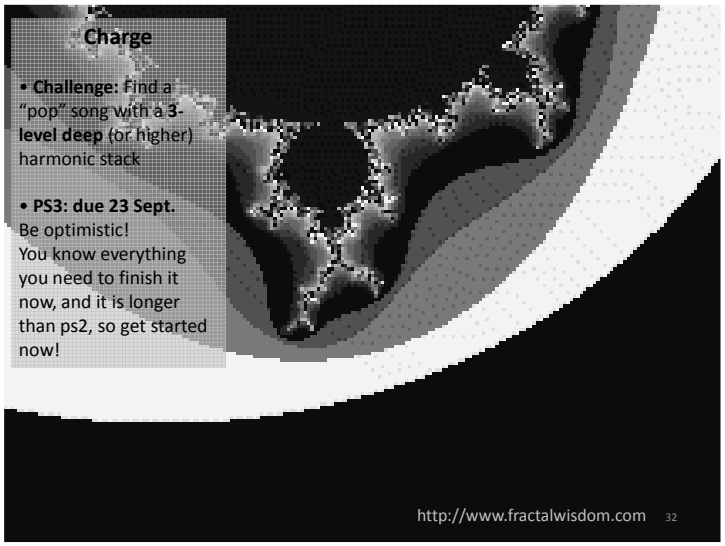
Music

- Almost All Music Is Like This
  - Pushes and pops the listener's stack, but doesn't go too far away from it
  - Repeats similar patterns in structured way
  - Keeps coming back to Tonic, and Ends on the Tonic
- Any famous Beatles song that doesn't end on Tonic?

"A Day in the Life" (starts on G, ends on E)

Charge

- **Challenge:** Find a "pop" song with a **3-level deep** (or higher) harmonic stack
- **PS3: due 23 Sept.**
  Be optimistic!
  You know everything you need to finish it now, and it is longer than ps2, so get started now!



http://www.fractalwisdom.com