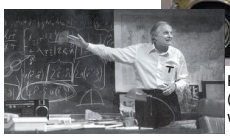
Lecture 8: Recursing Recursively

Now playing: JS Bach, *The Art of Fugue*



Richard Feynman's Van (parked outside the theater where QED is playing)

Alan Alda playing Richard Feynman in QED

CS150: Computer Science
University of Virginia
Computer Science

David Evans
http://www.cs.virginia.edu/evans

Menu

- Recursive Procedures
- GEB Chapter V
  - –Fibonacci Returns
  - –RTNs
  - –Music and Recursion

Defining Recursive Procedures

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. (This is the **base case**.)
3. Combine them to solve the problem.

Example

Define (find-closest goal numbers) that evaluates to the number in the list numbers list that is closest to goal:

```
> (find-closest 200 (list 101 110 120 201 340 588))
201
> (find-closest 12 (list 1 11 21))
11
> (find-closest 12 (list 95))
95
```


Find Closest Number

Be optimistic!

Assume you can define:

(find-closest-number goal numbers)

that finds the closest number to goal from the list of numbers.

What if there is one more number?

Can you write a function that finds the closest number to match from new-number and numbers?

Find Best Match

Strategy:

If the new number is better, than the best match with the other number, use the new number. Otherwise, use the best match of the other numbers.

Optimistic Function

```
(define (find-closest goal numbers)
  (if (< (abs (- goal (first numbers)))
         (abs (- goal
                 (find-closest goal (rest numbers)))))
      (first numbers)
      (find-closest goal (rest numbers))))
```


Defining Recursive Procedures

2. Think of the simplest version of the problem, something you can already solve.

If there is only one number, that is the best match.

The Base Case

```
(define (find-closest goal numbers)
  (if (= 1 (length numbers))
      (first numbers)
      (if (< (abs (- goal (first numbers)))
             (abs (- goal
                     (find-closest goal (rest numbers)))))
          (first numbers)
          (find-closest goal (rest numbers)))))
```

Same as before

Testing

```
(define (find-closest goal numbers)
  (if (= 1 (length numbers))
      (first numbers)
      (if (< (abs (- goal (first numbers)))
             (abs (- goal
                     (find-closest goal (rest numbers)))))
          (first numbers)
          (find-closest goal (rest numbers)))))
```

```
> (find-closest-number 200
    (list 101 110 120 201 340 588))
201
> (find-closest-number 0 (list 1))
1
> (find-closest-number 0 (list ))
first: expects argument of type <non-empty list>; given ()
```


Seen Anything Like This?

```
(define (find-best-match sample tiles color-comparator)
  (if (= (length tiles) 1)          ;;; If there is just one tile,
      (first tiles)                 ;;;    that tile is the best match.
      (pick-better-match            ;;; Otherwise, the best match is
        sample                      ;;; either the first tile in tiles,
        (first tiles)               ;;; or the best match we would find
        (find-best-match            ;;; from looking at the rest of the
          sample                    ;;; tiles.  Use pick-better-match
          (rest tiles)              ;;; to determine which one is better.
          color-comparator)
        color-comparator))))
```


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 (late lecture)
- DNA (next to last lecture)
- Sameness-in-differentness
- Game-playing algorithms (upcoming lecture)

Fibonacci's Problem

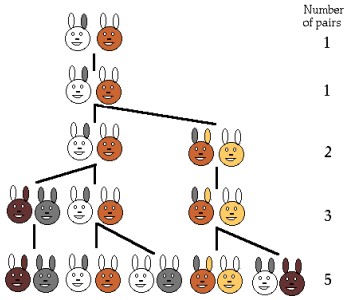
Filius Bonacci, 1202 in Pisa:

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

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

$$FIBO(1) = FIBO(2) = 1$$

Can we turn this into a Scheme procedure?

Note: SICP defines Fib with Fib(0)= 0 and Fib(1) = 1 for base case. Same function except for Fib(0) is undefined in GEB version.

Defining Recursive Procedures

Slide 3 Returns…

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. (This is the **base case**.)
3. Combine them to solve the problem.

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

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

$$FIBO(1) = FIBO(2) = 1$$

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)))))
```

$$FIBO(1) = FIBO(2) = 1$$

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

Fibo Results

```
> (fibo 2)
1
> (fibo 3)
2
> (fibo 4)
3
> (fibo 10)
55
> (fibo 100)
```

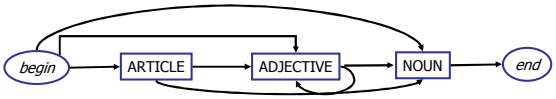
Still working after 4 hours…

Why can't our 1Mx Apollo Guidance Computer calculate (fibo 100)?

To be continued Monday (answer is in SICP, 1.2)

Recursive Transition Networks

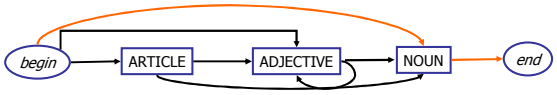
ORNATE NOUN



Can we describe this using Backus Naur Form?

Recursive Transition Networks

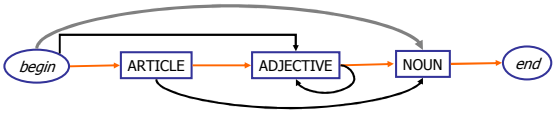
ORNATE NOUN



ORNATE NOUN ::= *NOUN*

Recursive Transition Networks

ORNATE NOUN

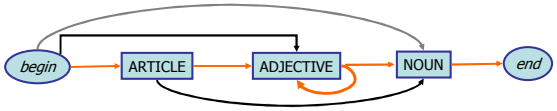


ORNATE NOUN ::= *NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE NOUN*

Recursive Transition Networks

ORNATE NOUN



ORNATE NOUN ::= *ARTICLE ADJECTIVE NOUN*

ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE NOUN*

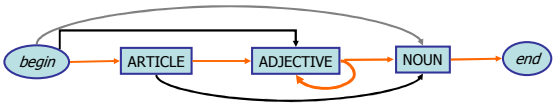
ORNATE NOUN ::= *ARTICLE ADJECTIVE ADJECTIVE ADJECTIVE NOUN*

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

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

Recursive Transition Networks

ORNATE NOUN



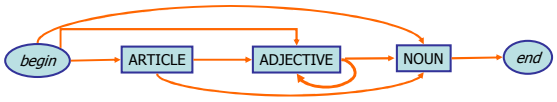
ORNATE NOUN ::= *ARTICLE ADJECTIVES NOUN*

ADJECTIVES ::= *ADJECTIVE ADJECTIVES*

ADJECTIVES ::=

Recursive Transition Networks

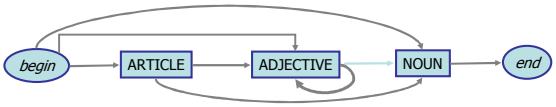
ORNATE NOUN



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

Recursive Transition Networks

ORNATE NOUN



ORNATE NOUN ::= [*ARTICLE*] *ADJECTIVE* NOUN*

Using extended BNF notation:

[*item* **]** *item* is optional (0 or 1 of them)
*item** 0 or more *item*s

Which notation is *better*?

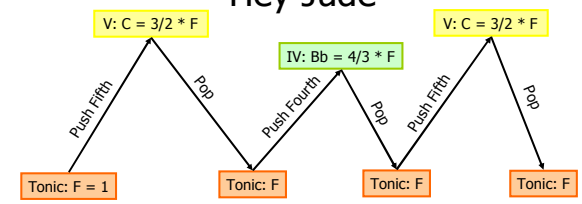
Music Harmony

Kleines Harmonisches Labyrinth
(Little Harmonic Labyrinth)

Hey Jude

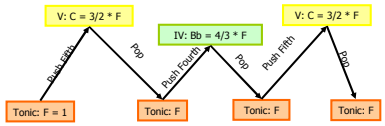
John Lennon and Paul McCartney, 1968

Hey Jude

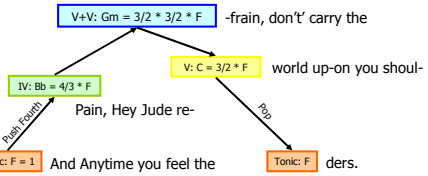


Tonic: Hey Jude, don't make it
V: bad. take a sad song and make it
Tonic: better Re-
IV: member to let her into your
Tonic: heart, then you can
V: start to make it bet-
Tonic: -ter.

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?

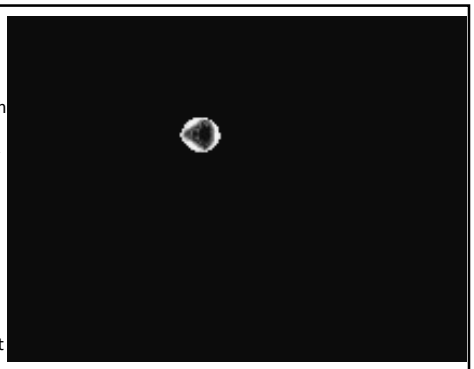
"A Day in the Life" (starts on G, ends on E)

Charge

- **Challenge:** Try to find a "pop" song with a 3-level deep harmonic stack
- **PS3:** due 10 days from today

Be optimistic!

You know everything you need to finish it now, so get started!



http://www.fractalwisdom.com/FractalWisdom/fractal.html