Recursion

CSE 215, Foundations of Computer Science

Stony Brook University

http://www.cs.stonybrook.edu/~cse215

Recursion: Sequences

- A sequence can be defined in 3 ways:
  - enumeration: -2,3,-4,5,...
  - general pattern: $a_n=(-1)^n(n+1)$, for all integers $n \geq 1$
  - recursion: $a_1=-2$ and $a_n=(-1)^{n-1}\, a_{n-1} + (-1)^n$
    - **define one or more initial values for the sequence AND**
    - **define each later term in the sequence by reference to earlier terms**
- A **recurrence relation** for a sequence $a_0, a_1, a_2, \ldots$ is a formula that relates each term a_k to certain of its predecessors $a_{k-1}, a_{k-2}, \ldots, a_{k-i}$, where i is an integer with $k-i \geq 0$
- The **initial conditions** for a recurrence relation specify the values of $a_0, a_1, a_2, \ldots, a_{i-1}$, if i is a fixed integer, OR
  $a_0, a_1, \ldots, a_m$, where m is an integer with $m \geq 0$, if i depends on k.

Recursion

- **Computing Terms of a Recursively Defined Sequence:**
  - Example:

initial conditions: $c_0 = 1$ and $c_1 = 2$

recurrence relation: $c_k = c_{k-1} + k\, c_{k-2} + 1$, for all integers $k \geq 2$

$c_2 = c_1 + 2\, c_0 + 1$ by substituting $k = 2$ into the recurrence relation

$= 2 + 2\cdot 1 + 1$ since $c_1 = 2$ and $c_0 = 1$ by the initial conditions

$= 5$

$c_3 = c_2 + 2\, c_1 + 1$ by substituting $k = 3$ into the recurrence relation

$= 5 + 3\cdot 2 + 1$ since $c_2 = 5$ and $c_1 = 2$

$= 12$

$c_4 = c_3 + 2\, c_2 + 1$ by substituting $k = 4$ into the recurrence relation

$= 12 + 3\cdot 5 + 1$ since $c_3 = 12$ and $c_2 = 5$

$= 33$

Recursion

- **Writing a Recurrence Relation in More Than One Way:**
  - Example:

initial condition: $s_0 = 1$

recurrence relation 1: $s_k = 3s_{k-1} - 1$, for all integers $k \geq 1$

recurrence relation 2: $s_{k+1} = 3s_k - 1$, for all integers $k \geq 0$

Recursion

- **Sequences That Satisfy the Same Recurrence Relation:**
  - Example:

initial conditions: $a_1 = 2$ and $b_1 = 1$

recurrence relations: $a_k = 3a_{k-1}$ and $b_k = 3b_{k-1}$ for all integers $k \geq 2$

$a_2 = 3a_1 = 3 \cdot 2 = 6$ $\quad$ $b_2 = 3b_1 = 3 \cdot 1 = 3$

$a_3 = 3a_2 = 3 \cdot 6 = 18$ $\quad$ $b_3 = 3b_2 = 3 \cdot 3 = 9$

$a_4 = 3a_3 = 3 \cdot 18 = 54$ $\quad$ $b_4 = 3b_3 = 3 \cdot 9 = 27$

Recursion

- Fibonacci numbers:
  1. We have one pair of rabbits (male and female) at the beginning of a year.
  2. Rabbit pairs are not fertile during their first month of life but thereafter give birth to one new male&female pair at the end of every month.

$$\begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k\end{bmatrix} = \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k-1\end{bmatrix} + \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs born}\\ \text{at the end}\\ \text{of month } k\end{bmatrix}$$

$$= \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k-1\end{bmatrix} + \begin{bmatrix}\text{the number}\\ \text{of rabbit}\\ \text{pairs alive}\\ \text{at the end}\\ \text{of month } k-2\end{bmatrix}$$

Recursion

- Fibonacci numbers:

The initial number of rabbit pairs: $F_0 = 1$

F_n : the number of rabbit pairs at the end of month n, for each integer $n \geq 1$

$\mathbf{F_n = F_{n-1} + F_{n-2}}$ **, for all integers $\mathbf{k \geq 2}$**

$F_1 = 1$, because the first pair of rabbits is not fertile until the second month

How many rabbit pairs are at the end of one year?

January 1st: $F_0 = 1$

February 1st: $F_1 = 1$

March 1st : $F_2 = F_1 + F_0 = 1+1= 2$

April 1st : $F_3 = F_2 + F_1 = 2+1= 3$

May 1st : $F_4 = F_3 + F_2 = 3+2= 5$

June 1st : $F_5 = F_4 + F_3 = 5+3= 8$

July 1st : $F_6 = F_5 + F_4 = 8+5= 13$

August 1st : $F_7 = F_6 + F_5 = 13+8= 21$

September 1st : $F_8=F_7+F_6=21+13=34$

October 1st : $F_9=F_8+F_7=34+21=55$

November 1st : $F_{10}=F_9+F_8=55+34=89$

December 1st : $F_{11}=F_{10}+F_9=89+55=144$

January 1st : $F_{12}=F_{11}+F_{10}=144+89=\mathbf{233}$

Recursion

- **Compound Interest:**
  - A deposit of \$100,000 in a bank account earning 4% interest compounded annually:

the amount in the account at the end of any particular year =

the amount in the account at the end of the previous year +

the interest earned on the account during the year

= the amount in the account at the end of the previous year +

0.04 · the amount in the account at the end of the previous year

$A_0 = \$100{,}000$

$A_k = A_{k-1} + (0.04) \cdot A_{k-1} = 1.04 \cdot A_{k-1}$, for each integer $k \geq 1$

$$A_1 = 1.04 \cdot A_0 = \$104{,}000$$

$$A_2 = 1.04 \cdot A_1 = 1.04 \cdot \$104{,}000 = \$108{,}160$$

...

Recursion

- **Compound Interest with Compounding Several Times a Year:**
  - An annual interest rate of i is compounded m times per year:

  the interest rate paid per each period is i/m

  P_k is the sum of the the amount at the end of the $(k - 1)$ period

  $+$ the interest earned during k-th period

  $P_k = P_{k-1} + P_{k-1} \cdot i/m = P_{k-1} \cdot (1 + i/m)$

  - If 3% annual interest is compounded quarterly, then the interest rate paid per quarter is $0.03/4 = 0.0075$

Compound Interest

- Example: deposit of \$10,000 at 3% compounded quarterly

For each integer $n \geq 1$, P_n = the amount on deposit after n consecutive quarters.

$P_k = 1.0075 \cdot P_{k-1}$

$P_0 = \$10{,}000$

$P_1 = 1.0075 \cdot P_0 = 1.0075 \cdot \$10{,}000 = \$10{,}075.00$

$P_2 = 1.0075 \cdot P_1 = (1.0075) \cdot \$10{,}075.00 = \$10{,}150.56$

$P_3 = 1.0075 \cdot P_2 \sim (1.0075) \cdot \$10{,}150.56 = \$10{,}226.69$

$P_4 = 1.0075 \cdot P_3 \sim (1.0075) \cdot \$10{,}226.69 = \$10{,}303.39$

The annual percentage rate (APR) is the percentage increase in the value of the account over a one-year period:

$APR = (10303.39 - 10000) / 10000 = 0.03034 = 3.034\%$

Recursive Definitions of Sum and Product

- The summation from i=1 to n of a sequence is defined using recursion:

$$\sum_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \sum_{i=1}^{n} a_i = \left(\sum_{i=1}^{n-1} a_i\right) + a_n, \quad \text{if } n > 1.$$

- The product from i=1 to n of a sequence is defined using recursion:

$$\prod_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \prod_{i=1}^{n} a_i = \left(\prod_{i=1}^{n-1} a_i\right) \cdot a_n, \quad \text{if } n > 1.$$

Sum of Sums

- For any positive integer n, if $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are real numbers, then

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i .$$

- Proof by induction

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i . \qquad \leftarrow P(n)$$

  - base step: $$\sum_{i=1}^{1}(a_i + b_i) = a_1 + b_1 = \sum_{i=1}^{1} a_i + \sum_{i=1}^{1} b_i$$

  - inductive hypothesis: $$\sum_{i=1}^{k}(a_i + b_i) = \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i . \qquad \leftarrow P(k)$$

Sum of Sums

- Cont.: We must show that:

$$\sum_{i=1}^{k+1}(a_i + b_i) = \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i. \qquad \leftarrow P(k+1)$$

$$\begin{aligned}
\sum_{i=1}^{k+1}(a_i + b_i) &= \sum_{i=1}^{k}(a_i + b_i) + (a_{k+1} + b_{k+1}) && \text{by definition of } \Sigma \\
&= \left(\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i\right) + (a_{k+1} + b_{k+1}) && \text{by inductive hypothesis} \\
&= \left(\sum_{i=1}^{k} a_i + a_{k+1}\right) + \left(\sum_{i=1}^{k} b_i + b_{k+1}\right) && \text{by the associative and cummutative laws of algebra} \\
&= \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i && \text{by definition of } \Sigma
\end{aligned}$$

Q.E.D.

Recursion

- **Arithmetic sequence**: there is a constant d such that

  $a_k = a_{k-1} + d$ for all integers $k \geq 1$

It follows that, $a_n = a_0 + d*n$ for all integers $n \geq 0$.

- **Geometric sequence**: there is a constant r such that

  $a_k = r * a_{k-1}$ for all integers $k \geq 1$

It follows that, $a_n = r^n * a_0$ for all integers $n \geq 0$.

Recursion

- A **second-order linear homogeneous recurrence relation** with constant coefficients is a recurrence relation of the form:

$$a_k = A * a_{k-1} + B * a_{k-2}$$

for all integers $k \geq$ some fixed integer

where A and B are fixed real numbers with $B = 0$.

Recursively Defined Sets

1. Identify a few core objects as belonging to the set AND
2. Give rules showing how to build new set elements from old

- A recursive definition for a set consists of:

I. BASE: A statement that certain objects belong to the set.

II. RECURSION: A collection of rules indicating how to form new set objects from those already known to be in the set.

III. RESTRICTION: A statement that no objects belong to the set other than those coming from I and II.

Recursive Definition of Boolean Expressions

- The set of Boolean expressions over a general alphabet is defined recursively:

I. BASE: Each symbol of the alphabet is a Boolean expression.

II. RECURSION: If P and Q are Boolean expressions, then so are:

(a) $(P \land Q)$ and

(b) $(P \lor Q)$ and

(c) $\sim P$.

III. RESTRICTION: There are no Boolean expressions over the alphabet other than those obtained from I and II.

Recursive Definition of Boolean Expressions

- Example: the following is a Boolean expression over the English alphabet {a, b, c, . . . , x, y, z}:

$$(\sim(p \land q) \lor (\sim r \land p))$$

(1) By I, p, q, and r are Boolean expressions.

(2) By (1) and II(a) and (c), $(p \land q)$ and $\sim r$ are Boolean expressions.

(3) By (2) and II(c) and (a), $\sim(p \land q)$ and $(\sim r \land p)$ are Boolean expressions.

(4) By (3) and II(b), $(\sim(p \land q) \lor (\sim r \land p))$ is a Boolean expression.

Recursive String Definitions

- A *string over S* (a finite set with at least one element) is a finite sequence of elements from S.
  - The elements of S are called *characters of the string*.
  - The *length of a string* is the number of characters it contains.
  - The *null string over S* is defined to be the "string" with no characters.
    - It is usually denoted **ε** (epsilon) and is said to have length 0.

Recursive String Definitions

- Example: the Set of Strings over an Alphabet:
  - Consider the set S of all strings in a's and b's - S is defined recursively as:

I. BASE: $\boldsymbol{\varepsilon}$ is in S, where $\boldsymbol{\varepsilon}$ is the null string.

II. RECURSION: If s ∈ S, then

(a) sa ∈ S and (b) sb ∈ S,

where sa and sb are the concatenations of s with a and b.

III. RESTRICTION: Nothing is in S other than objects defined in I and II above.

Derive the fact that ab ∈ S.

Recursive String Definitions

Derive the fact that ab ∈ S.

(1) By I, **ε** ∈ S.

(2) By (1) and II(a), **ε**a ∈ S. But **ε**a is the concatenation of the null string **ε** and a, which equals a. So a ∈ S.

(3) By (2) and II(b), ab ∈ S.

- The ***MIU-system***:

I. BASE: MI is in the MIU-system.

II. RECURSION:

a. If x I is in the MIU-system, where x is a string, then x I U is in the MIU-system = i.e., we can add a U to any string that ends in I. For example, since MI is in the system, so is MIU.

b. If Mx is in the MIU-system, where x is a string, then Mxx is in the MIUsystem = i.e., we can repeat all the characters in a string that follow an initial M. For example, if MUI is in the system, so is MUIUI.

c. If x I I I y is in the MIU-system, where x and y are strings (possibly null), then xUy is also in the MIU-system = i.e., we can replace I I I by U. For example, if M I I I I is in the system, so are MIU and MUI.

d. If xUUy is in the MIU-system, where x and y are strings (possibly null), then xUy is also in the MIU-system = i.e., can replace UU by U. For example, if MIIUU is in the system, so is MIIU.

III. RESTRICTION: No strings other than those derived from I and II are in the MIUsystem.

Derive the fact that MUIU is in the MIU-system:

(1) By I, MI is in the MIU-system.

(2) By (1) and II(b), M I I is in the MIU-system.

(3) By (2) and II(b), M I I I I is in the MIU-system.

(4) By (3) and II(c), MUI is in the MIU-system.

(5) By (4) and II(a), MUIU is in the MIU-system.

- Legal ***Parenthesis Structures***:

I. BASE: () is in P.

II. RECURSION:

a. If E is in P, so is (E).

b. If E and F are in P, so is EF.

III. RESTRICTION: No configurations of parentheses are in P other than those derived from I and II above.

Derive the fact that (())() is in P:

(1) By I, () is in P.

(2) By (1) and II(a), (()) is in P.

(3) By (2), (1), and II(b), (())() is in P.

Structural Introduction for Recursively Defined Sets

- Let S be a set that has been defined recursively, and consider a property that objects in S may or may not satisfy.

To prove that every object in S satisfies the property:

1. Show that each object in the BASE for S satisfies the property;
2. Show that for each rule in the RECURSION, if the rule is applied to objects in S that satisfy the property, then the objects defined by the rule also satisfy the property.

Because no objects other than those obtained through the BASE and RECURSION conditions are contained in S, it must be the case that every object in S satisfies the property.

Legal Parenthesis Structures

I. BASE: () is in P.

II. RECURSION:

a. If E is in P, so is (E).

b. If E and F are in P, so is EF.

III. RESTRICTION: No configurations of parentheses are in P other than those derived from I and II above.

- **Every configuration in P contains an equal number of left and right parentheses:**

Property: any parenthesis configuration has an equal number of left and right parentheses!

Show that each object in the BASE for P satisfies the property: The only object in the base for P is (), which has one left parenthesis and one right parenthesis.

Show that for each rule in the RECURSION for P, if the rule is applied to an object in P that satisfies the property, then the object defined by the rule also satisfies the property:

The recursion for P consists of two rules denoted II(a) and II(b).

Suppose E and F are parenthesis configurations that have equal numbers of left and right parentheses.

When rule II(a) is applied to E, the result is (E), so both the number of left parentheses and the number of right parentheses are increased by one → same number of parenthesis.

When rule II(b) is applied, the result is EF, which has an equal number, m(in E) + n(in F), of left and right parentheses.

Recursive Functions

- **McCarthy's 91 Function:** $M : \mathbf{Z}^+ \rightarrow \mathbf{Z}$

$$M(n) = \begin{cases} n - 10 \text{ if } n > 100 \\ M(M(n + 11)) \text{ if } n \leq 100 \end{cases}$$

$$\begin{aligned} M(99) &= M(M(110)) && \text{since } 99 \leq 100 \\ &= M(100) && \text{since } 110 > 100 \\ &= M(M(111)) && \text{since } 100 \leq 100 \\ &= M(101) && \text{since } 111 > 100 \\ &= 91 && \text{since } 101 > 100 \end{aligned}$$

Recursive Functions

- **The Ackermann Function:**

$A(0, n) = n + 1$ for all nonnegative integers n (1)

$A(m, 0) = A(m - 1, 1)$ for all positive integers m (2)

$A(m, n) = A(m - 1, A(m, n - 1))$ for all positive integers m and n (3)

$$
\begin{aligned}
A(1, 2) &= A(0, A(1, 1)) && \text{by (3) with } m = 1 \text{ and } n = 2 \\
&= A(0, A(0, A(1, 0))) && \text{by (3) with } m = 1 \text{ and } n = 1 \\
&= A(0, A(0, A(0, 1))) && \text{by (2) with } m = 1 \\
&= A(0, A(0, 2)) && \text{by (1) with } n = 1 \\
&= A(0, 3) && \text{by (1) with } n = 2 \\
&= 4 && \text{by (1) with } n = 3.
\end{aligned}
$$

$A(n, n)$ increases with extraordinary rapidity: $A(4, 4) \cong 2^{2^{2^{65536}}}$