コンピュータ概論 A/B

-- Mathematica での関数定義 --

数学科 栗野 俊一

2011/11/01 コンピュータ概論

伝言

私語は慎むように !!

- 教室に入ったら
  - 直に Note-PC の電源を入れておく
    - Network に接続し、当日の資料に目を通す
    - skype に Login する
    - Windows Update をしておこう
- やる気のある方へ
  - 今日の資料は、すでに上っています
    - どんどん、先に進んでかまいません
- mathematica
  - インストールができなかった人
    - 5 限に業者の方がきてくれそうなので、センターに行く

追加のお知らせ

- 講議録画
  - 毎週、講議の内容が録画され Network 経由で公開されます。
  - URL

    http://10.9.74.133/video (学内からのみ)

    http://edu-gw2.math.cst.nihon-u.ac.jp:10082 (学外からも[暫定])
  - 認証(メモってよい:その内、情報センターの ID/PW にします)
    - ID: m23b
    - PW: 3cdfa4747

先週の復習

- 先週の内容
  - 講義
    - Mathematica によるプログラミング基礎
  - 実習
    - [演習 1] Mathematica の変数の利用法
    - [演習 2] Mathematica の関数の作成方法
- 講義内容
  - コンピュータに命令を伝える為には「言語」が必要
    - 「言語」を定義付け : 構文(表現)と意味
  - Mathematica Notebook (*.nb)
    - Mathematica の計算結果を保存する形式
    - 人間の指示(In[##])とその結果(Out[##])が記録されている
    - 結果は %## で参照可能
  - 変数への代入と定義
    - 変数:値を保存でき、参照できる
    - 代入には二種類ある (「=」: 式を評価 /「:=」:式評価しない)
  - 関数の定義
    - 「関数名[引数_] := 式」で関数が定義できる

本日の予定

- 講義
  - Mathematica による関数定義
  - 数の定義
    - 自然数 (ペアノの公理)
    - 自然数を利用した、整数、有理数の定義
- 実習
  - [演習 1] 数を処理する関数
  - [演習 2] 課題の作成

本日の課題 (2011/11/01)

- 前回 (2011/10/25) の課題
  - 今週の課題にする
- 今週 (2011/11/01) の課題
  - 次のファイルを提出しなさい
    - 表題：ペアノの方法による「有理数の差」の関数 qsub を定義しなさい
    - ファイル名：20111025-QQQQ.nb (QQQQ は学生番号)
    - 詳しくは、配布した nat.txt/sample-20111101.nb の内容を参照

関数の利用

- □ 関数
  - ○ 関数とは
    - ▹ 「シンボル[引数列]」の形で表現され、値を持つ事ができる「もの」
    - ▹ cf. Sin[Pi] / f[3] / g[x,y^4]
  - ○ 関数の評価
    - ▹ 「シンボル[引数列]」の形を、その値に置き換える
- □ 引数のパターンマッチ
  - ○ 「_」(アンダースコア)を利用して「式」の「抽象パターン」が表現できる
  - ○ 例1
    - ▹ next[0] := 1
    - ▹ next[1] := 2
    - ▹ next[_] := 0
  - ○ 例2 (再帰的定義)
    - ▹ fib[1]=1
    - ▹ fib[2]=1
    - ▹ fib[x_]:=fib[x-1]+fib[x-2]

関数の再帰的定義

- 関数定義の基本
  - 既に定義済の関数を利用して、新しい関数を定義する
    - 例: myTan[x_] := Sin[x]/Cos[x]
    - Sin[x], Cos[x] は既に定義済
    - それを利用して myTan という新しい関数を定義
- 関数の再帰的定義
  - 関数の定義の内容の中にその関数自身を利用 (「=」を使ってはならない !!)
  - パターンマッチを利用し、「引数」が単純になるようにする
    - 例: pfac[0] := 1 / pfac[s[x_]] := pmul[s[x],pfac[x]]
    - pfac の定義(右辺)に pfac が表れている
    - 引数が単純になっている (左辺は s[x], 右辺は x) ので..
    - pfac[0] の定義がある点も重要

ユークリッドの互除法

- ユークリッドの互除法
  - 二つの自然数 m, n の最大公約数を求める方法
    - m > n の時に m を n で割った余りを r とする
    - r が 0 の時には n が最大公約数
    - r > 0 の時、m と n の最大公約数は、n と r の最大公約数となる
- ポイント
  - m,n の最大公約数が、n,r の最大公約数になっている
    - 定義が帰納的に行われている
- Mathematica での GCM の定義
  - GCM[m,0] := m
  - GCM[0,n] := m
  - GCM[m_,n_] := GCM[n,Mod[m,n]]
- 再帰的定義のポイント
  - 左辺より右辺が「小く」なっている (単調減少)
  - 最小の要素 (0) の結果は、再帰を使わずに求められる (下界の存在)

自然数

- ペアノの公理
  - 自然数を定義する公理
    - 0 は自然数である
    - n が自然数なら n+1 も自然数である
    - 上記の二つ以外に自然数はない
- Mathematica でペアノの公理の自然数を考える
  - 自然数
    - 0 は 0 で表現
    - n + 1 は s[n] で表現
  - 自然数の和
    - padd[0,x_] := x
    - padd[s[x_],y_] := s[padd[x,y]]
  - これを繰り返すと、分数まで表現できる

自然数による数の表現 (nat.txt)

- 自然数 (数の記号的な定義)
  - 0 と +1 (サクセッサー) のみで作る
    - 3 = 0+1+1+1 = s[s[s[0]]]
- 整数
  - 自然数の二つ組 (m,n) で整数 z を表現する
    - z = m - n => pp[m,n]
    - 同じ整数に対する異なる(m,n) 組があるので、同値類(～)を作る
    - 例 pp[4,2] ～ pp[3,1] ～ pp[2,0]
- 有理数
  - 自然数と整数の組 (z,n) で整数 q を表現する
    - q = z/n => qq[z,n]
    - 同じ有理数に対する異なる(z,n) 組があるので、同値類(～)を作る
    - 例 qq[18,12] ～ pp[9,4] ～ pp[6,2]
- 記号的な数
  - パターンマッチ + 再帰で処理できる

コーディング

- □ コーディングとは
  - ○「A を B でコーディングする」
    - ▹「A の要素 a を B の要素 b1,b2, .. の組み合わせと対応させる」
  - ○「b1,b2,..の組み合わせ」を「a の(B による..)表現」と呼ぶ
  - ○ 例 1: (A:漢数字(の集合), B:アラビア数字(の集合))

    「零」→「0」、「一」→「1」、「二」→「2」

    「十」→「10」、「百」→「100」、「千」→「1000」
  - ○ 例 2: (A:図形, B:ドット並び)

    「A] →「00010000

    00010100

    00111110

    00100010

    00100010」

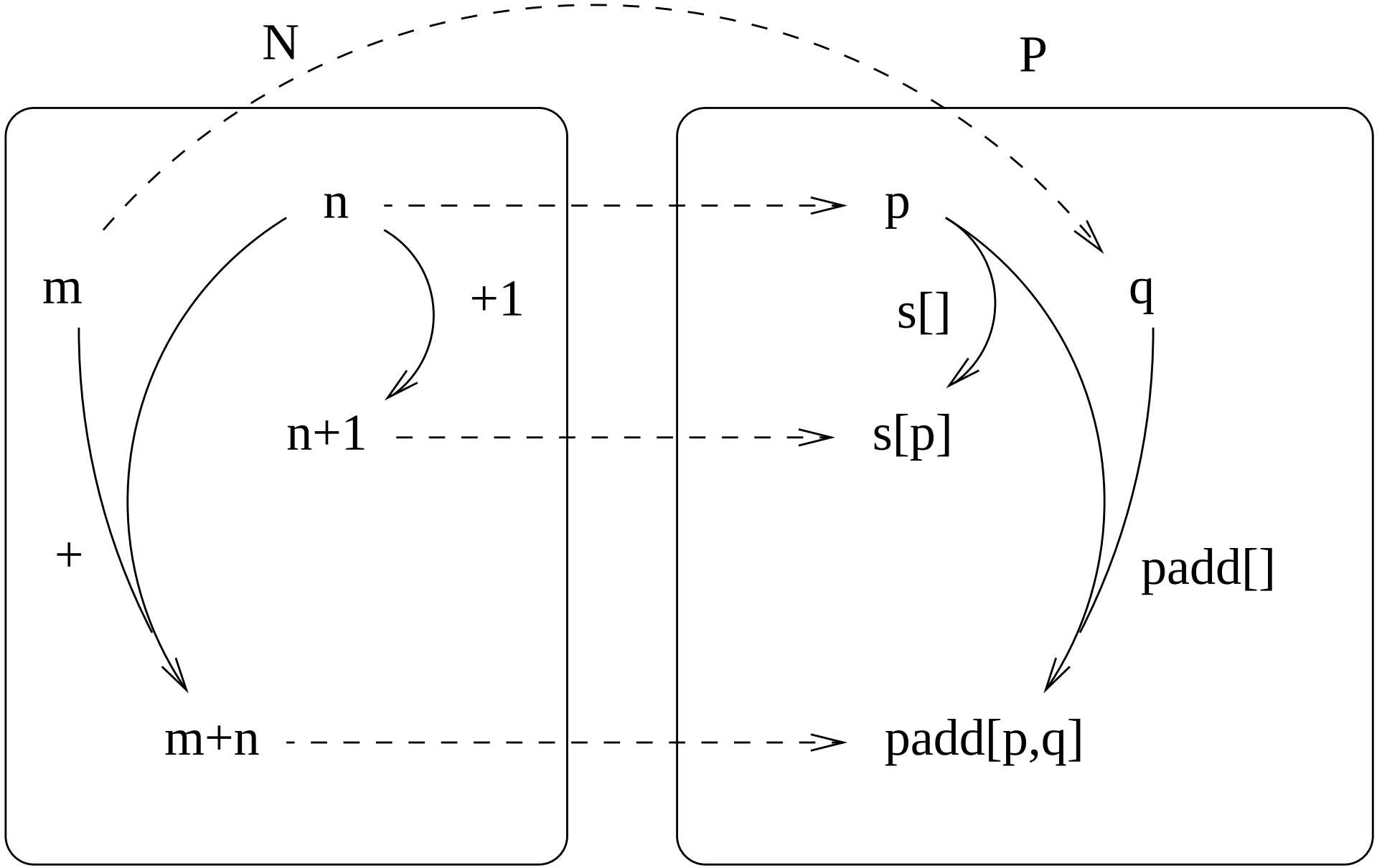
コーディングの仕方

- ○ 直積を作る (構文規則 : このように表現する)
  - ▹ 要素を増やす方法
- ○ 同値関係で割る (意味規則 : これとこれは同じ物)
  - ▹ 要素を減らす方法
- ○ 例 : N → Z

同型、コーディング、オブジェクト志向

- 「物(オブジェクト)」の考え方
  - それは「何であるか (What)」が本質だが...
  - それは「どう使えるか(How)」と捉える
    - 「家鴨の様によちよち歩き、家鴨の様にガーガー鳴くならそれは家鴨だ」
    - # 同じ様に振る舞う物は同じ物と考えて良い
- 「同型」
  - (今自分が着目している点に関して..) 振舞が同じ集合は(その観点で..)同じ
    - 「A と B が同型」という事は「A と B は異なる(what) だが同じ様に振る舞う(how)」
  - 「x と y が同値」: x と y が同じ振舞をするという事
- 「同値関係で割る」
  - 「同じ振舞をする物は同じ物とみなす」
- 「同値類」
  - 「同じ振舞をする物を区別しないで作った集合」

同型



同値類