コンピュータ概論 A/B

-- Mathematica Programming --

数学科 栗野 俊一 (TA: 佐藤 淳 [院生 1 年])

伝言

私語は慎むように !!

- 席は自由です (出席パスワード : 20151013)
  - できるだけ前に詰めよう
  - 教室にきたら直ぐにやる事
    - PC の電源 On / ネットワーク接続 / Web を参照する / skype を起動する
- 色々なお知らせについて
  - 栗野の Web Page に注意する事

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino
- 今週は「補習」はありません
- 担任より、お知らせ
  - 履修表を受け取ってない方は、TA に申し出て、受け取ってください
  - 今週から「Web 履修科目修正」が可能になります
  - 今週から教職科目の履修費の支払いが可能になります
    - 今年度教職科目を履修する人は、必ず支払う

前回(2015/10/06)の内容

- □ 講義
  - ○ Mathematica を利用してみる
    - ▹ 数式を入れ、[SE]すると、数式の計算をしてくれる(数式処理)
    - ▹ 微積：多倍長計算/文字式(展開・因数分解)/微分/積分/極限
    - ▹ 代機：ベクトル/行列計算/方程式の解/グラフ描画
  - ○「高機能」電卓として利用可能(それだけでも十分使える)
    - ▹「計算」だけなら、人間より優秀
  - ○ Mathematica で「何か(数学用語)」をしたければ..
    - ▹ 検索エンジンで「Mathematica 数学用語」で見付かる
- □ 演習
  - ○ 色々な Mathematica の機能

Mathematica の利用

- Mathematica での計算
  - 「式」を入れて [SE] ([Shift]+[Enter]) で計算(評価)が行われる
  - 三つの括弧の区別 (Mathematica では、働きが違うの区別が必要)
    - 「(」、「)」: 丸括弧/パーレン : 計算の優先順位を変更する(先に計算される部分を囲む)
    - 「[」、「]」: 角括弧/ブラケット : 関数適用を行う(関数の引数を囲む)
    - 「{」、「}」: 波括弧/ブレース、カーリー : 複数の要素を一つにまとめる(ベクトルの要素を囲む)
  - Mathematica で「色々な事」をするには (是非、色々試してみよう)
    - 「(Mathematica の)関数」を適用すれば良い (例: 「微分」は D[] を使う)
    - 「(凄く強力な)Help」がある / Web を探せば、沢山の解説ページが..
  - 「数学の計算は『何でも』Mathematica で計算できる」と思って良い
    - 「何」が出来るか : 数学の概念(専門用語)を知って居れば良い(数学、頑張ろう)
    - 「どうやって(どの関数を利用)」するか : 検索する「Mathematica 数学概念」
    - 「出来無い」と思うと「出来る事も出来無く」なる (できる、だから探せ)

本日(2015/10/13)の予定

- 講義
  - 「メタシステム」とは
  - 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
  - [演習 3] 課題の作成

本日の課題 (2015/10/13)

- 前回 (2015/10/06) の課題
  - 次のファイルを Mathematica で作成して CST Portal に提出してください
    - ファイル名 : 20151006-QQQQ.nb (QQQQ が学籍番号)
    - 内容 : Mathematica で色々やってみる
    - 形式 : nb 形式 (sample-20151006.nb を参照)
- 今回 (2015/10/13) の課題
  - 次のファイルを Mathematica で作成して CST Portal に提出してください
    - ファイル名 : 20151013-QQQQ.nb (QQQQ が学籍番号)
    - 内容 : 1 から n までの 3 乗和を計算する関数 cubeSum[n] の作成
    - 形式 : nb 形式 (sample-20151013.nb を参照)

システム

□「システム」とは

○「対象」と「操作」の対 (この二つは完全に区別される)

- ▷「対象(Object)」: 興味の「対象」となる集合の「要素」(広い意味での「数」/「何(What)」)
- ▷「操作(Operation)」:「対象」の要素を「(直接)扱う手段」(広い意味での「関数」/「どうする(How to)」)

表 1: システムの例

システム	対象	操作	事例
計算 (電卓)	数	演算 (四則/関数)	$1+1=2,\ \sin\left(\frac{\pi}{3}\right)=\frac{\sqrt{3}}{2}$
代幾	ベクト ル	行列 (線形変換)	$A\vec{v}=\vec{u}$
微積	関数	極限・ 微分・ 積分操作	$\int f(x)dx=F(x)+C$
数学 (入門)	数学の対象 (集合)	論理 (集合演算)	「 仮定」から「 結論」
TEX	文章/文字列	マクロ	`\frac{1}{2}` $=\frac{1}{2}$
Excel の計算	セルの値	式	他のセルの値から 値を 計算
Mathematica	数式 (数を 含む)	Mathematica の関数	$D[Sin[x],x]=Cos[x]$
OS	ファイル	ファイル操作	ファイルのコピー/削除
アプリ ケーション	データファイル (テキスト)	プログラム (エディタ)	ファイルの内容の変更
計算機	データ	プログラム	入力データから 出力データ
料理	食材 (素材の味)	調理 (調味量)	食物 (味) から 食物 (味)
刑事裁判	被疑者 (無実)	検事 (断罪)	どちらの主張が正しいかを 判断

□「学問」とは

○「『対象』を処理するために、『操作』の性質」をまとめたもの

システム図

- 個々の役割
  - 興味の対象は Object (対象)
  - 対象を操作するために Operation(操作方法)を利用
  - 「Operation の仕組みを知る」事が「学問する」事
- 「学習」の例
  - 「数」を操作するために「計算方法」を学ぶ
  - 「食事」を作るためには「料理法」を学ぶ
  - 「情報」を得るためには「情報検索方法」を学ぶ
- 応用
  - 学んだ内容結果を実地に適用する
  - 学んだ操作方法を利用して、対象を操作する

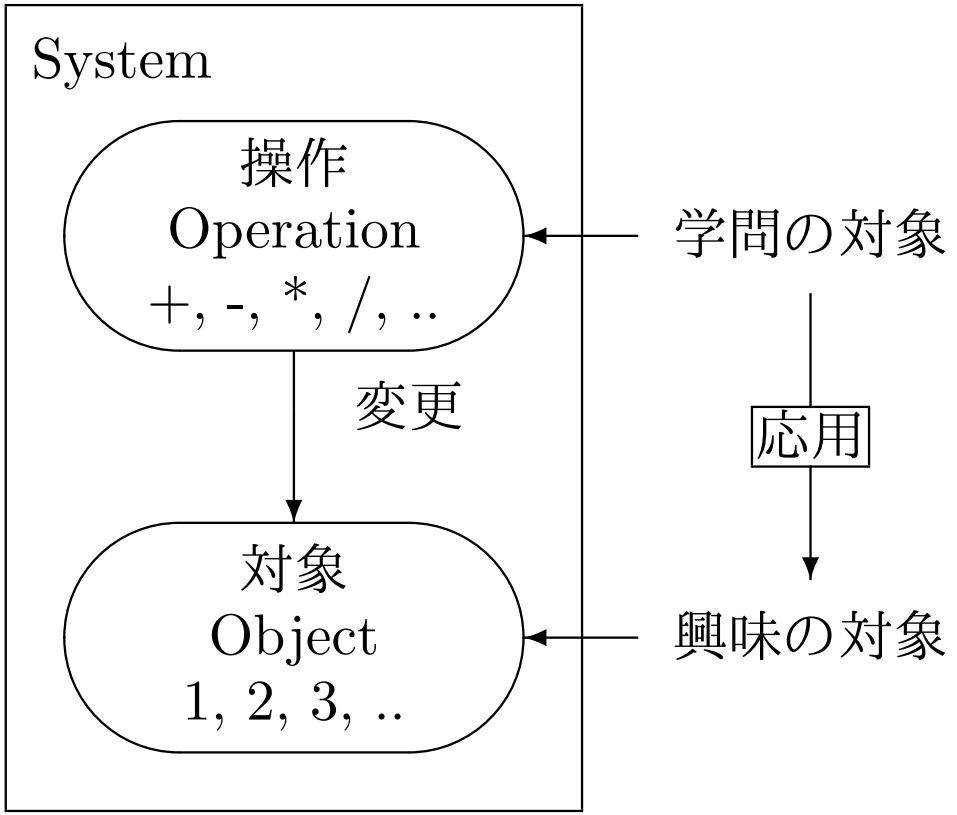


図 1: システム図

メタ概念

□ メタ〜(超〜)

○「『システムの操作』を操作する」仕組み

- ▷「メタ○○」は、「○○の○○」と置き換える事ができる(cf. メタ情報)

表 2: メタシステムの例

システム	対象	操作	メタ論
計算	数	演算 (四則/関数)	汎関数/関数操作 (微積/関数解析)
代幾	ベクトル	行列 (線形変換)	行列の計算 (線形空間)
数学	数学の対象	論理	帰納法, 背理法 (証明論/ヒルベルトのプログラム)
TEX	文章/文字列	マクロ	マクロ定義
Mathematica	数式	Mathematica の関数	関数定義 (今日の課題)
計算機	データ	プログラム	プログラミング
現実 (料理)	世界 (食物)	人物 (コック)	人物 (コック) の評判
会話	聞き手 (You)	話し手 (I)	観客 (they)
刑事裁判	被疑者	検事	裁判官 (cf. 最高裁)

○「対象を操作する操作方法」そのものを操作する

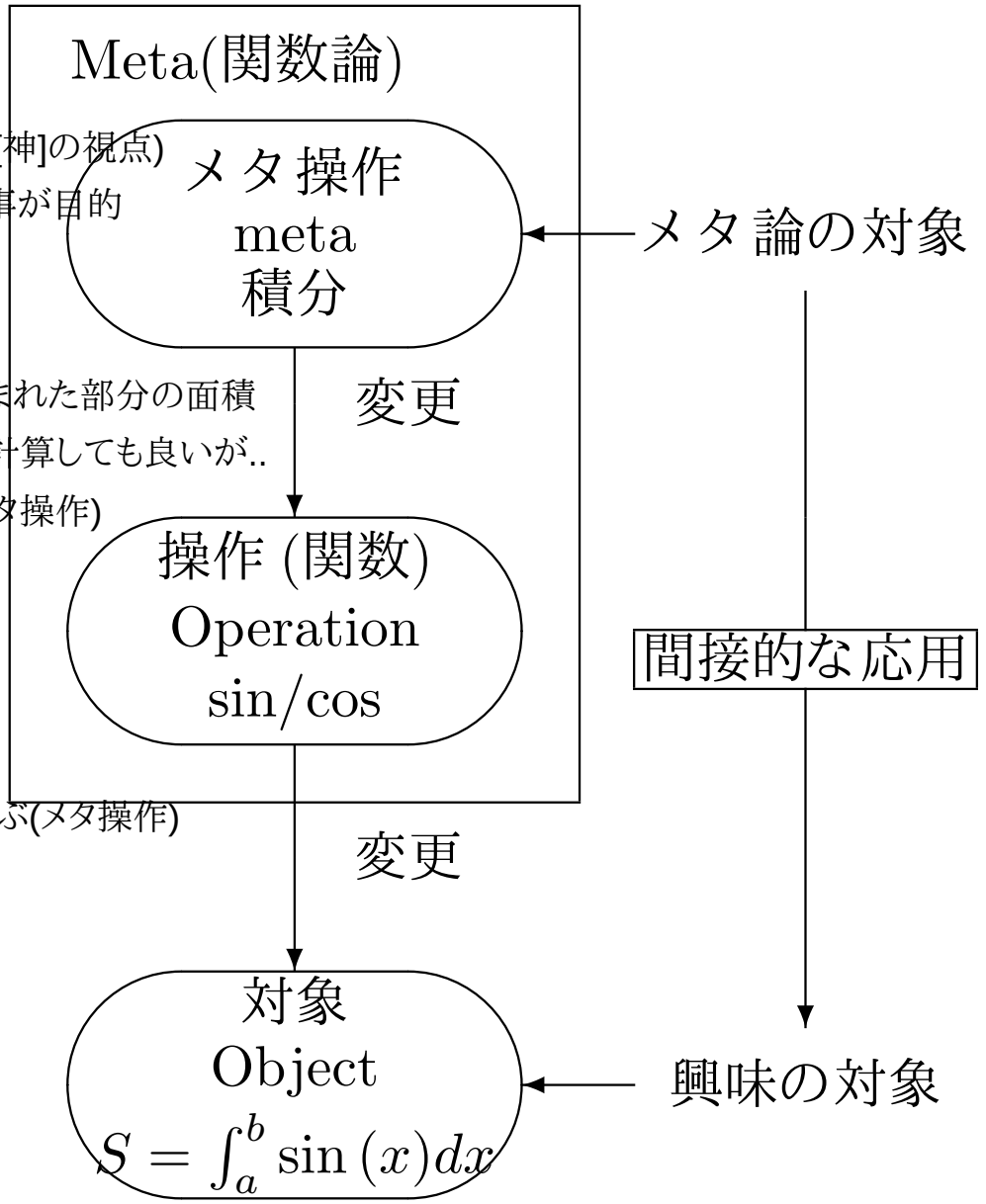
- ▷「対象の操作」を「操作方法の操作」で「間接的に操作」する
- ▷「間接」なので、「強力」だが「理解/応用するのは難しい」

○ メタ論の例

- ▷ あの議論(操作)の内容(対象)は解らないが、矛盾した議論(メタ)なので主張は正しくない(背理法)
- ▷ あの料理人(操作)の料理(対象)は食べた事はないが、三つ星のシェフ(メタ)なので、安心(評判)
- ▷ あの政治家(操作)の政策(対象)は知らないが、顔が嫌い(メタ)だから票は入れない(感情)

メタシステム図

- メタシステム
  - 特定なシステム上に構成される
  - システムの操作がメタシステムの対象 (鳥瞰/第三者[神]の視点)
  - 「操作の操作」により「『間接的に』対象を操作する」事が目的
- 例 1 : 関数論
  - 対象:数 / 操作:関数 / メタ:微分積分
    - 目的 : グラフ y=f(x) と x 軸、直線 x=a, x=b で囲まれた部分の面積
    - 操作 : 関数値を計算し、面積を細分化して総和を計算しても良いが..
    - メタ : 定積分で計算する → 原始関数を求める(メタ操作)
- 例 2 : 料理
  - 対象:食事 / 操作:料理 / メタ:シェフを呼ぶ
    - 目的 : 恋人に美味しい食事をご馳走したい
    - 操作 : 自分で、料理しても良いが..
    - メタ : 他人が料理した食事でも OK → シェフを呼ぶ(メタ操作)



「駄メタ(”)」論

- □ メタ論：強力過ぎる → 日常でも「乱用」されるきらいがある
- □ (こりゃ)「駄メタ(”)」論:「駄目」なメタ論の例 (S:システム/M:メタ)
  - ○「親(や先生)の言う事に従いなさい」と言う
    - ▹ S:言う事が正しいならそれに従う / M:親(や先生)の言う事は正しい
    - ▹ 問題：親(や先生)の言う事が「正しい」とは限らない
  - ○「消防所の『方(ほう)』から来ました。消火器を購入してください」と言う
    - ▹ S:消防署の所員なら消火器に詳しい / M: 消防署の所員なら消防所の『方(かた)』
    - ▹ 問題：消防署の方(かた)は、消防署の方(ほう)から来るが、逆は言えない
  - ○「お前のカーサン出べそ」という悪口を言う
    - ▹ S:相手を蔑む為に欠点を挙げる / M:親の形質(欠点)は子供に遺伝する
    - ▹ 問題：「出べそ」という形質は遺伝しない (産婦人科医がしくっただけ)
  - ○「答えを教えてください」と質問する
    - ▹ S:答を得る解法を学ぶ / M:解法の適用結果として正しい答えが得られる
    - ▹ 問題：学習の目的は「答」を知る事でなくて、「解法」を学ぶ事
  - ○「単位が取れた」とホットとする
    - ▹ S:学習状況を評価して、単位を与える / メタ：単位が得られれば状況は可
    - ▹ 問題：「可」だけでは、不充分、更なる学習が必要
  - ○ この OHP：駄洒落を言う (自己言及している)
    - ▹ S: 関連する事はいっしょに言う / M: 言葉が似ていれば関係がある
    - ▹ 問題：「音」が似ているからといって、「言葉」が似るわけではない

メタ関数と Mathematica プログラミング

- □ メタ関数とは
  - ○ 関数を引数、あるいは結果とする関数 (汎函數[はんかんすう]とも言う)
    - ▹ 普通の関数 (数を引数にして結果を数とする) の例 : Sin[x]
    - ▹ メタ関数の例 : D[Sin[x]] → Cos[x] (引数も結果も関数)
- □ Mathematica プログラミング
  - ○ メタ関数を定義する事
    - ▹ メタ関数には広義の意味で普通の関数を含めてしまう
- □ Mathematica は数式「処理言語」システム
  - ○ 新しい関数が定義できる
    - ▹「定義のための表現形式」が含まれている (言語システム)
  - ○「言語」システムの定義には、「構文規則」と「意味規則」が必要
    - ▹ 構文規則 : どのような「(表現)形:文字列」が適切か
    - ▹ 意味規則 : その「(表現)形」はどんな「意味」に対応するか
- □ Mathematica の関数定義とは
  - ○ 関数を表すシンボルに、その関数の意味を表す式を「割り当てる」事
    - ▹「add[x_,y_] := x+y」とすれば、「add[2,3]が5になる」ように定義した事になる

構文(表現)と意味

- □ 構文規則
  - ○「文字の並び」を「何の表現か?」と「判断するための規則」
    - ▹「123 + a」は、「123」、「a」、「+」に分解される
    - ▹「123」は、「1」、「2」、「3」という「数字列」なので「数値」
    - ▹「a」は、「a」という「英文字から始まる文字列」なので「変数」
    - ▹「+」は、「足し算」を表す「演算子」
    - ▹「123 + a」全体は、「式」になる
  - ○「どんな文字から構成されるか」という基準で区別される
    - ▹「何か」を表す、「表現方法」の規則
    - ▹規則に従って、「文字列」を見ると、幾つかの「構文要素」に分解される
- □ 意味規則
  - ○「構文要素」に対応した「意味」を決める規則
    - ▹「123」という「数字列」は、123 という「整数値」
    - ▹「a」という「シンボル」は、a という「変数」
    - ▹「123 + a」は、「二つの数を加える」という「文字式」
- □ 言語
  - ○構文規則 (文法)と、意味規則(「辞書」ならびに、「意味」) から成る
    - ▹「1 + 2」は「構文的」には、「「1」と「2」の「足し算」」を表す「式」となる
    - ▹「1 + 2」は「意味的」には、「3」(1, 2 は数値を表し、+ は足し算を表すから)

Mathematica(復習)

- □ Mathematica とは
  - ○ 数式処理言語システム
    - ▹ 「数式」を計算したり、数式の計算を行うプログラムが作れる
  - ○ 数式電卓
    - ▹ (文字を含む)数式の計算を行う
    - ▹ cf. 電卓：「数」の計算が出来る(数の式を入れると数の計算を行う)
    - ▹ [スタート]→[すべてのプログラム]→[アクセサリ]→[電卓]
- □ Mathematica の使い方
  - ○ ノートブックを開く
  - ○ 式を入力して [Shift]+[Enter] (以下 [SE] と表現)で評価開始
    - ▹ 計算に時間がかかりそうなら.. [Alt]+[,] で中断できる
  - ○ この講議では「Mathematica の導入」のみを扱う
    - ▹ 自分で色々調べて、試してみる (Help/チュートリアル)

Mathematica Notebook

- Mathematica Notebook (*.nb)
  - Mathematica の計算結果を保存する形式
  - 結果の保存方法
    - [ファイル]→[別名で保存] : 好きな名前で保存できる
    - [ファイル]→[保存] (Ctrl-S) : 今の名前で内容を更新する(古い内容は失われる)
  - 結果の利用方法
    - [ファイル]→[開く] (Ctrl-O) : 以前に保存した内容を読み込む
- 電卓としての利用
  - 「式」を入れて [Shift]+[Enter] (以下、[SE]) すると計算
    - 入力した式は In[番号] の形で表示される
    - 計算した結果は Out[番号] の形で表示される
    - 番号は、[SE] の順番につけられる
  - [SE] の効果
    - 「式」の評価が行われる
    - 「番号」が増える
    - In[番号]/Out[番号]が、それぞれ「定義」される

変数とその値

- □ シンボル (構文的な要素)
  - ○ シンボルとは
    - ▹「英字」から始まり「英数字」からなる文字列の事 (a, abc, a1, a12z)
    - ▹ (注意) 「英字」には、「ギリシャ文字」も含まれる (π , α)
    - ▹ cf.「数字」から始まる文字列はシンボルではない (2a は 2 × a となる)
  - ○ Mathematica は、大文字で始まるシンボルの幾つかを利用
    - ▹ (注意) 自分が利用する場合は、小文字で始まるシンボルを利用する事
- □ 変数 (意味的な要素)
  - ○ 変数とは
    - ▹ 名前として「シンボル」をもち、「値を持つ事ができるもの」
    - ▹「変数名」は、「変数」の名前で、それは「シンボル」である
  - ○ 変数の(即)値とは
    - ▹ (現時点で)変数に対応している「式(の評価結果)」の事
    - ▹「環境」は、「変数名」と「変数の値」の対応表をもっている
  - ○ 変数の値の参照
    - ▹「シンボル」の形を示す物を入れると、「その値」が評価され、その結果が表示される
    - ▹「値」が設定されていない場合は、その「変数自身」が「変数の値」となる
    - ▹ ?変数名で、「変数の値」である「式」を見る事ができる

変数への代入

- 評価代入(=)と未評価代入(:=)
  - 共に変数に「値」を「割り当て」る命令
  - 評価代入 :「変数 = 式」
    - 「式」は「評価」される。その「評価結果」が「変数の値」となる
    - 「値」は、何度でも参照できる (「値」に「名前」が付けられる)
  - 未評価代入 :「変数 := 式」
    - 「式」は「評価」されない。その「式(表現)」そのものが「変数の値」となる
  - 変数の値は何度でも変更(代入)できる
    - 変数に記録されているのは、最後の代入の結果(その前の値は失われる)
- 環境
  - 変数名とその値の対応表を持っている
    - 「代入」も「定義」も、その対応表を「書き換えて」いる
    - 書き換える値が違うだけ
- 「式」の評価(変数の場合)
  - 式の中に変数名が含まれていた場合に、それを変数の値に書き換える

関数の利用

- □ Mathematica の関数
  - ○ Mathematica の関数とは
    - ▹「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
    - ▹ fib[1]=1
    - ▹ fib[2]=1
    - ▹ fib[x_]:=fib[x-1]+fib[x-2]
    - ▹ <注意> fib[3] とすると、x に一時的に 3 が代入され、右辺(fib[x-1]+fib[x-2])の評価の時に利用される

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 でペアノの公理の形の「自然数」を考える
  - 自然数 (これを「ペアノ形式の表現」と呼ぶ事にする)
    - 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(有理数)」まで表現できる
    - 「実数」を表現するには、「収束概念」が必要になる

自然数による数の表現 (nat.txt)

- 自然数
  - 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]

「ペアノ形式」と普通の表現の関係

表 3:「S 表現」と普通の表現の関係

概念	普通の表現 (例)	ペアノ形式 (例)	コメント
0	0	0	0 は自然数
自然数 (0 以外)	1,2,3,..	s[0], s[s[0]], s[s[s[0]]], ..	x が自然数なら s[x] も自然数
n の次	n + 1	s[n]	s はサクセッサ (後継) 関数
和	1 + 2	padd[s[0],s[s[0]]]	
差	1 - 2	psub[s[0],s[s[0]]]	引く数の方が大きい場合は 0 にな
積	1 * 2	pmul[s[0],s[s[0]]]	
商	1 / 2	pdiv[s[0],s[s[0]]]	整数割り算 (小数点以下は切り捨て
最大公約数	gcd(1,2)	pgcd[s[0],s[s[0]]]	
整数	1, -1	pp[s[0],0], s[0,s[0]]	自然数の対を同値類で割った物が
整数の正規化		zbar	ペアのどちらか一方を 0 にする
整数の四則	+,-,*,/	zadd,zsub,zmul,zdiv	
有理数	1/2	qq[pp[s[0],0],s[s[0]]]	分子は整数で、分母が自然数の対
有理数の正規化		qbar	約分する
有理数の四則	+,-,*,/	qadd,qsub,qmul,qdiv	

- □ 可換性(well defined)
  - ○ 通常の自然数とペアノ形式の自然数は ntop, pton で同型になっている
  - ○ 個々のペアノ形式の関数は ntop, pton に関して可換になっている
    - ▹ pton[padd[ntop[1],ntop[2]]]=3=1+2