愛知教育大学教育実践総合センター紀要
第4号，pp. 75～78（March，2001）

算術式のためのGUI作成

清　水　秀　美
（愛知教育大学教育実践総合センター）

GUI Development for Arithmetic Expression

Hidemi SHIMIZU
(Center for Research, Training and Guidance in Educational Practice, Aichi University of Education)

要約　算術コンピュータ・シミュレーションにおいて，コンピュータ算術式でなく，いわゆる学校教育で使用される算術式で画面表示することが，式の直観的理解という点で重要である。ここで開発されるのは，コンピュータ算術式を，普通用いられるグラフィックな算術式に変換するGUIプログラム部分である。そのために用いられるアルゴリズムは，DCG規則に，各項の移動，縮小拡大を行う補強項を付加して，算術式の構文解析を行うことにある。ここでの出力結果はJava言語とリンクして，Windows上に表示される。
Keywords：算術式，コンピュータ・シミュレーション，GUI，DCG

はじめに

コンピュータに人間と同じ様な方法で計算させる，いわゆる算術コンピュータ・シミュレータの作成（清水，1999）において問題の一つは，算術式の表現である。コンピュータでの算術式は我々が小学校以来馴染んできた算術式ではない。ここでは両者を区別する上で，前者をコンピュータ算術式（Computer Arithmetic Expression: CAE），後者を人間の視覚・直観に訴えるという意味でグラフィック算術式（Graphic Arithmetic Expression: GAE）として区別することにする。コンピュータ算術式では，全てを一行で表現しようとするのに対して，後者ではできるだけ直観的に理解できるように記号を用いて2次元的表現を採用している。例えば両者は以下のようになる（図1）。

Computer Arithmetic Expression: CAE	Graphic Arithmetic Expression: GAE
1/2+1/3+1/4	$\frac{1}{2}+\frac{1}{3}+\frac{1}{4}$
sqrt(1/3^2+1/4^2)	$\sqrt{\frac{1}{3^2}+\frac{1}{4^2}}$

図1．コンピュータ算術式とグラフィック算術式

要するにCAEは線的であるのに対してGACは面的であるといえる。CAEは一定の規則に従って線的に表現され，一方，GAEも一定の規則に従い面的に記述されるので，CAEからGAEへの変換の可能性は容易に予測される。小学校と中学校1年生までの算術・代数式を対象にその実現を試みる。

本稿の目的は，CAEからGAEの表現変換の方法を明確にすることであるが，このことは教科書との表現形式に合せるという意味以上に，算数学習を促進させる上でGAE表現は重要である。すでにコンピュータ・ソフトでもGAE表現を与える試みがなされてきた。TEXを始めとする多くのワード・プロセッサー・ソフトでGAE表現が可能となっている。しかし，それらではCAEからGAEへ自動的に変換されるようにはなっていない。数学記号を手作業で配置するか，それに準じた作業を必要とする。もちろん，数式処理アプリケーションソフトMaple V（井上，1997，p88-122）では入力代数式と処理結果にGAEを与えている。残念ながら，処理の途中結果は明示されないようである。当然，算術式の途中結果を出力しない。算術シミュレータを作成する上でCAEからGAEへの自動変換ソフトの自主開発は避けて通れない課題である。

CAEからGAEへの自動変換アルゴリズム

DCG規則による構文解析　算術式も代数式を含む数式と同様に文法規則に従い記述され，CAEもGAEも例外ではない。従って算術式を構成する各要素とそれらの相互の関連は構文解析で把握できる。構文解析は自然言語処理用のプログラミングツールとしてDCG（Define Clause Grammar）を用いて行われるのが一般的となっている（畝見，1985; Coelho, Cotta & Pereira, 1982, p121）。ここでもこれに従い，算術式の構文規則を図1のように表現することができる。ここでは，帯分数，小数点，三角法，対数，累乗の各表現に対処することとし，さらに（），‖，[] や代数表現の処理ができるようにした。次に，図2の構文解析で分離され関連づけられた構成要素について，補強項を用いてGAE対応の各構成要素の座標を算出するよう

にした（図２参照）。

```
expr(B) --> expr(X),['+'],term(Y),{exprDraw(X,'+',Y,B)}.
expr(B) --> expr(X),['-'],term(Y),{exprDraw(X,'-',Y,B)}.
expr(A) --> term(A).
term(B) --> term(X),['*'],term(Y),{exprDraw(X,'*',Y,B)}.
term(B) --> term(X),[':'],term(Y),{exprDraw(X,':',Y,B)}.
term(B) --> term(X),['/'],term(Y),{fractDraw(X,Y,B)}.
term(B) --> number(X),['&'],number(Y),['/'],number(Z),{numfDraw(X,Y,Z,B)}.
term(B) --> factor(X),['^'],term(Y),{powerDraw(X,Y,B)}.
term(A) --> factor(A).
factor(B) --> ['('],expr(A),[')'],{bracketDraw('(',A,B)}.
factor(B) --> ['{'],expr(A),['}'],{bracketDraw('{',A,B)}.
factor(B) --> ['['],expr(A),[']'],{bracketDraw('[',A,B)}.
factor(B) --> ['-'],factor(A),{signDraw('-',A,B)}.
factor(B) --> ['('),number(X),['$'],[')'],{degreeDraw(X,B)}.
factor(B) --> ['+'],factor(A),{signDraw('+',A,B)}.
factor(B) --> [sqrt],factor(A),{sqrtDraw(sqrt,A,B)}.
factor(B) --> [sin],factor(A),{signDraw(sin,A,B)}.
factor(B) --> [cos],factor(A),{signDraw(cos,A,B)}.
factor(B) --> [tan],factor(A),{signDraw(tan,A,B)}.
factor(B) --> [log],factor(A),{signDraw(log,A,B)}.
factor(B) --> number(A),{atomDraw(A,B)}.
factor(B) --> alpha(A),{atomDraw(A,B)}.
number(A) --> [N],{number(N),A=N}.
alpha(A) --> [D],{atom(D),A=D}.
```

図２．　CAEからGAEに変換するためのDCG表現

内部表現　DCGを実行するに際してはBUP（Bottom Up Parser）トランスレータで実行可能なProlog言語に変換した。BUPを用いた理由は左再帰を回避することに加えて，各算術式の要素の位置を原点からスタートさせ，順次移動，縮小・拡大を繰り返し，最終表現を得るためである。各要素の終端で与えられる(x,y) 座標値は（0,0）で，以下の様な内部表現が与えられる。

[要素,要素の文字長,文字サイズ（初期値=16），
x座標値,y座標値]

例えば，123+45の場合，この算術式の構成要素は"123"，"+"，"45"の３つに分けられ，内部表現"123"は始めにfactor（B）の補強項{atomDraw（A,B）}により，[[123,3,16,0,0]]が与えられる。同様に"45"に対して[[45,2,16,0,0]]が与えられる。次にterm（A）--> factor（A）.とexpr（A）--> term（A）.より，expr（B）-->expr（X），['+'], term（Y），{exprDraw(X,'+',Y,B）} のX,Yが確定し，補強項{exprDraw(X,'+',Y,B）}のBが決定され，B=[[[123,3,16,0,0]],[+,1,16,24,0], [[45,2,16,32,0]]]となる（16ポイントの文字サイズの高さと幅を考慮して，x座標に沿って平行移動がなされる）。

12/345の場合，" 12" と" 345" は先ず始めにそれぞれ[[12,2,16,0,0]]，　[[345,3,16,0,0]]として表現される。そして，補強項{fractDraw（X,Y,B）}で，それぞれ移動を行い，かつ分数を表す長さの調節された水平直線を加えて，B=[[[12,2,16,8,-8]],[____,4,16,0,-8], [[345,3,16,4,8]]]が得られる。

13^2の場合，"13"と"2"に対して同様に始め[[13,2,16,0,0]] と [[2,1,16,0,0]] として表現され，"2"に対して縮小と移動を行い、最終的にB=[[[13,2,16,0,0]], [[2,1,8,16,-8]]]を得る。

例外処理　上述のDCGによる分析では処理は左から進行するので，1÷2/3や4*5/6は，以下の様に表現される。

$$\frac{1 \div 2}{3} \qquad \frac{4 \times 5}{6}$$

これを改めて，

$$1 \div \frac{2}{3} \qquad 4 \times \frac{5}{6}$$

と表現する変更は補強項で行われる。また，分数式では，一般に括弧表現は行われない。CAEでは（1+2）/3と表現するが，GAEでは括弧は用いられない。これらの細かな例外の多くは補強項で処理する。

代数式への対処　算術式のみでなく初歩的な代数式にも対処できるようにした。そのために，数値以外にアルファベットの処理を可能にするために，DCG規則に以下を追加した。

alpha（A）-->[D],{atom（D）,A=D}.

例えば2*a/（3*b+4*c）は以下の様に表現される。

$$\frac{2a}{3b+4c}$$

代数式での×の記号の省略は補強項で行い，÷の処理はDCG規則による構文解析以前の段階で分数式に変換される。

数式入力エラーに対する処理　上記のDCG規則で算術式の構文解析を進めるわけであるから，この範囲外のものはすべてエラーとして排除する。そのために上記DCG規則から補強項を除いたものをエラー・チェック用DCGとして利用する（図3）。

```
t_expr --> t_expr,['+'],t_term.
t_expr --> t_expr,['-'],t_term.
t_expr --> t_term.
t_term --> t_term,['*'],t_term.
t_term --> t_term,[':'],t_term.
t_term --> t_term,['/'],t_term.
t_term --> t_number,['&'],t_number,['/'],t_number.
t_term --> t_factor,['^'],t_term.
t_term --> t_factor.
t_factor --> ['('],t_expr,[')'].
t_factor --> ['{'],t_expr,['}'].
t_factor --> ['['],t_expr,[']'].
t_factor --> ['-'],t_factor.
t_factor --> ['('],t_number,['$'],[')'].
t_factor --> ['+'],t_factor.
t_factor --> [sqrt],t_factor.
t_factor --> [sin],t_factor.
t_factor --> [cos],t_factor.
t_factor --> [tan],t_factor.
t_factor --> [log],t_factor.
t_factor --> t_number.
t_factor --> t_alpha.
t_number --> [N],{number(N)}.
t_alpha --> [D],{atom(D)}.
```

図3．算術式入力エラー・チェック用DCG規則

コンピュータ算術式入力記号について　+，-に加えて以下のようにコンピュータ入力記号を設定した。×→*，÷→:，分数記号→/，帯分数つなぎ記号→&，累乗記号→^，角度表示記号（__$），根号記号→sqrt，およびsin，cos，tan，logである。ただし，等号記号＝も利用できるようにした。

PrologからWindows表示へ　以上の処理プログラムは全てProlog言語で記述される。この言語の特徴は他の言語に比べて比較的簡単に思想をコーディングできる点にある。しかし，処理結果をWindows上に提示することを想定した言語ではないので，結果の視覚的提示には限界がある。今日では、この限界を克服するためProlog言語に異種間言語とのリンク機能を付与する場合が多い。例えばソフネック社のAZ-PrologではC言語とのインターフェイスが提供されている。今日ではJava言語とのインターフェイスの提供が盛んであり，より簡単に結果のグラフィック表示が可能になってきている。しかし、問題はJava言語の歴史は浅く，1・2年ごとに改定されるので，それにつれてProlog言語の改定が行われる必要がある。ここで暫定的に試用したのは日本語対応のIF-Prolog社製の，Minerva 1.01である。

実行結果

図 4 はJava言語による算術式のためのGUI画面を示す。画面の右側に画面全体のコントロール・スイッチ類を配し，算術式入力は画面上部のテキストフィールドで行う。入力は分数式である。

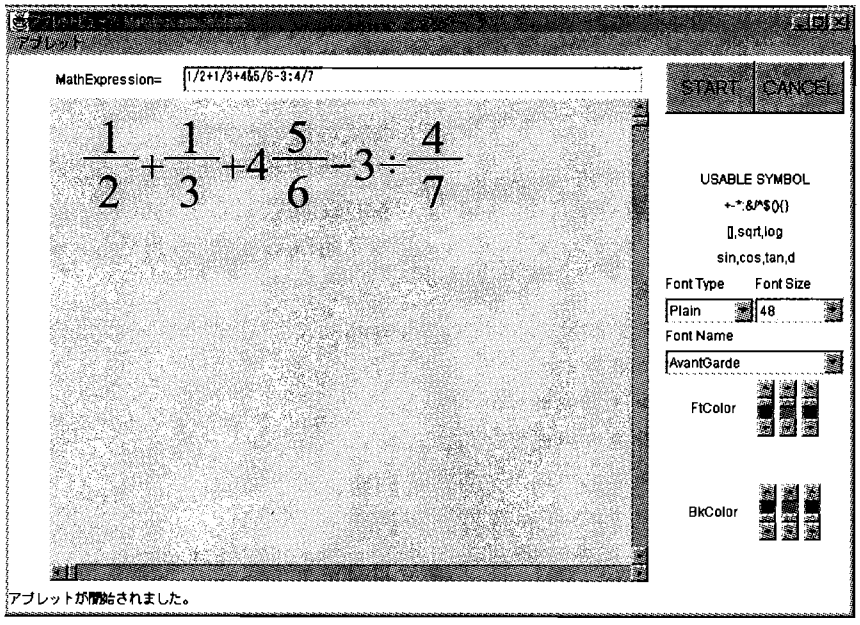

図４．GUI全体画面と分数式の表示

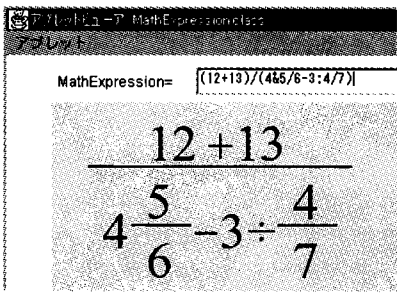

図５．繁分数の表示

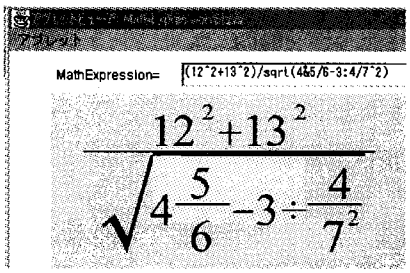

図６．根号記号と累乗表示

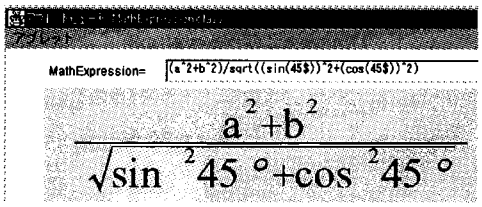

図７．代数表現と三角法記号の表示

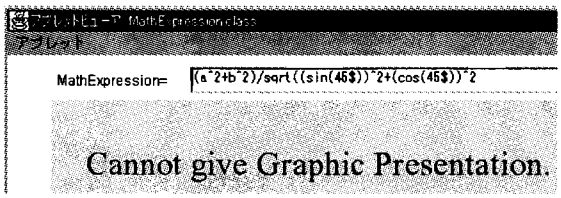

図８．算術式入力エラー表示

図 5 は繁分数の表示，図6は根号記号と累乗の表示，図 7 は代数表現と三角法記号の表示である。図8は右括弧不足に伴う算術式入力エラーの表示例である。その他，文字サイズや文字種の変更，フォント・カラーやバックグランド・カラーの変更，小数点入力への対応も十分であった。

参考文献

Blackman, N. R. & Mossinghoff, M. J.（井上他訳）1997 MAPLE V リファレンスブック。 オーム社

Coelho, H., Cotta, J.C., & Pereira, L .M. 1982 How to solve it with Prolog. Ministerio da habitacao e obras publicas.

畝見達夫 1985 自然言語処理への応用。［溝口文雄監修 Prologとその応用 2 総研出版］

清水秀美 1999 小学算数計算シミュレータの作成 愛知教育大学教育実践総合センター紀要 第2号 39-46。