数理解析研究所講究録
1038巻 1998年 198-204



# 数式処理システム「Mathematica」にインタラクティブ機能を組み込む (Add Interactive Facility to Mathematica)

宮地 力
Chikara Miyaji
筑波大・体育科学系
University of Tsukuba

**Abstract.** Symbolic Computation Systems, like Mathematica are basically interactive. But it is not interactive to general events from the users. For that reason, these System lack the ability of expansion to meet users special needs. In this article, an system which add interactive facility to Mathematica is introduced. It enables to build special Graphical User Interface to it and adds the mechanism to communicate with other Mathematica sessions. This system will help to use Mathematica in a educational field.

## 1. はじめに

ほとんどの数式処理システムは，式を読みそれを評価するという点では，基本的にインタラクティブである．しかし，一般的なユーザのイベントに対してインタラクティブではない．ユーザからのイベントは，マウスクリック，メニュー選択など様々であり，それらすべてに対応することは難しい．そのため，数式処理システムを応用的に利用する時，例えば教育用のツールとして利用するときなど，拡張性に欠けるという問題がでてくる．本論では，既存の数式処理システムとしてMathematicaを取り上げ，それを改良しイベントを受けてインタラクティブに対応出来るようにした試みを紹介する[1]．このシステムによって，Mathematicaをマウスやメニューに対しダイナミックに反応するように拡張できた．また，遠隔教育のための道具としてMathematicaを利用できるようになった．

## 2. システムの基本構成

このシステムは，ネットワークを利用し，いくつかのプログラムが共同で働くことによってでき上がっている．プログラムは，基本的に2つの部分からなる．1つは，Serializerで，Mathematicaのフロントエンドとカーネルの間で受け渡される式を中継するプログラムである．もう一つは，ユーザの多様なイベント，入出力に対応するプログラムである．このプログラムは，ユーザからのイベントを式にして，やはりSerializerに送る．そこで，その

式もカーネルで評価される．この仕組によって，いろいろなインターフェイスをもったプログラムから，カーネルに式を送ることができる．

また，Serializerは，インターフェイスプログラムからイベントを受けるだけではなく，複数のSerializerとも通信ができる．それによって，複数のフロントエンドを利用して1つのカーネルを使うことや，Mathematicaのセッションどうしで式をやりとりすることが可能になる．

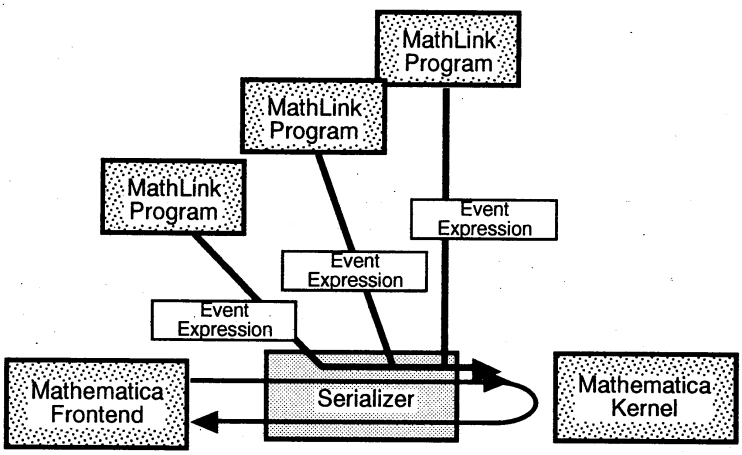


Fig. 1. Serializerは，フロントエンドとカーネルの間で式をカーネルに送る

## 3. MathLink

これらのプログラム間の通信は，MathLinkというプロトコルを用いている．MathLinkは，式を基本の要素にした通信のプロトコルである[2, 3]．例えばフロントエンドとカーネルはMathLinkを使って通信をしている．フロントエンドはカーネルに式を送り，カーネルはそれを評価し，また式の形でフロントエンドに送り返す．このように，式を基本の要素として通信をしているので，複雑な構造の情報をやりとりすることができる．また，C言語によるMathLink通信のライブラリがあるので，Cのプログラムと Mathematicaのカーネルとの間でも式のやり取りができる．

## 4. MathLinkテンプレート

MathLinkテンプレートを使うと，MathematicaからCのプログラムを呼び出し計算をし，その結果の値をMathematicaに返す通信の仕組を簡単に作成することができる．これを利用すると，非同期にMathematicaからCのプログラムのある関数を実行することができる．

例えば，Cの関数addtwo(int x,int y)があり，x+yを返す．この関数を利用して和を計算するMathematicaの関数AddTwo[]を作る．それは，テンプレートを用いて以下のように定義する．そして，この定義からでき上がるCプログラム，これをMathLinkプログラムと呼ぶ．

```
:Begin:
:Function:  addtwo
:Pattern:  AddTwo[i_, j_]
:Arguments:    i, j
:ArgumentTypes:    Integer, Integer
:ReturnType:  Integer
:End:
```

Fig. 2.　AddTwo のテンプレート

はじめにこの MathLink プログラムを Mathematica と接続する．そして AddTwo[2,3] を評価すると，2, 3 の値がネットワークを経由して MathLink プログラムに送らる．そして，addtwo() が実行され，和の値 5 が Mathematica に返される．MathLink テンプレートを利用すると，関数の実行をするだけで，ネットワークを意識せずに，このようなネットワークを利用した計算ができる．

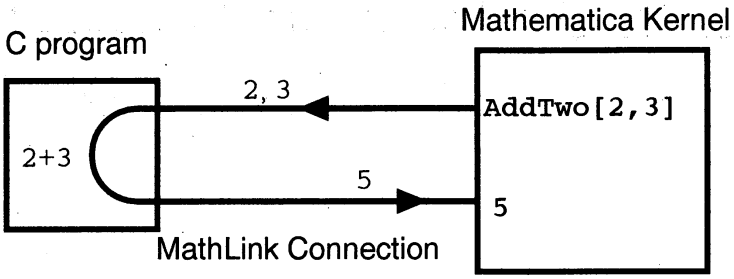


Fig. 3.　AddTwo の計算の過程

本システムの Serializer，ユーザとのインタラクションをする MathLink プログラムは，このテンプレートを利用してつくった．そして，Serializer には，テンプレートによる MathLink 通信に加えて，フロントエンドとカーネルとの通信部分を付加した．またインタラクションをする MathLink プログラムも，テンプレートを利用して作り，それにイベントを Serializer に送る通信部分を付け加えた．この方法により，Serializer や他の MathLink プログラムは Mathematica からテンプレートの関数を利用してコントロールすることが出来る．

## 5. オブジェクト指向プログラミングの利用

Graphical User Interface の部品やウィンドウをプログラムする時，オブジェクト指向プログラミングが適している．そこで，本システムでもオブジェクト指向によってプログラムを作った．ただし，Mathematica には，オブジェクト指向の枠組みがないので，関数閉包を利用してオブジェクト指向を実現した．その方法を以下に説明する．この関数 New[dog] は，dog クラスの定義である．

New[dog] を実行すると，新しいオブジェクトのインスタンスができる．そのインスタンスは，内部に self という関数を持ち，ローカル変数 weight がある．koro インスタンスに

```
New[dog] :=
Module[weight, self,
self[setweight, w_] := weight = w;
self[getweight] := w;
self]
```

Fig. 4. dog クラスの定義 New[dog]

setweight メッセージを送ると，ローカル変数 weight の値を変更できる．

```
In[1] := koro = New[dog];
In[2] := koro[setweight, 50]
Out[2] = 50
In[3] := koro[getweight]
Out[3] = 50
```

Fig. 5. koro インスタンスへメッセージを送る

ここで用いたオブジェクト指向は，上記の方法を発展させ，多重継承や，外部メソッド定義，内部メソッド定義が出来るようにしたものである．

## 6. メッセージの流れ

ユーザがプログラム上のウィンドウや Graphical User Interface 部品に対し，あるアクションをする．そうすると，それは，ある式に対応する．例えば，ウィンドウのインスタンス self$123 のクローズボタンが押されたとき，self$123[dispose] というメッセージが対応する．そして，その式がカーネルに送られる．そのメッセージによって，テンプレートで定義した関数が実行されて MathLink プログラムのウィンドウデータが消去される．

このようなメッセージの流れでオブジェクトを作ることによって，Mathematica を利用してダイナミックにオブジェクトの生成や関数の再定義などできるようになった．例えば，ボタンを押したときに実行する関数は，Mathematica の関数として定義してあるので，自由に変更することができる．

また，この方法は，システムの仕事を適当にネットワーク上に分散して分担することを可能にした．MathLink プログラムは，インターフェイスの表示と入力の受け付けを分担し，実際にどのようなことをするかは Mathematica 上の関数として定義する．これは，SmallTalk でのプログラム設計の際に提唱されている，Model-View-Contoroller に対応するものである．

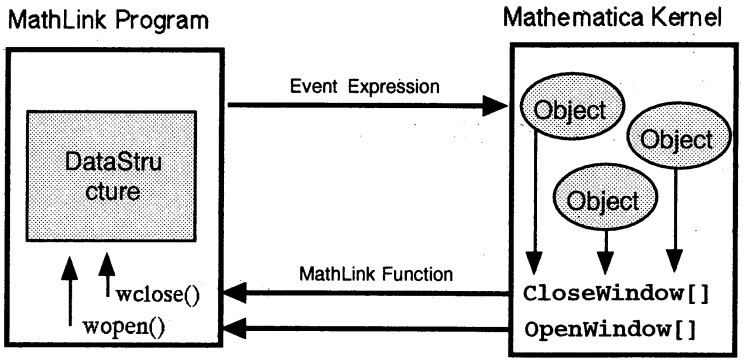


Fig. 6. MathLinkプログラムとMathematica上のオブジェクトの関係

## 7. プログラムの実行例

Mathematicaのライフゲームのプログラムのインターフェイスを本システムにより作成する例を紹介する.

```
In[1] := w = New[Window];
```

この式を実行することによりウィンドウが一つ開く. そのウィンドウ上にBitMapオブジェクトを1つ置く.

```
In[2] := bm = New[BitMap, 100,100, w];
```

このbmがライフの計算をするように新しいメソッドをbmに付加する.

```
In[3] := bm[life] :=
(bm[settable, dolife[bm[gettable]]]; bm[draw])
```

ここでdolife[table]はtableのデータに対しライフゲームによる世代交代を一回する関数である. bm[gettable]は, bmからデータを持ってくるメッセージ, settableはデータを設定するメッセージである. bmオブジェクトは, Mathematica上で関数として定義されているので, 新たに関数定義をbmに追加したことになる.

ボタンを表わすButtonObjectと, テキストエリアを表わすTextEditオブジェクトを追加する.

```
In[4] := btn = New[ButtonObject, w];
In[5] := txt = New[TextEdit, w];
```

以上でウィンドウ上に3つのオブジェクトが現われる. これをマウスで適当な位置に配置し直す. この操作も内部ではイベント式がカーネルに送られ, それによって位置が変化している.

ボタンが押されたとき, テキストエリアにある回数だけライフゲームの世代交代をするように設定する. それには, dobutton[]関数を定義し, それをボタンのアクションにすればよい.

```
In[6] := dobutton[] :=
Do[bm[life], i, ToExpression[txt[gettext]]];

In[7] := btn[setaction, Hold[dobutton[]]];
```

以上でライフゲームのインターフェイスが作成できた．関数の定義を行いながらインターフェイスを作ることで，対話的かつダイナミックにインターフェイスを組みたてることができる．これは，Mathematicaがインタープリタ型言語であることによるメリットである．また，このようなインターフェイスがあれば，マウスを使うことによって，簡単にいろいろなセルのパターンを調べることができる．これは，Graphical User Interfaceの長所である．このようなインターフェイスがいろいろな分野に合わせて作られるなら，数式処理システムの長所を活かし，なおかつ使いやすいシステムが構築できるだろう．また，これらのインターフェイスが複数あって，それらがお互いに補うことによって，より使いやすい環境を組みたてられる．

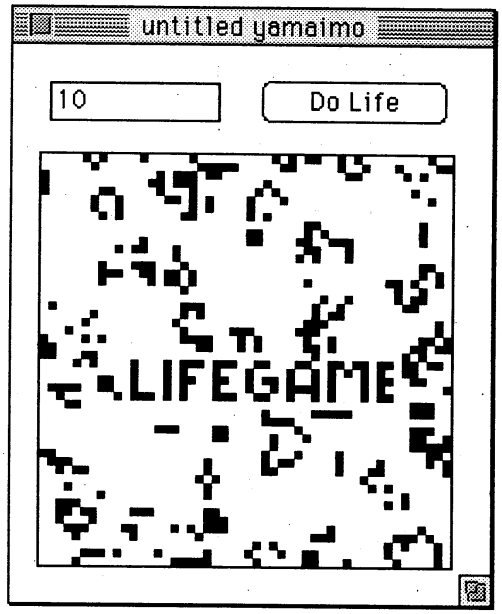


Fig. 7.　完成したライフゲームのインターフェイス

## 8. Serializerどうしの式のやりとり

Serializerも，1つのMathLinkプログラムとして他のSerializerにつなぐことができるようにした．これによって，ある式を自分の接続してるカーネルではなく，他のSerializerの先にあるカーネルで評価することができる．これは，Mathematica上での分散処理を可能にする．そして，分散処理だけではなく，自分のノートブック上の式やグラフを相手のノートブック上に送るということも可能になる．

Mathematicaのノートブック上に表示されている文字や絵も，Mathematicaの式で表現されている．そこで，あるノートブック上の式を，他のSerializerに送り，それを，あちら

のカーネルで評価すれば，あちらのノートブック上に，自分のノートブックに見えているものを送ることができる．

この機構によって，Mathematicaのセッションをしているどうしが式をやり取りすることが可能になった．そして，ネットワーク通信を利用しているので，場所的な制限はなく，コミュニケーションをすることができる．このような道具を組みあわせることで，リモート教育の道具をつくることが可能になるであろう．

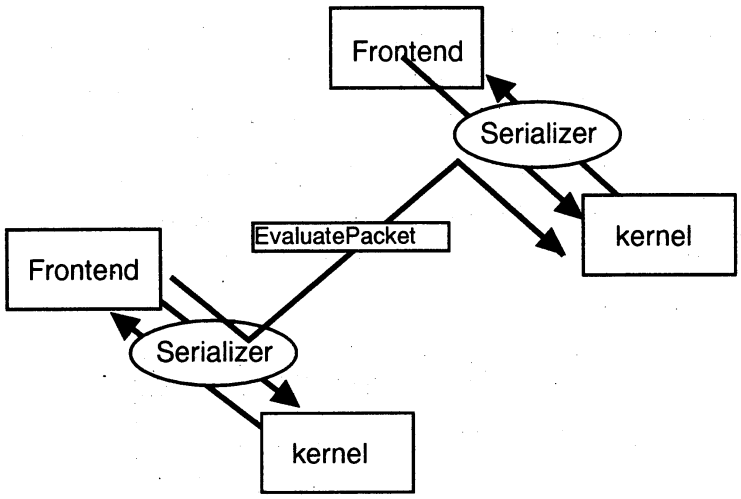


Fig. 8. Serializerどうしが式を送る

## 9. まとめ

数式処理システムは今後いろいろな分野で利用されていく道具になる．その時，さまざまなインターフェイスを簡単に付加すること，ネットワーク的な利用やプログラムができることは，その道具の利用範囲を広げる上で重要である．本システムはMathematicaにそのような機能を付加して，新しい利用が可能であることを示した．

## 参 考 文 献

[1] MIYAJI, Chikara and KIMURA, Hiroshi : Writing Graphical User Interface using Mathematica and MathLink, Innovation in Mathematics, Proceedings of Second International Mathematica Symposium, Computational Mechanics Pub., 307-314, 1997

[2] 宮地 力：Mathematicaによるネットワークプログラミング，岩波コンピュータサイエンス，岩波書店，1998.

[3] Wolfram, Stephen : The Mathematica Book, 3rd Edition, Cambridge U. Press, 630-674, 1996.