Python を用いた Asir 実行環境の実装

An implementation of Asir execution environment by Python

神戸大学理学部数学科　　池田和暉 *1
Kazuki Ikeda
Department of Mathematics, Faculty of Science
Kobe University

神戸大学・大学院理学研究科数学専攻　　高山信毅 *2
Nobuki Takayama
Department of Mathematics, Faculty of Science
Kobe University

Abstract

We develope a graphical user interface for Risa/Asir computer algebra system. By utilizing Kivy on Python and OpenXM protocols, the interface is robust and its code side is small.

1　はじめに

Risa/Asir はおもに多項式計算に特化した数学ソフトウエアである. 1990 年代に富士通の国際研で開発がはじまり[1], その後富士通ライセンスの元のオープンソースソフトウエアとして開発が進められてきた. Asir (以下 Risa/Asir を 単に Asir と呼ぶ) は OpenXM プロトコル[2]をサポートしているので, 他のソフトウエアに容易にかつロバストな形で統合することが可能である. Asir 本体はコマンドラインインターフェスを持つのみで, 標準的な GUI は持たない. Asir に対して今までさまざまなインターフェースが構築されてきた. texmacs インタフェース, MacOS 専用の cfep/asir[3], emacs インタフェース[4]などである.

さてここ数年のプログラミング言語の人気ランキングでは Python が一位の座を守っている[5]. それに伴いライブラリの充実度も指数関数的と言えるであろう. となると, Python で Risa/Asir の GUI を作ってみよう, というのは当然の流れである.

Asir は C 言語に似たユーザ言語をもつシステムで初心者に優しく, C 言語に熟達した上級者にも心地良いユーザ言語を提供している. 筆者 (高山) は神戸大学理学部数学科の科目, 計算数学 I で, “Risa/Asir で入

*1 〒 657-8501 神戸市灘区六甲台町 1-1　E-mail: 1643081s@stu.kobe-u.ac.jp
*2 〒 657-8501 神戸市灘区六甲台町 1-1　E-mail: takayama@math.kobe-u.ac.jp

[1] そのあたりの経過については, “斎藤, 竹島, 平野, RISA/ASIR 日本で生まれた数式処理ソフト, SEG 出版” が詳しい.

[2] http://www.openxm.org

[3] http://www.math.kobe-u.ac.jp/Asir より配布されている

[4] openxm.org が配布している openxm debian パッケージに同梱されている. use-asir-mode.sh --local-yes コマンドでこの機能が使える. Windows 版はインストール説明書に従い設定する. MacOS 版は上記 Asir 配布サイトにある.

[5] https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019

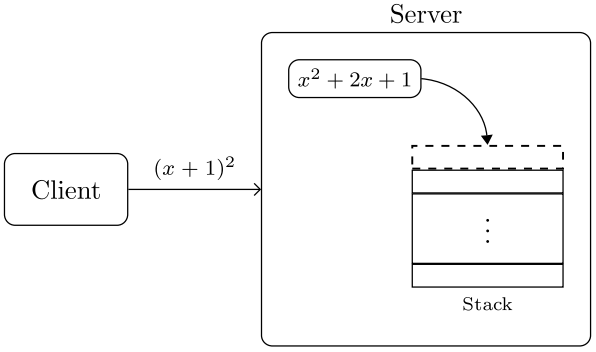
門, それから C 言語を少し", という組み立てで講義を行っている[6]. この講義の実習教材として使えるような Asir の Python による GUI を開発目標に本プロジェクトはスタートした.

2 OpenXM

OpenXM プロジェクトは数学ソフトウエア間の通信のためのさまざまなプロトコルを提案している. 我々は "野呂, 高山, OpenXM の設計と実装: クライアントサーバモデルと数学共通表現 (OpenXM-RFC 100), 1997–2002" [7] で仕様が定義されている, 通信プロトコル (通信規約) に従い Python と Asir の接続を行うこととした. OpenXM-RFC 100 では数学データの効率的通信のために bignum (整数 **Z**) や多項式の表現なども規定されているが, GUI との通信のためには文字列型のみを利用するものとした. また Python から Asir を呼び出すために, OpenXM プロジェクトが配布している, OpenXM プロトコルをサポートする C 言語のライブラリである `ox_toolkit` を shared library として Python に読み込ませて接続することとした. このライブラリを用いることによりプロトコルが少々複雑な計算の中断機構 (`ox_reset`) も容易に利用することができる. 文字列型のデータを Python から送信し, Risa/Asir に評価させて, 結果をまた文字列型で得るための C 言語と `ox_toolkit` によるサンプルプログラムは OpenXM のソースコード[8]の `OpenXM/src/ox_toolkit/sample5.c` である.

さて, OpenXM-RFC 100 の概略を説明しよう. このプロトコルでは計算サーバは stack machine であり, クライアントと OX メッセージをやりとりする. OX メッセージには COMMAND, DATA, などの種類がある. COMMAND 型の OX メッセージは, stack machine に対するコマンドを運ぶカプセルであり, スタックマシンに対する命令としては `SM_popString`, `SM_shutdown`, ... などがある. DATA 型の OX メッセージは, さまざまなデータを運ぶカプセルであり, データ型としては `CMO_STRING`, `CMO_INT32`, `CMO_LIST`, ... などがある.

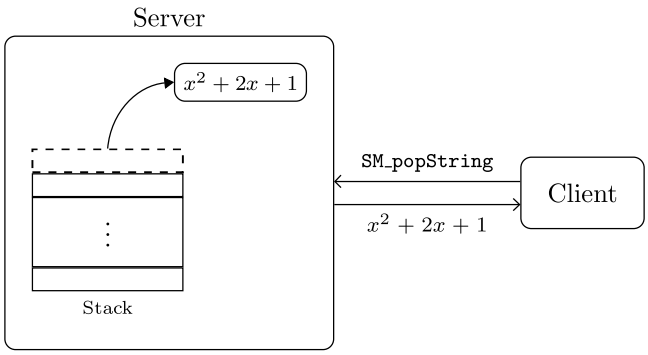
OX サーバーは計算の Request が来ると結果を内部の Stack に積む (この時点では結果を Client に返却しない). Client から `SM_popString` 命令が来て初めて Stack から直前の計算結果を取り出して返す. このようにすることで, 複数のサーバに計算を任せる際にうまく並列化できるようになる.



[6] 野呂, 高山, Risa/Asir ドリル 2017

[7] `http://www.math.kobe-u.ac.jp/OpenXM/Current/OpenXM-RFC.html`

[8] `http://www.math.kobe-u.ac.jp/OpenXM` にダウンロードのリンクがある.



3 ctypes

ctypes [9] は Python から shared library をロードするためのライブラリである. たとえば `libsample.so` を読み込むには

```
import ctypes
lib = ctypes.cdll.LoadLibrary("libsample.so")
```

とする. 関数を呼ぶためには引数の型と返り値の型を指定する必要がある. `int` を 2 つ受け取って `char*` を返す関数

```
char* myfunc(int a, int b);
```

に対して

```
lib.myfunc.argtypes = (ctypes.c_int, ctypes.c_int)
lib.myfunc.restype = ctypes.c_char_p
```

とすることで `lib.myfunc(12, 3)` のように使える.

この機能を用いて我々は OpenXM による通信を行う `ox_toolkit` を python に shared library としてロードして利用している.

4 Kivy

Kivy [10] は Python で GUI アプリケーションを作成するためのライブラリである. 特徴として

- KV language によって宣言的なレイアウト構築が可能
- オープンソースで開発されており, コミュニティが活発

などがある.

たとえば, ボタンを 2 つ縦に並べるレイアウトは Kivy で次のようにして実現できる.

main.py

```
from kivy.app import App
```

[9] https://docs.python.org/ja/3/library/ctypes.html
[10] https://kivy.org/

```
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button

class TwoButtonsApp(App):
    def build(self):
        root = BoxLayout(orientation="vertical")
        btn1 = Button(text="Hello")
        btn2 = Button(text="World")
        root.add_widget(btn1)
        root.add_widget(btn2)
        return root
app = TwoButtonsApp()
app.run()
```

KV language を使う場合, 以下のようになる. 注意点として, KV language を記述するファイル名を Python のソースコード内のクラス名に対応させる必要がある.

main.py

```
# from ...
class TwoButtonsApp(App):
    def build(self):
        return BoxLayout()
# app = ...
```

twobuttons.kv

```
<BoxLayout>:
    orientation: "vertical"
    Button:
        text: "Hello"
    Button:
        text: "World"
```

アプリケーションが大きくなると「画面全体では Widget (UI を構成する要素; ボタンなど) を縦に並べるが, 上部では横方向に Widget を配置したい」というようなネストしたレイアウトを扱いたい場合がある. Python のみで実現する方法は Widget 同士の親子関係を分かりづらくして, メンテンナンス性が損なわれる. KV language ではインデントの深さとネストの深さが対応しているため, 親子関係は明確である.

5 主な機能とソースコードの読み方

開発した GUI アプリケーションが提供している機能として

- `#define` を展開
- 計算の中断

- `print_png_form()` [11] の結果をプレビュー (図 1)
- エラー行を検出 (図 2)
- ソースコード整形 (図 3, 図 4)
- ソースコードの HTML 出力 (図 5)

などがある.

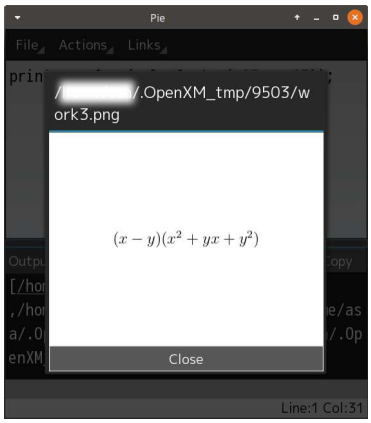


図 1: `poly_factor(x^3 - y^3)`

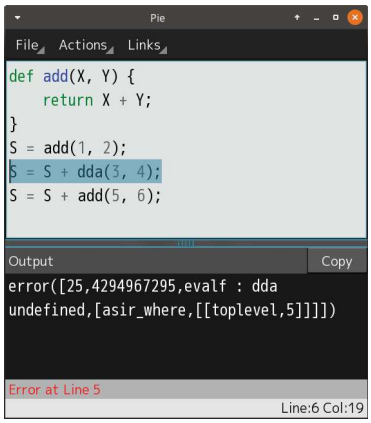


図 2: 未定義の関数を呼んでいるエラー

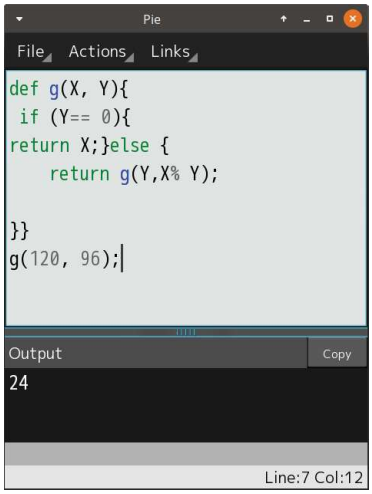


図 3: ソースコード整形前

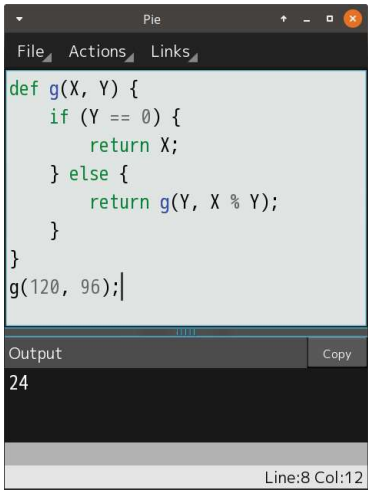


図 4: ソースコード整形後

ここでは, 最も基本的な「入力されたソースコードを Asir のプログラムとして解釈して計算を行い, その結果を表示する機能」の実装を説明する. 以降のファイル名や行番号は `b2d750f4a8` で表される commit 時点でのソースコード [12] に対応している. 最新版は `https://github.com/ia7ck/pie-editor` を参照してほしい.

[11] `http://www.math.kobe-u.ac.jp/OpenXM/1.2.1/doc/asir-contrib/html-ja/cman-ja_41.html#SEC41`

[12] `https://github.com/ia7ck/pie-editor/tree/b2d750f4a86e6ceb4e774e3a31dd13ec3cb81e55`

図 5: HTML 出力したソースコードをブラウザで開いた様子

アプリケーションは Editor クラスのインスタンスを 1 つ持ち, Editor クラスでは Header クラス (各機能を実行するためのボタンを管理するクラス) や Result クラス (計算結果の文字列を表示するためのクラス) のインスタンスを持っている. ヘッダー部の “Run” ボタンを押すとイベントに紐付けられた関数 run_source_code が呼ばれる.

pie.kv

```
67  on_release: root.editor.coderunner.run_source_code()
```

run_source_code では主に

1. エディタに入力されたソースコードを文字列として取得
2. その文字列を Asir サーバに送信

を行う. 1. について, 入力は SourceCode クラスの text 属性に保持されているため次のようにして得られる.

coderunner.py

```
19  text = selection if len(selection) > 0 else e.source_code.text
```

条件分岐は「選択した箇所のみを評価する機能」を実現するために必要であるが, いまは気にしなくてよい. #define の展開などを経た text を server_input とする. 2. の処理に対応する箇所は以下である.

coderunner.py

```
25  sv.execute_string(server_input)
```

execute_string は最終的には ox_toolkit の ox_execute_string を呼び出す.

送信後は 1 秒ごとにサーバに計算結果を問い合わせる. 計算が終わっていた場合, 次のようにして結果を受け取る.

coderunner.py

```
43  res = sv.pop_string()
```

pop_string は ox_toolkit の ox_popString と似た処理を行っている．受け取った文字列をもとにして結果を表示する部分を更新する．

coderunner.py

```
45  e.update_result(res)
```

Result クラスは SourceCode クラスと同様に text 属性を持っているため, Result クラスのインスタンスを通して文字列をそこにセットすればよい．実際に, update_result 内では次のようにしている．

main.py

```
299  r.text = res
```

6 終わりに

Python 製の GUI フレームワーク Kivy を用いて Asir 実行環境を実装した．次のステップとしては, 実際に実習の時間で学生に今回開発したアプリケーションを使ってもらい改善していくこと, また数学研究のツールとしての使いやすさも改善していくことなどがある[13]．また, ターミナルエミュレータの termux [14] を用いて Android タブレット上で動作させる試みも考えられるだろう．

謝　辞

This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University. The second author is supported by JSPS KAKENHI Grant Number 17K05279.

文献は本文中の footnote に記載している．文献表は省略する．

[13] 現状でも print_png_form 関数の呼び出しで数式を読みやすく表示する機能が搭載されている．小技ではあるが便利である．

[14] https://termux.com/