РОЗШИРЕННЯ МОЖЛИВОСТЕЙ WEB-СКМ SAGE

С.В. Шокалюк, Г.Ю. Руденко
Україна, м. Кривий Ріг, Криворізький державний педагогічний університет
ksv_ipm@mail.ru

SAGE (*Software for Algebra and Geometry Experimentation* – ПЗ для агебраїчних та геометричних досліджень) – нова вільно поширювана Web-орієнтована СКМ, що об'єднує в собі такі відомі системи GAP, GSL, Matplotlib, Maxima, MWRANK, NetworkX, NTL, Numpy, PARI, Singular та багато інших засобами Python, Lisp, Fortran 95, та C/C++.

SAGE має власне символьне ядро, проте виступає переважно як інтегратор різних систем, надаючи їм єдиний Web-інтерфейс. Можливість виконання на Web-сторінках, генерованих SAGE, програм мовами Python, Lisp, Java, надає їм високого рівня інтерактивності, порівняного з традиційними СКМ (Mathematica, MathCAD, Maple), без суттєвих вимог до апаратних ресурсів комп'ютера користувача (необхідні лише браузер та мережне з'єднання).

Відкритий характер системи дає можливість додавання до неї нових функцій, типів та класів, створювати нові бібліотеки та інтегрувати у неї нові програми як: 1) сценарії SAGE; 2) сценарії на Python з використанням бібліотеки SAGE; 3) програми на C/C++, інтегрованими засобами Cython; 4) код Cython; 5) програма мовою СКМ (наприклад, сценарій Maxima); 6) будь-яка комбінація з пп. 1–5.

Найпростіший спосіб додання нової функції до SAGE – вставка коду Cython. Якщо розпочати введення командою `%cython`, то при виконанні вводу він: а) зберігається у файлі; б) транслюється у мову C; в) компілюється у динамічну бібліотеку (.so), що завантажується надалі при повторному виклику уведеного.

Застосування Cython дозволяє за однакового з Python коду (додається лише %cython) досягти на обчислювальних операціях більш ніж 200-кратного прискорення. Наприклад, код для швидкого піднесення до степеня двійки [1]

```
{{{
%cython
def is2pow(unsigned int n):
  while n != 0 and n%2 == 0:
    n = n >> 1
  return n == 1

time [n for n in range(10^5) if is2pow(n)]
}}}
```

дає результат

```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384,
32768, 65536]
CPU time: 0.05 s, Wall time: 0.06 s
```

Перетворення його на код Python (видаленням `%cython`) показує вже

```
CPU time: 13.04 s, Wall time: 15.08 s
```

Інтеграція існуючого ПЗ до SAGE виконується за допомогою псевдотермінального інтерфейсу: єдиною вимогою до інтегрованого ПЗ є можливість його роботи із стандартними потоками введення/виведення. Взаємодія між інтегрованою програмою та SAGE виконується неблокованими (асинхронними) каналами введення/виведення за допомогою методів `_send`, `_so_far` та `_get`.

Для інтеграції власного математичного ПЗ у SAGE ми пропонуємо наступний шаблон інтерфейсу модельної програми `mysystem`:

```
"""
[[Наводиться опис ПЗ, до якого створюється інтерфейс]]
EXAMPLES:
[[Приклади застосування нового інтерфейсу у SAGE]]
AUTHORS:
    Ваші прізвище та ініціали
"""

from expect import Expect, ExpectElement, ExpectFunction, FunctionElement
from sage.misc.misc import verbose

class MySystem(Expect):
    """
    [[Наводиться допомога з mysystem]]
    """
    def __init__(self,
                 maxread=100000, script_subdirectory=None,
                 logfile=None,
                 server=None,
                 server_tmpdir=None):
        Expect.__init__(self,
                        name = 'mysystem', #ім'я системи
                        # Шаблон регулярного виразу для програмного
                        # вводу (чим складніше, тим краще)
                        prompt = '>> ',
                        # Команда запуску інтегрованої програми
                        command = "mysystem",
                        maxread = maxread,
                        server=server,
                        server_tmpdir=server_tmpdir,
                        script_subdirectory = script_subdirectory,
                        # Якщо цей прапор=true, то при натисканні
                        # Ctrl-C перевантажується весь інтерфейс
                        restart_on_ctrlc = False,
```

```
                    # Якщо цей прапор=true, друкує повідомлення
                    # перед початком виконання команди
                    verbose_start = False,
                    logfile=logfile,
                    # Максимальний розмір введення, при
                    # перевищенні якого вивід іде у файл
                    eval_using_file_cutoff=1024)
    self.__seq = 0

def _repr_(self):
    return 'Інтерпретатор MySystem'

def __reduce__(self):
    return reduce_load_mysystem, tuple([])

def __getattr__(self, attrname):
    if attrname[:1] == "_":
        raise AttributeError
    return MySystemFunction(self, attrname)

# тут вкажіть команду завершення роботи mysystem
def _quit_string(self):
    raise NotImplementedError

# тут вкажіть команду mysystem для читання з файлу filename
def _read_in_file_command(self, filename):
    raise NotImplementedError

# має повернути список з всіх та ідентифікаторів mysystem
def trait_names(self):
    raise NotImplementedError

# тут реалізуйте читання з файлу filename у mysystem
def read(self, filename):
    raise NotImplementedError

def kill(self, var):
    # команда знищення змінної із заданим ім'ям
    pass

def console(self): # виконує консольну команду (див. нижче)
    pass

def version(self): # отримує версію системи (див. нижче)
    pass

def _object_class(self):
    return MySystemElement

# визначає символ для позначення істини у mysystem
def _true_symbol(self):
    raise NotImplementedError

# визначає символ для позначення хиби у mysystem
def _false_symbol(self):
```

```
        raise NotImplementedError

    # визначає символ для позначення еквівалентності у mysystem
    def _equality_symbol(self):
        raise NotImplementedError

    def help(self, command): # допомога по заданій команді
        raise NotImplementedError

class MySystemElement(ExpectElement):
    # Опишіть тут елементи інтегрованої системи
    def trait_names(self):
        return self.parent().trait_names()

class MySystemFunctionElement(FunctionElement):
    def _sage_doc_(self):
        M = self._obj.parent()
        return M.help(self._name)

class MySystemFunction(ExpectFunction):
    def _sage_doc_(self):
        M = self._parent
        return M.help(self._name)

def is_MySystemElement(x):
    return isinstance(x, MySystemElement)

# Створення об'єкту
mysystem = MySystem()

def reduce_load_MySystem():
    return mysystem

import os
def mysystem_console(): # Виконує процес
    os.system('mysystem')

def mysystem_version():
    """
    Приклад:          sage: mysystem.version()
    """
    raise NotImplementedError
```

Більш простим варіантом інтеграції є випадок, коли система завантажується лише для виконання команди, поданої на її вхід з командного рядка чи файлу (як це зроблено у [2] для інтерпретатора Lisp).

Література:

1. Stein, W., Joyner, D. Sage Programming Guide. – 2008. – 86 p.

2. Теплицький І.О., Семеріков С.О. Дослідження дидактичних можливостей мови Ліспу як засобу побудови інтелектуальних систем у шкільному курсі інформатики // Проблеми сучасного підручника: Зб. наук. праць. – К.: Педагогічна думка, 2004. – Вип. 5., Ч. ІІ. – С. 183-191.