数理解析研究所講究録
第1930巻 2015年 93-100

The current state of computer algebra system on tablet devices

Mitsushi Fujimoto
Fukuoka University of Education *

1 Introduction

Infty project[1] developed and released some useful software including InftyReader – an OCR system for mathematical documents. InftyEditor is one of the products developed by Infty project, and is a mathematics typesetting tool. The author, a core member of Infty project, built into InftyEditor a computing function for mathematical expressions[2]. In 2003, AsirPad[3], a computer algebra system with a handwriting interface on PDA, was developed using the technology of InftyEditor. AsirPad can communicate with Risa/Asir[4] through OpenXM protocol[5], and calculate mathematical expressions inputted by handwriting. This is an application for Zaurus that is a PDA with 400MHz CPU, 64MB memory, Linux OS, 3.7inch VGA screen and the weight 250g. The mainstream of the current mobile devices is shifting to smartphones or tablet devices and some computer algebra systems for these platforms are developed. In this article we explore the current state of computer algebra system on tablet devices.

2 Mathellan project and tablet devices

AsirPad was used to present a lecture on RSA cryptography at a junior high school[6]. Ordinary calculator is not available because encryption and decryption in RSA use division of large numbers. The students learned how to encrypt/decrypt their messages through calculations by AsirPad. They could input mathematical expressions and calculate without any special training. We could get a result for usefulness of PDA and handwriting interface from this lecture.

In 2010, we started *Mathellan* Project, a mobile Math e-Learning project, to go the next step. The current version of AsirPad can not provide math quizzes. Also, the current e-Learning systems/LMS are not good at displaying/inputting mathematical expressions and checking student answers to math quizzes. The aim of this project is to develop a Math e-Learning system for pen-based mobile devices. We call our Math e-Learning system Mathellan and consider the following as the basic requirements[7]:

- displaying math expressions
- inputting math expressions
- including graphs of elementary functions

*fujimoto@fue.ac.jp

- freehand drawing
- automated checking of student answers.

We need a new platform instead of Zaurus in order to realize the above requirements. And we chose tablet devices as a client of Mathellan.

3 Various Tablet devices

At present, a lot of manufacturers release tablet devices. The operating systems of them are Android, Blackberry, iOS, webOS, Windows 8, and so on. However, the technical specifications of current tablet devices are almost same as follows:

- CPU: ARM architecture(v7), 32bit
- CPU core: 2 or 4
- CPU speed: 600MHz – 1.4GHz
- FPU: Hardware
- GPU: Hardware
- Memory: 512MB – 1GB
- Screen resolution: 1024x600 – 2048x1536
- Sensor : Multi-touch, Accelerometer, Gyroscope, GPS

The specification has improved very much compared with Zaurus. Thus, it is sufficient to implement computer algebra systems on tablet devices.

4 Use of CAS from Tablet Devices

There are four methods to access to computer algebra systems from tablet devices. The first method is to use a native application including a computer algebra engine on tablets (Method 1). The second method is to access to a computer algebra system on the other machine through an application on tablets (Method 2). The third method is to access to a computer algebra system on the other machine through a web browser on tablets (Method 3). The last method is to use a work sheet including a computer algebra kernel (Method 4).

The following is a table for computer algebra systems on tablets we investigated.

Method	App name	CAS engine	OS	Paid
Method 1 (Native CAS App)	MathStudio	original	Android/iOS	✓
	Mathomatic	Mathomatic	Android/iOS	✓
	PariDroid	PARI	Android	
	Maxima on Android	Maxima	Android	
	JavaYacas	Yacas	Android	
	Yacas for iPhone[1)]	Yacas	iOS	✓
	iCAS	Reduce	iOS	✓
	PocketCAS	Giac/Xcas	iOS	✓
	Pi Cubed	original	iOS	✓
	Python Math	SymPy	iOS	
Method 2 (CAS through App)	WolframAlpha	Mathematica	Android/iOS	✓
	SageMath	GAP, Maxima etc	Android/iOS	
Method 3 (CAS through Web)	WolframAlpha	Mathematica	N/A	
	Omega	Maxima	N/A	
	Geogebra	MathPiper etc	Java	
Method 4 (CAS work sheet)	Maple Player	Maple	iOS	
	Wolfram CDF Player	Mathematica	iOS (not yet)	

5 Implementation of a CAS engine for tablet devices

We adapt Method 1 for the client of Mathellan, and need the components, i.e. CAS engine / GUI / communication mechanism with CAS engine / internal form for mathematical expressions, to realize it. This method is also suitable for UNIX-based computer algebra systems. In this section, we concentrate on building a CAS engine.

It is common to build console applications for UNIX-based OS by the following steps:

1. Downloading a source of an application:
   e.g., `wget http://***.sourceforge.org/***/***-2.0.tar.gz`
2. Extracting the source:
   e.g., `tar zxvf ***-2.0.tar.gz`
3. Generating a Makefile for building the application:
   e.g., `cd ***-2.0;./configure`
4. Building the application:
   e.g., `make`
5. Installing the application:
   e.g., `make install`

[1)] This is now unavailable for downloading from App Store.

We shall explain the compiling of the Risa/Asir's source code for Android by (i) a self-build environment and (ii) a cross-build environment. Of course, we have to modify a portion of the source code with respect to the OS and CPU architecture of tablets before building.

5.1 Self build environment

Most of computer algebra systems need some external libraries, e.g., garbage collector or arbitrary-precision arithmetic library. Some of them are not presumed to be built by cross-build environments. In such a case, self-build environment is the best way to avoid troubles with building. Risa/Asir needs a garbage collector Boehm-GC. We made a binary of Risa/Asir for Zaurus as the CAS engine of AsirPad by this way since a self-build environment for Zaurus was available. Unfortunately, this way is unavailable for iOS because the terminal application for iOS is not provided and we cannot access to the file system of iOS unless we remove the limitations of iOS by a hack.

On the other hand, terminal applications for Android are available through Google Play Store and users can access to the file system of Android without a rooted account. Furthermore, a self-build environment including gcc is provided by a user community. The following is a self-build process for Risa/Asir on Android.

1. Download and installation of 'TerminalIDE' through Google Play Store. This is not only a terminal application but also includes a lot of command line tools.

2. Installation of GCC according to `http://rwiki.sciviews.org/doku.php?id=getting-started:installation:android`.

3. Compiling Boehm-GC.

```
wget http://www.hpl.hp.com/personal/Hans_Boehm/gc/gc_source/gc.tar.gz
tar zxvf gc.tar.gz
export CFLAGS="-D IGNORE_DYNAMIC_LOADING"
./configure --prefix=$HOME --disable-threads
make
make install
```

4. Compiling of Risa/Asir.

```
./configure --prefix=$HOME
make
make install
make install-lib
```

However, you will have a lot of collisions in header files. And you have to fix all collisions by modifying source codes or header files to get a binary.

5.2 Cross build environment

Android NDK, a cross-build environment to develop C/C++ applications for Android, is provided by Google. However, its C library is Bionic LIBC and is different from the standard C library GLIBC. Therefore, developers need to modify a lot of source codes to use Bionic LIBC. Fortunately, some third-party companies provide cross-build environments based on GLIBC. The following is a cross-build process for Risa/Asir using 'Sourcery CodeBench Lite Edition'.

1. Download and installation of 'Sourcery CodeBench Lite Edition' via the developer's site [2] to your Linux PC.

2. Setting environment variables.

```
export CROSS_COMPILE=arm-none-linux-gnueabi-
export ARCH=arm
export CFLAGS="-I/opt/Sourcery_G++_Lite/arm-none-linux-gnueabi/
  include -static"
export LDFLAGS="-L/opt/Sourcery_G++_Lite/arm-none-linux-gnueabi/
  lib -static"
export CC=arm-none-linux-gnueabi-gcc
export CXX=arm-none-linux-gnueabi-g++
export STRIP=arm-none-linux-gnueabi-strip
export AR=arm-none-linux-gnueabi-ar
export AS=arm-none-linux-gnueabi-as
export LD=arm-none-linux-gnueabi-ld
export RANLIB=arm-none-linux-gnueabi-ranlib
export OBJDUMP=arm-none-linux-gnueabi-objdump
export NM=arm-none-linux-gnueabi-nm
```

3. Compiling Boehm-GC for Android on a Linux machine.

```
wget http://www.hpl.hp.com/personal/Hans_Boehm/gc/gc_source/gc.tar.gz
tar zxvf gc.tar.gz
export CFLAGS="-D IGNORE_DYNAMIC_LOADING -I/opt/Sourcery_G++_Lite/
  arm-none-linux-gnueabi/include -fsigned-char -static"
./configure --prefix=$HOME --disable-threads --host=
  arm-none-linux-gnueabi --build=i686-pc-linux-gnu
make
make install
```

4. Compiling of Risa/Asir for Android on a Linux machine.

```
./configure --prefix=$HOME --host=arm-none-linux-gnueabi
  --build=i686-pc-linux-gnu
```

[2] http://www.mentor.com/embedded-software/sourcery-tools/sourcery-codebench/editions/lite-edition/

```
make
make install
make install-lib
```

In case of this method, it is important to use static libraries by `-static` option for GCC because Android OS does not have GLIBC dynamic libraries.

6 Transfer to a target device and testing

Generally, binaries created by a cross-build environment are tested on an emulator. The Android emulator on PC is not fast. If the binary is a Java application, we can use Intel x86 Emulator Accelerator(HAXM) which is faster than the normal emulator. However, the CAS binary created by the above environment cannot be executed on HAXM since it is not Java application. In such a case, it is effective to transmit to the target device and to test on it.

Transferring a binary to a target device is done as follows:

1. Connecting a target device to PC with a USB cable.
2. Executing the following commands in a terminal application on PC:

```
adb shell mkdir /sdcard/asir
adb push ./asir /sdcard/asir/
adb shell chmod 755 /sdcard/asir/asir
```

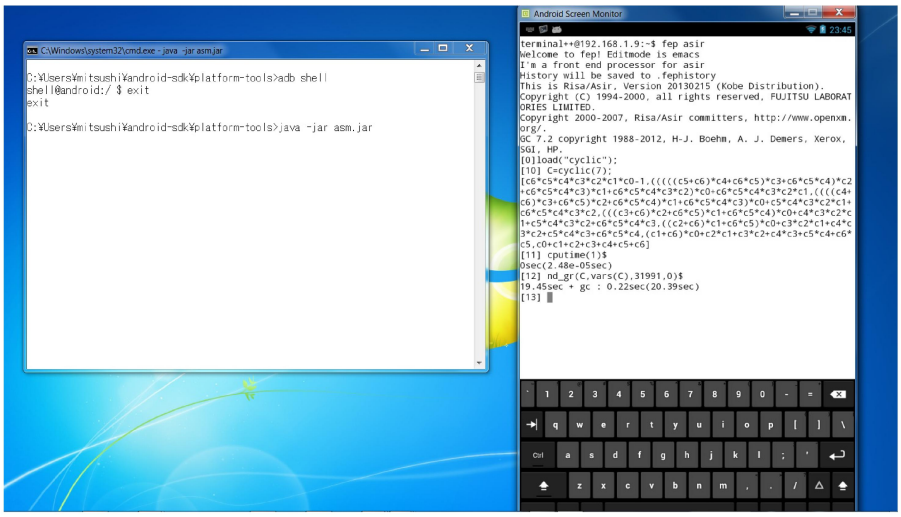
It is very useful to display the screen of the target device on PC for the debugging and presentation. We can use Android Screen Monitor[8] to realize it.

1. Connecting a target device to PC with a USB cable.
2. Executing the following commands in a terminal application on PC:

```
adb shell
exit
java -jar asm.jar
```

3. Selecting the target device from the Android Device window.

The following screenshot is the screen of Nexus 7 monitored on PC using this tool.



The Risa/Asir binary for Android obtained by the above process is available from `http://www.fue.ac.jp/~fujimoto/asiroid/`.

7 Conclusion

In this article, we reported the current state of computer algebra systems on tablets and described methods to build a computer algebra system with CUI from source codes. Console based computer algebra systems are useful as CAS engines. However, a user needs a terminal application and a software keyboard on the tablet to use them. The interaction with CAS engine through terminal is too inefficient. A user have to switch the software keyboard over and over again to complete the input operation. We need a GUI for computer algebra system on tablets.

Y. Honda developed a GUI of Maxima using WebView, and released it as 'Maxima on Android' on Google Play Store[9]. This application uses MathJax to display mathematical expressions, and can output computational results beautifully. WebView is a core class in the WebKit. This method can be used to develop applications for iOS having WebKit as a standard framework. We think that applications should be developed by the method not depending on platform like this. On the other hand, the GUI of AsirPad was developed using Qt. Qt is a cross-platform application framework, and can be used to build applications for various operating systems: Windows, MacOS X, Linux, Solaris, QNX etc. The next version of Qt seems to support Android and iOS. We are planning to use Qt to build the GUI for a client of Mathellan.

References

[1] Infty project, `http://www.inftyproject.org`

[2] M. Fujimoto, T. Kanahori and M. Suzuki, Infty Editor – A Mathematics Typesetting Tool with a Handwriting Interface and a Graphical Front-End to OpenXM Servers, Computer Algebra – Algorithms, Implementations and Applications, RIMS Kokyuroku vol.1335 (2003) pp. 217–226.

[3] M. Fujimoto and M. Suzuki, AsirPad - A Computer Algebra System with a Pen-based Interface on PDA, Proceedings of the Seventh Asian Symposium on Computer Mathematics, Korea Institute for Advanced Study (2005) pp. 259–262.

[4] M. Noro, et al., *A computer algebra system Risa/Asir*, `http://www.math.kobe-u.ac.jp/Asir/asir.html`

[5] M. Maekawa, M. Noro, N. Takayama, Y. Tamura and K. Ohara, The Design and Implementation of OpenXM-RFC 100 and 101, Computer Mathematics (Proceedings of the Fifth Asian Symposium on Computer Mathematics), World Scientific (2001) pp. 102–111.

[6] M. Fujimoto, M. Suzuki and T. Kanahori, On a classroom experiment using PDA and handwriting interface (in Japanese), IPSJ Symposium Series vol.2006, no.8 (2006) pp. 331–338.

[7] M. Fujimoto and S. M. Watt, An Interface for Math e-Learning on Pen-Based Mobile Devices, Proceedings of the Workshop on Mathematical User-Interfaces, `http://www.activemath.org/workshops/MathUI/10/proc/FujimotoWatt.html` (2010).

[8] Android Screen Monitor, `http://code.google.com/p/android-screen-monitor/` (2009).

[9] Y. Honda, Maxima on Android, `https://sites.google.com/site/maximaonandroid/` (2012).

Mitsushi Fujimoto
Department of Mathematics, Fukuoka University of Education
1-1 Akamabunkyo-machi, Munakata 811-4192, JAPAN.
Web: `http://www.fue.ac.jp/~fujimoto/`
E-mail: `fujimoto@fue.ac.jp`