Li Xuanji

NUS High School of Math and Science
20 Clementi Avenue 1 Singapore 129957
`http://www.highsch.nus.edu.sg/`

Phone:	96959264
Email:	`xuanji@gmail.com`
Github:	`github.com/zodiac`
Web:	`xuanji.li`

Education

NUS High School of Math and Science	2006-2011.

Work Experience

Institute for Infocomm Research	Mar-Nov 2009
DSO National Laboratories	Nov-Dec 2009
.	Oct-Dec 2010

Programming

These are the languages I am comfortable with:

Python

C/C++

Scheme

Side Projects

Interactive SICP

Etc

ankiResource

A set of web apps written in Django, designed to make a convenient way to share sentences, resources, etc to aid in learning languages.

ankiResource is based on the Anki (`http://ankisrs.net/`) Spaced Repitition System, an electronic flashcard system which shows flashcards to the user and lets him grades how easy it was to answer the card. An algorithm is used to determine when the user will next see the card.

ankiResource was designed to allow users to share decks of sentences with each other. We chose sentences as the basic unit that could be shared because it's reccommended by the popular system All Japanese All The Time. Features include automatically parsing and de-conjugating sentences for search with the MeCab morphological analyzer.

screenshots: `https://docs.google.com/present/view?id=dfbtczd8_23gtg4b2j2`
url: `github.com/bombpersons/ankiResource/`

Research

Following is a list of the projects I have completed.

Directional Localization of Objects via 2.4Ghz Wireless Signals

In many industrial and office buildings knowing the position of an object is helpful for tracking individuals and moving goods. Previous research involving radio frequency (RF) wireless localization achieved this by creating a fingerprint of signal strengths for each location with several routers with dipole (omnidirectional) antennas and matching observed signal strengths to the location with the most similar fingerprint. However, wireless fingerprinting is unable to determine the direction the user is facing, and hence will not be very helpful in situations where the user is lost or needs directions, e.g. in museum tours or navigating large buildings. In this paper we present a method of which a user, using a directional patch antenna, can use wireless fingerprinting techniques to determine the direction he/she is facing.

Smatch: Static Analysis for C

This paper describes the authors' experience working on an open-source static checker, smatch. We do so by taking real-life bugs from the Linux kernel and running smatch on it. If smatch was not able to detect it (a false negative) we try to write a check or modify an existing one so that it can. We believe that this allows us to write better, more realistic checks. Also, we encountered many bugs for which it was not feasible to write a checker for, and discuss them. Examples of checkers written in smatch are given.
url: `http://repo.or.cz/w/smatch.git/`

LEEK: State Merging for Automatic Test Generation

Automatic test generation is the problem of generating test cases in a given program for the purposes of vulnerability detection and program verification. One method of achieving this is Symbolic Execution, which executes the program with *symbolic* variables to quickly test every possible execution path that the program may take. Often symbolic execution creates many execution paths that are similar; our project focuses on determining when these paths can be merged to cut down on repetitive operations. In addition, we have developed a heuristic to avoid merging when the merging algorithm performs worse. We prove our algorithm's soundness and correctness with respect to the base system and demonstrate its practical usefulness by generating test cases for real world programs. We observe speed up from exponential to linear time for certain programs.
url: `https://github.com/zodiac/klee-nush/`

Python development

I sporadically look at the Python issue tracker and work on patches and have gotten a few of them accepted. You can find me by searching for my full name.

Competitive Programming

I learned most of my theoretical CS stuff through preparing for and joining competitive programming contests which require (high-school students) to write programs to accomplish various tasks which is then graded on secret test data; a simple example would be “find all anagrams in a given list of words." Standard algorithms we are expected to know include dynamic programing, binary search, depth-first and breadth-first search, Dijkstra's algorithm, Kruskal's algorithm, suffix array etc. We are also often expected to design new algorithms for difficult problems.

Last updated: February 10, 2014