UCF | College of Engineering and Computer Science

UNIVERSITY OF CENTRAL FLORIDA

APPLeSEEd@UCF

Free Software Enables Free Science

Paul Gazzillo and Joshua Santana

@paul_gazzillo

pgazz.com

About Me

https://paulgazzillo.com

Assistant Professor of Computer Science at University of Central Florida

Started Fall 2018

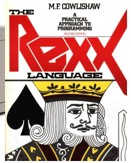
Research areas: programming languages, software engineering, security

Some contributions

- Parsing C without preprocessing
- Detection of side channel vulnerabilities
- Makefile configuration discovery

My Path to Free Software

Childhood



M.F. COWLISHAW

THE

A PRACTICAL APPROACH TO PROGRAMMING

SECOND EDITION

Rexx

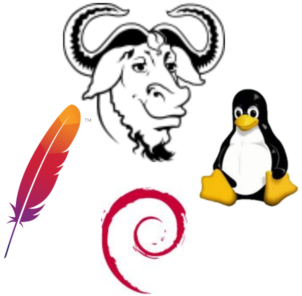
LANGUAGE

College and work

UNIX®

Grad school

Since grad school



Future: services, mobile, hardware

?

I Love My Job!

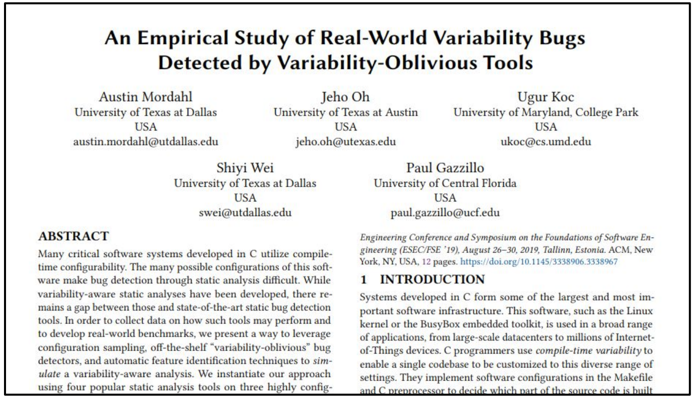
Research

- Articulate good problems
- Develop novel solutions
- Make and disseminate new knowledge and tools

Teaching

- Teach computer science principles
- Prepare students for industry and/or academia
- Convey the joy of programming and computing

Working with Students



An Empirical Study of Real-World Variability Bugs Detected by Variability-Oblivious Tools

Austin Mordahl
University of Texas at Dallas
USA
austin.mordahl@utdallas.edu

Jeho Oh
University of Texas at Austin
USA
jeho.oh@utexas.edu

Ugur Koc
University of Maryland, College Park
USA
ukoc@cs.umd.edu

Shiyi Wei
University of Texas at Dallas
USA
swei@utdallas.edu

Paul Gazzillo
University of Central Florida
USA
paul.gazzillo@ucf.edu

ABSTRACT

Many critical software systems developed in C utilize compile-time configurability. The many possible configurations of this software make bug detection through static analysis difficult. While variability-aware static analyses have been developed, there remains a gap between those and state-of-the-art static bug detection tools. In order to collect data on how such tools may perform and to develop real-world benchmarks, we present a way to leverage configuration sampling, off-the-shelf "variability-oblivious" bug detectors, and automatic feature identification techniques to *simulate* a variability-aware analysis. We instantiate our approach using four popular static analysis tools on three highly config-

Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '19), August 26–30, 2019, Tallinn, Estonia. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3338906.3338967

1 INTRODUCTION

Systems developed in C form some of the largest and most important software infrastructure. This software, such as the Linux kernel or the BusyBox embedded toolkit, is used in a broad range of applications, from large-scale datacenters to millions of Internet-of-Things devices. C programmers use *compile-time variability* to enable a single codebase to be customized to this diverse range of settings. They implement software configurations in the Makefile and C preprocessor to decide which part of the source code is built

UCF's Campus Is Beautiful

Incentives in Academia

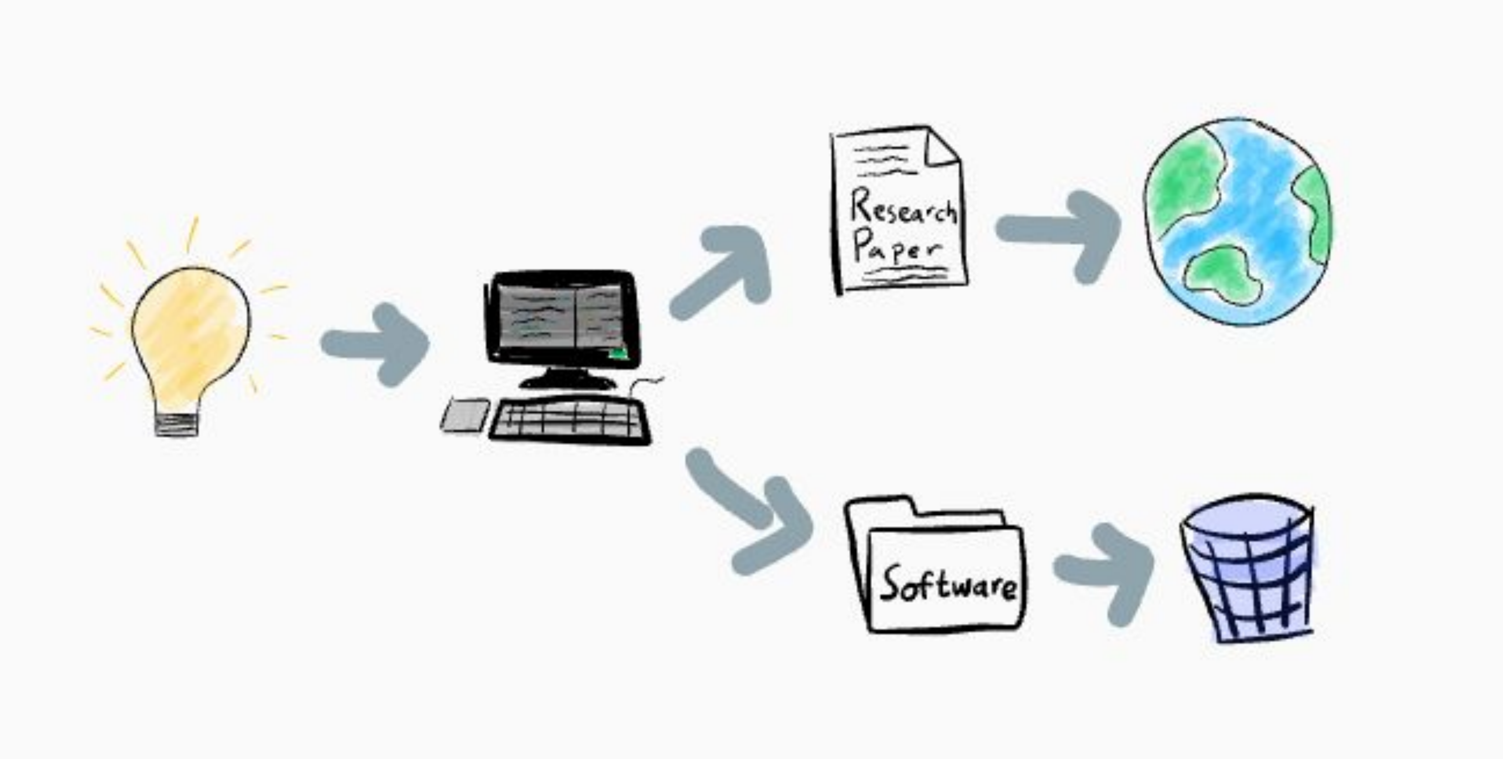
Publish (papers) or perish

Artifacts sometimes gets left behind (about 40% of SE papers release artifacts)

Research is (often) publicly funded

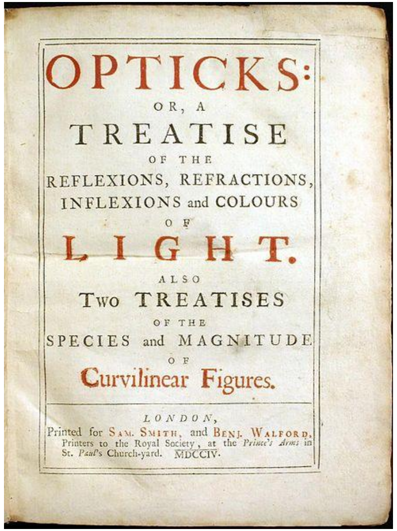
ROSE Festival
Recognizing and Rewarding Open Science in Software Engineering

Oversimplified View of Research



Science Depends on Good Tools

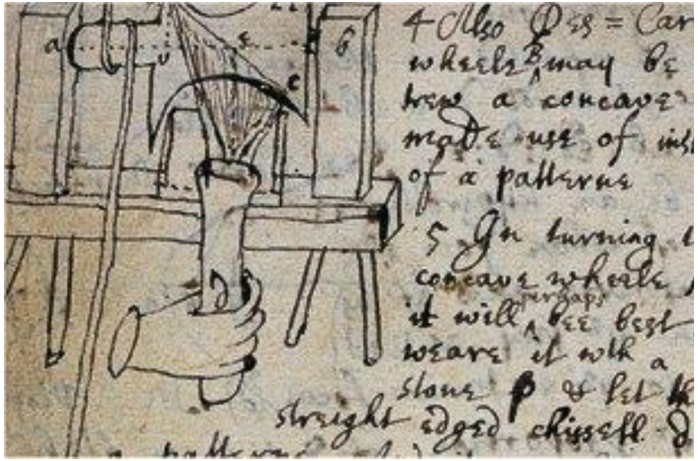
Newton's Opticks



OPTICKS:
OR, A
TREATISE
OF THE
REFLEXIONS, REFRACTIONS,
INFLEXIONS and COLOURS
OF
LIGHT.
ALSO
Two TREATISES
OF THE
SPECIES and MAGNITUDE
OF
Curvilinear Figures.

LONDON,
Printed for SAM. SMITH, and BENJ. WALFORD,
Printers to the Royal Society, at the *Prince's Arms* in
St. *Paul's* Church-yard. MDCCIV.

Newtons' Notes on Lens Grinding



4 Also ... = Car
wheels B may be
~~here~~ a concave
made use of ins
of a patterns

5 In turning t
concave wheels
it will perhaps bee best
weare it with a
stone p & let
straight edged chissell

"If it disagrees with experiment, it's wrong." -Feynman

How to Have Real-World Impact: Five Easy Pieces

by Emery Berger on Oct 29, 2019 | Tags: industrial adoption, real-world impact

1. Scratch an itch

2. Build real systems

3.Embed yourself

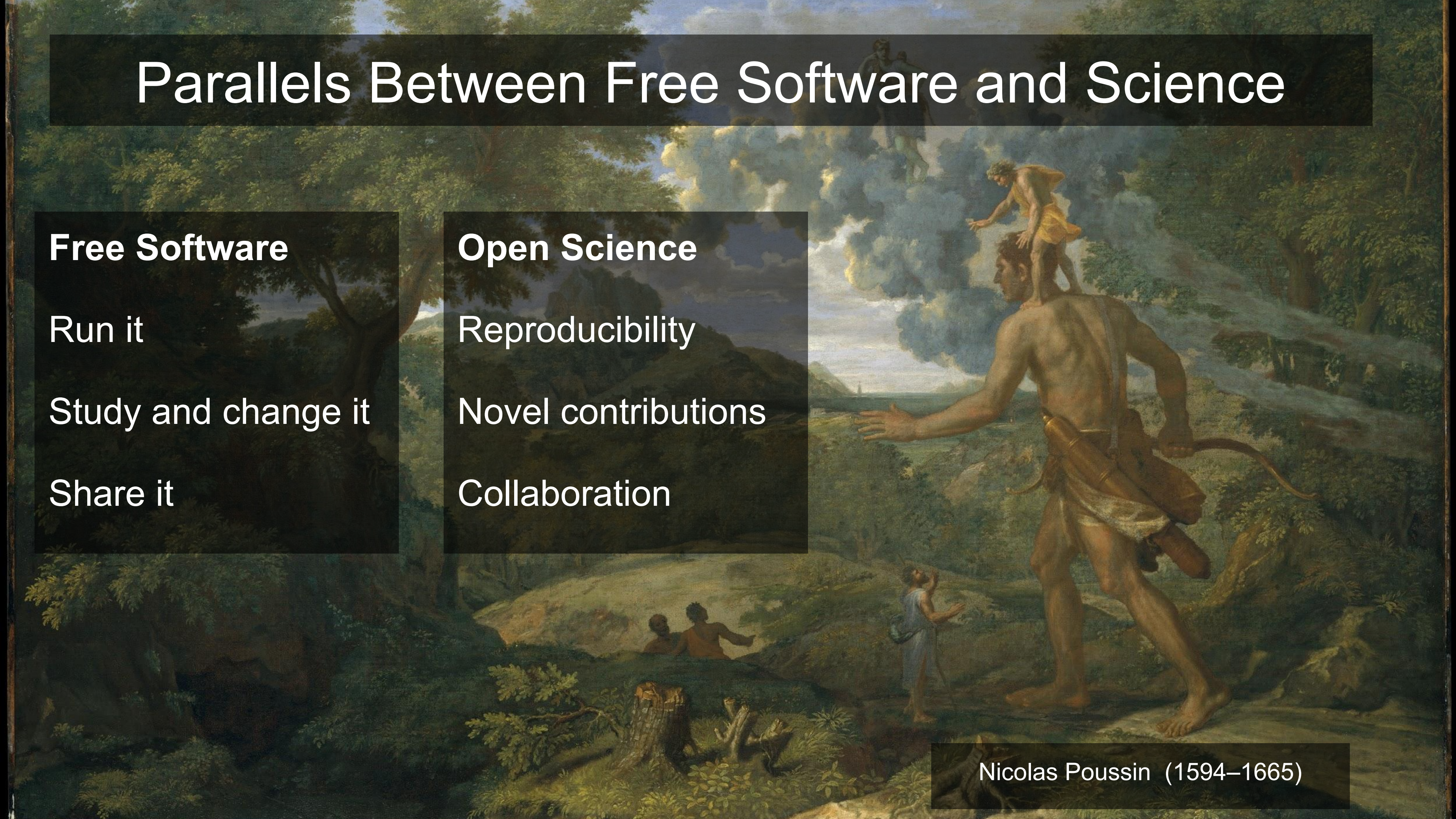
4. Give great talks

5. Go to the mountain

- Emery Berger

Parallels Between Free Software and Science

Free Software	Open Science
Run it	Reproducibility
Study and change it	Novel contributions
Share it	Collaboration



Nicolas Poussin (1594–1665)

Free Software in Research

Run It: My Main Tools for Research

Systems and programming: emacs, git, gcc, python, emacs, bash, coreutils

Research libraries: z3, picosat, sat4j

Writing: emacs, latex

Collaboration: google docs, slack, overleaf, office365 (school requirement)

Our Research Group Collaboration

Now:

- Google docs: notes, outlines, etc
- Slack: chat, video chat
- Skype: video chat (UCF license)
- Office365: calendar (UCF license)

Next to try:

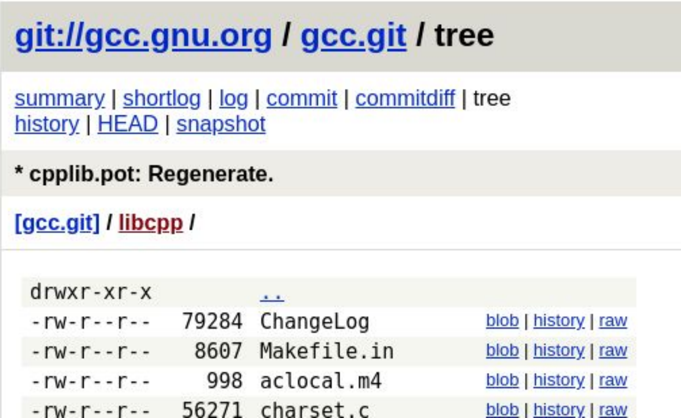
- Nextcloud: notes, shared calendars
- Matrix: chat
- Jitsi, mumble: group chat

Study and Change It: Reading GCC Source Code

Research problem: parse C across all #ifdefs

Why? bug-finding, security, etc

Needed to replicate how part of GCC worked for C



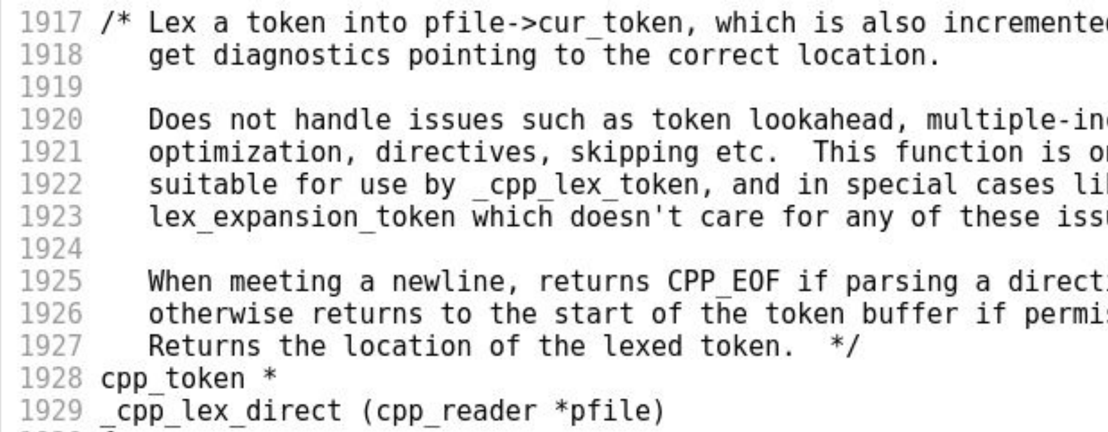
git://gcc.gnu.org / gcc.git / tree

summary | shortlog | log | commit | commitdiff | tree
history | HEAD | snapshot

*** cpplib.pot: Regenerate.**

[gcc.git] / libcpp /

```
drwxr-xr-x          ..
-rw-r--r--   79284 ChangeLog      blob | history | raw
-rw-r--r--    8607 Makefile.in    blob | history | raw
-rw-r--r--     998 aclocal.m4     blob | history | raw
-rw-r--r--   56271 charset.c      blob | history | raw
```



```
/* Lex a token into pfile->cur_token, which is also incremente
   get diagnostics pointing to the correct location.

   Does not handle issues such as token lookahead, multiple-in
   optimization, directives, skipping etc.  This function is o
   suitable for use by _cpp_lex_token, and in special cases li
   lex_expansion_token which doesn't care for any of these iss

   When meeting a newline, returns CPP_EOF if parsing a direct
   otherwise returns to the start of the token buffer if permi
   Returns the location of the lexed token.  */
cpp_token *
_cpp_lex_direct (cpp_reader *pfile)
```

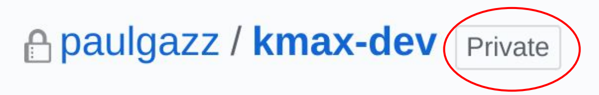
Share It: Community Makes the Research Better

Kept further development in private repository.

My thinking: who would use this anyway?

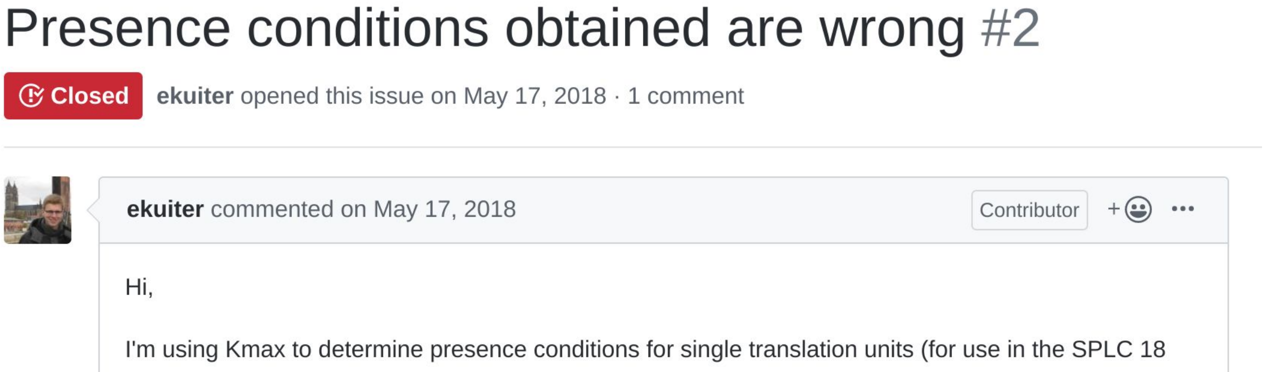
This guy found bugs!

paulgazz / **kmax**



paulgazz / **kmax-dev** Private

Elias Kuiter
ekuiter



Presence conditions obtained are wrong #2

Closed **ekuiter** opened this issue on May 17, 2018 · 1 comment

ekuiter commented on May 17, 2018 Contributor

Hi,

I'm using Kmax to determine presence conditions for single translation units (for use in the SPLC 18

Share It: Others Can Run, Study, and Change It

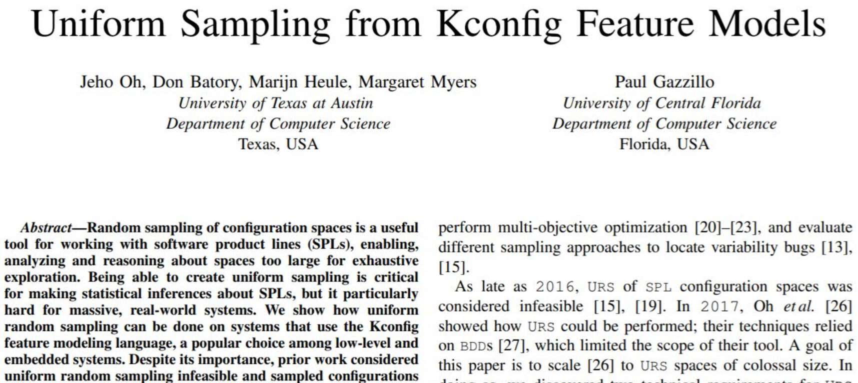
Team at another university working on separate project

Met at conference

Working together on tool improvements, publication

Now co-advising

Jeho Oh, PhD Student
University of Texas at Austin



Uniform Sampling from Kconfig Feature Models

Jeho Oh, Don Batory, Marijn Heule, Margaret Myers
University of Texas at Austin
Department of Computer Science
Texas, USA

Paul Gazzillo
University of Central Florida
Department of Computer Science
Florida, USA

***Abstract*—Random sampling of configuration spaces is a useful tool for working with software product lines (SPLs), enabling, analyzing and reasoning about spaces too large for exhaustive exploration. Being able to create uniform sampling is critical for making statistical inferences about SPLs, but it particularly hard for massive, real-world systems. We show how uniform random sampling can be done on systems that use the Kconfig feature modeling language, a popular choice among low-level and embedded systems. Despite its importance, prior work considered uniform random sampling infeasible and sampled configurations**

perform multi-objective optimization [20]–[23], and evaluate different sampling approaches to locate variability bugs [13], [15].

As late as 2016, URS of SPL configuration spaces was considered infeasible [15], [19]. In 2017, Oh *et al.* [26] showed how URS could be performed; their techniques relied on BDDs [27], which limited the scope of their tool. A goal of this paper is to scale [26] to URS spaces of colossal size. In doing so, we discovered two technical requirements for URS

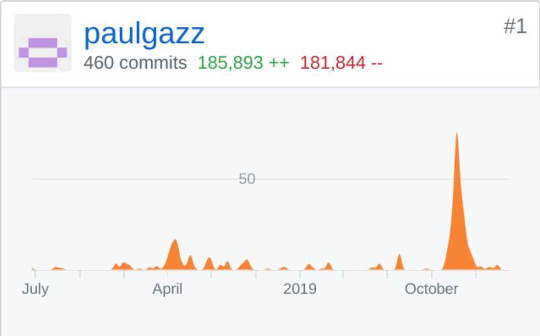
Share It: Others Can Run, Study, and Change It

A researcher interested in the same area

He ad an idea to improve performance and scalability

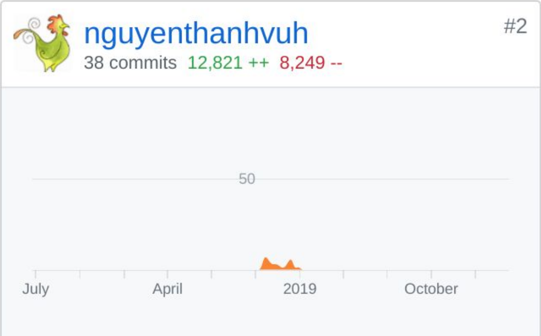
He spent time reengineering the tool

This is enabling our future collaboration



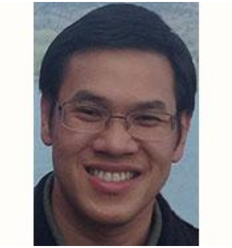
paulgazz #1

460 commits 185,893 ++ 181,844 --



nguyenthanhvuh #2

38 commits 12,821 ++ 8,249 --



Prof. Vu Nguyen
University of Nebraska--Lincoln

Share It: Using Best Practices

Close collaborators willing to put up with ad-hoc prototypes

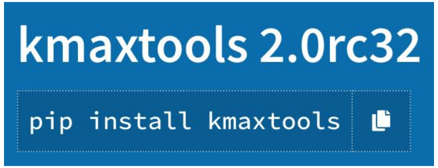
Others should not have to

Good engineering takes time

Not very incentivized in academia

But worth it

python™
Package
Index



kmaxtools 2.0rc32

```
pip install kmaxtools
```

Share It: Enabling Real-World Applications

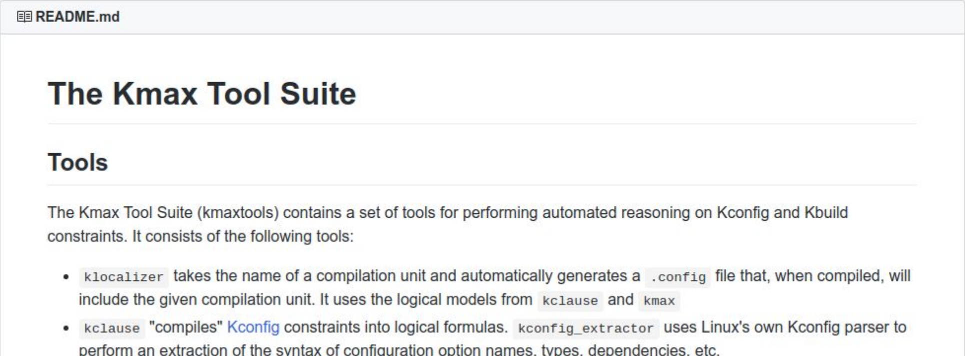
Kernel developer tried the tooling out

Incredibly good feedback from user point of view

Finding areas of need and applications

More collaboration opportunities

Julia Lawall
Inria/LIP6
Researcer and Linux Kernel Dev



README.md

The Kmax Tool Suite

Tools

The Kmax Tool Suite (kmaxtools) contains a set of tools for performing automated reasoning on Kconfig and Kbuild constraints. It consists of the following tools:

- `klocalizer` takes the name of a compilation unit and automatically generates a `.config` file that, when compiled, will include the given compilation unit. It uses the logical models from `kclause` and `kmax`
- `kclause` "compiles" Kconfig constraints into logical formulas. `kconfig_extractor` uses Linux's own Kconfig parser to perform an extraction of the syntax of configuration option names, types, dependencies, etc.

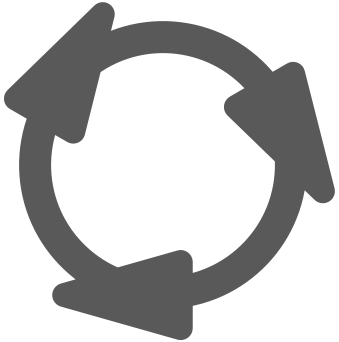
Free Software Enables Science

Study and change it

Novel contributions

Run it

Reproduce results



Share it

Collaborate

Free Software in Education

Teaching Philosophy

A guide to students' own intellectual growth

Students learn self-reliance in the safety of school

Students as empty vessels

Teacher as guide

Challenge: Mass Education

UCF has thousands of CS students

Some classes have hundreds of students

Impossible to be a personal guide to everyone all the time

What helps: consistency, clear expectations, automation, communication, free content

Free Software as a Standard

Instead of one proprietary system, knowledge that generalizes

Endless opportunities exploration

Endless community support

Many students from low income families, first gen college, more economical

Spirit of self learning, do it yourself, community support

Learning GNU/Linux taught me more about OS, PL, SE

Teaching Systems Software

Compilers, Linkers, Loaders, etc

Mostly compilers for undergraduates

Hard topic, project in C

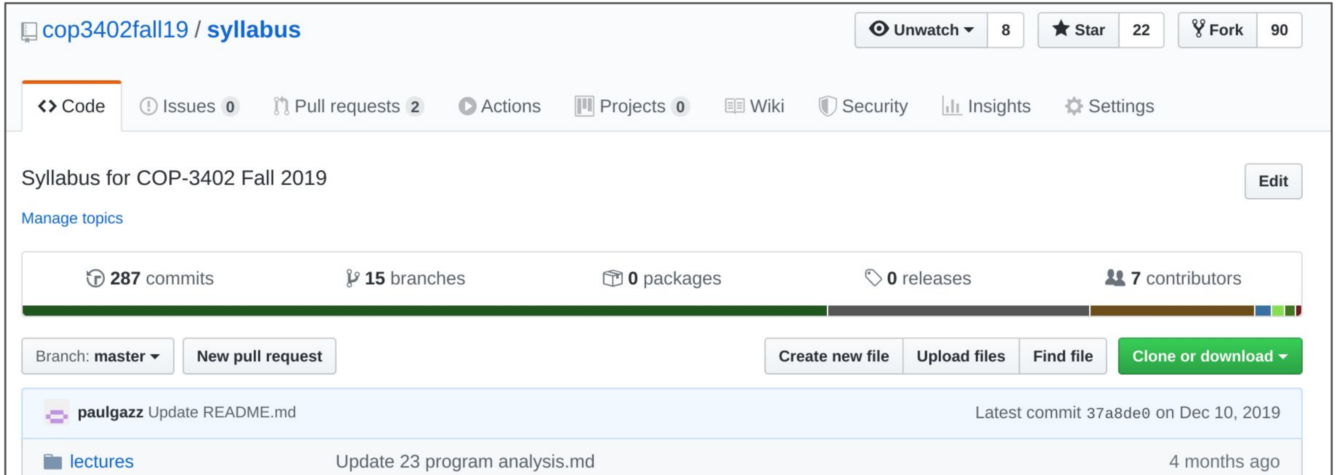
Automation essential for 200 students

Standardized on GNU/Linux, C, Makefiles, and git

All software submitted via git (GitHub classrooms)

Must compile with make on GNU/Linux

Gave a publicly-available test suite for self-grading



Syllabus for COP-3402 Fall 2019

paulgazz Update README.md — Latest commit `37a8de0` on Dec 10, 2019

lectures — Update 23 program analysis.md — 4 months ago

Many Students Lack Command-Line Experience

I assumed students would know or figure out git and bash

Irony: little systems software used in Systems Software course

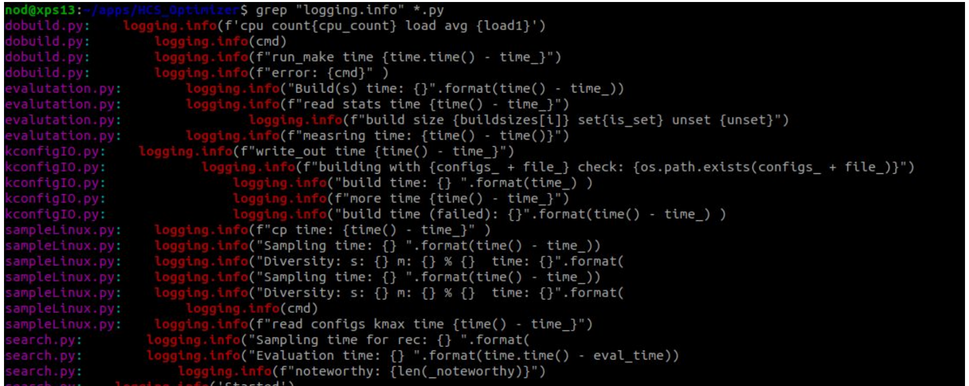
Spent much time in office hours on extra tutoring

Many students had never used the command-line on any OS

Useful for any computing job

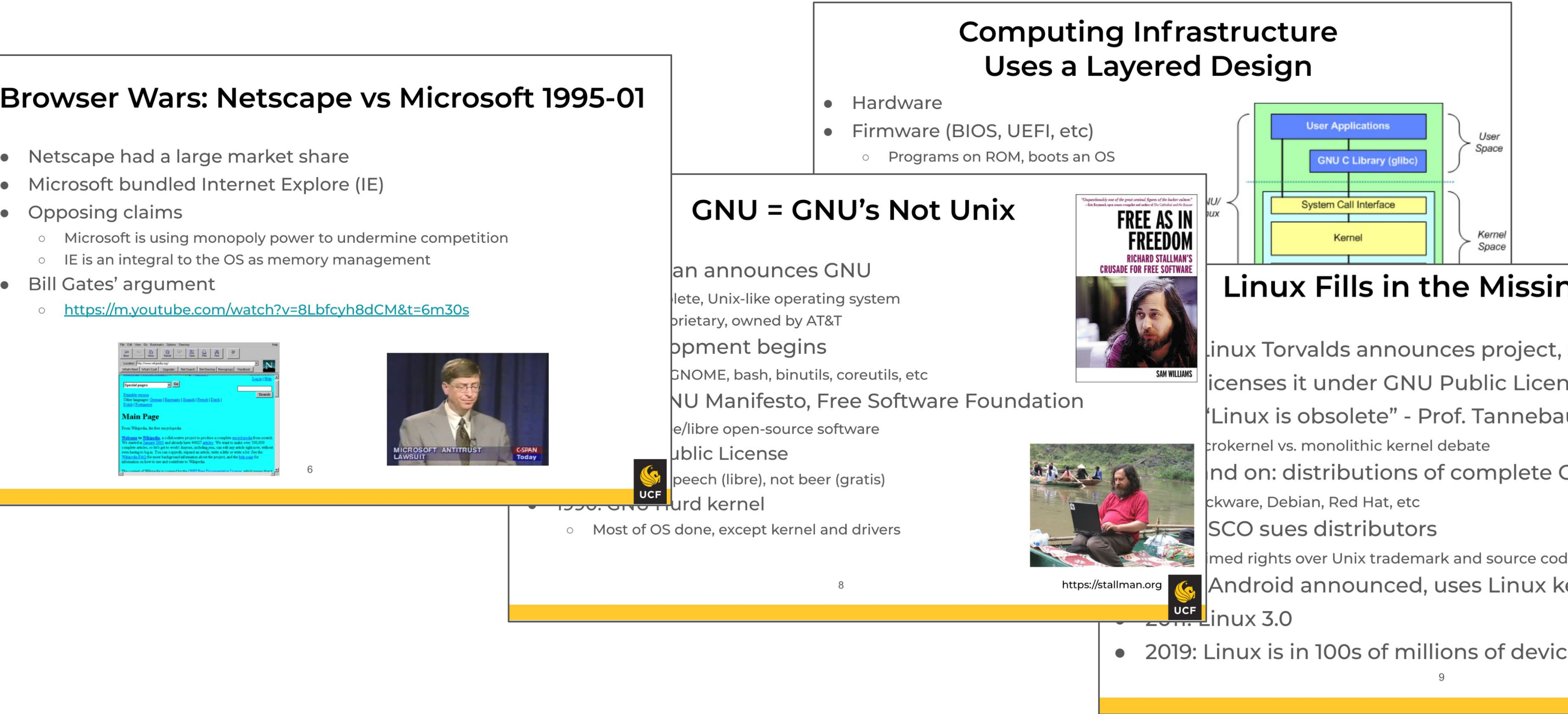
Empowers students

Gives control over device



```
nod@xps13:~/apps/HCS_Optimizer$ grep "logging.info" *.py
dobuild.py:    logging.info(f'cpu count{cpu_count} load avg {load1}')
dobuild.py:        logging.info(cmd)
dobuild.py:        logging.info(f"run_make time {time.time() - time_}")
dobuild.py:        logging.info(f"error: {cmd}" )
evalutation.py:        logging.info("Build(s) time: {}".format(time() - time_))
evalutation.py:        logging.info(f"read stats time {time() - time_}")
evalutation.py:                logging.info(f"build size {buildsizes[i]} set{is_set} unset {unset}")
evalutation.py:        logging.info(f"measring time: {time() - time()}")
kconfigIO.py:    logging.info(f"write_out time {time() - time_}")
kconfigIO.py:            logging.info(f"building with {configs_ + file_} check: {os.path.exists(configs_ + file_)}")
kconfigIO.py:                logging.info("build time: {} ".format(time_) )
kconfigIO.py:                logging.info(f"more time {time() - time_}")
kconfigIO.py:                logging.info("build time (failed): {}".format(time() - time_) )
sampleLinux.py:    logging.info(f"cp time: {time() - time_}" )
sampleLinux.py:    logging.info("Sampling time: {} ".format(time() - time_))
sampleLinux.py:    logging.info("Diversity: s: {} m: {} % {}  time: {}".format(
sampleLinux.py:    logging.info("Sampling time: {} ".format(time() - time_))
sampleLinux.py:    logging.info("Diversity: s: {} m: {} % {}  time: {}".format(
sampleLinux.py:        logging.info(cmd)
sampleLinux.py:    logging.info(f"read configs kmax time {time() - time_}")
search.py:        logging.info("Sampling time for rec: {} ".format(
search.py:        logging.info("Evaluation time: {} ".format(time.time() - eval_time))
search.py:            logging.info(f"noteworthy: {len(_noteworthy)}")
search.py:    logging.info('Started')
```

Incorporated More OS Basics and History



Browser Wars: Netscape vs Microsoft 1995-01

- Netscape had a large market share
- Microsoft bundled Internet Explore (IE)
- Opposing claims
  - Microsoft is using monopoly power to undermine competition
  - IE is an integral to the OS as memory management
- Bill Gates' argument
  - https://m.youtube.com/watch?v=8Lbfcyh8dCM&t=6m30s

Computing Infrastructure Uses a Layered Design

- Hardware
- Firmware (BIOS, UEFI, etc)
  - Programs on ROM, boots an OS

GNU = GNU's Not Unix

an announces GNU

lete, Unix-like operating system

prietary, owned by AT&T

opment begins

GNOME, bash, binutils, coreutils, etc

NU Manifesto, Free Software Foundation

e/libre open-source software

ublic License

peech (libre), not beer (gratis)

- 1990: GNU Hurd kernel
  - Most of OS done, except kernel and drivers

https://stallman.org

Linux Fills in the Missin

inux Torvalds announces project,

icenses it under GNU Public Licen

'Linux is obsolete" - Prof. Tannebau

crokernel vs. monolithic kernel debate

nd on: distributions of complete C

ckware, Debian, Red Hat, etc

SCO sues distributors

imed rights over Unix trademark and source cod

Android announced, uses Linux ke

- 2011: Linux 3.0
- 2019: Linux is in 100s of millions of devic

Lots of Windows and Mac Users: Virtualization

Introduced a VM as an easier way to fulfill course requirements

More opportunity to teach OS basics and command-line

Some students end up installed GNU/Linux on bare metal

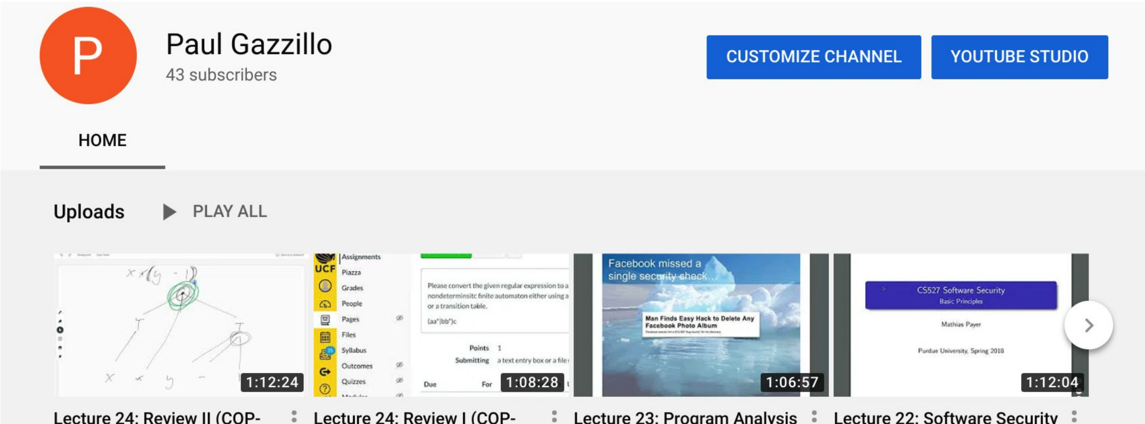
Producing Free and Open Course Content

Ease to screen capture lectures (uncomfortable at first)

Important resource for self-study

Less anxiety about missing content during class

Some (all?) students need repetition and review



Lecture 24: Review II (COP-

Lecture 24: Review I (COP-

Lecture 23: Program Analysis

Lecture 22: Software Security

Lots of Existing Sources for Self-Study

Education without paying for college

VE RI TAS
HARVARD
Extension School

Anonymous Feedback from Students

"github process is awesome for helping us understand how it works and using it for personal projects and in our careers later on"

"it has literally made it easier for me to get a research position after mentioning that I know git"

"Very, very meticulously organized - github was utilized beautifully - and everything laid out clearly."

"have a much clearer process on how to set up your environment for doing programming assignments. How to setup the virtual machine, how to use test cases, etc…"

"Lecture a bit slower with some more layman explanations."

"Having more in-class functions or examples for the programming assignments."

"A lot of students come in knowing a lot of the material already presented so the coding assignments come relatively easy for them."

Praise and Criticism

"I liked how clearly he teaches, and he obviously loves what he teaches. His projects are well organized, and he is so personable and truly cares about how his students are doing and how they are doing in the course. He takes their success as a reflection on his teaching and organization"

"I think the course is perfect as is."

"he's boring to listen to"

"The assignments were insanely difficult, so hard to understand. Please make them easier."

"finding a way to make the material more interesting"

Free Software Enables Education

Free software as a standard for computer science education

Students learn self-reliance, control over computing devices

Students learn the community support and sharing ethic

Enables mass education

Enables low-income, first-generation students (Libre *and* Gratis)

I need more free software services and content

- Suggestions for free-software-based learning platforms (git, bash, etc)?
- Suggestions for easy-to-use content creation services?

A Student's Perspective

About me

Undergrad at UCF, pursuing a Bachelors in Computer Science

Getting into Free Software

Free Software is defacto

Using free software in research

Becoming aware; not taking freedom for granted

Picture Credits

Christiaan Colen - https://www.flickr.com/photos/christiaancolen/
https://www.allaboutcircuits.com/news/how-compaqs-clone-computers-skirted-ibms-patents-and-gave-rise-to-eisa/
https://www.rexxla.org/rexxlang/mfc/trl.html
https://www.lifewire.com/operating-systems-why-unix-2200178
https://gnu.org
https://www.debian.org/logos/
https://www.apache.org/foundation/press/kit/
https://www.linuxfoundation.org/
https://commons.wikimedia.org/wiki/File:Keramik-Neues-Museum-01.JPG
https://www.nps.gov/wica/learn/management/seasonal-employment-at-wind-cave-national-park.htm
https://commons.wikimedia.org/wiki/File:Checkmark.svg
https://sites.google.com/site/dapforteachers/home/undegraduate-tips-cognitive-development
https://en.wikipedia.org/wiki/File:Vagrant_Logo.svg
https://en.wikipedia.org/wiki/File:Green_Dragon_Book_(front).jpg

What Are Your Questions?