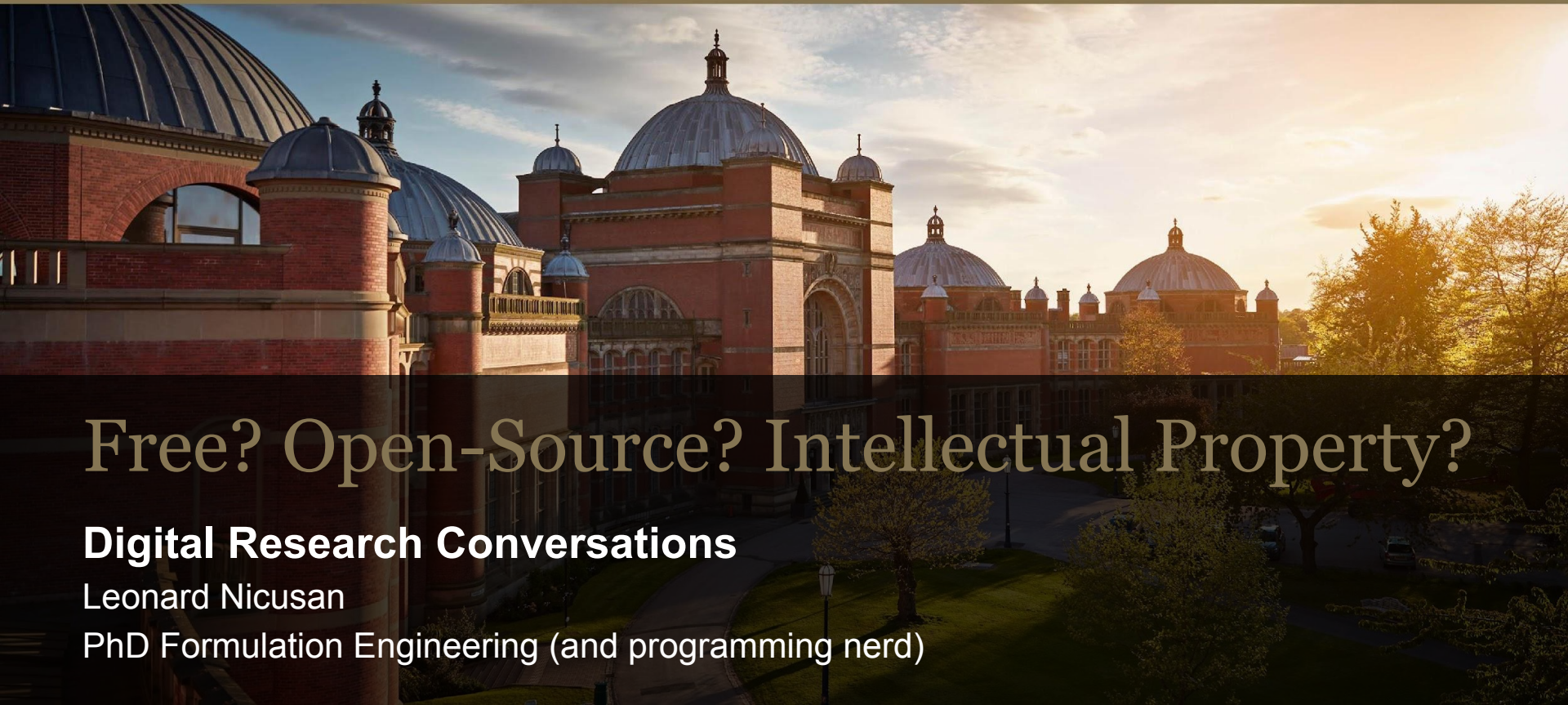

UNIVERSITY OF BIRMINGHAM

Free? Open-Source? Intellectual Property?

Digital Research Conversations

Leonard Nicusan
PhD Formulation Engineering (and programming nerd)

Free

- As in “beer”: no **price** required
- As in “speech”: have **the liberty** to run, copy, modify, distribute code

Nowadays used interchangeably (FOSS: Free and Open Source Software)

1960 - 1980s

- Code started to migrate from assembly to higher-level languages (Fortran, C, Forth, BCPL, Cobol)
- Architectures and OSs were “Terra Incognita” - almost every computer had different instructions, hardware interaction, etc.
- MIT group of “hackers” started a software sharing movement

- The Unix OS was very expensive in the 1970s - along with compilers and everything else.
- The GNU project started writing an open-source foundation for software

1980s - 2000s

- The GNU Compiler Collection (GCC) formed the basis of most new software coming in later years
- Linux, Android, etc.
- New compilers (LLVM) -> healthy competition, more research, faster software
- LLVM produced a new generation of programming languages: Julia, Rust, Go, Swift
- Now we can’t imagine a world without these FOSS tools!

How Different Would the World be without FOSS?

- Need to pay for a C compiler for a specific operating system
  - Need to pay for a Python interpreter for Windows
  - Pay more to get it on Mac too
- Pay for a code editor / IDE
- Pay for NumPy and each library
- Each company would have a different, incompatible computer architecture
- No Android, iOS, Linux servers
- Computers programs would be immensely slower

(Almost) All of our coding stack is FOSS

- The x86_64 CPU architecture - and ARM
- The entire interface to hardware (the C standard library, on any OS)
- The C/C++ compilers building Python, MATLAB, Windows
- The Python, Julia, R interpreters
- The libraries we use: NumPy, SciPy, OpenFOAM, LAMMPS, VTK
- The standards we use: MPI, OpenMP, OpenCL, ~CUDA

Other side of the coin: story from granular mechanics

- LAMMPS is the most popular (and often fastest) molecular dynamics program, developed since 1995 - FOSS-licensed
- New additions are almost always submitted and included in the main codebase
- Around 2010, it was forked, and an Austrian PhD added models to simulate granular materials - named it LIGGGHTS
- Development diverged, LAMMPS had billions in funding from entire US national labs, LIGGGHTS was developed by a few scientists
- The original LIGGGHTS contributor formed a company, stopped developing it, renamed it to Aspherix and asked for £20,000 a year to use it

Other side of the coin

- “I spent 4 years developing a library for everyone to use, why should a company include it in their commercial suite and profit off it?”

- But we’re also building everything on FOSS - how can we balance this?

Also: linking

If we “import numpy as np”, we’re not bound by NumPy’s license - we’re merely using it, not copying it into our codebase.

But in compiled languages, using a library is different:

- Dynamic linking (.dll, .dylib, .so) - simply “hooking” into a pre-compiled library when starting the program. Can easily break.
- Static linking (.a -> .exe) - actually including all code needed into our standalone program.

Licensing?

Licenses

- Many software licenses available today:
  - MIT, Apache, GNU v2 / v3, BSD
  - Some only need attribution: MIT, Apache
  - Some require derivatives to remain open-source: GNU

- Are creative-commons licenses applicable to code? Why, why not?
- As researchers writing code, what should we use and why?