The big idea
Compare and contrast

• How lawyers interpret legal texts
• How computers interpret software
More specifically

1. Use concepts from the philosophy of law — speech acts, interpretation, etc. — to give a rigorous account of how software works.

2. Use that account to illuminate questions in legal doctrine: e.g., how should judges interpret smart contracts?

3. Use that account to illuminate questions in legal theory: e.g., is the ideal judge a computer?
Software speech acts
Legal speech acts

• “Be it hereby enacted that …” is a *speech act*

• It has the *illocutionary force* of changing the law (and possibly also of commanding subjects to comply and officials to act.)

• Other legal speech acts: contracts, wills, ToS

• They have their own illocutionary forces
Software speech acts

- `print(2+2)` is also a kind of speech act
- When uttered to a Python interpreter, it causes the computer to display 4
- We could talk about this mechanistically, deny that the computer understands anything, and deny that communication is taking place
- But this overlooks the ways in which `print(2+2)` is linguistically meaningful
Law thinks that software is speech

- *E.g., Bernstein v. DoJ*: software can be First-Amendment-covered speech
- *E.g., Computer Associates v. Altai*: software can be copyrightable
- Neither of these cases is intelligible if software is inherently only a functional artifact
- For better or for worse, we program computers with words that have meaning to humans
Who is the interpreter?

- Legal texts are addressed to *people*: citizens, counterparties, guests, and especially judges
  - They mean what they mean to people
- Programs are addressed to *computers*: they consist of a series of commands to execute
  - Do they mean (only) what they cause computers to do?
Types of meaning

• Program meaning: what a program causes a computer to do

• Programmer meaning: what a bug-free version of the program would do

• Incidental meaning: what else a program’s text conveys to other programmers who read it

• User meaning: what a program communicates to a user
from itertools import repeat
for feet in [3,3,2,2,3]:
    print " ".join("DA–DA–DUM"
        for dummy in [None]
        for foot in repeat(metric", feet))

Types of meaning

- **Program meaning**: (syntax error)
- **Programmer meaning**: print “DA-DA-DUM…”
- **Incidental meaning**: the source is a limerick
- **User meaning**: the output is a limerick
Applications, e.g.
Unauthorized access

• Many programs implicitly communicate to users the scope of permission to use them
  
  • *United States v. Morris*: what is the “intended function” of Sendmail?
  
  • What would a reasonable user understand as the programmer meaning of this program?
Is the ideal of a judge another programmer who helps the legislature test and debug its code?

Or is the ideal of a judge a reliable computer who correctly executes the legislature’s code?

Program meaning shows that nearly discretionless interpretation is possible

But not even the most rigid versions of textualism go that far
Legal drafting and software development

• Can effective software development techniques be pulled back into law?
• Textual aspects: type-safe languages, modular designs
• Toolchains: editors, version control, etc.
• Program analysis and debugging
Questions
Questions for you

• What should I call it?

• “The Jurisprudence of Software” is boring

• What should I read?

• Philosophy of language, law, and CS

• Tech-law theory: “code is law,” algorithmic decision-making, computable law
Questions for me