The Structure and Legal Interpretation of Computer Programs

James Grimmelmann

Yale ISP Ideas Lunch
February 1, 2018

Three vignettes

Video poker


GAME KING
MULTI-GAME
MULTI-GAME
Earn & Win!
WEDNESDAYS IN JUNE
INSERT TICKETS
GAME KING
MULTI-GAME
EARN
1000 POINTS
$20 FREE SLOT PLAY
WIN
ROYAL FLUSH NO DEUCES
4 DEUCES
1800
1200
2100
1400
2400
1600
2700
1800
3000
2000

Website scraping

```
User-agent: *
Disallow: /private/
```

```
User-agent: *
Dsallow: /private/
```

The DAO

"The terms of The DAO Creation are set forth in the smart contract code existing on the Ethereum blockchain at `0xbb9bc244d798123fde783fcc1c72d3bb8c189413`. Nothing in this explanation of terms or in any other document or communication may modify or add any additional obligations or guarantees beyond those set forth in The DAO's code."

Software with legal effects

- Software can convey permission to use it
- Obvious analogies: statutes, licenses, etc.
- These have their own legal interpretive rules
- *What are the interpretive rules for software?*

Code is law?

Who is the interpreter?

- Legal texts are addressed to *people*: citizens, counterparties, guests, and especially judges
  - So we care about their meaning to people
- But software is addressed to *computers*: it consists of a series of commands to execute
  - I.e., the functional effects of a program derive from its meaning to a computer

Proposition:
functional meaning ≠ legal meaning

- Interpretive strategy: *strict functional meaning*
  - Let the computer interpret the code for you
  - What the code allows is what the law allows
  - Legal meaning = functional meaning
- This is obviously insufficient as a theory
  - Computers malfunction; software is buggy

Proposition:
functional meaning ~ legal meaning

- Not anything goes!
  - Video poker is not video backgammon
  - The DAO is incoherent unless there is some determinate content to "the smart contract code existing on the Ethereum blockchain at `0xbb9bc244d798123fde783fcc1c72d3bb8c18941`"
- Legal meaning is grounded in functional meaning

Literal functional meaning

Specification and semantics

- What does `2**2` mean in a programming language?
- Three answers:
  - Use a program: a *reference implementation* whose behavior is by stipulation treated as correct
  - Use natural language: a *specification* that defines the behavior of a correct implementation
  - Use mathematics: a *formal semantics* that identifies programs with abstract entities

Two questions

- Where do specifications and semantics come from?
  - Some people got together to define them
- What language are we running?
  - “Python” 2.7 is different from “Python” 3.6
- These questions can be answered only by reference to a community of programmers and users

Fixing functional meaning

- A technical community agrees on a process for deriving a functional meaning from texts
- Developers implement that process on different computers, with different tools, etc.
- Most of the time, running a program on most implementations yields the same result
- A program's *literal functional meaning* is what a standardized implementation would do with it

Ordinary functional meaning

The price we pay

- Running a program produces *a* result—but not necessarily the right result
- This is characteristic of literal functional meaning as an interpretive strategy: specifying in advance the resolution of all possible ambiguities is a recipe for predictably getting many of them wrong
- The concept of a "bug" assumes a distinction between actual and intended program behavior

Ordinary meaning

- The ordinary legal meaning of a text is the meaning a reasonable audience would give it
  - If a program's audience consists of its users …
    - … they expect that a program contains bugs
- A program's *ordinary functional meaning* is what reasonable people in the position of its users would expect it to do, if it were free of bugs

Three vignettes, redux

Video poker

- Reasonable video poker players understand:

  (1) They are expected and allowed to play as skillfully as they can to improve their payouts

  (2) Quitting and returning to a game after a hand has been played probably wasn't intended to change the payout multiplier

- The case looks hard because of the conflict between (1) and (2). But ordinary functional meaning controls, and the payout trick looks like a bug, not a feature.

Robots.txt

- Reasonable web scrapers understand:

  (1) `Disallow` was probably intended; `Dsallow` isn't a valid keyword in the robots exclusion standard

  (2) The standard is designed for bots to process automatically, not (primarily) for humans to read

- Literal functional meaning is appropriate because of (2). Without specific knowledge (maybe even with it), bot operators don't need to respect `Disallow`.

The DAO

- Reasonable blockchain investors understand:

  (1) The DAO contract was buggy

  (2) The DAO's legal instruments purported to make the contract judicially unreviewable

  (3) The DAO depends on Ethereum

- Whether (2) successfully selects literal functional meaning is a question of offline contract law. But (3) makes even literal functional meaning ambiguous!

Questions?