Specification languages for computable laws versus basic legal principles: Tension Table

ALEIX SOLÉ SÁNCHEZ, ERIC SANCHO ADAMSON, MARTA SORIA HEREDIA, JOOST J. JOOSTEN

Transposing a regulation written in natural language into an computable algorithm raises a set of problems with regards to legal principles, with few previous studies to help us conceptualize such problems.

These problems notably affect the legal principles of legal certainty, accountability and contestability, and affects any computable algorithm meant to enforce a legal provision. By changing the language in which regulations are to be written from natural to semi-formal or to formal languages some legal principles may fare better while others may fare worse. In this paper, we describe the problems that affect the legal principles as a function of the language used to draft legislation.

TENSION TABLE:

Computable laws:

Language, software paradigm and legal principles

More accurate and exact but less undertandable for the general public

Specification Language	Programming paradigm	Legal Principles: Legal Certainty	Legal Principles: Accountability	Legal Principles: Contestability
Natural Language	Not Formally Verified	Decisions will probably not be consistent with the established legal framework. The text will be accessible and comprehensible to the public and authorities.	Automated decision won't be reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, which will be unreliable.
Technical Language	Not Formally Verified	Decisions will likely not be consistent with the established legal framework. The text will be less comprehensible to public and authorities.	Automated decision will be barely reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, which will be mostly unreliable.
Formal Language	Not Formally Verified	Decisions will probably be consistent with the established legal framework. The text will only be accessible to experts.	Automated decision will be quite reliable and explainability will be difficult: the software is not comprehensible to the public, challenging the principle of transparency.	Right to contest turns almost impossible since authorities can't explain software decisions, yet they will probably be working according to the law
Formal Language	Formally Verified	Decisions will be consistent with established legal framework. The text will only be accessible to experts	Automated decision will be reliable and explainability will be difficult, but it will be guaranteed that the software is the exact reproduction of its specification	Right to contest will be difficult since authorities can't explain software decisions, yet those are working according to the law

DESIDERATA

We want to ensure that:

- Laws, written by legislators and intended to be implemented by software are executed as the law states in writing.
- Different software products replicating the same law have equivalent behavior.
- Software does not have internal bugs that lead to unexpected behavior and untraceable arbitrary outputs.
- Transparency in the application (correspondence between the legal text, the output of the program and access to the reasons for the sanctions)
- The design of the software is not left to the interpretation that the programmer happens to have of the law.

STANDARD 1

A law that intends to be computerized, should be written both in natural language and an isomorphic version in formal language (the exact specification for the programmer: Meaning the sole computational, logico-mathematical interpretation). With this, the possibility of different applications of the same law decreases. Also it gives a solid basis for formal verification of software.

STANDARD 2

If the program is formally verified through Coq, Agda, Isabelle or any other proof assistant, we can ensure the program behaves exactly as instructed and avoid bugs altogether. Thus, (a) the program is an exact representation of the law and (b) there won't be unexpected behavior coming from internal software design flaws or bugs whatsoever.

Proyecto PID2020-115774RB-I00: PERSONALIZACION DE LOS SERVICIOS PUBLICOS, SESGOS E INTELIGENCIA ARTIFICIAL: HACIA LA CONSOLIDACION DE LOS DERECHOS DIGITALES EN LAS ADMINISTRACIONES PUBLICAS, financiado por:

Proyecto 2019PDPD-107667GB-I00: Worlds and Truth Values: Challenges to Formal Semantics (MUNVAL) financiado por:

SOFTWARE DE FALLO CERO. Innovación para el sector del transporte. Proyecto RTC-2017-6740-7 financiado por MCIN/AEI/ 10.13039/501100011033 y por FEDER Una manera de hacer Europa