Normative Systems

The meeting point between Jurisprudence and Information Technology?

Luigi Logripppo

Université du Québec en Outaouais
We shall see that Jurisprudence and IT have some commonalities of concepts and issues. They may be slowly pulling together.
Normative Systems

- The term *normative system* is being used in the literature with different definitions.
- A much cited book by Alchourron and Bulygin bears this title, and claims application to social sciences only.
  - Loosely defines norms as statements that relate cases to solutions.
General importance of normative system

- Jones and Sergot wrote in 1990:
  - “at the appropriate level of abstraction, law, computer systems, and many other kinds of organisational structure may be viewed as instances of normative systems
  - “we use the term to refer to any set of interacting agents whose behaviour may be usefully regarded as governed by norms
  - “norms prescribe how the agents ought to behave and specify how they are permitted to behave and what their rights are
Two corrections, perhaps?

- Jones and Sergot wrote in 1990:
  - Normative systems:
    - “we use the term to refer to any set of interacting agents whose behaviour may be usefully regarded as governed by norms
    - “norms prescribe how the agents ought to behave and specify how they are permitted to behave and what their rights are

Set of norms?

Excessive reliance on deontic concepts?
The behavior of computer systems is of increasing legal relevance
- Security
- E-commerce, E-contracts
- IT governance

Ideally, it should be possible for law and regulations to be directly implemented in computer policies,
- these should automatically change as the law changes

This will force the law to be more precise, at least in certain areas
More forces

- Computer networks will be like social systems, with their own norms (policies)
Deontic Logic

- Deontic logic is a modal logic of obligation and permission.
- Based on the observation that the De Morgan laws apply to these concepts:

  not obligatory not P = P is permitted
  not permitted not P = P is obligatory

  Def.: forbidden P = P is not permitted
  Def.: X has a right = State has obligation to X
Deontic logic in normative systems

- It is often assumed that norms are expressed in deontic logic
  - See previous statement by Jones and Sergot
- BUT…
The study of elementary normative forms

- As biologists can learn much by studying elementary life forms, we can learn much by studying elementary normative forms
  - Firewalls
  - Hammurabi code
If any one steals cattle or sheep, or an ass, or a pig or a goat, if it belong to a god or to the court, the thief shall pay thirty fold; if they belonged to a freed man of the king he shall pay tenfold; if the thief has nothing with which to pay he shall be put to death.

This code is written strictly in Event-Condition-Action (ECA) style.
If any one steals cattle or sheep, or an ass, or a pig or a goat, if it belong to a god or to the court, the thief shall pay thirty fold.

A question is whose action this is: The judge’s? The thief’s?
A rule in a Linux *router* to drop packets having any ("all") protocol, that come from node "nuisance.com" and go anywhere

Also trigger-condition-action
Thus, the most elementary normative systems are simply made of *rules*:

- Given such a behaviour, and such a situation, such is the resulting action
- **Norms can exist without the notion of obligation**
Enter deontic logic with Moses’ law

8. Thou shalt not steal

We have gained abstraction (this covers a dozen articles from Hammurabi code)

But lost specificity
- What happens if one steals?
- How to enforce?

This is a requirement to be implemented
Rules and Requirements

- We have identified two normative styles
  - Rule style
  - Requirement style
- This is consistent with the distinction between *requirement* and *implementation* in Software Engineering
- There are of course other styles
Consistency

Are there incompatible norms for the same situations?
Cases...

- Inconsistency between requirements
- Inconsistency between rules and requirements
- Inconsistency between rules

  - The second case is often solved by giving the priority to the requirement
Inconsistency in law

- Inconsistency is one of the major issues for lawyers and judges
- It is often dealt with by showing that apparently incompatible rules deal with different cases
  - Although its origin may be an error...
Inconsistency in sets IT policies: it’s an error

- It can be an implementation error
  - In the spec or in the implementation
    - The method to avoid these has been to rigorously check specs and implementations
      - Software Engineering, Formal methods

- Or it can be a Feature Interaction problem
  - Methods have been ad-hoc
    - We’ll get back to this
What does inconsistency mean in norms?

- In classical logic, a single inconsistency invalidates the whole system, anything becomes derivable
  - \((A \text{ and not } A) = \text{False}\) and anything can be derived from False
    - Which btw means that an inconsistent system is complete!

- However in practice inconsistencies in sets of rules are dealt with by trying to ‘isolate and fix’ the inconsistent rules
  - Logics to justify this exist
Detection of inconsistency

- Theorem provers
- Satisfaction algorithms
- Algorithms are NP-complete (or worse) but a lot can be done if few variables are involved
  - In many practical cases we have seen, the problem was treatable
Completeness

Are all cases covered?
Examples of incompleteness

- A set of rules can be incomplete if some aspects of the requirements are not covered
- E.g. Canadian charter of rights protects the right to life
  - However Canada has no law about abortion
    - Is Canada’s law incomplete wrt requirements?
- Requirements can be *implicit*
  - E.g. does the Hammurabi code cover *all* cases of theft?
    - This question makes sense even though Hammurabi did not know Moses’ law, because he covers several cases of theft
    - Similarly, in common law requirements are induced from cases, i.e. rules
Incompleteness in IT

- IT has standard ways to deal with incompleteness:
  - The default solution
    - For every program, set of rules, etc. we know what will happen in the case where none of the specified conditions is true
  - However this might not correspond to the specification or the *intention* of the user
Incompleteness in law

- The lawyer’s reasoning wrt incompleteness is totally different
- There will be attempts to derive rules
  - From requirements
  - From similar rules
    - Which means inducing the requirements from similar rules
- Only if this fails, then the IT approach is taken
  - Situation not covered by law, nothing to do
Some common research topics

- Defeasible logic and meta-rules
- Feature interactions
- Ontologies
Defeasible Logic

Applies to both consistency and completeness
Priority among norms in firewalls

- In firewalls, the rules are scanned top-down
  - The first applicable norm is applied and all following ones are ignored
- So is solved the problem of several applicable rules (policy interaction)
- This can’t be justified easily:
  - The order of axioms is not important in logic
  - The order of norms is not important in law
    - although later norms can abrogate earlier ones
Defeasible Logic

- A non-monotonic logic proposed by Donald Nute. In defeasible logic, there are three types of propositions:
  - **Hard rules**
    - specify that a fact is *always* a consequence of another;
      - *All birds have wings*
  - **Defeasible rules**
    - specify that a fact is *typically* consequence of another;
      - *All birds fly*
  - **Defeaters**
    - specify *exceptions* to defeasible rules.
      - *Ostriches don’t fly*

*Before applying a defeasible rule, check for defeaters!*
Defeasible logic by priorities

- R1: Professor(X) => Tenured(X)
- R2: Visiting(X) => Non-Tenured(X)

Is a Visiting Professor tenured?

Which one is the defeater?

- One common way to answer is to give priorities to rules, most probably here R2>R1
Firewall example

- In a firewall, the first applicable rule defeats all following ones
  - R1 > R2 > R3 …
- So all rules are defeasible by a previous one
  - Legal theory and IT have independently discovered the same problem, and solved it in similar ways
Meta-rules

- A normative system can also include meta-rules, to decide which rule(s) should be defeated in case of inconsistency
  - Priority rule can be considered a meta-rule
  - In XACML: access control language
  - It is possible to specify *combining algorithms*
    - Deny override
    - Permit override
    - Etc.
Meta-rules in law

- lex specialis derogat legi generali
- lex posterior derogat legi priori
- lex superior derogat legi inferiori

A law can be overridden by
- a more special one,
- a posterior one,
- or a superior one
Another application: *Closure* norm

- A closure norm is a norm that makes a system complete, e.g.
  - In Cisco firewalls, all packets for which there is no rule are rejected
    - Similar to a legal system where all behaviours that are not explicitly allowed are forbidden
  - In Linux firewalls, the rule is opposite
    - A ‘more liberal’ legal system
      - *Nulla poena sine lege*
Closure norm as defeasible norm

- In defeasible logic, a closure norm is a norm that exists in the system, but can be defeated by any other norm (G. Governatori)
  - It applies only if no other norm applies
- If defeasible logic is not used, it is a norm that applies when the negation of the premises of all other norms holds
  - Difficulty in constructing this negation, it changes as the set of norms changes
Feature Interactions
1. A calls B
2. B forwards to C
3. A gets connected to C

OCS: Originating Call Screening
CF: Call Forward

A has C in OCS list
B has CF to C
C

OCS goal is violated.
Feature Interaction

- Multi-user feature interaction, i.e. resolution of conflicts between agents resulting from conflicting goals, is precisely the subject of law!
- This suggests that in order to solve FIs in IT systems we’ll have to develop the equivalent of generally recognized laws.
Wired-in solution

- The law, even common sense, knows perfectly how to deal with this, why don’t we?
  - If Alice lends a book to Bob, and Bob wants to lend it to Carla, of course he must check first with Alice!
  - If Alice delegates a task to Bob, and Bob wants to delegate it to Carla, of course he must check with Alice.

- In computing we are haven’t really developed a culture yet...

- Very slowly, we’ll have to develop *principles*:
  - *Ownership, delegation*...
  - Who owns a connection, when can it be delegated...
Trusted third party (TTP)

- In ‘real life’, arbitrators, judges, notaries are essential to prevent and solve feature interaction
- And so they must be in computer communications
  - TTPs to apply FI resolution policies
- In some implicit way, connecting parties will have to recognize the jurisdiction of a TTP
OCS-CF Interaction with TTP

- Parties will keep TTP informed of their intentions, asking for approvals
- CF will be ‘disapproved’ by TTP
TTP Present and Future

- At present, TTPs are not much used, except for authentication
- Users tend to trust the other party they are dealing with, which often has conflicting interests
- Application areas:
  - Web services
  - E-commerce, E-contracts in particular
Ontologies (in CS sense…)

- In legal systems, just as in IT policies, there is yet another type of norm, the *definitional* norm.
  - Wikipedia: *An ontology is typically a hierarchical data structure containing all the relevant entities and their relationships and rules within that domain (e.g. a domain ontology).*
Ontologies as generators

- We can have a norm saying that theft is punished in a certain way, then definitions saying that certain behaviours are theft
  - Another way to bridge betw. Moses and Hammurabi...
- In a company, we can program the switchboard with the company’s organizational tree
  - Then we can have a rule such as:
    - *When an employee is absent, calls for him go to the supervisors*
    - This can generate dozens of rules
- Enterprise security systems are built on enterprise ontologies
  - E.g. Role-based Access Control (RBAC)
Conclusions

- Many concepts are common between Jurisprudence and IT
- Forces exist that will draw the two areas closer in the long run
- Conceptual consolidation is desirable and will surely occur
- Much is to be learned from such consolidation, in both fields