On Legal Contracts, Imperative and Declarative Smart Contracts, and Blockchain Systems

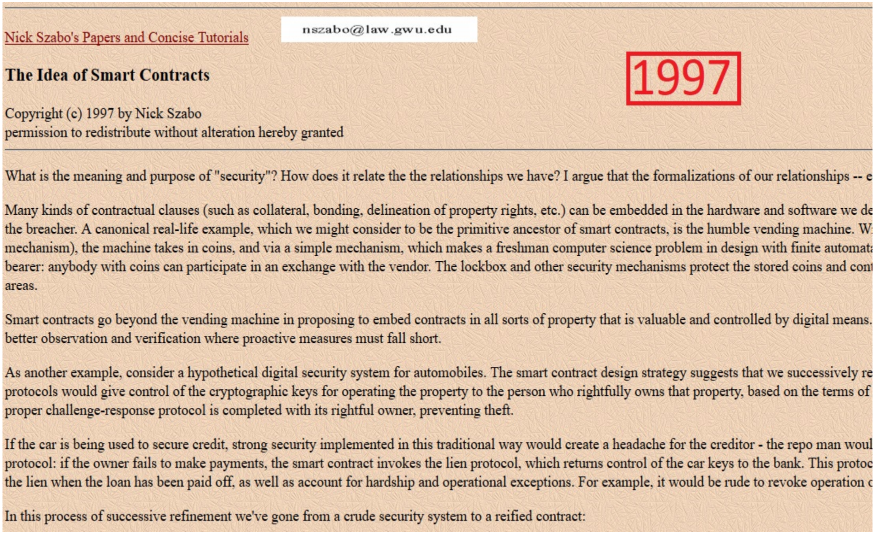
Guido Governatori[1], **Régis Riveret**[1], Xiwei Xu[1]
Florian Idelberger[2], Giovanni Sartor[2],
Zoran Milosevic[3]

[1]Data61, CSIRO

[2]European University Institute

[3]Deontik

July 4, 2018



Nick Szabo's Papers and Concise Tutorials

nszabo@law.gwu.edu

The Idea of Smart Contracts

1997

Copyright (c) 1997 by Nick Szabo
permission to redistribute without alteration hereby granted

What is the meaning and purpose of "security"? How does it relate the the relationships we have? I argue that the formalizations of our relationships -- e

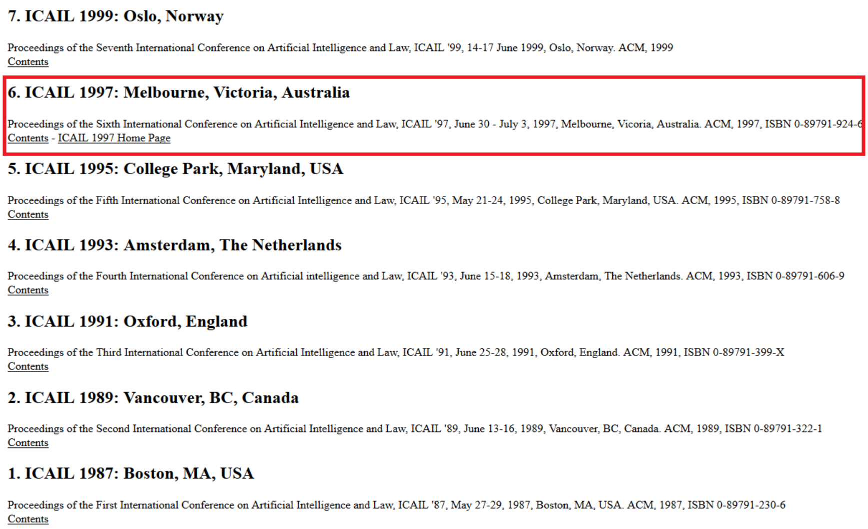
Many kinds of contractual clauses (such as collateral, bonding, delineation of property rights, etc.) can be embedded in the hardware and software we de
the breacher. A canonical real-life example, which we might consider to be the primitive ancestor of smart contracts, is the humble vending machine. W
mechanism), the machine takes in coins, and via a simple mechanism, which makes a freshman computer science problem in design with finite automat
bearer: anybody with coins can participate in an exchange with the vendor. The lockbox and other security mechanisms protect the stored coins and con
areas.

Smart contracts go beyond the vending machine in proposing to embed contracts in all sorts of property that is valuable and controlled by digital means.
better observation and verification where proactive measures must fall short.

As another example, consider a hypothetical digital security system for automobiles. The smart contract design strategy suggests that we successively re
protocols would give control of the cryptographic keys for operating the property to the person who rightfully owns that property, based on the terms of
proper challenge-response protocol is completed with its rightful owner, preventing theft.

If the car is being used to secure credit, strong security implemented in this traditional way would create a headache for the creditor - the repo man woul
protocol: if the owner fails to make payments, the smart contract invokes the lien protocol, which returns control of the car keys to the bank. This protoc
the lien when the loan has been paid off, as well as account for hardship and operational exceptions. For example, it would be rude to revoke operation c

In this process of successive refinement we've gone from a crude security system to a reified contract:



7. ICAIL 1999: Oslo, Norway

Proceedings of the Seventh International Conference on Artificial Intelligence and Law, ICAIL '99, 14-17 June 1999, Oslo, Norway. ACM, 1999
Contents

6. ICAIL 1997: Melbourne, Victoria, Australia

Proceedings of the Sixth International Conference on Artificial Intelligence and Law, ICAIL '97, June 30 - July 3, 1997, Melbourne, Vicoria, Australia. ACM, 1997, ISBN 0-89791-924-6
Contents - ICAIL 1997 Home Page

5. ICAIL 1995: College Park, Maryland, USA

Proceedings of the Fifth International Conference on Artificial Intelligence and Law, ICAIL '95, May 21-24, 1995, College Park, Maryland, USA. ACM, 1995, ISBN 0-89791-758-8
Contents

4. ICAIL 1993: Amsterdam, The Netherlands

Proceedings of the Fourth International Conference on Artificial intelligence and Law, ICAIL '93, June 15-18, 1993, Amsterdam, The Netherlands. ACM, 1993, ISBN 0-89791-606-9
Contents

3. ICAIL 1991: Oxford, England

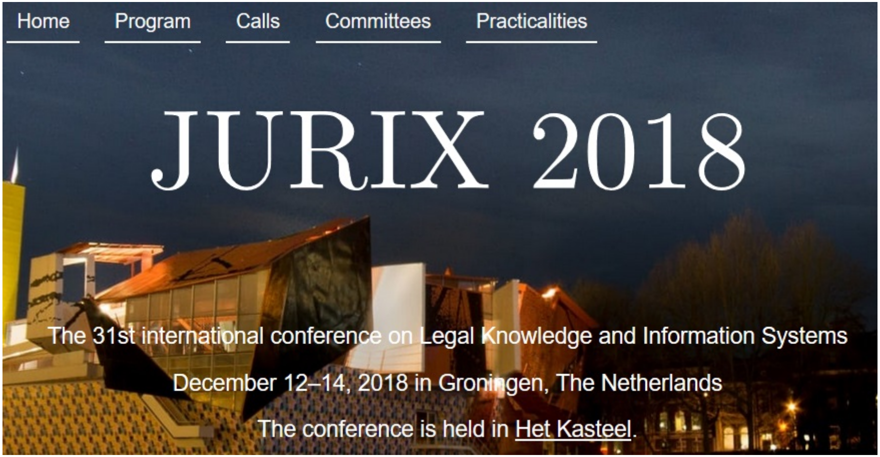
Proceedings of the Third International Conference on Artificial Intelligence and Law, ICAIL '91, June 25-28, 1991, Oxford, England. ACM, 1991, ISBN 0-89791-399-X
Contents

2. ICAIL 1989: Vancouver, BC, Canada

Proceedings of the Second International Conference on Artificial Intelligence and Law, ICAIL '89, June 13-16, 1989, Vancouver, BC, Canada. ACM, 1989, ISBN 0-89791-322-1
Contents

1. ICAIL 1987: Boston, MA, USA

Proceedings of the First International Conference on Artificial Intelligence and Law, ICAIL '87, May 27-29, 1987, Boston, MA, USA. ACM, 1987, ISBN 0-89791-230-6
Contents

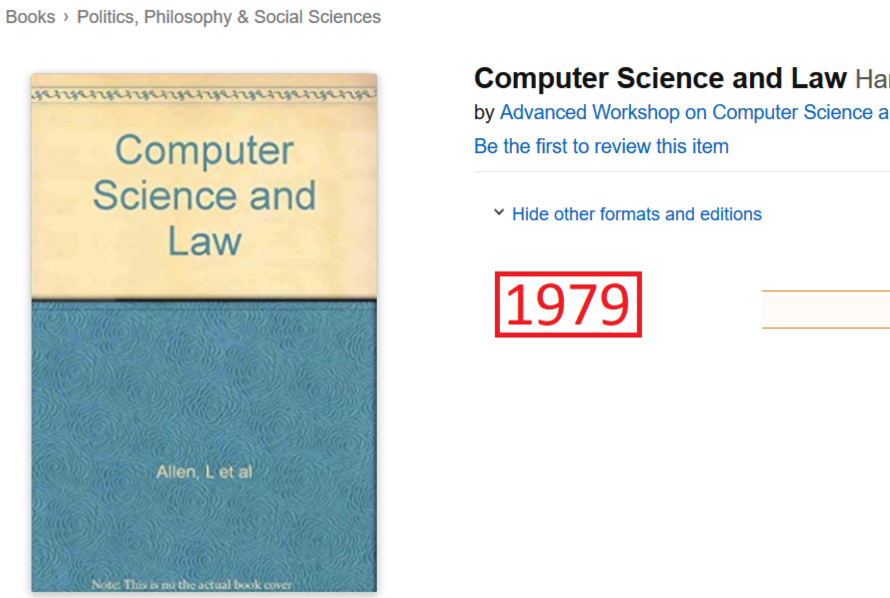


JURIX 2018

The 31st international conference on Legal Knowledge and Information Systems

December 12–14, 2018 in Groningen, The Netherlands

The conference is held in Het Kasteel.


Books › Politics, Philosophy & Social Sciences

Computer Science and Law Ha

by Advanced Workshop on Computer Science a

Be the first to review this item

Hide other formats and editions

1979

1 July 1646 – 14 November 1716

Overview of the project - in short

- Old problem: how to capture normative statements (laws, contracts, etc.) with computer code, and how to make that legally binding?

Overview of the project - in short

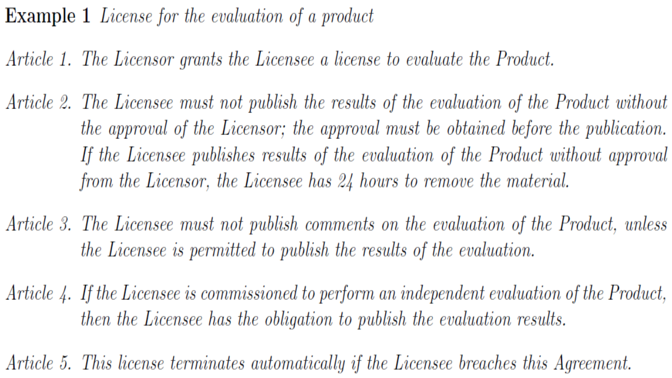
- Old problem: how to capture normative statements (laws, contracts, etc.) with computer code, and how to make that legally binding?
- New problem: how to turn legal contracts into smart contracts (and vice versa)?

Overview of the project - in short

- Old problem: how to capture normative statements (laws, contracts, etc.) with computer code, and how to make that legally binding?
- New problem: how to turn legal contracts into smart contracts (and vice versa)?

- For both problems, two approaches
  - imperative approach, e.g. Solidity;
  - declarative approach, e.g. Logic Programming.

Example



Example 1 *License for the evaluation of a product*

Article 1. The Licensor grants the Licensee a license to evaluate the Product.

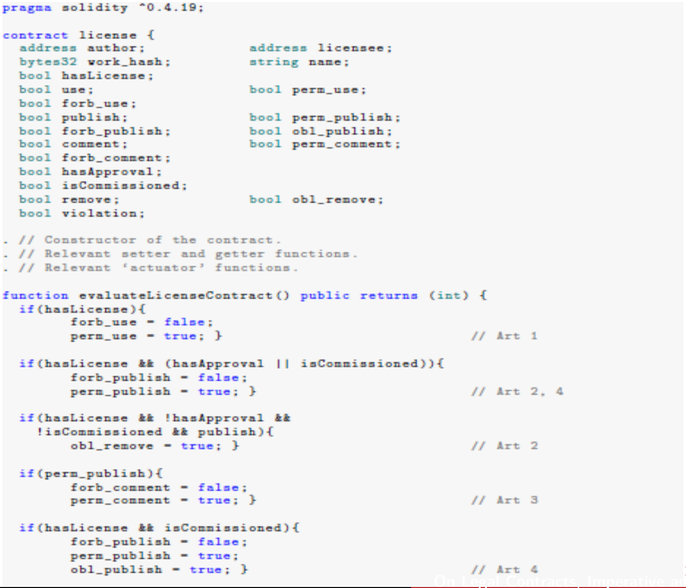
Article 2. The Licensee must not publish the results of the evaluation of the Product without the approval of the Licensor; the approval must be obtained before the publication. If the Licensee publishes results of the evaluation of the Product without approval from the Licensor, the Licensee has 24 hours to remove the material.

Article 3. The Licensee must not publish comments on the evaluation of the Product, unless the Licensee is permitted to publish the results of the evaluation.

Article 4. If the Licensee is commissioned to perform an independent evaluation of the Product, then the Licensee has the obligation to publish the evaluation results.

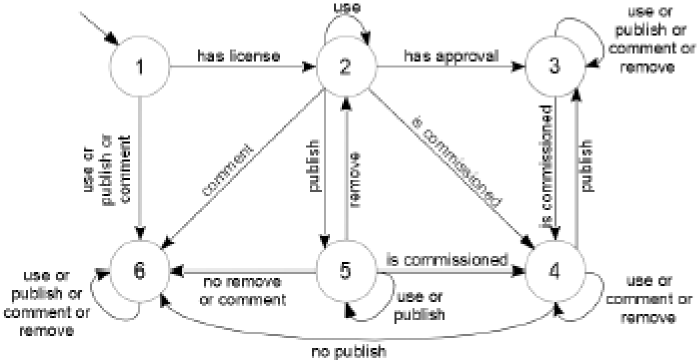
Article 5. This license terminates automatically if the Licensee breaches this Agreement.

Example



```
pragma solidity ^0.4.19;

contract license {
  address author;            address licensee;
  bytes32 work_hash;         string name;
  bool hasLicense;
  bool use;                  bool perm_use;
  bool forb_use;
  bool publish;              bool perm_publish;
  bool forb_publish;         bool obl_publish;
  bool comment;              bool perm_comment;
  bool forb_comment;
  bool hasApproval;
  bool isCommissioned;
  bool remove;               bool obl_remove;
  bool violation;

. // Constructor of the contract.
. // Relevant setter and getter functions.
. // Relevant 'actuator' functions.

function evaluateLicenseContract() public returns (int) {
  if(hasLicense){
        forb_use = false;
        perm_use = true; }                              // Art 1

  if(hasLicense && (hasApproval || isCommissioned)){
        forb_publish = false;
        perm_publish = true; }                          // Art 2, 4

  if(hasLicense && !hasApproval &&
    !isCommissioned && publish){
        obl_remove = true; }                            // Art 2

  if(perm_publish){
        forb_comment = false;
        perm_comment = true; }                          // Art 3

  if(hasLicense && isCommissioned){
        forb_publish = false;
        perm_publish = true;
        obl_publish = true; }                           // Art 4
```

Example


use
use or publish or comment or remove
1
has license
2
has approval
3
use or publish or comment
comment
publish
remove
is commissioned
is commissioned
publish
use or publish or comment or remove
6
no remove or comment
5
is commissioned
4
use or comment or remove
use or publish
no publish

Example

```
Art1.0: => [Forb_licensee] use
Art1.1: hasLicense => [Perm_licensee] use
Art2.1: => [Forb_licensee] publish [Compensated] [Obl_licensee] remove
Art2.2: hasLicense, hasApproval => [Perm_licensee] publish
Art3.1: => [Forb_licensee] comment
Art3.2: [Perm_licensee] publish => [Perm_licensee] comment
Art4.0: hasLicense, isCommissioned => [Obl_licensee] publish
Art5.1: violation => [Forb_licensee] use
Art5.2: violation => [Forb_licensee] publish

% Superiority relation
Art1.1 > Art1.0,
Art2.2 > Art2.1, Art3.2 > Art3.1,
Art5.1 > Art1.1, Art5.2 > Art4.0.
```

Does blockchain actually change anything?

Elements of legal contracts

- Agreement
- Consideration
- Competence and capacity
- Legal object and purpose

Does blockchain actually change anything?

Legal interpretation

- Content
- Interpretation
- Open-textured terms
- Implied terms

Does blockchain actually change anything?

Lifecyle

- Negotiation and formation
- Contract storage and notarizing
- Performance/execution
- Modification
- Dispute resolution
- Termination

Lesson learnt so far...

- Solidity may not be convenient to write a logic engine for smart contracts.

Reference

- Guido Governatori, Florian Idelberger, Zoran Milosevic, Regis Riveret, Giovanni Sartor, Xiwei Xu. **On Legal Contracts, Imperative and Declarative Smart Contracts, and Blockchain Systems**, Artificial Intelligence & Law (2018).