Logical English as an Executable Computer Language

Robert Kowalski
Imperial College London

Together with

Fariba Sadri
Miguel Calejo
Jacinto Davila
Vesko Karadotchev

Conclusion

Cc: B Kowalski computers.

IT Link Inc. Critique
Planner No. 5R-155
12xY – 600

APR 1993

ROMEIKE GROUP

OUT WITH THE PROGRAMMERS... ...IN WITH THE LAWYERS?

Who would we expect to be the best person to write a computer program? A trained programmer, most of us would reply. Perhaps an unusually inventive end-user, given access to tools such as code generators, and fourth generation languages. But a lawyer ...

Well, according to Bob Kowalski, a professor of computational logic at Imperial College London, lawyers are just the people. Kowalski, you may recall, devised the principles of logic programming, on which the PROLOG

language. No matter that it may be a rather specialised and stilted form of English, it is still much more accessible to the average human than any mathematical formalism.

An example, according to Kowalski, is the emergency notice found in London Underground trains:

> *Press the alarm signal to alert the driver. The driver will stop immediately if any part of the train is in a station. If not, the train will proceed*

Logical English (LE)

- modelled on the language of law.
- understandable by English speakers
  without special training.
- suitable for general computation, i.e.
  programming, databases and knowledge representation.
- prototypes in Prolog

The Language of Law	The majority of computer languages
declarative	imperative
describes the legal consequences of actions and events	assignment statements
logical conditionals and logical all/every/exists	if-then-else, iteration and loops
avoids ambiguity, but embraces vagueness (open texture)	unambiguous by design
passively judges the behaviour of external agents	actively support the goals of external agents

Smart contracts
actively support the goals of legal agreements.

Examples from

- The British Nationality Act
  logic programs.

- A simplified loan agreement
  deterministic finite automata.

- Rock, Paper, Scissors
  (Lessons and insights from a cryptocurrency lab)
  imperatives.

- ISDA Master Agreement
  (International Swaps and Derivatives Association)
  obligations.

THE BRITISH NATIONALITY ACT AS A LOGIC PROGRAM

The formalization of legislation and the development of computer systems to assist with legal problem solving provide a rich domain for developing and testing artificial-intelligence technology.

M. J. SERGOT, F. SADRI, R. A. KOWALSKI, F. KRIWACZEK, P. HAMMOND, and H. T. CORY

British Nationality Act
Acquisition at Birth

English

1.-(1) A person born in the United Kingdom after commencement shall be a British citizen if at the time of the birth his father or mother is

(a) a British citizen; or

(b) settled in the United Kingdom.

Logical English (basic level)

British Nationality Act
Acquisition at Birth

English

1.-(1) A person born in the United Kingdom after commencement
shall be a British citizen
if at the time of the birth
his father or mother is
(a) a British citizen; or
(b) settled in the United Kingdom.

Logical English (basic level)

A person acquires british citizenship by subsection 1.1 at a time
if *the person is born in the uk at the time*
and *the time is after commencement*
and *another person is the father of the person*
or *another person is the mother of the person*
and *the other person is a british citizen at the time*
or *the other person is settled in the uk at the time.*

“a”, “an” or “another” indicates the first occurrence of a logical variable
“the” indicates a later occurrence of the same logical variable in the same sentence.
No “all”, “every” or “some”.

No “all”, “every” or “some” in Basic Logical English

All payments that are specified in some transaction are obligatory.

needs to be rewritten.

For example:

A payment is obligatory
if the payment is specified in a transaction.

Or:

It is an obligation that a payment is made
if the payment is specified in a transaction.

Examples from

- The British Nationality Act
  logic programs.

- A simplified loan agreement
  deterministic finite automata.

- Rock, Paper, Scissors
  (Lessons and insights from a cryptocurrency lab)
  imperatives.

- ISDA Master Agreement
  (International Swaps and Derivatives Association)
  obligations.

OFFICE OF FINANCIAL RESEARCH

WORKING PAPER

15-04 | March 26, 2015
Revised March 27, 2017

Contract as Automaton: The Computational Representation of Financial Agreements

Mark D. Flood
Office of Financial Research
mark.flood@ofr.treasury.gov

Oliver R. Goodenough
Office of Financial Research and Vermont Law School
oliver.goodenough@ofr.treasury.gov
ogoodenough@vermontlaw.edu

Agreement

This loan agreement dated June 1, 2014, by and between Lender Bank Co. ("Lender") and Borrower Corp. (Borrower), will set out the terms under which Lender will extend credit in the principal amount of $1,000 to Borrower with an un-compounded interest rate of 5% per annum, included in the specified payment structure.

1. The Loan

At the request of Borrower, to be given on June 1, 2014, Lender will advance $1,000 to Borrower no later than June 2, 2014. If Borrower does not make such a request, this agreement will terminate.

2. Repayment

Subject to the other terms of this agreement, Borrower will repay the loan in the following payments:

(a) Payment 1, due June 1, 2015, in the amount of $550, representing a payment of $500 as half of the principal and interest in the amount of $50.

(b) Payment 2, due June 1, 2016, in the amount of $525, representing a payment of $500 as the remaining half of the principal and interest in the amount of $25.

3. Representations and Warranties

The Borrower represents and warrants, at the execution of this agreement, at the request for the advance of funds and at all times any repayment amount shall be outstanding, the Borrower's assets shall exceed its liabilities as determined under an application of the FASB rules of accounting.

4. Covenants:

The Borrower covenants that at the execution of this agreement, at the request for the advance of funds and at all times any repayment amount shall be outstanding it will make timely payment of all state and federal taxes as and when due.

5. Events of Default

The Borrower will be in default under this agreement upon the occurrence of any of the following events or conditions, provided they shall remain uncured within a period of two days after notice is given to Borrower by Lender of their occurrence (such an uncured event an "Event of Default"):

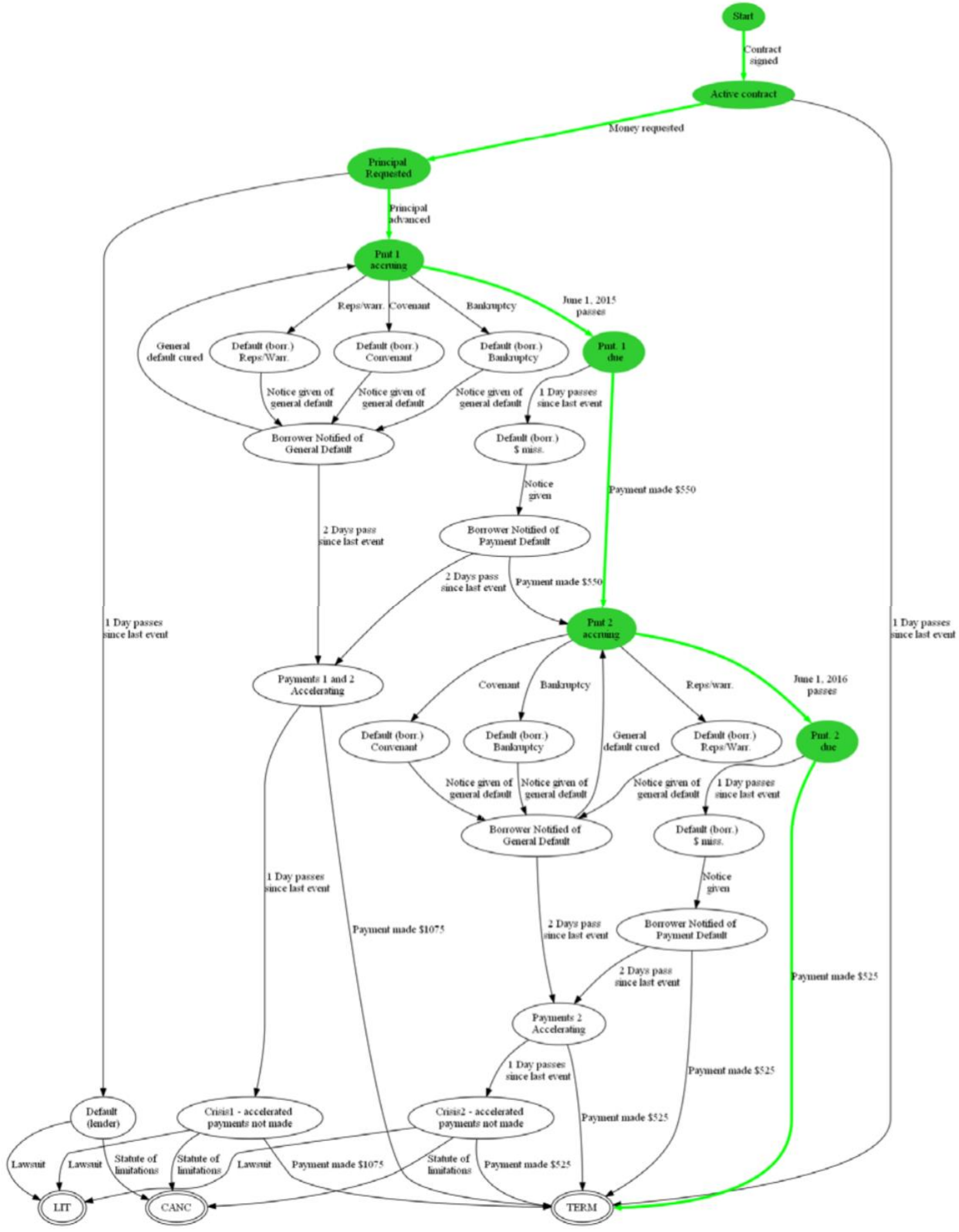
(a) Borrower shall fail to make timely payment of any amount due to Lender hereunder;

(b) Any of the representation or warranties of Borrower under this agreement shall prove untrue;

(c) Borrower shall fail to perform any of its covenants under this agreement;

(d) Borrower shall file for bankruptcy or insolvency under any applicable federal or state law.

A default will be cured by the Borrower (i) remedying the potential event of default and (ii) giving effective notice of such remedy to the Lender. In the event of multiple events of default,


Start
Contract signed
Active contract
Money requested
Principal Requested
Principal advanced
Pmt 1 accruing
Reps/warr.
Covenant
Bankruptcy
June 1, 2015 passes
General default cured
Default (borr.) Reps/Warr.
Default (borr.) Convenant
Default (borr.) Bankruptcy
Pmt. 1 due
Notice given of general default
1 Day passes since last event
Borrower Notified of General Default
Default (borr.) $ miss.
Notice given
Payment made $550
2 Days pass since last event
Borrower Notified of Payment Default
1 Day passes since last event
Pmt 2 accruing
Payments 1 and 2 Accelerating
June 1, 2016 passes
Default (borr.) Convenant
Default (borr.) Bankruptcy
Default (borr.) Reps/Warr.
Pmt. 2 due
Borrower Notified of General Default
Default (borr.) $ miss.
Payment made $1075
Payment made $525
Payments 2 Accelerating
Default (lender)
Crisis1 - accelerated payments not made
Crisis2 - accelerated payments not made
Lawsuit
Statute of limitations
LIT
CANC
TERM

Agreement

This loan agreement dated June 1, 2014, by and between Lender Bank Co. ("Lender") and Borrower Corp. (Borrower), will set out the terms under which Lender will extend credit in the principal amount of $1,000 to Borrower with an un-compounded interest rate of 5% per annum, included in the specified payment structure.

<u>1. The Loan:</u>

At the request of Borrower, to be given on June 1, 2014, Lender will advance $1000 to Borrower no later than June 2, 2014. If Borrower does not make such a request, this agreement will terminate.

Agreement

This loan agreement dated June 1, 2014, by and between Lender Bank Co. ("Lender") and Borrower Corp. (Borrower), will set out the terms under which Lender will extend credit in the principal amount of $1,000 to Borrower with an un-compounded interest rate of 5% per annum, included in the specified payment structure.

1. The Loan:

At the request of Borrower, to be given on June 1, 2014, Lender will advance $1000 to Borrower no later than June 2, 2014. If Borrower does not make such a request, this agreement will terminate.

It is an obligation that the lender advances $1000 to the borrower at a time
and the time is before the end of 2014/6/2
if the borrower has requested $1000 on 2014/6/1.

Or

The lender becomes liable to litigation at a time
if it is the end of 2014/6/2 at the time
and the borrower has requested $1000 on 2014/6/1
and it is not the case that
the lender has advanced $1000 at another time and
the other time is before the end of 2014/6/2.

1. The Loan:

At the request of Borrower, to be given on June 1, 2014, Lender will advance $1000 to Borrower no later than June 2, 2014. If Borrower does not make such a request, this agreement will terminate.

The contract becomes terminated
when it is the end of 2014/6/1
and it is not the case that
the borrower has requested the loan.

"conclusion when conditions"
means
"conclusion at a time
if conditions at the time"

The contract becomes terminated at a time
if it is the end of 2014/6/1 at the time
and it is not the case that
the borrower has requested the loan at the time.

translated into Prolog/LPS
(Logical Production System)

```
end_of_day(2014/6/1)
initiates terminated
if not requested(borrower, 1000, 2014/6/1).
```

LPS implemented in SWI Prolog running online in SWISH

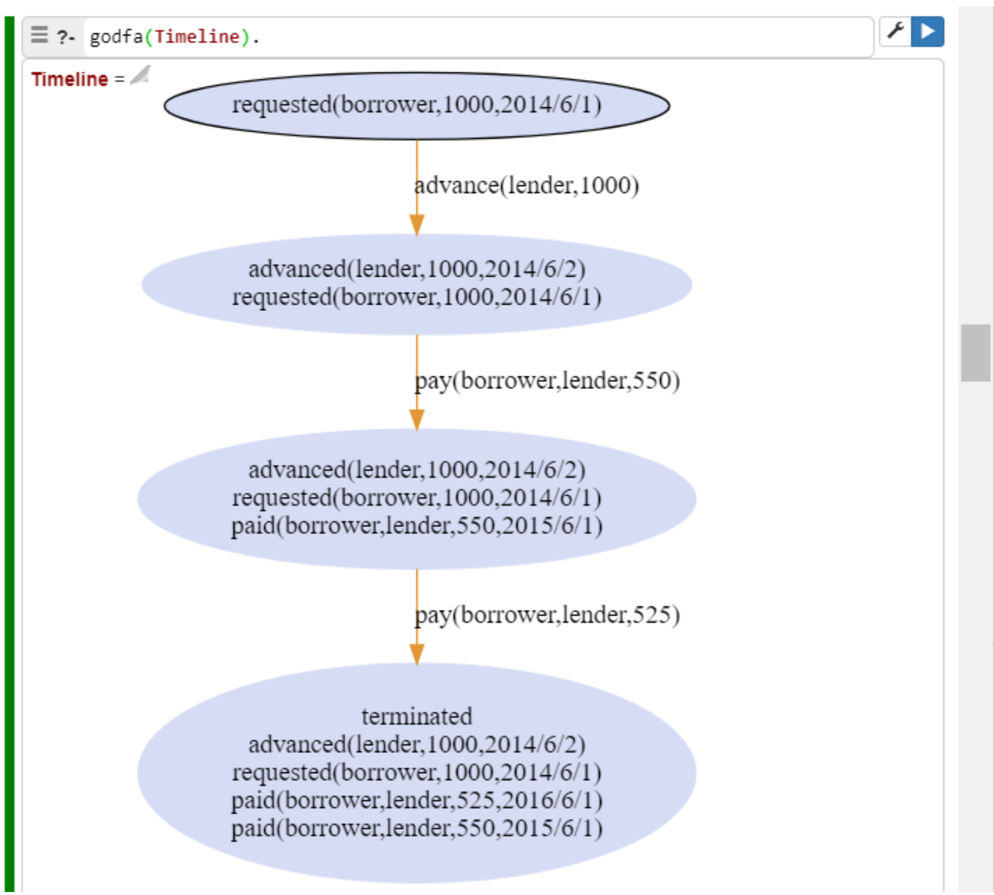
Using SWISH to realise interactive web based tutorials for logic based languages 2019, Jan Wielemaker, Fabrizio Riguzzi, Robert Kowalski, Torbjorn Lager, Fariba Sadri, Miguel Calejo. In Theory and Practice of Logic Programming, 19(2), 229-261.

```
end_of_day(2014/6/2)
initiates liable_to_litigation(lender)
if requested(borrower, 1000, 2014/6/1),
not advanced(lender, 1000).

end_of_day(2014/6/1)
initiates terminated
if not requested(borrower, 1000, 2014/6/1).
```

The history of states and events can be visualised as a state transition diagram (or derministic finite automaton)

```
observe request(borrower, 1000) at '2014-06-01T15:00'. % at 15:00.
observe advance(lender, 1000) at '2014-06-02T18:00'.
observe pay(borrower,lender, 550) at '2015-06-01T12:00'.
observe pay(borrower,lender, 525) at '2016-06-01T06:00'.
```



Examples from

- The British Nationality Act
  logic programs.

- A simplified loan agreement
  deterministic finite automata.

- Rock, Paper, Scissors
  (Lessons and insights from a cryptocurrency lab)
  imperatives.

- ISDA Master Agreement
  (International Swaps and Derivatives Association)
  obligations.

Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab

K **Delmolino**, M Arnett, A Kosba, A Miller… - … Conference on Financial …, 2016 - Springer

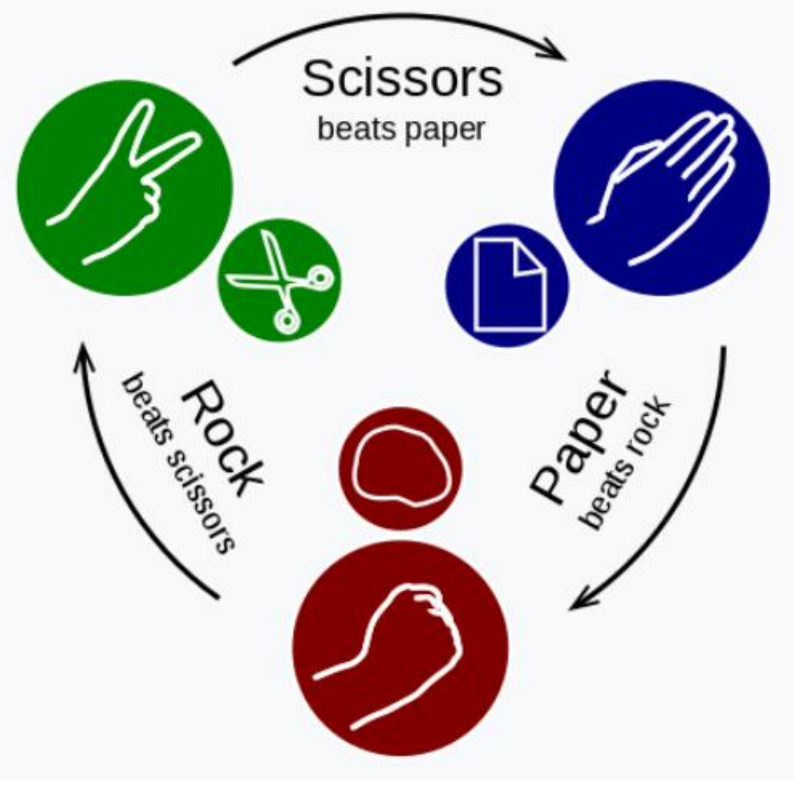
We document our experiences in teaching smart contract programming to undergraduate students at the University of Maryland, the first pedagogical attempt of its kind. Since smart contracts deal directly with the movement of valuable currency units between contractual …

☆ 99 Cited by 259 Related articles All 11 versions

Rock–paper–scissors

From Wikipedia, the free encyclopedia

Each player simultaneously forms one of three shapes with an outstretched hand:



Rock-paper-scissors in an “early” variant of Logical English

Logical Contracts

File Edit Examples Help

loan | life | deliveryDelay | loan | RockPaperScissorsBaseEN

```
then the reward that is a number becomes the number plus the value.

When a player pays a prize
then the reward that is a number becomes the number minus the prize.

If a first player has played a first choice at a first time
and a second player has played a second choice at the first time
and the first player is different from the second player
and the first choice beats the second choice
and it is not the case that the game is over at the first time
then initiate the game is over from the first time to a fourth time
and the reward is a prize at the first time
and the first player pays the prize from the first time to a second time.

If a first player has played a first choice at a first time
and a second player has played a second choice at the first time
and the first player is different from the second player
and it is not the case that the first choice beats the second choice
and it is not the case that the second choice beats the first choice
and it is not the case that the game is over at the first time
then initiate the game is over from the first time to a fourth time
and the reward is a prize at the first time
and a number is half the prize
and the first player pays the number from the first time to a second time
and the second player pays the number from the first time to the second time.
")
```

Rock-paper-scissors in LPS on the Ethereum blockchain

```
beats(scissors, paper).
beats(paper, rock).
beats(rock, scissors).

prolog_events e_transaction(latest,_From,_Input,_Wei,_To). % Generate events from the blockchain

e_transaction(latest,From,Input,Wei,To) initiates played(From,Input,Wei) if
    lps_my_account(To), Wei>0, not played(From,_,_).

fluents played(_Player,_Choice,_Value), gameOver.

reward(R) at T if
    balance(V) at T,
    R is round(V*0.9). % keep 10% :-)

balance(B) at T if
    findall(V,played(_,_,V) at T,L), sum_list(L,B).

num_players(N) at T if
    findall(P, played(P,_,_) at T, L), length(L,N).

false num_players(N), N>2.

pay(Player,Prize) from T1 to T3 if    % plan / macro action on the blockchain
    lps_my_account(Us),
    e_sendTransaction(Us,Player,Prize,PaymentTx) from T1 to T2,
    e_existsTransactionReceipt(PaymentTx) at T3.

if played(P0,Choice0,_) at T1, played(P1,Choice1,_) at T1, P0\==P1, beats(Choice0,Choice1), not gameOver at T1
then initiate gameOver from T1, reward(Prize) at T1, pay(P0,Prize) from T1 to T2.

if played(P0,Choice,_) at T1, played(P1,Choice,_) at T1, P0 @> P1, not gameOver at T1
then initiate gameOver from T1, reward(Prize) at T1, Half is Prize/2, pay(P0,Half) from T1, pay(P1,Half) from T1.
```

Reactive rules in LPS behave like imperative programs

```
if  played(P0,Choice0), played(P1,Choice1), P0\==P1,
beats(Choice0,Choice1), not gameOver, reward(Prize)
then initiate gameOver, pay(P0,Prize).

```

Imperative English

If a player has played a choice and another player has played another choice,
and the choice beats the other choice ,
and the game is not over, and the reward is a prize
then end the game and pay the player the prize.

Declarative English

If a player has played a choice and another player has played another choice,
and the choice beats the other choice ,
and the game is not over, and the reward is a prize
then the game becomes over and the player is paid the prize.

ANALYSIS 18 3 JANUARY 1958

IMPERATIVE AND DEONTIC LOGIC

By P. T. GEACH

The logic of proper imperatives is, I think, fairly trivial. For every proper imperative, there is a future-tense statement whose 'coming true' is identical with the fulfilment of the imperative. This is the source of everything that can be said about the inferability, incompatibility, etc. of imperatives; their being imperatives does not affect these logical interrelations.

Examples from

- The British Nationality Act
  logic programs.

- A simplified loan agreement
  deterministic finite automata.

- Rock, Paper, Scissors
  (Lessons and insights from a cryptocurrency lab)
  imperatives.

- ISDA Master Agreement
  (International Swaps and Derivatives Association)
  obligations.

ISDA

International Swaps and Derivatives Association, Inc.

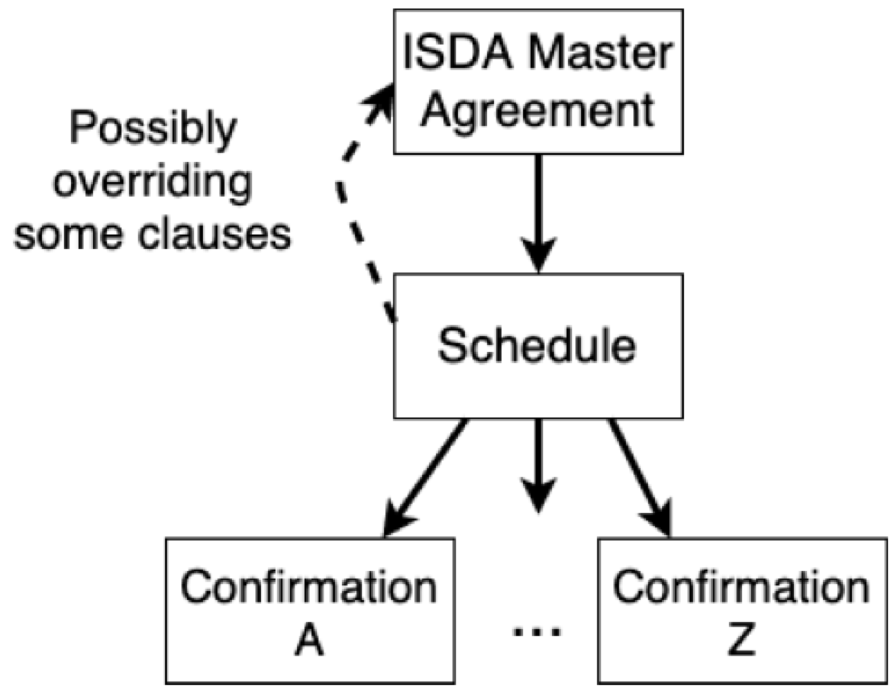
2002 MASTER AGREEMENT

dated as of March 22, 2011

Bank of America, N.A. and **LKQ Corporation**

have entered and/or anticipate entering into one or more transactions (each a "Transaction") that are or will be governed by this 2002 Master Agreement, which includes the schedule (the "Schedule"), and the documents and other confirming evidence (each a "Confirmation") exchanged between the parties or otherwise effective for the purpose of confirming or evidencing those Transactions. This 2002 Master Agreement and the Schedule are together referred to as this "Master Agreement".

Accordingly, the parties agree as follows:—



Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

First Steps Towards Logical English

Author:
Vesko Karadotchev

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing Science of Imperial College London

September 2019

2. Obligations

(a) ***General Conditions.***

(i) Each party will make each payment or delivery specified in each Confirmation to be made by it, subject to the other provisions of this Agreement.

It is an obligation that a party performs an action
if the action is specified in a confirmation
and the action is a payment
or the action is a delivery
and it cannot be shown that
it is not the case that it is an obligation that the party performs the action.

An obligation is satisfied
if the obligation is that the party performs an action
and the party performs the action.

An obligation is violated
if it is not the case that the obligation is satisfied.

(c) ***Netting of Payments.*** If on any date amounts would otherwise be payable:—

(i) in the same currency; and

(ii) in respect of the same Transaction,

by each party to the other, then, on such date, each party's obligation to make payment of any such amount will be automatically satisfied and discharged and, if the aggregate amount that would otherwise have been payable by one party exceeds the aggregate amount that would otherwise have been payable by the other party, replaced by an obligation upon the party by which the larger aggregate amount would have been payable to pay to the other party the excess of the larger aggregate amount over the smaller aggregate amount.

It is not the case that
 it is an obligation that a party pays to a counterparty
 an amount in a currency for a transaction on a date
if it is an obligation that the party pays to the counterparty
 a net amount in the currency for the transaction on the date.

It is an obligation that a party pays to a counterparty
 a net amount in a currency for a transaction on a date
if the net amount is a larger aggregate amount minus a smaller aggregate amount
and the larger aggregate amount is the sum of each amount of each payment by the party to the counterparty in the currency for the transaction on the date
and the smaller aggregate amount is the sum of each amount of each payment by the counterparty to the party in the currency for the transaction on the date.

Legislation as Logic Programs*

Robert A. Kowalski

Department of Computing
Imperial College of Science, Technology and Medicine
London SW7 2BZ, UK

January 1991
Revised June 1992

Abstract. The linguistic style in which legislation is normally written has many similarities with the language of logic programming. However, examples of legal language taken from the British Nationality Act 1981, the University of Michigan lease termination clause, and the London Underground emergency notice suggest several ways in which the basic model of logic programming could usefully be extended. These extensions include the introduction of types, relative clauses, both ordinary negation and negation by failure, integrity constraints, metalevel reasoning and procedural notation.

Conclusion

cc: B Kowalski computers.

IT Link Inc. Critique
Planner No. 5R-155
12xY – 600

APR 1993

ROMEIKE GROUP

OUT WITH THE PROGRAMMERS... ...IN WITH THE LAWYERS?

Who would we expect to be the best person to write a computer program? A trained programmer, most of us would reply. Perhaps an unusually inventive end-user, given access to tools such as code generators, and fourth generation languages. But a lawyer ...

Well, according to Bob Kowalski, a professor of computational logic at Imperial College London, lawyers are just the people. Kowalski, you may recall, devised the principles of logic programming, on which the PROLOG

language. No matter that it may be a rather specialised and stilted form of English, it is still much more accessible to the average human than any mathematical formalism.

An example, according to Kowalski, is the emergency notice found in London Underground trains:

> *Press the alarm signal to alert the driver. The driver will stop immediately if any part of the train is in a station. If not, the train will proceed*

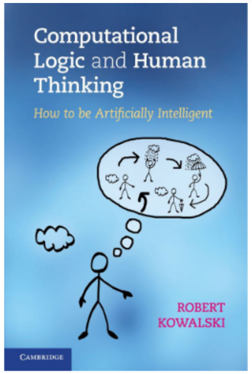
For more information and opinion

doc.ic.ac.uk/~rak/

Robert Kowalski

[Computational Logic and Human Thinking: How to be Artificially Intelligent]

This earlier draft of a book of the same title, published in July 2011 by Cambridge University Press, presents the principles of Computational Logic, so that they can be applied in everyday life. I have written the main part of the book informally, both to reach a wider audience and to argue more convincingly that Computational Logic is useful for human thinking. However, I have also included a number of additional, more formal chapters for the more advanced reader.

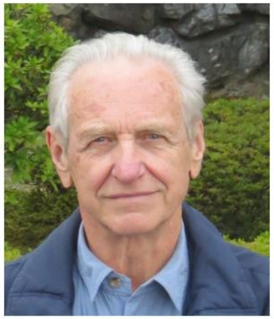


s and Distinguished Research Fellow

nputing
ondon

London SW7 2BZ, UK.

uk



Imperial College London

Department of Computing

First Steps Towards Logical English

Author:
Vesko Karadotchev

Not secure | lps.doc.ic.ac.uk

Imperial College London

Department c

LPS

Logic Production Systems

Home

LPS aims to close the gap

Not secure | logicalcontracts.com

Acknowledgements

Fariba Sadri	for BNA + LPS
Miguel Calejo	for LPS on SWISH + Logical Contracts
Jacinto Davila	for preliminary version of Logical English
Vesko Karadotchev	for most recent development of Logical English

Thank you

Additional background slides

A Survey and Classification of Controlled Natural Languages

Tobias Kuhn*
ETH Zurich and University of Zurich

What is here called ***controlled natural language (CNL)*** *has traditionally been given many different names. Especially during the last four decades, a wide variety of such languages have been designed. They are applied to improve communication among humans, to improve translation, or to provide natural and intuitive representations for formal notations. Despite the apparent differences, it seems sensible to put all these languages under the same umbrella. To bring order to the variety of languages, a general classification scheme is presented here. A comprehensive survey of existing English-based CNLs is given, listing and describing 100 languages from 1930 until today. Classification of these languages reveals that they form a*

British Nationality Act 1981

1981 CHAPTER 61

An Act to make fresh provision about citizenship and nationality, and to amend the Immigration Act 1971 as regards the right of abode in the United Kingdom.
[30th October 1981]

BE IT ENACTED by the Queen's most Excellent Majesty, by and with the advice and consent of the Lords Spiritual and Temporal, and Commons, in this present Parliament assembled, and by the authority of the same, as follows:—

PART I

BRITISH CITIZENSHIP

Acquisition after commencement

Acquisition by birth or adoption.

1.—(1) A person born in the United Kingdom after commencement shall be a British citizen if at the time of the birth his father or mother is—

(*a*) a British citizen; or

(*b*) settled in the United Kingdom.

(2) A new-born infant who, after commencement, is found abandoned in the United Kingdom shall, unless the contrary is shown, be deemed for the purposes of subsection (1)—

(*a*) to have been born in the United Kingdom after commencement; and

(*b*) to have been born to a parent who at the time of the birth was a British citizen or settled in the United Kingdom.

Robert Kowalski on criticism of a project to put the British Nationality Act into Prolog

How the logic of the law is put on trial

LOGIC programming arouses strong reactions. Supporters, like me, find it hard to contain their enthusiasm. Others, like Philip Leith, reported by Brian Bloomfield (Computer Guardian, March 26), find it hard to control their hostility.

At first, from its beginning in 1972, logic programming led a quiet and relatively unnoticed existence, until the Japanese identified it as the key to their fifth generation computing project. Most professional computer scientists were taken by surprise and judged the Japanese proposals to be confused and naive. Software engineers decried the lack of attention to orthodox software engineering principles. Artificial Intelligence researchers criticised the attention given to the logic programming language Prolog in preference to the more popular language Lisp. The major responses to the Japanese challenge, such as MCC in the USA, Alvey in Britain and Esprit in the EEC, did their best to avoid imitating the Japanese.

Slowly over the past six years attitudes have begun to change without any special promotion outside Japan, logic programming has become accepted into the mainstream of computing.

But logic programming is still vulnerable to attack, because unlike other approaches to computing it also aims to give insight into problem-solving by humans and not only by computers. In his article Brian Bloomfield reports Philip Leith's attack on our use of logic programming at Imperial College to formalise the 1981 British Nationality Act. These attacks are based upon a misrepresentation of our work.

Our use of logic to formalise legislation is based on the thesis that much human knowledge and belief can usefully be formulated as ... ical content which can be subjected to the rigorous application of logical deductions. Many of us believe that such logical analysis can profitably be applied to written legislation, whether the resulting deductions are then performed by human or by computer. We believe that logical analysis can help to clarify and simplify legislation, identify inconsistencies and determine its logical implications. We believe that such clarity and rigour can be achieved without sacrificing flexibility and compassion.

In our work we have used Prolog as a means of testing our analysis of legislation by means of computers. We found Prolog to be a powerful, though far from perfect, tool for this purpose.

There are very significant differences between logic, logic programming and Prolog. These differences are too complex to do justice to them in so little a space. I shall concentrate, therefore, entirely on explaining our views about the role of logic.

Bloomfield's article states that our work on the British Nationality Act was supported by the Government with a grant from the Alvey programme. On the contrary, our work on the British Nationality Act was started before the Alvey programme came into existence. As we explained in our May 1986 Communications of the ACM article, we chose the British Nationality Act for purely academic reasons, to test our theories about the value of logical analysis applied to law.

Compared with other legislation it is relatively self-contained and yet sufficiently complex that its implications are hard to determine by common sense reasoning alone. We were aware of claims that the British ... plications. It was our belief that a rigorous logical analysis of the Act would help to make any such implications apparent. It is especially regrettable, therefore, that unjustified insinuations of a racist character to our work have distracted attention from more substantial issues.

Philip Leith further misrepresents our position as identifying the legal process with the rigid application of rules embodying a single interpretation of legislation. Nowhere do we ourselves make such a claim. We are fully aware that the provisions laid down in legislation are only one source of the law. Contrary to the claims of our critics, we have always maintained that our British Nationality Act formalisation can only be used to determine what consequences follow from a given interpretation of the Act.

At no time have we ever proposed that our program could be used to decide questions of British citizenship autonomously by computer.

We have always emphasised that the application of logic and logic programming, in any context, needs to be embedded within a framework for assimilating knowledge, for revising beliefs and for comparing alternative systems of belief. It is precisely because reasoning in law requires such great flexibility that we have regarded it as an ideal domain to test and compare alternative theories of formal reasoning.

But even restricting our attention, as we did in our study of the British Nationality Act, to the formalisation of written legislation, we believe that logical analysis can help identify and eliminate unintended ambiguities without forcing us to resolve ones which are deliberate.

... actual legislation deliberately incorporates vague concepts such as being of "good character" and "intending to reside in the United Kingdom." Indeed it is largely through the presence of such concepts, admitting different interpretations, that law achieves much of its flexibility and apparent resistance to logical analysis.

Our thesis is that legislation can be formalised in logical terms without needing to specify any interpretation of vague concepts.

For example, we can represent the fact that being of good character and intending to reside in the United Kingdom are necessary conditions for entitlement to naturalise as a British citizen. But we do not need to define the meaning of the conditions in order to do so. Logical consequences derived from such a representation of legislation invariably contain such conditions as explicit qualifications. Thus we would not be able to derive an unqualified conclusion that Mary is entitled to naturalise as a British citizen, for example. But we might be able to derive that according to the legislation Mary would be entitled to naturalise as a British citizen if she were judged to be of good character. We believe that in such a way we can account both for the flexibility of legislation as well as for its fundamentally logical character.

There are other kinds of ambiguity in legislation which do not have such a benign character and which are generally not intended by the legislator. This is exemplified by rules such as "a person is entitled to benefit X if he has at least one dependent child and he has been unemployed for six months or he is incapable of work." Here the ambiguity concerns whether or not a ... entitled to benefit X if he does not have any children. This ambiguity needs to be resolved before the rule can be logically formalised. We believe that the identification and elimination of such ambiguities, which do not stand up to logical analysis, is valuable in its own right.

No one seriously doubts that logic is an essential tool for solving scientific and engineering problems. But even in these domains it needs to be combined with the creative discovery of appropriate assumptions and tempered by an awareness and a concern for the human implications of the conclusions. Logic does not determine what we should take as our assumptions. However, once the assumptions have been given it helps us to determine their logical consequences. Similarly, logic does not decide whether a given consequence is or is not acceptable. But if the consequence is judged to be unacceptable it helps us to identify the assumptions that must be rejected or modified.

Without clear but flexible rules applied with due attention to logic, bureaucrats and judges would be free to reach arbitrary, inscrutable, and unjustified decisions. Politicians could exercise power without accounting for their deeds.

We believe that there are powerful practical and ethical arguments in favour of bringing more logic to bear on human affairs. These arguments deserve to be analysed and vigorously debated. To the extent, however, that such debate consists largely of misrepresentation, it distracts attention from more important matters.

Robert Kowalski is Professor of Computational Logic at Imperial College of Science and Technology, University of London

My learned box of tricks

'26/2/93 "the Times"

Lawyers are about the last people most of us would think of putting in charge of our computers because they are often thought verbose and given to obscurantism.

Robert Kowalski, professor of computational logic at London's Imperial College of Science, Technology and Medicine, disagrees. In fact he thinks that the next generation of computer scientists should be recruited from the law, not mathematics.

At their best, says Professor Kowalski, who has just published a paper drawing parallels between legislation and logic programming, lawyers are capable of a clarity and precision of expression that matches that of mathematicians. But lawyers have an advantage over those same mathematicians, he says, in that they communicate in English, a language accessible to us all rather than only to a closed circle of the initiated.

Best of all, he says, are legal draftsmen. What they are really doing when they draft laws is writing programs expressed in human language to be executed by

the train is in a station" — is, in effect, a wonderful piece of programming.

Most computer scientists automatically fall back on mathematics to specify what a program should do, says the professor, but it may also be possible to achieve the same results in plain English.

Take the specification for sorting a sequence. Rendered in the kind of English which a legal draftsman might be at home with rather than in mathematical symbols, one might say, suggests professor Kowalski, that "the result of sorting a sequence should be an ordered permutation of the input sequence".

> 'Lawyers would make ideal program writers'

Concepts such as "order" and "permutation" in turn can also be defined in natural language rather than maths. Thus a sequence might be regarded as ordered "if for every pair of elements in the sequence the earlier one is smaller than or equal to the later element in the sequence".

Professor Kowalski says: "If you gave the

ATTORNEY-GENERAL'S DEPARTMENT

Office of Legislative Drafting

15 April 1993

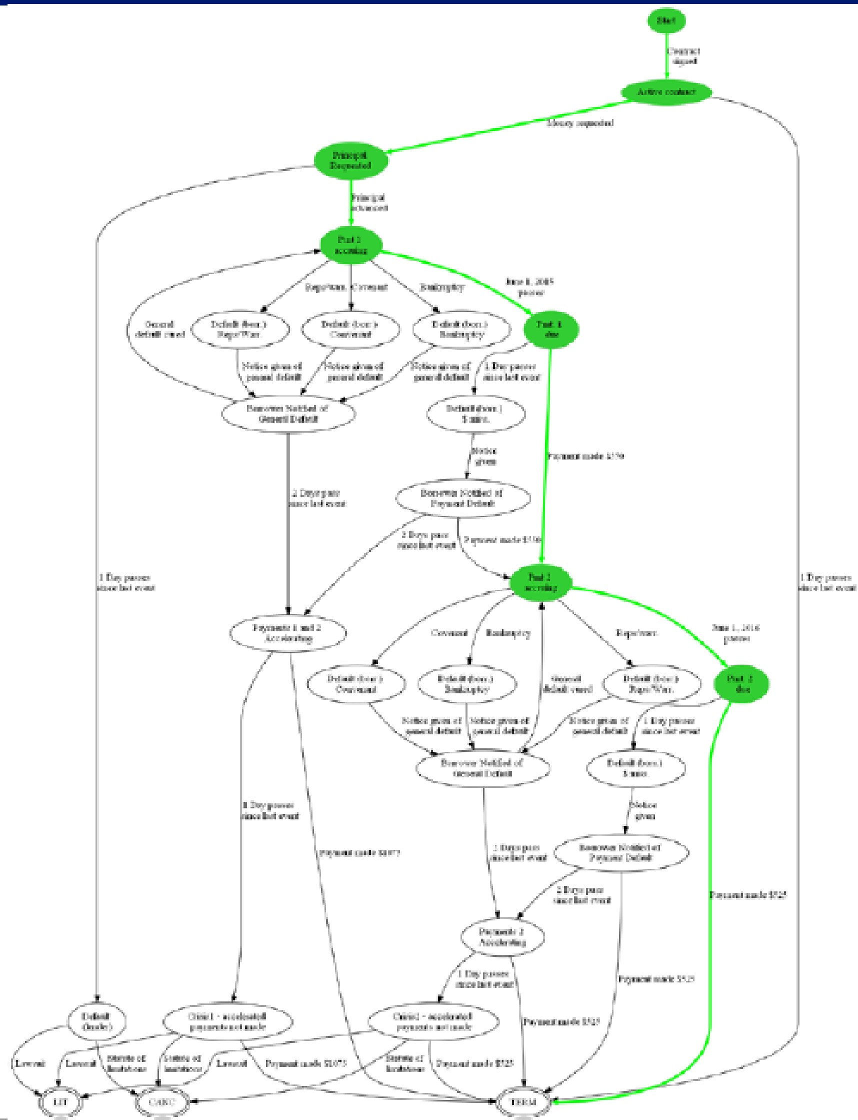
Professor Robert Kowalski
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate
London SW7 2BZ
ENGLAND

Dear Professor Kowalski

LEGISLATION AS LOGIC PROGRAMS

for drafting of delegated legislation
approaches between them vary considerably. I was interested to see that your redrafted versions of subsections 1 (1) and 1 (2) of the British Nationality Act 1981 were more like the way this Office would draft provisions of that kind than were the original provisions.

Contract as Automaton

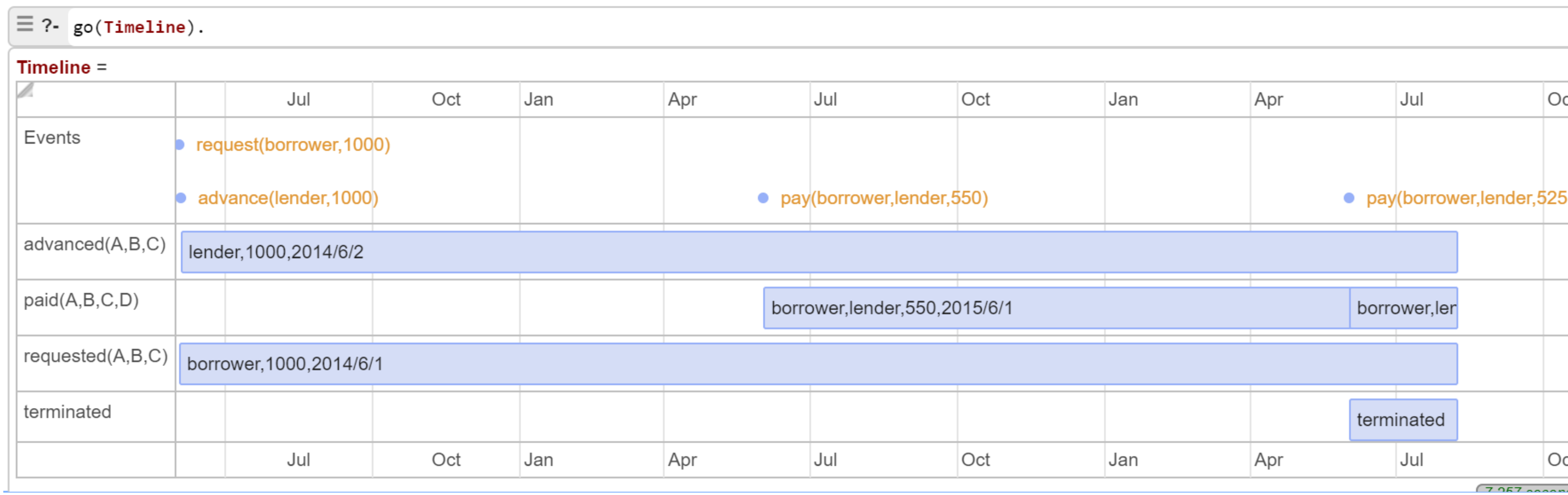


Deterministic Finite Automaton (DFA) as a chain of event and consequence:

- Start state (q_0) at the top
- Terminal states (3) at bottom
- "Happy" or intended path traced in green
- More "interesting" ramifications traced in black

The history of states and events can be visualised as a timeline (or Gantt chart)

```
observe request(borrower, 1000) at '2014-06-01T15:00'. % at 15:00.
observe advance(lender, 1000) at '2014-06-02T18:00'.
observe pay(borrower,lender, 550) at '2015-06-01T12:00'.
observe pay(borrower,lender, 525) at '2016-06-01T06:00'.
```



A smart contract in an imperative programming language

```
def player_input(choice):
  if num_players < 2 and msg.value == 1000:
    reward += msg.value
    player[num_players].address = msg.sender
    player[num_players].choice = choice
    num_players = num_players + 1
    return(0)
  else:
    return(-1)
def finalize():
  p0 = player[0].choice
  p1 = player[1].choice
  # If player 0 wins
  if check_winner[p0][p1] == 0:
    send(0,player[0].address, reward)
    return(0)
  # If player 1 wins
  elif check_winner[p0][p1] == 1:
    send(0,player[1].address, reward)
    return(1)
  # If no one wins
  else:
    send(0,player[0].address, reward/2)
    send(0,player[1].address, reward/2)
    return(2)
```

Not secure | lps.doc.ic.ac.uk

Imperial College
London

Department of Comp

LPS

Logic Production Systems

Home

LPS aims to close the gap between logical and imperative computer languages, by performin
actions to generate models to make goals of the logical form *if antecedent then consequen*
Model generation serves as a global imperative, which generates commands to make *conse*
true whenever *antecedents* become true.

Contact Us   Logical Contracts Server   Logical Contracts at RuleML+RR 2018

LOGICAL CONTRACT

A logical representation of a legal document that is close to natural, human language, but executable by computer.

It can be used to:

- monitor compliance of the parties to a contract.
- enforce compliance, by issuing warnings and remedial actions.
- explore logical consequences of hypothetical scenarios.
- query and update the Ethereum blockchain