Logic and Smart Contracts

Robert Kowalski, Imperial College London
Miguel Calejo, Interprolog.com
Fariba Sadri, Imperial College London

Outline

Imperative languages for
smart contracts

Logic for
legal documents

Combining logical and imperative semantics for
smart contracts

Artif Intell Law
https://doi.org/10.1007/s10506-018-9223-3

On legal contracts, imperative and declarative smart contracts, and blockchain systems

Guido Governatori[1] · Florian Idelberger[2] · Zoran Milosevic[3] ·
Regis Riveret[1] · Giovanni Sartor[2] · Xiwei Xu[4]

The term 'smart contract' was initially proposed in the early 90s for e-commerce applications (Szabo 1997) but has recently been widely used in the context of distributed ledger technologies and in particular blockchain technologies ….

In this context, a smart contract is any self-executing program running in the distributed ledger environment, and it is often meant to implement automated transactions agreed by the parties….

While not every smart contract has legal significance, many smart contracts are linked to legal contracts, namely with agreements meant to have legal effect.

DECISIONCAMP ON MONDAY, SEPTEMBER 17TH

Dan Selman and *Jerome Simeon*
Accord Project for Smart Legal Contracts

Bas Janssen and *Stijn van Dooremalen*
Smart contracts from legal text: interpretation, analysis and modelling !!

Thursday, September 20th

16:00-17:30	Session 35: Tutorial: LegalRuleML (RuleML+RR)

Monica Palmirani (Bologna), Guido Governatori (Data61/CSIRO).

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

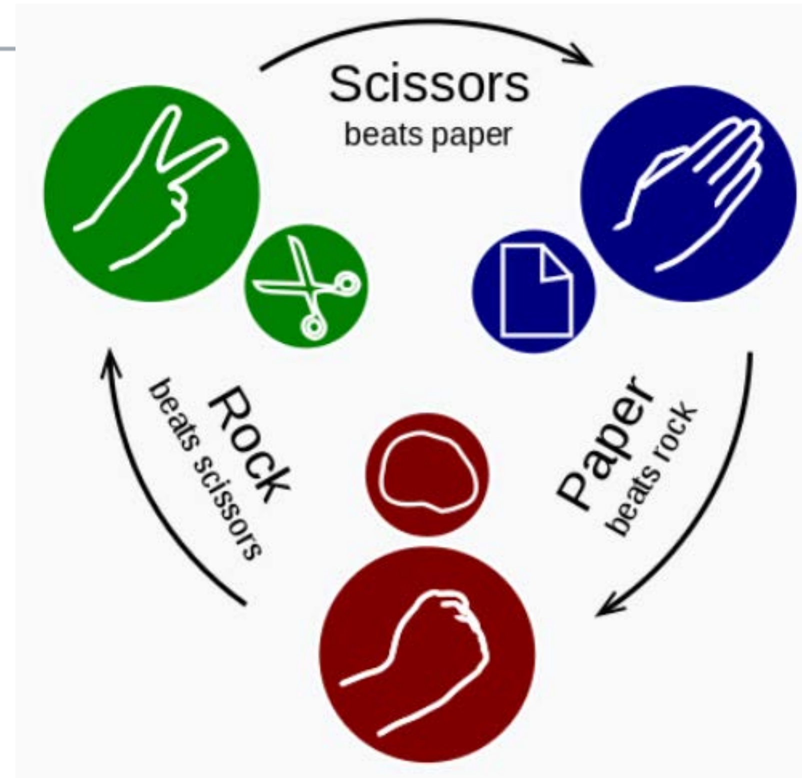
K Delmolino, M Arnett, A Kosba, A Miller… - … Conference on Financial …, 2016 – Springer

… pitfalls in programming safe **smart contracts**. The student is presented with the **buggy** version of the **contract** and asked to fix the bugs in a step-by-step, guided manner.

Cited by 118 Related articles All 11 versions

Rock–paper–scissors

From Wikipedia, the free encyclopedia



Rock–paper–scissors is
a zero-sum game
played between two people.

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

rock ✊
paper ✋
scissors ✌️

```
19     else:
20       return(-1)
21   def finalize():
22     p0 = player[0].choice
23     p1 = player[1].choice
24     # If player 0 wins
25     if check_winner[p0][p1] == 0:
26       send(0,player[0].address, reward)
27       return(0)
28     # If player 1 wins
29     elif check_winner[p0][p1] == 1:
30       send(0,player[1].address, reward)
31       return(1)
32     # If no one wins
33     else:
34       send(0,player[0].address, reward/2)
35       send(0,player[1].address, reward/2)
36       return(2)
```

Figure 3: **Pitfalls in smart contract design.** This buggy contract illustrates a few pitfalls: **Pitfall 1** (Lines 19 and 20): If a third player attempts to join the contract, his money effectively vanishes into a blackhole.

A typical smart contract is written in an imperative programming language with condition-action rules (sometimes called “conditional logic”).

accordproject.org

Accord Project

a consortium of lawyers and organizations,
developing a “functional programming language”, **Ergo**

“for the formation and execution of smart legal contracts
in a blockchain-agnostic standard implementation”.

accordproject.org

Accord Project

Specification

The Late Delivery And Penalty clause in the typical legal contract looks like this:

> Late Delivery and Penalty. In case of delayed delivery **except for Force Majeure cases**, the Seller shall pay to the Buyer for every **2 weeks** of delay penalty amounting to **10.5%** of total value of the Equipment whose delivery has been delayed. Any fractional part of **a week** is to be considered **a full week**. The total amount of penalty shall not, however, exceed **55%** of the total value of the Equipment involved in late delivery. If the delay is more than **10 weeks**, the Buyer is entitled to terminate this Contract.

```
namespace org.accordproject.latedeliveryandpenalty
import org.accordproject.common.*
import org.accordproject.latedeliveryandpenalty.*
// Declare a contract over a template model
contract LateDeliveryAndPenalty over TemplateModel {
  // Clause checking for late delivery and calculating penalty
  clause latedeliveryandpenalty(request : LateDeliveryAndPenaltyRequest) : LateDeliveryAndPenaltyResponse throws Error {
    // Guard against calling late delivery clause too early
    define variable agreed = request.agreedDelivery;
    enforce momentIsBefore(agreed,now()) else
    throw new Error{ message : "Cannot exercise late delivery before delivery date" }
    ;
    // Guard against force majeure
    enforce !contract.forceMajeure or !request.forceMajeure else
    return new LateDeliveryAndPenaltyResponse{
      penalty: 0.0,
      buyerMayTerminate: true
    }
    ;
    // Calculate the time difference between current date and agreed upon date
    define variable diff = momentDiffDays(now,agreed);
    // Penalty formula
    define variable penalty =
      (diff / contract.penaltyDuration.amount) * contract.penaltyPercentage/100.0 * request.goodsValue;
    // Penalty may be capped
    define variable capped = min([penalty, contract.capPercentage/100.0 * request.goodsValue]);
    // Return the response with the penalty and termination determination
    return new LateDeliveryAndPenaltyResponse{
      penalty: capped,
      buyerMayTerminate: diff > contract.termination.amount
    }
```

OFR WORKING PAPER
OFFICE OF FINANCIAL RESEARCH

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

Key Messages

- Financial contracts are structured internally as state-transition systems.
- Discrete finite automata (DFA) are an adequate formalism to represent this structure as finite sets of states, events, and transitions.

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,

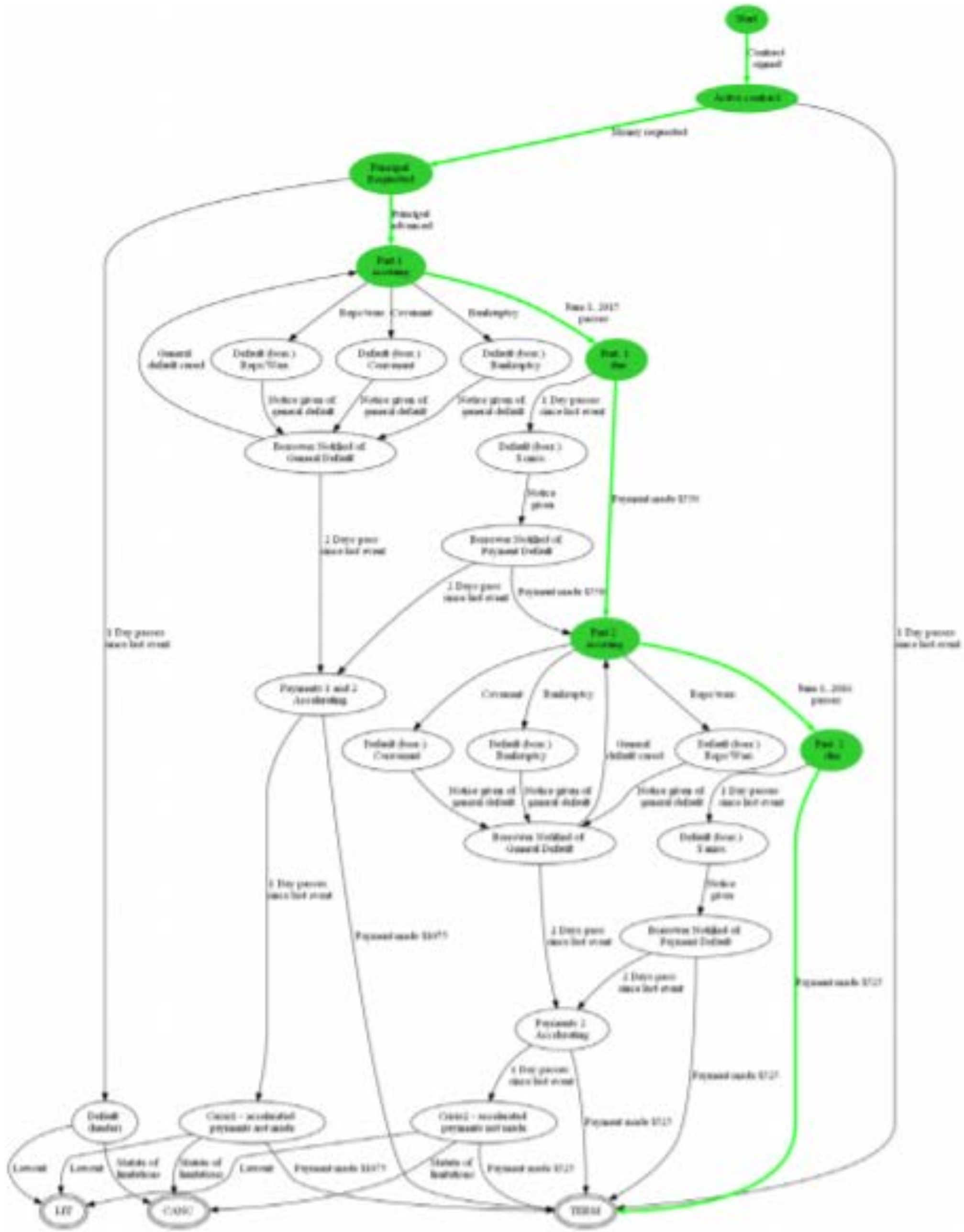


Figure 1: Graphical Representation of the Deterministic Finite Automaton (DFA) for the Streamlined Contract

Outline

Imperative languages for
smart contracts

Logic for
legal documents

Combining logical and imperative semantics for
smart contracts

Allen, Layman E.

Professor Emeritus of Law

808 Legal Research
734.764.9339
E-mail laymanal@umich.edu

Layman E. Allen has been a pioneer in the use of mathematical logic as a tool of analysis in law as well as in the use of computers in the field of legal research. He

Symbolic logic: a razor-edged tool for drafting and interpreting legal documents, Yale Law J. 66 (1957) 833–879.

Allen & Saxon (1984)

"**The University may terminate this lease** when the Lessee, having made application and executed this lease in advance of enrollment, is not eligible to enroll or fails to enroll in the University or leaves the University at any time prior to the expiration of this lease, or for violation of any provisions of this lease, or for violation of any University regulation relative to Resident Halls, or for health reasons, **by providing the student with written notice of this termination 30 days prior to the effective time of termination**; unless life, limb, or property would be jeopardized, the Lessee engages in the sales or purchase of controlled substances in violation of federal, state or local law, or the Lessee is no longer enrolled as a student, or the Lessee engages in the use or possession of firearms, explosives, inflammable liquids, fireworks, or other dangerous weapons within the building, or turns in a false alarm, **in which cases a maximum of 24 hours notice would be sufficient**".

The clause consists of a single sentence
with the ambiguous form:

```
A if A1 and A2 or A3 or A4 or A5 or A6 or A7
unless B1 or B2 or B3 or B4 or B5 in which cases B.
```

Allen & Saxon (1984) identify approximately 80 questions to disambiguate between all possible interpretations.

They conclude that the intended interpretation is:

```
((A IF ((A1 AND (A2 OR A3)) OR A4 OR A5 OR A6 OR A7))
IF NOT (B1 OR B2 OR B3 OR B4 OR B5))
AND (IF (B1 OR B2 OR B3 OR B4 OR B5) THEN B).
AND (IF NOT (B1 OR B2 OR B3 OR B4 OR B5) THEN NOT B
```

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.

The British Nationality Act as a logic program. 1986
Sergot, Sadri, Kowalski, Kriwaczek, Hammond. and Cory
Communications of the ACM, *29*(5), pp.370-386.

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.

Logic Program

X acquires british citizenship by subsection 1.1 at time T
if X is born in the uk at time T
and T is after commencement
and Y is father of X **or**
Y is mother of X
and Y is a british citizen at time T **or**
Y is settled in the united kingdom at time T.

The University of Michigan lease termination clause as a logic program

```
A if A1 and A2 and not exception.
A if A1 and A3 and not exception.
A if A4 and not exception.
A if A5 and not exception.
A if A6 and not exception.
A if A7 and not exception.
exception if B1.
exception if B2.
exception if B3.
exception if B4.
exception if B5.
B if exception.
```

AI, Logic and Law – State of the Art – notable examples

Defeasible deontic logic - related to **LegalRuleML.**

Legal Specification Protocol working group.

Accord Project, developing the **Ergo** language.

Contract Definition Language, at Stanford.

Ergo of **Coherent Knowledge**.

Legalese developing the **L4** language.

Neota "Logic".

Objects, Logic and English (OLE)

Outline

Imperative languages for smart contracts

Logic for legal documents

Combining logical and imperative semantics for smart contracts.

AI, Logic and smart contracts – next steps

On legal contracts, imperative and declarative smart contracts, and blockchain systems

Guido Governatori[1] · Florian Idelberger[2] · Zoran Milosevic[3] · Regis Riveret[1] · Giovanni Sartor[2] · Xiwei Xu[4]

To sum up our comparison of imperative and declarative smart contracts, the declarative approach has significant advantages over its imperative counterpart.

However, it is arguable that a full representation of a smart contract has to explicitly establish and link the normative effects (rights, obligation, transfers of entitlement) resulting from the contract with the procedure for implementing these rights and obligations through the computational actions performed by the contract, in the given infrastructure.

Hence, a hybrid approach combining imperative and declarative components would help to bridge the gap between smart contracts and their legal counterparts.

lps.doc.ic.ac.uk

Imperial College London

Department of Computing

LPS

Logic Production Systems

Home

reactive rule

logic program clause

LPS aims to close the gap between logical and imperative computer languages, by performing actions to generate models to make goals of the logical form *if antecedent then consequent* true.

Model generation serves as a global imperative, which generates commands to make *consequents* true whenever *antecedents* become true.

LPS also includes beliefs of the logical form *conclusion if conditions*. In addition to their logical interpretation, beliefs also have an imperative interpretation as procedures, which make or determine whether a *conclusion* is true by making or determining whether the *conditions* are true.

Attributing Mental Attitudes to Social Entities: Constitutive Rules are Beliefs, Regulative Rules are Goals

Guido Boella[1] and Leendert van der Torre[2]

[1] Dipartimento di Informatica - Università di Torino- Italy. E-mail: guido@di.unito.it
[2] CWI Amsterdam and TU Delft - The Netherlands. E-mail: torre@cwi.nl

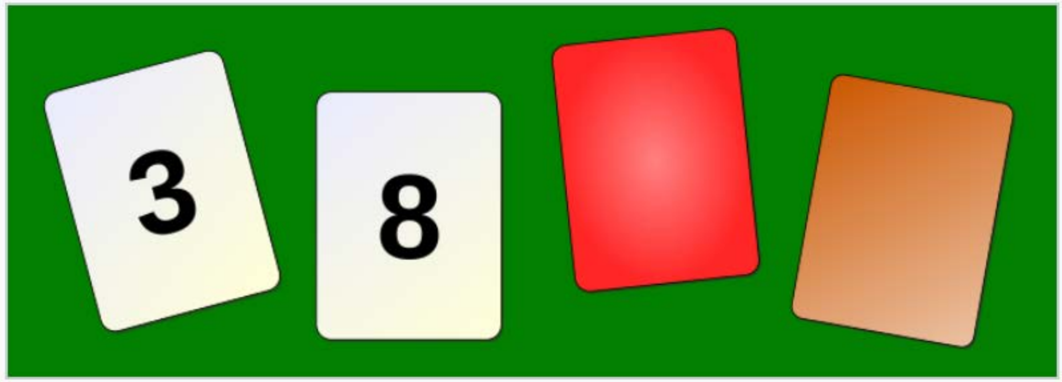
Abstract. In this paper, we propose a model of constitutive and regulative norms in a logical multiagent framework. We analyze the relationship between these two types of rules and explain similarities between them, using the metaphor of considering social entities - like normative systems, groups and organizations - as agents and of attributing them mental attitudes as well as an autonomous behavior. We argue that while constitutive norms expressing "counts-as" relations are modelled as the beliefs of social entities, regulative norms, like obligations, prohibitions and permissions, are modelled as their goals.

2004. *Proceedings of Collective Intentionality CollInt*, *4*.

Goals and Beliefs: It can be hard to tell the difference

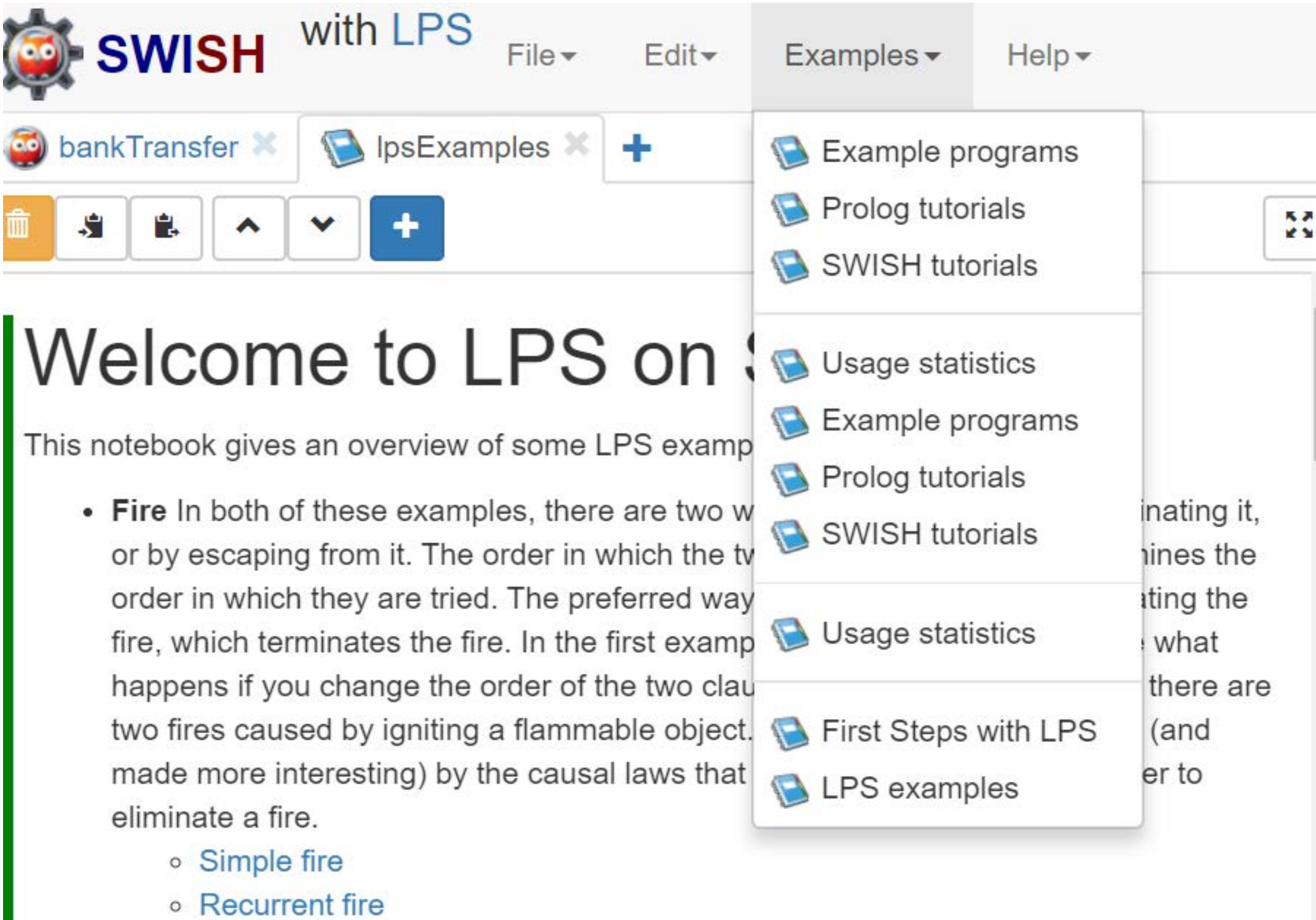
Wason selection task

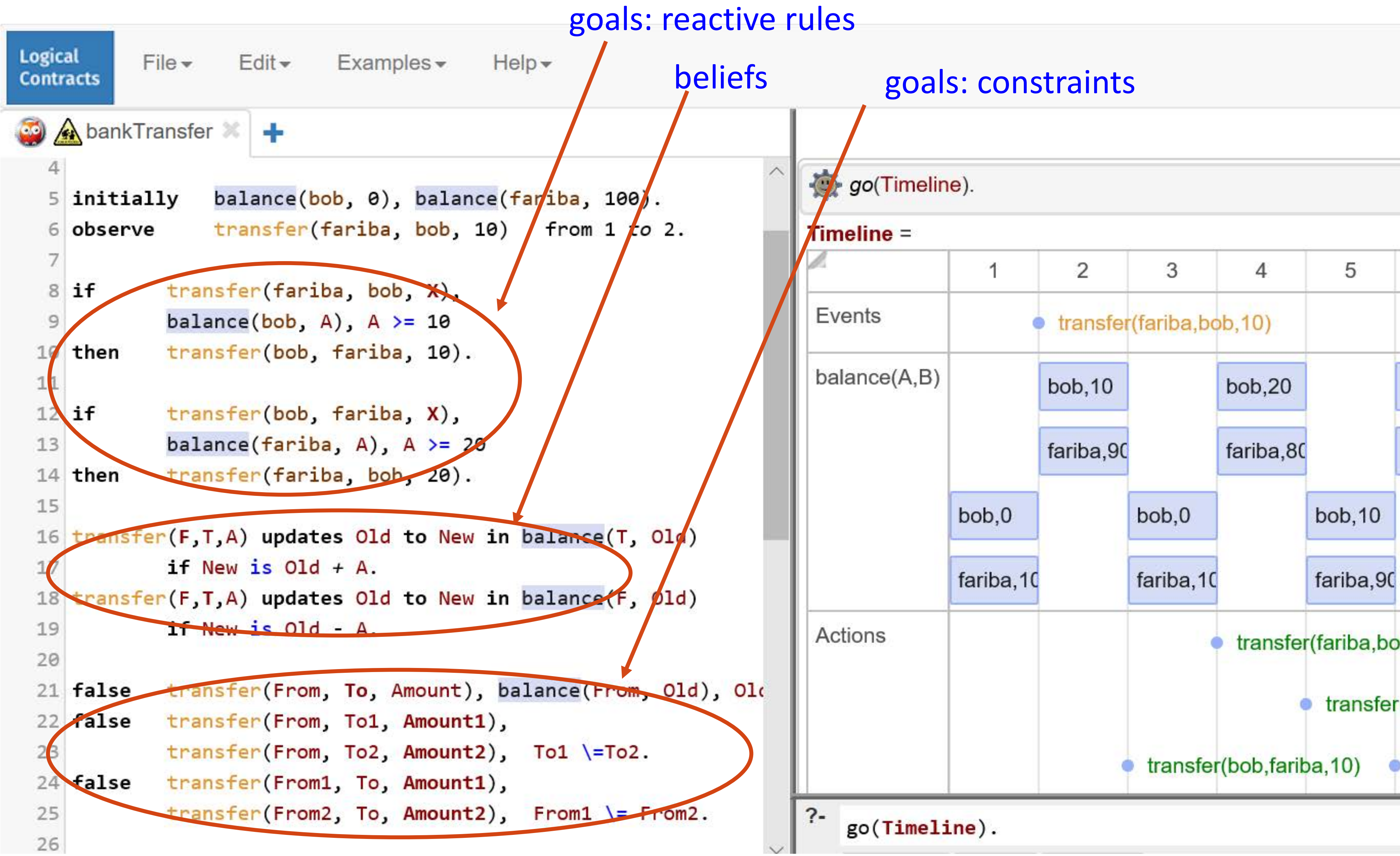
From Wikipedia, the free encyclopedia



Each card has a number on one side, and a patch of color on the other. Which card or cards must be turned over to test the idea that if a card shows an even number on one face, then its opposite face is red?

Each card has an age on one side, and a drink on the other. Which card(s) must be turned over to test the idea that if you are drinking alcohol then you must be over 18?


SWISH with LPS
File
Edit
Examples
Help
bankTransfer
lpsExamples
Example programs
Prolog tutorials
SWISH tutorials
Usage statistics
Example programs
Prolog tutorials
SWISH tutorials
Usage statistics
First Steps with LPS
LPS examples
Welcome to LPS on
This notebook gives an overview of some LPS examp
Fire In both of these examples, there are two w
inating it,
or by escaping from it. The order in which the tw
ines the
order in which they are tried. The preferred way
ting the
fire, which terminates the fire. In the first examp
what
happens if you change the order of the two clau
there are
two fires caused by igniting a flammable object.
(and
made more interesting) by the causal laws that
er to
eliminate a fire.
Simple fire
Recurrent fire


goals: reactive rules
beliefs
goals: constraints
Logical Contracts
File
Edit
Examples
Help
bankTransfer
initially balance(bob, 0), balance(fariba, 100).
observe transfer(fariba, bob, 10) from 1 to 2.
if transfer(fariba, bob, X),
balance(bob, A), A >= 10
then transfer(bob, fariba, 10).
if transfer(bob, fariba, X),
balance(fariba, A), A >= 20
then transfer(fariba, bob, 20).
transfer(F,T,A) updates Old to New in balance(T, Old)
if New is Old + A.
transfer(F,T,A) updates Old to New in balance(F, Old)
if New is Old - A.
false transfer(From, To, Amount), balance(From, Old), Ol
false transfer(From, To1, Amount1),
transfer(From, To2, Amount2), To1 \=To2.
false transfer(From1, To, Amount1),
transfer(From2, To, Amount2), From1 \= From2.
go(Timeline).
Timeline =
1
2
3
4
5
Events
transfer(fariba,bob,10)
balance(A,B)
bob,10
bob,20
bob,0
bob,0
bob,10
Actions
transfer(bob,fariba,10)
?- go(Timeline).

Timeline =

	1	2	3	4	5	6	7	8	9	10	11
Events		transfer(fariba,bob,10)									
balance(A,B)	bob,0 fariba,100	bob,10 fariba,90	fariba,100 bob,0	fariba,80 bob,20	bob,10 fariba,90	fariba,70 bob,30	fariba,80 bob,20	fariba,60 bob,40	bob,30 fariba,70	fariba,50 bob,50	
Actions			transfer(bob,fariba,10)	transfer(fariba,bob,20)	transfer(bob,fariba,10)	transfer(fariba,bob,20)	transfer(bob,fariba,10)	transfer(fariba,bob,20)	transfer(bob,fariba,10)	transfer(fariba,bob,2	
	1	2	3	4	5	6	7	8	9	10	11

?-

```
go(Timeline).
```

Examples▴ History▴ Solutions▴ table results Run!

Animation in LPS

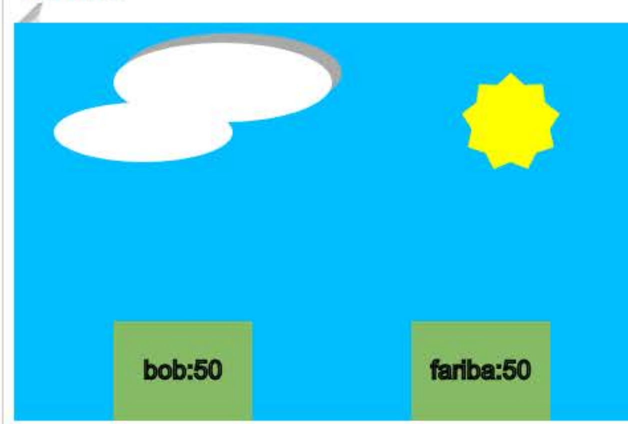
Logical Contracts File Edit Examples Help

bankTransfer

```
initially   balance(bob, 0), balance(fariba, 100).
observe     transfer(fariba, bob, 10)   from 1 to 2.

if      transfer(fariba, bob, X),
        balance(bob, A), A >= 10
then    transfer(bob, fariba, 10).

if      transfer(bob, fariba, X),
        balance(fariba, A), A >= 20
then    transfer(fariba, bob, 20).

transfer(F,T,A) updates Old to New in balance(T, Old)
        if New is Old + A.
transfer(F,T,A) updates Old to New in balance(F, Old)
        if New is Old - A.

false   transfer(From, To, Amount), balance(From, Old), Old < Amount.
false   transfer(From, To1, Amount1),
        transfer(From, To2, Amount2),  To1 \=To2.
false   transfer(From1, To, Amount1),
        transfer(From2, To, Amount2),  From1 \= From2.
```

go(Timeline).

Timeline =




```
?- go(Timeline).
```

Rock-paper-scissors in LPS

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

initially reward(0).

observe transaction_from(miguel,rock,1000) from 1 to 2.
observe transaction_from(bob,paper,1000) from 1 to 2.
observe transaction_from(alex,paper,1000) from 2 to 3. % one player too many!

transaction_from(From,Input,Wei) initiates played(From,Input).
transaction_from(_Player,_,X) updates Old to New in reward(Old) if  New is Old+X.

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.

pay(_,Prize) updates Old to New in reward(Old) if New is Old-Prize.

false transaction_from(_From,_Input,Wei), Wei=<0.
false transaction_from(From,_Input,_Wei), played(From,_).
false num_players(N), N>2.

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

```
mmount).
_Total), gameOver.




00) from 1 to 2.
) from 1 to 2.
0) from 2 to 3. % one player too many!

tes played(From,Input).
ld to New in reward(Old) if  New is Old+X

,Choice1) at T1, P0\==P1, beats(Choice0,C
Prize) at T1, pay(P0,Prize) from T1 to T2

hoice) at T1, P0 @> P1, not gameOver at T
Prize) at T1, Half is Prize/2,
 T1.

rd(Old) if New is Old-Prize.

), Wei=<0.
), played(From,_).


ed(P,_) at T, L), length(L,N).
```

go(Timeline).

Rejected observations [transaction_from(alex,paper,1000)] attempting to satis

Timeline =

	-1	0	1	2	3
Events				transaction_from(bob,paper,1	
				transaction_from(miguel,rock,	
gameOver					gameOver
played(A,B)				miguel,rock	
				bob,paper	
reward(A)			0		0
				2000	
Actions					pay(bob,2000)

?- go(Timeline).

Examples▴ History▴ Solutions▴

Rock-paper-scissors on Etherium 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.
```

Not secure | demo.logicalcontracts.com/p/Ballot.pl

Logical Contracts | File | Edit | Examples | Help

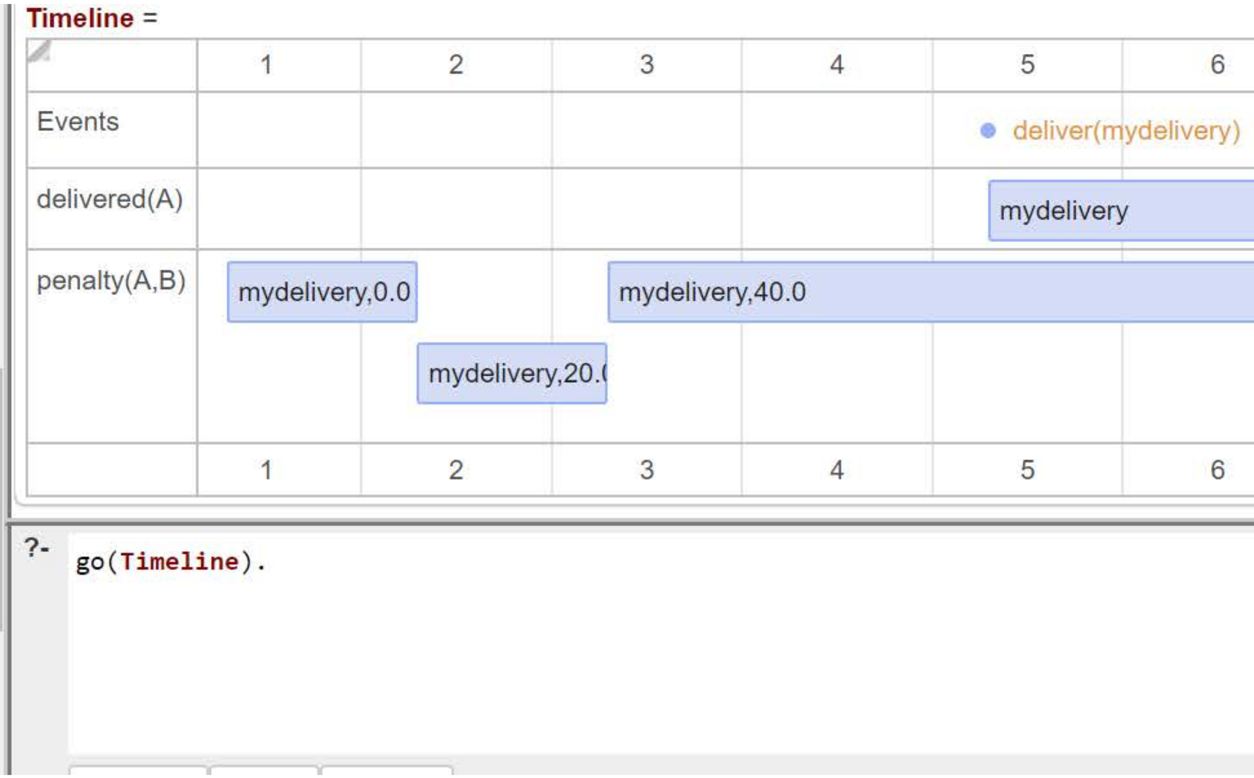
Ballot

```
events
    ballot(_Chairman, _Proposals), giveRightToVote(_Chairman, _Voter),
    delegate(_FromVoter, _ToVoter), vote(_Voter, _Candidate).

fluents chairman(_Chairman), voter(_Voter, _Weight), voted(_Voter, _Candida
    delegateOf(_Voter,_D), voteCount(_Candidate, _Votes).

observe ballot(chair, [trump, clinton]) from 1 to 2.
observe giveRightToVote(chair, miguel),
    giveRightToVote(chair, fariba),
    giveRightToVote(chair, bob), giveRightToVote(chair, jacinto)  from 3 to
observe delegate(bob, miguel)  from 4 to 5.
observe vote(miguel, clinton)  from 5 to 6.
observe delegate(jacinto,bob) from 6 to 7.
observe delegate(fariba, miguel)  from 7 to 8.

ballot(_Chairman, Proposals) initiates voteCount(Candidate, 0) if
    member(Candidate, Proposals).
ballot(Chairman, _Proposals) initiates voter(Chairman,1).
ballot(Chairman, _Proposals) initiates chairman(Chairman).

% the ballot is new:
```

The Accord Project delayed delivery example

```
initially penalty(mydelivery, 0.0).
deliver(Order) initiates delivered(Order).

end_of_day(Date2) updates Old to New in penalty(Order, Old) if
  latest_delivery(Order, Date1),
    not delivered(Order),
    real_date_add(Date1,Delay,Date2),
    not force_majeure(_),
    not terminated(Order),
   total_value(Order, Value),
   penalty_percentage(Order, PenaltyPercent),
   percentage_cap(Order, CapPercent),
   New is PenaltyPercent*Value*(Delay+1),
   Cap is CapPercent*Value,
   New =< Cap.
```

```
latest_delivery(mydelivery, 2018/4/1).
total_value(mydelivery, 100).
penalty_percentage(mydelivery, 0.20).
percentage_cap(mydelivery, 0.50).

observe deliver(mydelivery)
at'2018-04-04T15:00'.

end_of_day(Date2) updates Old to New i
     latest_delivery(Order, Date1),
     not delivered(Order),
     real_date_add(Date1,Delay,Date2),
     not force_majeure(_),
     not terminated(Order),
    total_value(Order, Value),
    penalty_percentage(Order, PenaltyPe
    percentage_cap(Order, CapPercent),
    New is PenaltyPercent*Value*(Delay+
    Cap is CapPercent*Value,
    New =< Cap.
```



The loan agreement embedded in a SWISH notebook

The FG and LPS representations employ a similar, "more precise" approach to the representation of violable obligations,

```
If an obligation is violated,
then some associated suboptimal state of affairs,
penalty, remedy or new obligation arises.
```

Although the first sentence of clause 1 expresses an obligation, nowhere in the contract is there any mention of a remedy
however, require an explicit representation of a remedy, or at least some representation of the less than ideal resulting st
in which the end of the day on June 1 2014 is an event that causes the lender to be liable to litigation. We can represent
the implicit intention of the contract that the contract terminates (correctly) if the borrower does not request the loan on th

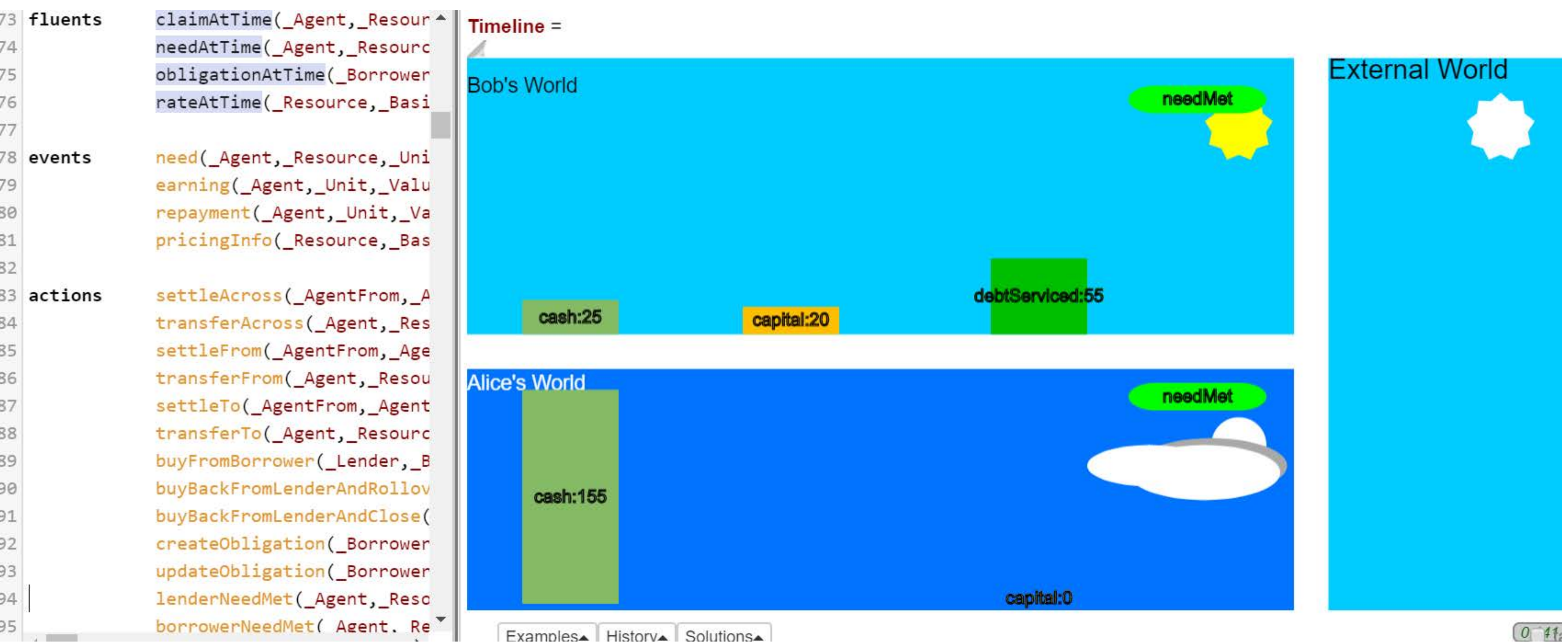
```
end_of_day(2014/6/2)
initiates liable_to_litigation(lender)
if requested(borrower, 1000, 2014/6/1),
not advanced(lender, 1000).

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

Adaptive Smart contracts using R3 Corda

Enabling Shift from Financial Products to Consumer Experiences

CordaCon, London 2018

Avinash Patil
Banking & Financial Services Practice
TCS



Secure | https://lps.js.org/sandbox/

lps.js The LPS runtime in JS

```
maxTime(10).
action(transfer(From, To, Amount)).
fluent(balance(Person, Amount)).

initially([balance(bob, 0), balance(fariba, 100)]).
observe(transfer(fariba, bob, 10), 1).

transfer(fariba, bob, X, T1, T2), balance(bob, A, T2), A >= 10 ->
        transfer(bob, fariba, 10, T2, T3).

transfer(bob, fariba, X, T1, T2), balance(fariba, A, T2), A >= 20 ->
        transfer(fariba, bob, 20, T2, T3).
```

Run Program | Example Programs

Time	1 (10 ms)	2 (11 ms)	3 (10 ms)	4 (10 ms)
Events	• transfer(fariba, ...			
Actions		• transfer(bob, far...	• transfer(fariba, ...	• transfe
Fluents	balance(bob, 0) balance(fariba, 100)	balance(bob, 10) balance(fariba, 90)	balance(bob, 0) balance(fariba, 100)	balance(bob, 20) balance(fariba, 80)

lps.js has been extended to build a desktop application, LPS Studio, to visualise LPS programs for interactive storytelling using the Electron framework.

Conclusions

Gap between two approaches:
software engineering and formal methods
logic, AI and Law.

Need a logic that combines
goals and
beliefs.

Need a logic that combines
declarative and
imperative sentences.

The language of well-written legal documents
can be an example for the computer languages of the future.

Complementary Slides

More about smart contracts

More from the Legal Specification Protocol

More from Logic, AI and Law

More from LPS

Google

"smart contract"

All Images News Videos Books More Settings Tools

About 2,610,000 results (0.61 seconds)

Smart contracts | Smart Contract Review | solidified.io

Ad www.solidified.io/

Join the market leader for **smart contract** security code audits now! Become an auditor. Audit Score.

What Are Smart Contracts? A Beginner's Guide to Smart Contracts

https://blockgeeks.com/guides/smart-contracts/

As Vitalik Buterin, the 22-year-old programmer of Ethereum, explained it at a recent DC Blockchain Summit, in a **smart contract** approach, an asset or currency is ...

People also search for

smart contract use cases	smart contract platforms
blockchain smart contracts pdf	smart contracts insurance
real world examples of smart contracts	the first smart contract platform is r3 corda

Smart contract - Wikipedia

https://en.wikipedia.org/wiki/Smart_contract

A **smart contract** is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation https://blockgeeks.com/guides/smart-contracts/

Smart Contracts: The Blockchain Technology That Will Replace Lawyers

#Beginners #Blockchain 101 #Blockchain for business #Blockchain startups

s 136 j 1K f 1K o

A Beginner's Guide to Smart Contracts

The best way to describe smart contracts is to compare the technology to a vending machine.

Ordinarily, you would go to a lawyer or a notary, pay them, and wait while you get the document.

With smart contracts, you simply drop a bitcoin into the vending machine (i.e. ledger), and your escrow, driver's license, or whatever drops into your account.

More so, smart contracts not only define the rules and penalties around an agreement in the same way that a traditional contract does, but also automatically enforce those obligations.

https://blockgeeks.com/guides/smart-contracts/

Example

Suppose you rent an apartment from me. You can do this through the blockchain by paying in cryptocurrency. You get a receipt which is held in our virtual contract; I give you the digital entry key which comes to you by a specified date. If the key doesn't come on time, the blockchain releases a refund. If I send the key before the rental date, the function holds it releasing both the fee and key to you and me respectively when the date arrives. The system works on the If-Then premise and is witnessed by hundreds of people, so you can expect a faultless delivery. If I give you the key, I'm sure to be paid. If you send a certain amount in bitcoins, you receive the key. The document is automatically canceled after the time, and the code cannot be interfered by either of us without the other knowing since all participants are simultaneously alerted.

Logic for legal documents

Imperative languages for smart contracts on blockchains

Distributed Ledger Technology: beyond block chain

A report by the UK Government Chief Scientific Adviser

Smart contracts are contracts whose terms are recorded in a computer language instead of legal language. Smart contracts can be automatically executed by a computing system, such as a suitable distributed ledger system. The potential benefits of smart contracts include low contracting, enforcement, and compliance costs.

Home > Volume 2, Number 9 - 1 September 1997 > **Szabo**

Formalizing and Securing Relationships on Public Networks

Nick Szabo

Abstract

Smart contracts combine protocols with user interfaces to formalize and secure relationships over computer networks.

Objectives and principles for the design of these systems are derived from legal principles, economic theory, and theories of reliable and secure protocols.

Similarities and differences between smart contracts and traditional business procedures based on written contracts, controls, and static forms are discussed, …..

From: Developing a Legal Specification Protocol: Technological Considerations and Requirements

LSP Working Group
Draft of July 28, 2018

Time for a Legal Specification Protocol (LSP)

In many domains of human activity, the application of electronic computing has made once cumbersome tasks quicker and easier.

The automation of portions of legal and regulatory processes holds the similar promise of delivering faster and better service at lower cost.

From: Developing a Legal Specification Protocol: Technological Considerations and Requirements

Interacting with Blockchain and Distributed Ledger Technology

A point of particular importance will be the ability of computational contracts and other machine executable legal specification to interact with blockchains and other distributed ledger technologies.

The current “smart contract” initiatives, … , have been developed largely in the blockchain context,

and while their use has largely been to specify relatively simple payment trigger transactions, they have the potential for much more complex specification.

Developing a Legal Specification Protocol: Technological Considerations and Requirements

The boom in blockchain development has included initiatives around the so-called "smart contract."

Although the rather grand name would seem to imply that the LSP has already occurred, as currently employed most blockchain smart contracts are executable scripts of relatively low contractual expressivity.

For the most part, they are single trigger links between some event or instruction and the delivery of cryptocurrency funds from one holder to another.

As Vitalik Buterin, a leading Ethereum programmer has described it, the smart contract sets up a conditional payment instruction in code:

> "and the program runs this code and at some point it automatically validates a condition and it automatically determines whether the asset should go to one person or back to the other person, or whether it should be immediately refunded to the person who sent it or some combination thereof."

Logic for
legal documents

Imperative languages for
smart contracts
on block chains

propositional logic - Layman Allen	- 1950s
logic programming (LP) - British Nationality Act	- 1980s
extensions of LP and other logics	- 3rd millenium

Rules and exceptions

English

40.-(2) The Secretary of State may
by order deprive a person of a citizenship status
if the Secretary of State is satisfied that
deprivation is conducive to the public
good.

40.-(4) The Secretary of State may not
make an order under subsection (2)
if he is satisfied that
the order would make the person stateless.

Logic program

*The Secretary of State may by order
deprive a person of a citizenship status*
if *the Secretary of State is satisfied that
deprivation is conducive to
the public good*

and *the Secretary of State is*
not *satisfied that
the order would make the person
stateless.*

Logic, AI and Law – what went wrong?

Lack of commercialisation,
due in part to open textured predicates,
such as “good character”, “reasonable effort”,
and the resulting apparent need for case-based reasoning.
(cured by precise definitions or human judgement)

Prolog infinite loops.
(cured by tabling in XSB, forward reasoning in Datalog and grounding in ASP).

Competition from and confusion with production systems,
e.g. Oracle Policy Management (OPM)
http://www.oracle.com/industries/government/pdfs/oracle-haley-enterprisepublic-sector-ds.pdf.
(cured by tabling in XSB?)

Confessions of a production rule vendor (part 2)

Apr 2nd, 2018 by paul@haleyAI.com.
http://haleyai.com/wordpress/2018/04/02/confessions-of-a-production-rule-vendor-part-2/#more-1169

"XSB Prolog provides more powerful reasoning than production systems.

It can handle logical inconsistences unlike theorem provers.

It is more practical and easier to use than other Prologs.

XSB Prolog is commercial, open-source.
It has been used in IBM Watson and by U.S. Customs."

AI, Logic and Law – Current State of the Art – selected examples

Accord Project, a consortium of lawyers and organizations, developing a functional programming language, **Ergo** "for the formation and execution of smart legal contracts in a blockchain-agnostic standard implementation".

Legal Specification Protocol working group developing "a coordinated, interoperable standard for embodying contracts and other legal formulations as executable computer code". A draft white paper highlights the use of **deterministic finite automata**. In the DFA approach, all clauses of a contract are represented in the form *current state -> event -> next state*.

The **R3** consortium of 70 global financial institutions has a conceptual framework, **Corda,** for smart contracts in finance. Corda is an alternative to conventional blockchain systems. Smart contracts in Corda are compiled into a Java Virtual Machine, standardising on a bytecode set, which is neutral about the language.

AI, Logic and Law – State of the Art – notable examples

Defeasible deontic logic extends LP with defeasible rules, defeaters and priorities. Deontic concepts are expressed by modal operators. Related to **LegalRuleML** (Athan, Governatori, Palmirani, Paschke, & Wyner. Proc. of the 11th Reasoning Web Summer School, 2015.)

Accord Project, a consortium of lawyers and organizations, using a functional programming language, **Ergo** "for the formation and execution of smart legal contracts in a blockchain-agnostic standard implementation".

Legal Specification Protocol working group developing "a coordinated, interoperable standard for embodying contracts and other legal formulations as executable computer code". A draft white paper highlights the use of **deterministic finite automata**. In the DFA approach, all clauses of a contract are represented in the form *current state -> event -> next state*.

AI, Logic and Law – State of the Art – notable examples

Contract Definition Language, part of the computable contracts project at Stanford, used to model several U.S. Federal Acts. It uses LP to encode legal regulations and causal laws. Deontic concepts are encoded using meta-predicates.

Ergo of **Coherent Knowledge**, implemented in XSB Prolog, used in a POC to automate US Federal Reserve Regulation W, which regulates transactions between banks,. The automation is an extended database of facts and legal rules, with queries that evaluate whether the facts comply with the rules. Deontic concepts are encoded using meta-predicates.

Legalese is a company developing the **L4** language for drafting “legal documents the way programmers develop software”. L4 is a modal language, with modal operators for time. It aims to support formal verification of contracts, using model checking techniques.

AI, Logic and Law – Current State of the Art – selected examples

Neota "Logic" is a commercial production rule system for legal applications. Rules represent legal expertise, rather than legal documents.

Objects, Logic and English (OLE) is designed for "managers, legal and financial professionals to read and write their smart contracts in a manner close to their own professional practice using their own language". The language includes logical rules, transitional rules and constraints, with an English-like, object-oriented syntax. A compiler from OLE to Solidity is being developed for the Ethereum blockchain.

The syntax of LPS

Goals include reactive rules in First-order logic:

for all X [*antecedent* → *there exists Y consequent*]

or *if antecedent then consequent.*

Beliefs are clauses in logic programming form:

for all X [*there exists Y conditions* → *conclusion*]

or *conclusion if conditions.*

All humans are mortal. ← Belief:
mortal(X) if human(X).

All humans are virtuous. ← Goal:
if human(X) then virtuous(X).

Psychological studies show that people have trouble reasoning with conditionals. e.g. the Wason selection task.

Secure | https://lps.mauris.sg

Logic Production Systems   About

Hello there!

Run

Welcome to LPS JavaScript Implementation Demo. Enter the LPS source code on the right and press "Run" to run the LPS program.

The LPS program execution currently happens on the server-side. Browser-side execution is possible, but will only be demonstrated once the project becomes open source.

Example Programs

Below are some preloaded LPS programs that you can load and get started.

- mark
- mark-hiccup
- fireSimple
- fireRecurrent
- mapColouring

```
maxTime(10).
action(transfer(From, To, Amount)).
fluent(balance(Person, Amount)).

initially([balance(bob, 0), balance(fariba, 100)]).
observe(transfer(fariba, bob, 10), 1).

transfer(fariba, bob, X, T1, T2), balance(bob, A, T2), A >= 10 ->
        transfer(bob, fariba, 10, T2, T3).

transfer(bob, fariba, X, T1, T2), balance(fariba, A, T2), A >= 20 ->
        transfer(fariba, bob, 20, T2, T3).

updates(transfer(F, T, A), balance(T, Old), balance(T, Old + A)).
updates(transfer(F, T, A), balance(F, Old), balance(F, Old - A)).

% <- transfer(From, To, Amount), balance(From, Old), Old < Amount.
% <- transfer(From, To1, Amount1), transfer(From, To2, Amount2), To1 != To2.
% <- transfer(From1, To, Amount1), transfer(From2, To, Amount2), From1 != From2.
```

Time	1 (0 ms)	2 (5 ms)	3 (5 ms)	4 (7 ms)
Events	• transfer(fariba, bob, 10)			
Actions		• transfer(bob, fariba, 10)	• transfer(fariba, bob, 20)	
Fluents	balance(bob, 0)	balance(bob, 10) balance(fariba, 90)	balance(fariba, 100) balance(bob, 0)	balance balance

ttps://lps.mauris.sg

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