Logical Contracts

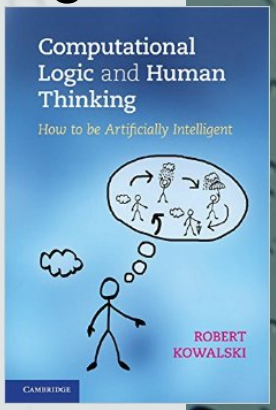
Logical Contract Server Preview

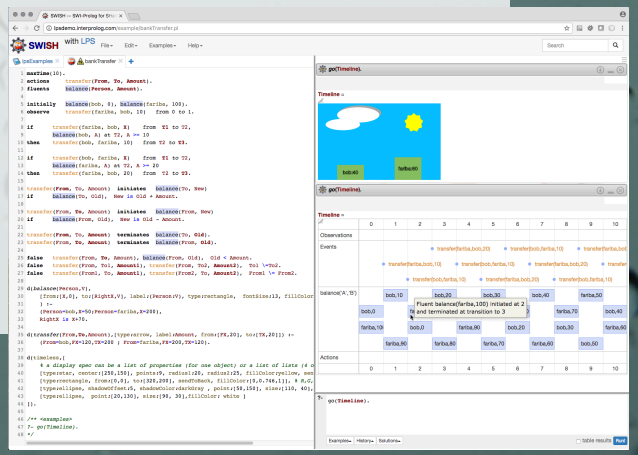
RuleML Webinar, Nov 24, 2017

Miguel Calejo (*with LC team*)

Preamble: Logical Production Systems (LPS)

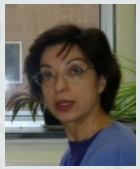
- Research by Kowalski and Sadri at Imperial College
  - Legal Reasoning, British National Act, Obligation as goal satisfaction, etc.
  - *Computational Logic for Human Thinking book*
  - Logical agents with Dávila
  - *Computational Logic for Use in Teaching* with Calejo
  - Several LPS implementations
- http://lps.doc.ic.ac.uk





Logical Contracts?



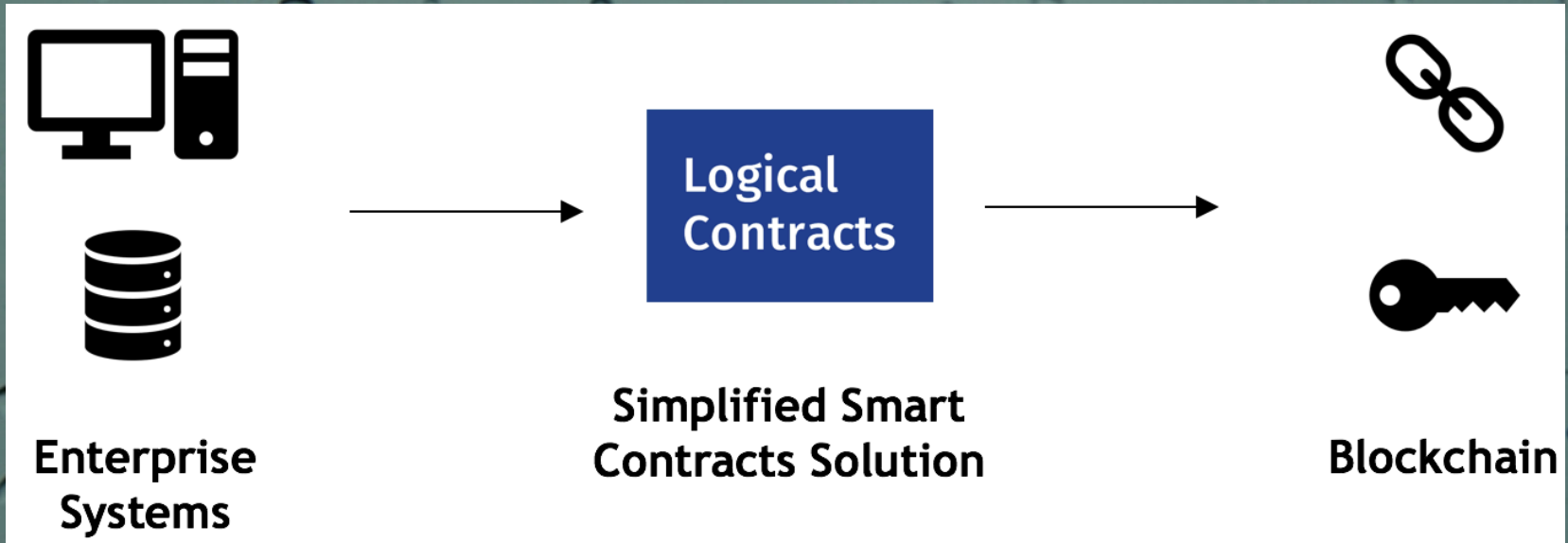




- Miguel Calejo, CTO
- Bob Kowalski, Chief Scientist
- Jacinto Dávila, Senior Engineer
- Fariba Sadri, External Research
- Alex Garcia, Business Dev

imperial innovations

- Imperial College London spinoff (*now courting investors*)
- Enhance and apply LPS to…
  - Smart contracts: a good fit

Sales pitch du jour...



Integrating enterprise systems with smart contracts

Imperative logic in plain English

Vision through example

- *Rock-Paper-Scissors gambling game*
- *1 click:*
  - *simulation is displayed, graphical and narrative*
- *1 click:*
  - *contract executing on cloud with new Ethereum address*
- *We're done:*
  - *bets are received, game is decided, prize paid*
  - *Ethereum retains history and contract reference*

RSP gambling (simulated)

Logical Contracts

Logical Contracts -- Alpha Der

Not Secure | demo.logicalcontracts.com/example/RockPaperScissorsBase.pl

Logical Contracts | File | Edit | Examples | Help | Search | Login

RockPaperScissorsBase | fintechExamples

```
3
4  beats(scissors, paper).
5  beats(paper, rock).
6  beats(rock, scissors).
7
8  events transaction_from(_From,_Input,_Ammount).
9  fluents played(_Player,_Choice),reward(_Total), gameOver.
10 actions pay(_Player,_Prize).
11
12 initially reward(0).
13
14 % simulate input events:
15 observe transaction_from(miguel,rock,1000) from 1 to 2.
16 observe transaction_from(bob,paper,1000) from 1 to 2.
17 observe transaction_from(alex,paper,1000) from 2 to 3. % one player too many!
18
19 transaction_from(From,Input,Wei) initiates played(From,Input) if
20     Wei>0, not played(From,_).
21
22 num_players(N) at T if
23     findall(P, played(P,_) at T, L), length(L,N).
24
25 false num_players(N), N>2.
26
27 transaction_from(_Player,_,X) updates Old to New in reward(Old) if
28     New is Old+X.
29
30 pay(_,Prize) updates Old to New in reward(Old) if New is Old-Prize.
31
32 if  played(P0,Choice0) at T1, played(P1,Choice1) at T1, P0\==P1, beats(Choice0,Choice1), not gameOver at T1
33 then initiate gameOver from T1, reward(Prize) at T1, pay(P0,Prize) from T1 to T2.
34
35 if played(P0,Choice) at T1, played(P1,Choice) at T1, P0 @> P1, not gameOver at T1
36 then initiate gameOver from T1, reward(Prize) at T1, Half is Prize/2, pay(P0,Half) from T1, pay(P1,Half) from T1.
```

go(Timeline).

Rejected observations [transaction_from(alex,paper,1000)] attempting to satisfy prospective preconditions

Timeline =

	-2	-1	0	1	2	3	4	5	6	7
Observations										
Events						pay(bob,2000)				
						initiate gameOver				
					transaction_from(bob,pa					
					transaction_from(migue					
gameOver						gameOver				
played('A','B')					miguel,rock					
					bob,paper					
reward('A')			0			0				
					200					
Actions										
	-2	-1	0	1	2	3	4	5	6	7

0.125 seconds cpu time

?- go(Timeline).

Examples | History | Solutions | table results | Run!

LPS for logic programmers:

- LPS is a Prolog superset, adding explicit time…
  - "Time" as a sequence of discrete cycles
- Literals can be **timeless** as usual… or not:
  - **Fluents**: true over a cycle
  - **Events/actions**: happen in cycle transitions
- Extra syntax:
  - Fluent and event rules and declarations, external observations
  - Post conditions (actions changing fluents), integrity constraints
  - Reactive rules
- A Prolog program executes "instantly". But a LPS program executes over time cycles:
  - Reactive rules introduce *parallel* (AND) goals
  - Fluents and events/actions mean.. delay

Timeless truth:

```
beats(scissors,paper)
```

Event:

```
happens(transaction_from(bob,paper,1000),1,2)
```

Action:

```
pay(bob,2000) from 2 to 3
```

Fluents:

```
holds(reward(2000),2)
```

```
gameOver at 3
```

Blockchain 101

- "Global" database, or "distributed ledger"
  - Grows monotonically with new versions ("blocks") every few seconds
  - Contains account balances and transactions
- Ethereum, Hyperledger, …
  - Can also contain and execute code (smart contracts): Solidity, Javascript
- Ethereum interface for LPS
  - LPS in cloud off blockchain; flexibility, abstraction from blockchain du jour
  - e_getBalance(+Account,+Block,-Value)
  - e_transaction(+Block,?From,?Input,?Wei,?To)
  - e_sendTransactionWithAtom(+From,+To,+Value,+Message,-Tx)
  - e_existsTransactionReceipt(+Tx)
  - For anything else doable with Ethereum RPC:
    - e(method(Argument1,Argument2,..Result)

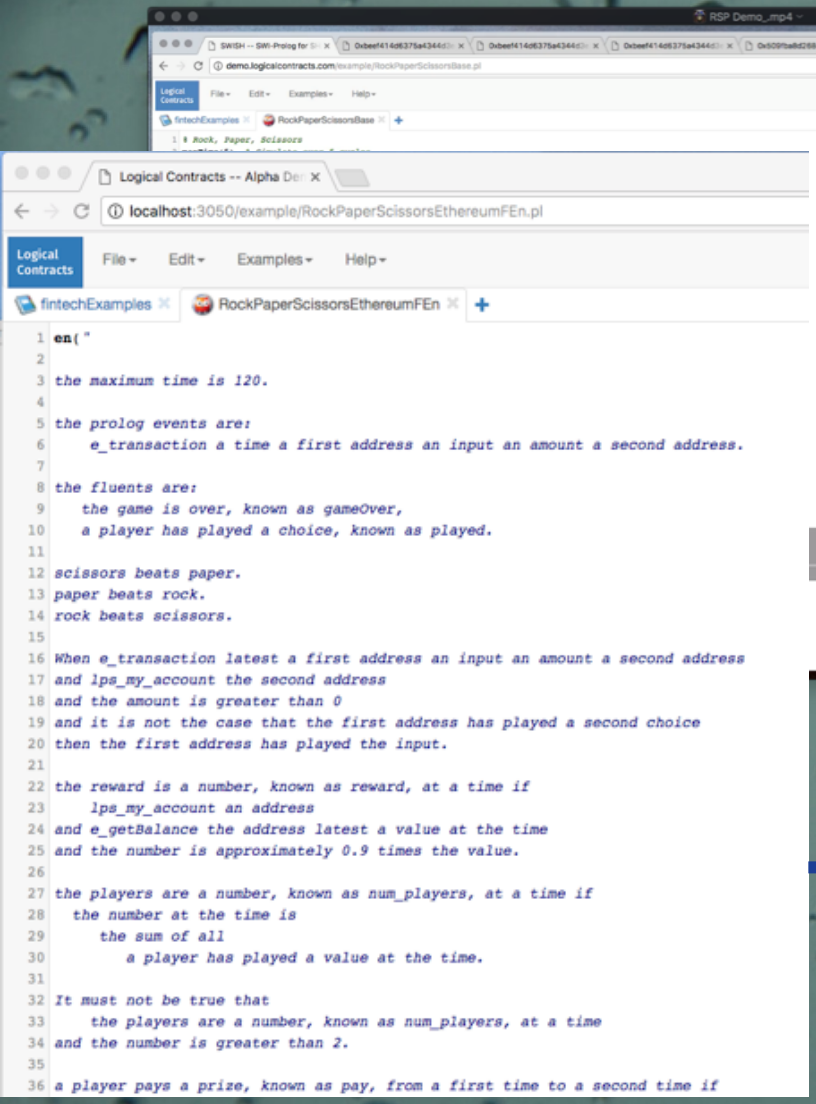
RSP on Ethereum blockchain

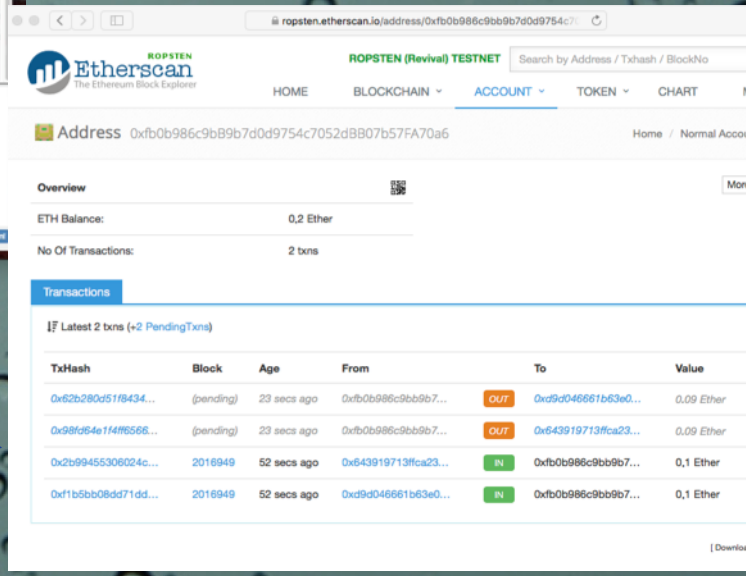
Logical Contracts

```
% Rock, Paper, Scissors gambling on Ethereum!
%  or in hexadecimal: '0x726f636b','0x7061706572', '0x736369736f7273'
maxRealTime(120). % 2 minutes game lifetime
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) if
    lps_my_account(To), Wei>0, not played(From,_).

fluents played(_Player,_Choice), gameOver.

reward(R) at T if    % intensional fluent obtained from the blockchain
    lps_my_account(A), e_getBalance(A,latest,V) at T,
    R is round(V*0.9). % keep 10% for gas

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.
```

RSP on Ethereum blockchain

Logical Contracts





Demo link at http://logicalcontracts.com/server/

Conclusion

- Demo and more info at http://logicalcontracts.com/server
  - LPS multi contract server, implemented with SWI Prolog and SWISH
  - Web editor, visualizations
  - Preliminary formal English, explanator
  - Web services (event injection, remote actions)
  - Hibernation (a contract can suspend and resume in another engine)
  - Ethereum logical API (via a local geth node)
  - Demo playground with Ethereum testnet accounts
- Upcoming
  - Pilot projects, to validate LPS for smart contracts
  - Formal English improvements, other languages
- Open source pledge☺
  - …except some dev tools and external interfaces

THANKS! mc@logicalcontracts.com