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](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 College London spinoff (*now courting investors*)
- Enhance and apply LPS to…
  - Smart contracts: a good fit
- Open source engine and basic environment
Sales pitch du jour...

Integrating enterprise systems with smart contracts
Imperative logic in plain English
PoC 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
maxTime(5). \# Simulate over 5 cycles

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

events transaction_from(_From, _Input, _Amount).

fluenets played(_Player, _Choice), reward(_Total), gameOver.

actions pay(_Player, _Prize).

initially reward(0).

\$ simulate input events:
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).

false transaction_from(_From, _Input, Wei), Wei \leq 0.
false transaction_from(From, _Input, _Wei), played(From, _).

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

false num_players(N), N > 2.

transaction_from(_Player, _, X) updates Old to New in reward(Old) if
New is Old + X.

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

if played(P0, Choice0) at T1, played(P1, Choice1) at T1, P0 \leq 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.
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

**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 all account balances and transactions
• Ethereum, Hyperledger, …
  – Can also contain and execute code (smart contracts): Solidity, JavaScript
• `ethereum` interface for LPS
  – LPS contracts in cloud servers 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) )`
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.
RPS on Ethereum blockchain

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

• Demo and more info at http://logicalcontracts.com/server
  – LPS contract server, implemented on SWI Prolog
  – Web editor, visualizations, on SWISH
  – Preliminary formal English, explanator
  – Web services (LPS event injection, remote LPS actions)
  – Hibernation (a contract can suspend and resume in another engine)
  – Ethereum logical API (via geth), LPS contracts as Ethereum accounts

• Upcoming
  – More examples from AI+Law literature: escrow, loans, …
  – Bridging to existing logics etc.
  – Pilot projects
  – Formal English improvements, other languages

• Open source pledge😊
  – Collaborations welcome

THANKS! mc@logicalcontracts.com