accordproject.org

Smart Legal Contracts: A Standardized Approach

Houman Shadab
Co-Director, the Accord Project
houman@accordproject.org

What is the Accord Project?

- Sets the legal and technical foundation for **smart legal contracts** by interfacing with leading lawyers, industry organizations, and technologists
- Addresses the lack of a common approach for smart legal contracts and the widely divergent, potentially incompatible, approaches that are emerging
- Producing **open source core** for smart legal contracts that embodies a collaborative techno-legal foundation and meets the needs of the legal industry

IN COLLABORATION WITH

Clio · IEEE · IACCM · ISO International Organization for Standardization · HYPERLEDGER

Working Groups

Supply Chain

MSAs and ancillary documents, tracking data standardization, upstream vs. downstream coordination, real-time system integration, secure data exchange, supply chain visibility, IoT standards

Financial Services

Real-time incorporation of market data, dynamic pricing and collateralization, fund structures, clearing and settlement infrastructure, claim types, coverage adjustment, use of telematics

Intellectual Property

Automating digital rights management, IP registration in a global database, automating the grant, refusal, termination, and assignment of IP, incorporation of real-time data about infringement

Venture and Token Sales

Automation and integration of investment documents and connection to milestones; integration with equity holding platforms; automation of various forms of blockchain token sales and governance frameworks

Dispute Resolution

Preventing and resolving disputes involving smart legal contracts; divergence between law and code; automated and distributed dispute resolution; smart ADR clauses; relationship with online dispute resolution.

Members and partners...

CLYDE&CO

legalzoom

DAMCO

Baker McKenzie.

KOCH INDUSTRIES INC

MIT Connection Science
the technology of innovation

LexisNexis

Bolero
SAFER, SMARTER AND FASTER GLOBAL TRADE

DENTONS

Mishcon de Reya

IEEE

SLAUGHTER AND MAY

Holland & Knight

Standard Chartered

C/M/S/

McDermott Will & Emery

BakerHostetler

NORTON ROSE FULBRIGHT

ashurst

Freshfields

Linklaters

...and many more.

Goals of the Accord Project

Open Source Community

Grow a community to develop freely available code, documentation, and other deliverables supporting the use of smart legal contracts globally across a wide variety of industries, use cases, and platforms. Subject to the Apache-2 license to ensure that individuals and companies have wide latitude in using the code for commercial, educational, and private purposes.

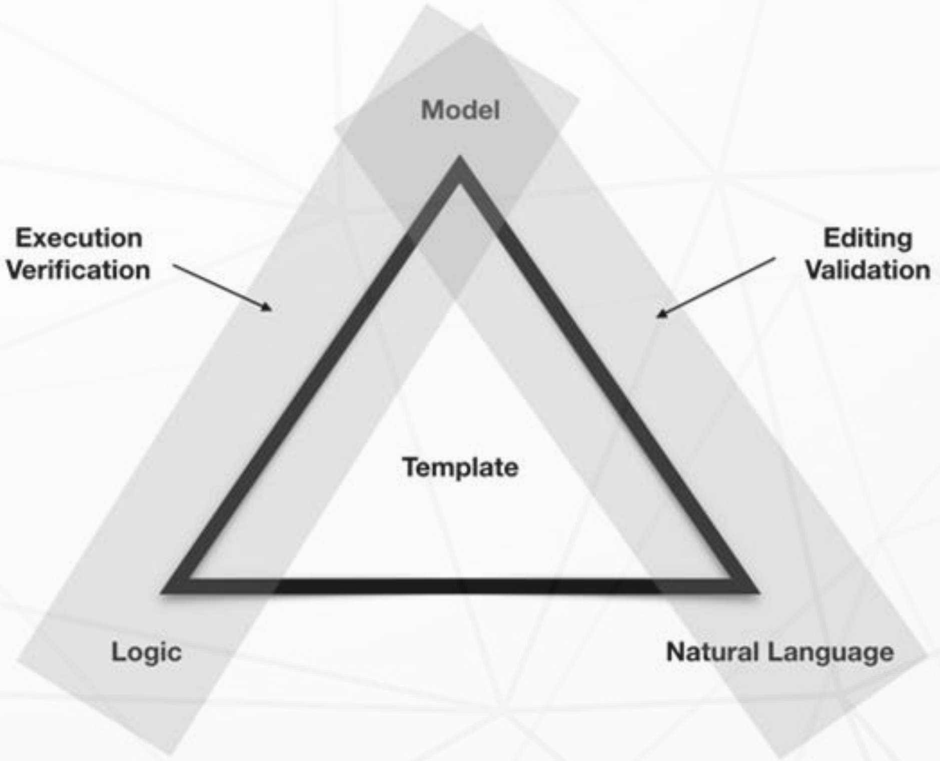
Smart Legal Contract Templating and Modeling

Develop a universally accessible and widely used open source library of modular smart legal contract and smart clause templates and models that reflect input from transactional attorneys and other experts that meets the needs of technology-enabled enterprises and specific business requirements. Built according to the Cicero specification.

Legal Contracting Language

Develop a domain specific language for smart legal contract execution that is accessible to non-technical professionals, compatible with a variety of execution targets such as SaaS platforms and distributed ledgers, and meets security, modularity, and other requirements. Built according to the Ergo language specification.

What is a Smart Legal Contract?



Built on Three Pillars

Legal Expertise

Technical Specifications

Open Source Software

Step-by-step

Progressive migration/evolution of existing legal and contract management practice:

1. Text

2. Text with digital signature

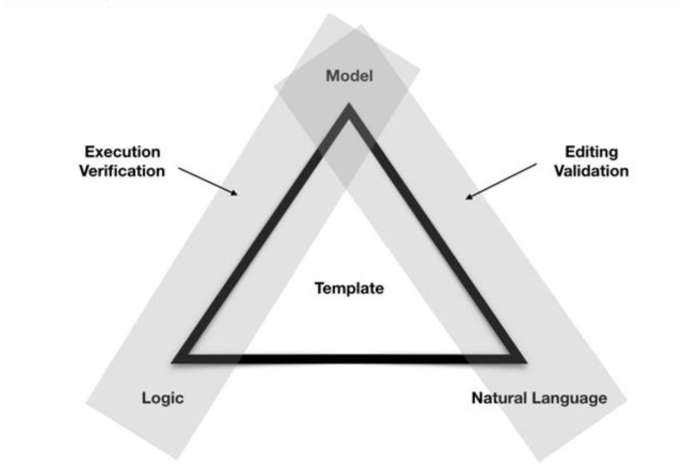
1. Text with variables (a model!) with digital signature

2. Text with variables and logic, with digital signature
   a. Automated handling of notifications and contract obligations

3. Distributed execution of contractual logic

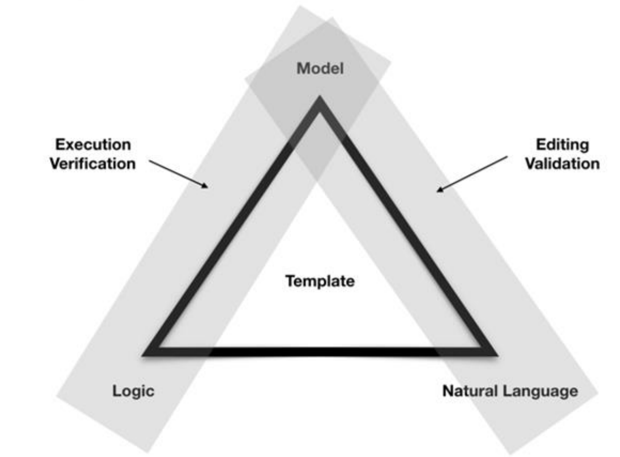
Natural Language

```
**Late Delivery and Penalty.** In case of delayed delivery[{" except for Force
Majeure cases,":? forceMajeure}] the Seller shall pay to the Buyer for every
[{penaltyDuration}] of delay penalty amounting to [{penaltyPercentage}]% of
the total value of the Equipment whose delivery has been delayed. Any
fractional part of a [{fractionalPart}] is to be considered a full
[{fractionalPart}]. The total amount of penalty shall not however, exceed
[{capPercentage}]% of the total value of the Equipment involved in late
delivery. If the delay is more than [{termination}], the Buyer is entitled to
terminate this Contract.
```



Model

```
concept SupplyModel {
 /** Does the clause include a force majeure provision? */
 o Boolean forceMajeure optional
 /* For every penaltyDuration that the goods are late */
 o Duration penaltyDuration
 /* Seller pays the buyer penaltyPercentage % of the value of the goods */
 o Double penaltyPercentage
 /** Up to capPercentage % of the value of the goods */
 o Double capPercentage
 /* If the goods are >= termination late then the buyer
    may terminate the contract */
 o Duration termination
 /* Fractional part of a ... is considered a whole ... */
 o TemporalUnit fractionalPart
}
```

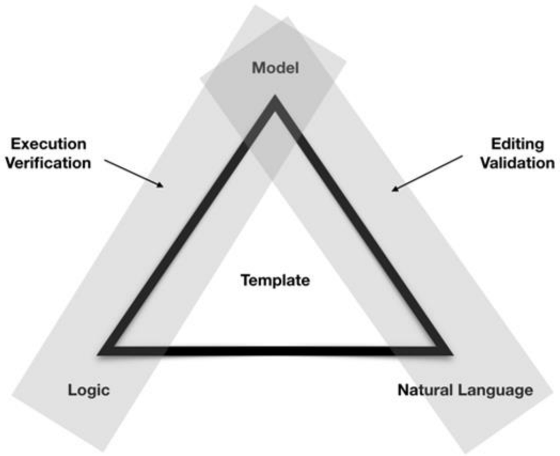


Logic

```
contract SupplyAgreement over SupplyModel {
  clause lateDeliveryAndPenalty(request: Request): Response {
   // Guard against force majeure
   enforce !contract.forceMajeure;
   define constant penalty =
     (diff / contract.penaltyDuration.amount)
        * contract.penaltyPercentage / 100.0 * request.goodsValue;

   // Penalty may be capped
   define constant capped =
    min([penalty,
         contract.capPercentage * request.goodsValue / 100.0]);

   // Return the response with the penalty
      and termination determination
   return Response {
     penalty: capped,
     buyerMayTerminate: diff > contract.termination.amount
   }
  }
}
```

ERGO

http://github.com/accordproject/ergo



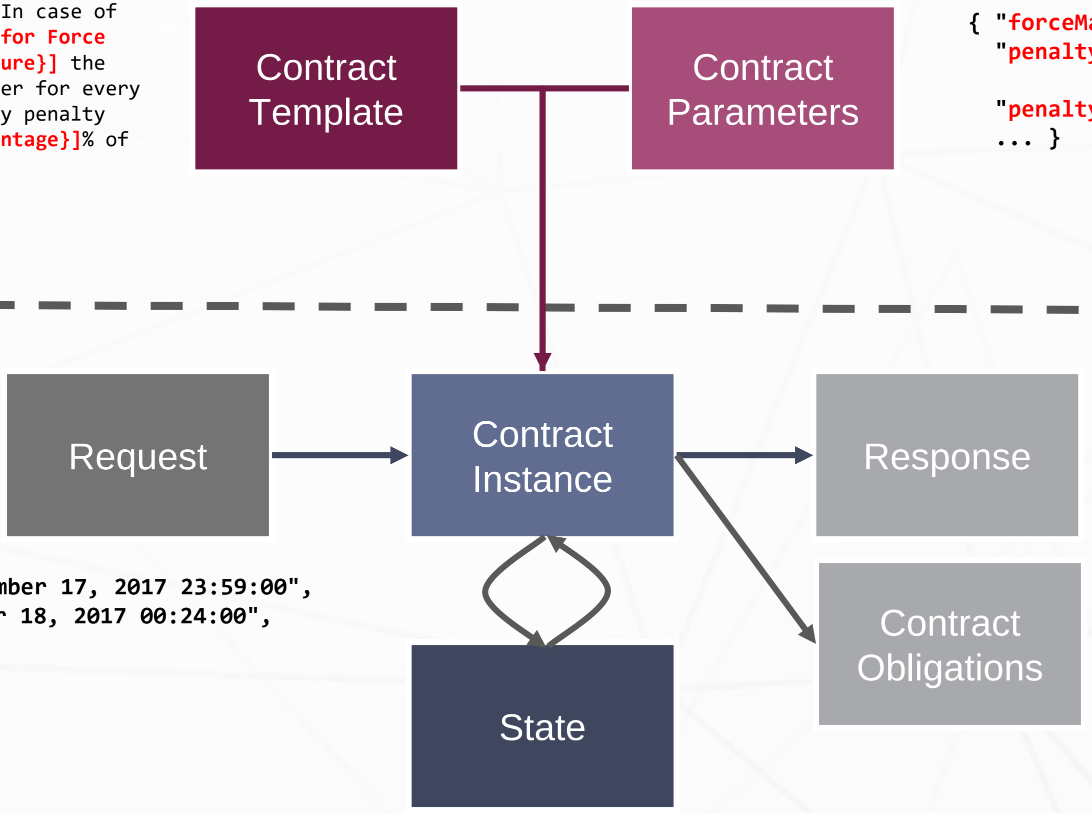
Programming Model

```
Late Delivery and Penalty. In case of
delayed delivery[{" except for Force
Majeure cases,":? forceMajeure}] the
Seller shall pay to the Buyer for every
[{penaltyDuration}] of delay penalty
amounting to [{penaltyPercentage}]% of
the ...
```

```
{ "forceMajeure" : false,
  "penaltyDuration" : { amount :2,
                        unit : “days” },
  "penaltyPercentage" : 10.5,
  ... }
```

Contract Creation

Contract Execution

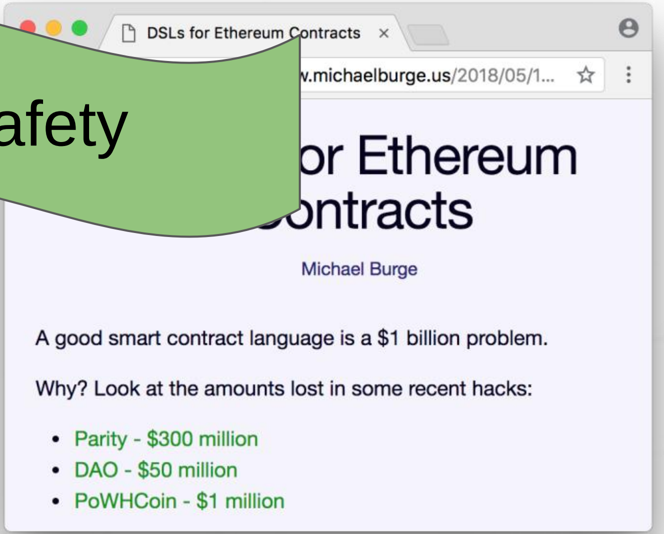


```
{ "penalty": 110.00000000000001,
  "buyerMayTerminate": true }
```

```
{ "agreedDelivery": "December 17, 2017 23:59:00",
  "deliveredAt": "December 18, 2017 00:24:00",
  "goodsValue": 200.00 }
```

Ergo’s Goals

Safety



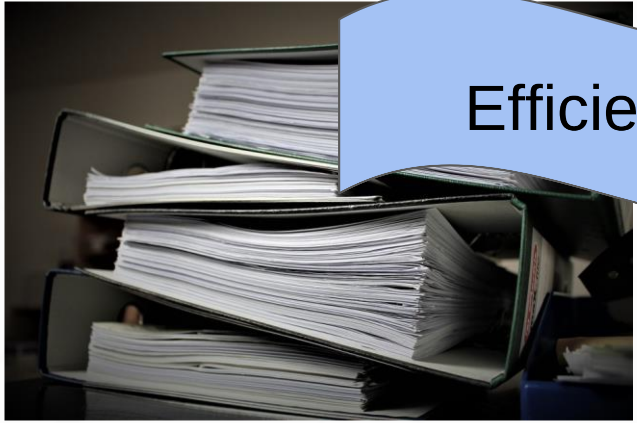
or Ethereum
ontracts

Michael Burge

A good smart contract language is a $1 billion problem.

Why? Look at the amounts lost in some recent hacks:

- Parity - $300 million
- DAO - $50 million
- PoWHCoin - $1 million



Efficiency

Apache v2 License

```
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and
distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright
owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities
that control, are controlled by, or are under common control with that entity.
For the purposes of this definition, "control" means (i) the power, direct or
indirect, to cause the direction or management of such entity, whether by
contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

                                                         ntity exercising

                                                         modifications, including
                                                         source, and configuration

                                                         anical transformation or
                                                         d to compiled object code,
                                                         a types.

                                                         Source or Object form, made
                                                         ght notice that is included
                                                         the Appendix below).

                                                         rce or Object form, that
                                                         he editorial revisions,
```

How to apply this license

Create a text file (typically named LICENSE or LICENSE.txt) in the root of your source code and copy the text of the license into the file.

Note: *The Apache Foundation recommends taking the additional step of adding a boilerplate notice to the header of each source file. You can find the notice at the very end of the license in the appendix.*

Required

- 🔵 License and copyright notice
- 🔵 State Changes

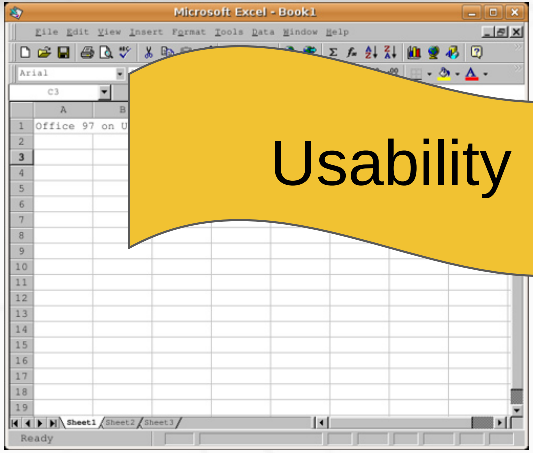
Permitted

- 🟢 Commercial Use
- 🟢 Modification
- 🟢 Distribution
- 🟢 Sublicensing
- 🟢 Patent Grant

Forbidden

- 🔴 Use Trademark
- 🔴 Hold Liable

Openness



Usability

Programming Model in Ergo

```
clause ::
  Request × State ⟶
( Response × State
          × Obligation[] )
| Error
```

```
clause late(req : LateRequest)
 : LateResponse {

 emit BillingObligation
   {amount: req.weeks * 5.0};

 enforce req.weeks > 0.0
 else throw CheatError{};

 set state PenaltyPaid{};

 return LateResponse{};
}

call late(LateRequest{weeks:2.0});
```

Why a New Language?

- *Domain specific* meant for legal contract logic
- Integral with Accord Project specification: CML and Templates
- Ease of use for legal-tech (template) developers
- Portable, compiles to various runtimes (e.g., nodejs) or DLTs (e.g., Fabric, EVM)
- Formally specified, no run-time errors, all contract calls terminate, deterministic
- Suitable for analysis & verification (contract property, cost bounds)
- “Modern language”: Distributed as Node.js package, Tooling (mode for various code editors, REPL), Documentation, Modularity, Error reporting, Performance…

Ergo Contracts as Classes

https://blog.colony.io/a-simple-agreement-for-future-tokens-or-equity-b8ef08608347

```
contract Safte over SafteContract {
  clause tokenSale(request : TokenSale) : TokenShare {
    let discountRate = (100.0 - contract.discount) / 100.00;
    let discountPrice = request.tokenPrice * discountRate;
    return TokenShare{ tokenAmount : contract.purchaseAmount / discountPrice }
  }

  clause equityFinancing(request : EquityFinancing) : EquityShare {
    let discountRate = (100.0 - contract.discount) / 100.00;
    let discountPrice = request.sharePrice * discountRate;
    return EquityShare{ equityAmount : contract.purchaseAmount / discountPrice }
  }

  clause disolutionEvent(request : DissolutionEvent) : PayOut {
    return PayOut{ amount : contract.purchaseAmount }
  }
}
call dissolutionEvent(DissolutionEvent{ cause : "Cold feet" });
call tokenSale(TokenSale{ tokenPrice: 3.14 });
call equityFinancing(EquityFinancing{ sharePrice: 2.98 });
```

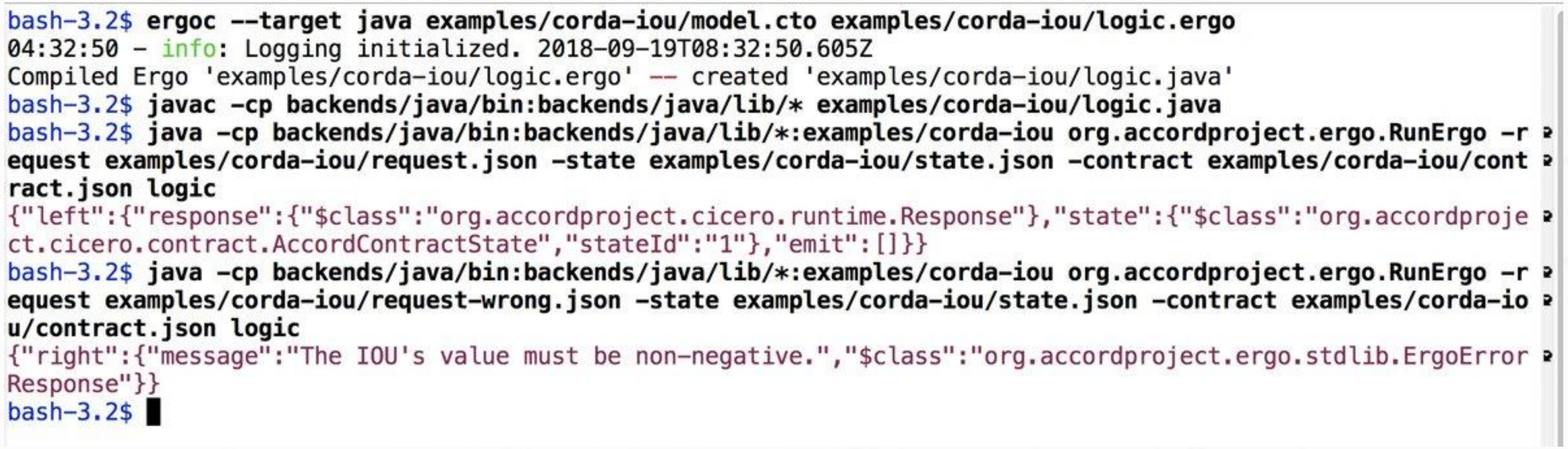
Ergo Contracts as Rules

https://blog.colony.io/a-simple-agreement-for-future-tokens-or-equity-b8ef08608347

```
contract Safte over SafteContract
  rule tokenSale when TokenSale do
    let discountRate = (100.0 - contract.discount) / 100.00;
    let discountPrice = request.tokenPrice * discountRate;
    return TokenShare{ tokenAmount : contract.purchaseAmount / discountPrice }
  ;

  rule equityFinancing when EquityFinancing do
    let discountRate = (100.0 - contract.discount) / 100.00;
    let discountPrice = request.sharePrice * discountRate;
    return EquityShare{ equityAmount : contract.purchaseAmount / discountPrice }
  ;

  rule disolutionEvent when DissolutionEvent do
    return PayOut{ amount : contract.purchaseAmount }
  ;

send DissolutionEvent{ cause : "Cold feet" };
send TokenSale{ tokenPrice: 3.14 };
send EquityFinancing{ sharePrice: 2.98 };
```

Blockchain Agnostic

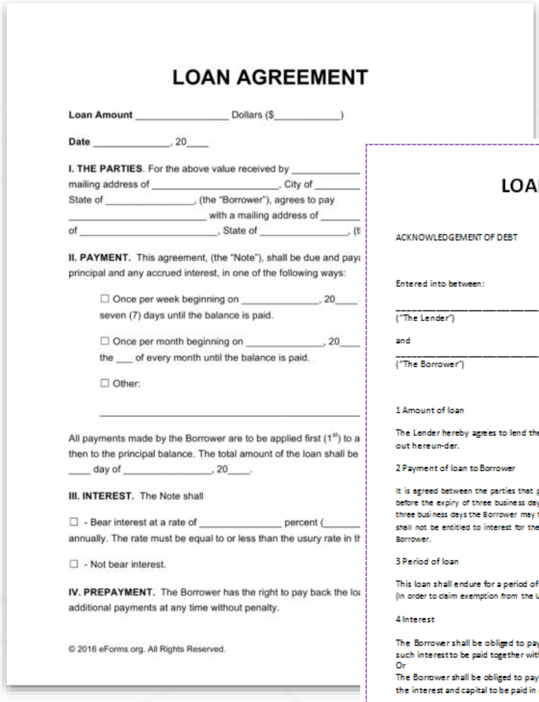
Corda blockchain's IOU implemented as logic in Ergo



```
bash-3.2$ ergoc --target java examples/corda-iou/model.cto examples/corda-iou/logic.ergo
04:32:50 - info: Logging initialized. 2018-09-19T08:32:50.605Z
Compiled Ergo 'examples/corda-iou/logic.ergo' -- created 'examples/corda-iou/logic.java'
bash-3.2$ javac -cp backends/java/bin:backends/java/lib/* examples/corda-iou/logic.java
bash-3.2$ java -cp backends/java/bin:backends/java/lib/*:examples/corda-iou org.accordproject.ergo.RunErgo -r
equest examples/corda-iou/request.json -state examples/corda-iou/state.json -contract examples/corda-iou/cont
ract.json logic
{"left":{"response":{"$class":"org.accordproject.cicero.runtime.Response"},"state":{"$class":"org.accordproje
ct.cicero.contract.AccordContractState","stateId":"1"},"emit":[]}}
bash-3.2$ java -cp backends/java/bin:backends/java/lib/*:examples/corda-iou org.accordproject.ergo.RunErgo -r
equest examples/corda-iou/request-wrong.json -state examples/corda-iou/state.json -contract examples/corda-io
u/contract.json logic
{"right":{"message":"The IOU's value must be non-negative.","$class":"org.accordproject.ergo.stdlib.ErgoError
Response"}}
bash-3.2$ █
```

Future Work: Contract Composition

- Most contracts include various standard “reusable” or “boilerplate” clauses
- Examples: Installment payments, interest calculations, jurisdiction, etc.
- What is the right model to compose clauses in Ergo?
  - Clauses = Traits?
  - Clauses = Rules?



LOAN AGREEMENT

Loan Amount ________________ Dollars ($____________)

Date _______________, 20____

I. THE PARTIES. For the above value received by ____________
mailing address of _________________________, City of __________
State of _________________, (the “Borrower”), agrees to pay
___________________________ with a mailing address of _________
of ___________________________, State of _________________, (t

II. PAYMENT. This agreement, (the “Note”), shall be due and paya
principal and any accrued interest, in one of the following ways:

☐ Once per week beginning on _______________, 20____
seven (7) days until the balance is paid.

☐ Once per month beginning on _______________, 20____
the ___ of every month until the balance is paid.

☐ Other:

____

All payments made by the Borrower are to be applied first (1st) to a
then to the principal balance. The total amount of the loan shall be
____ day of _________________, 20____.

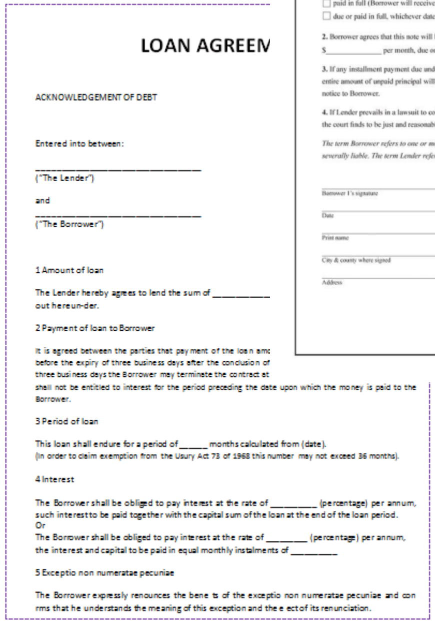
III. INTEREST. The Note shall

☐ - Bear interest at a rate of ________________ percent (________
annually. The rate must be equal to or less than the usury rate in th

☐ - Not bear interest.

IV. PREPAYMENT. The Borrower has the right to pay back the loa
additional payments at any time without penalty.

© 2016 eForms.org. All Rights Reserved.



LOAN AGREEM

ACKNOWLEDGEMENT OF DEBT

Entered into between:

____
("The Lender")

and

____
("The Borrower")

1 Amount of loan

The Lender hereby agrees to lend the sum of ___________
out hereun-der.

2 Payment of loan to Borrower

It is agreed between the parties that pay ment of the loa n amc
before the expiry of three business days after the conclusion of
three busi ness days the Borrower may terminate the contract at
shall not be entitled to interest for the period preceding the date upon which the money is paid to the Borrower.

3 Period of loan

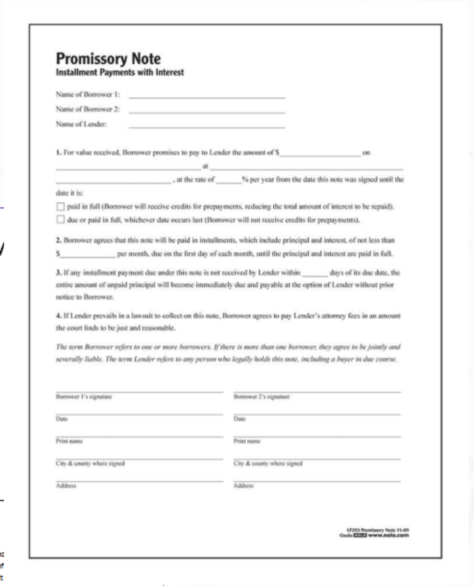
This loan shall endure for a period of______ months calculated from (date).
(In order to claim exemption from the Usury Act 73 of 1968 this number may not exceed 36 months).

4 Interest

The Borrower shall be obliged to pay interest at the rate of _________ (percentage) per annum, such interest to be paid together with the capital sum of the loan at the end of the loan period.
Or
The Borrower shall be obliged to pay interest at the rate of ________ (percentage) per annum, the interest and capital to be paid in equal monthly instalments of _________

5 Exceptio non numeratae pecuniae

The Borrower expressly renounces the bene ts of the exceptio non numeratae pecuniae and con rms that he understands the meaning of this exception and the e ect of its renunciation.



Promissory Note
Installment Payments with Interest

Name of Borrower 1: ______________________________
Name of Borrower 2: ______________________________
Name of Lender: ______________________________

1. For value received, Borrower promises to pay to Lender the amount of $__________________ on ______________________________ at ______________________________________ ________________________, at the rate of ______% per year from the date this note was signed until the date it is:

☐ paid in full (Borrower will receive credits for prepayments, reducing the total amount of interest to be repaid).
☐ due or paid in full, whichever date occurs last (Borrower will not receive credits for prepayments).

2. Borrower agrees that this note will be paid in installments, which include principal and interest, of not less than $____________ per month, due on the first day of each month, until the principal and interest are paid in full.

3. If any installment payment due under this note is not received by Lender within ______ days of its due date, the entire amount of unpaid principal will become immediately due and payable at the option of Lender without prior notice to Borrower.

4. If Lender prevails in a lawsuit to collect on this note, Borrower agrees to pay Lender’s attorney fees in an amount the court finds to be just and reasonable.

The term Borrower refers to one or more borrowers. If there is more than one borrower, they agree to be jointly and severally liable. The term Lender refers to any person who legally holds this note, including a buyer in due course.

Borrower 1’s signature	Borrower 2’s signature
Date	Date
Print name	Print name
City & county where signed	City & county where signed
Address	Address

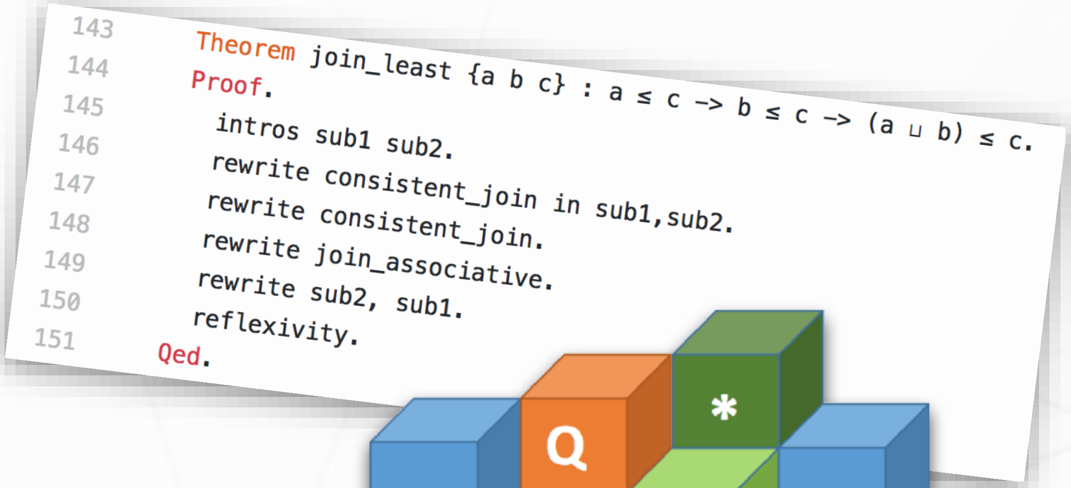
LF293 Promissory Note 11-09
www.nolo.com

Future Work: More on Verification

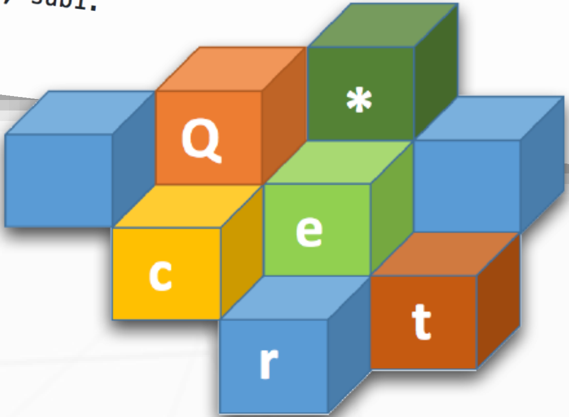
Typed Ergo programs should
(a) always terminate
(b) without any runtime errors

The Good News

Ergo is written in Coq, and built on Q*Cert which gives us:
- Data model
- Type foundations
- Optimization framework
- Proofs!



```
Theorem join_least {a b c} : a ≤ c -> b ≤ c -> (a ⊔ b) ≤ c.
Proof.
  intros sub1 sub2.
  rewrite consistent_join in sub1,sub2.
  rewrite consistent_join.
  rewrite join_associative.
  rewrite sub2, sub1.
  reflexivity.
Qed.
```



accord-project.slack.com

@accordhq

www.accordproject.org

houman@accordproject.org