Smart digital contracts:
Contract analysis and some open problems

Fritz Henglein

Email: henglein@diku.dk, henglein@deondigital.com

OPLSS 2019, 2019-06-27
Recall

Agents: Persons, companies, robots, devices that sign events and their evidence.
Events: Significant real-world events that update the state of the (business) world.
Resources: Physical (goods, services) and digital (money, rights) resources, represented by free vector space over resource types.
Ownership states: Map of which owner owns which (compound) resource, represented by coproduct of resource space indexed by agents.
Resource transfers: Changes to ownership states that sum to 0.
Resource manager: System that maintains ownership states subject to credit limit policy (admissible ownership states), admitting only updates by resource transfers.
Contract: Set of happy paths (acceptable event sequences).
Contract specification: A syntactic object denoting a contract.
Contract specification language: Language for contract specifications.
Contract (life-cycle) management: Program that receives a contract specification and then processes a stream of events in accordance with the semantics of the contract specification.
Today

- Contract analysis
- Some exercises and open problems
**Contract properties**

**Definition**

A *contract property* is a predicate (Boolean function) $P$ on contract specifications.

It is *extensional* if $C[C_1] = C[C_2] \implies (P(C_1) \iff P(C_2))$.

It is *universal* if $P(C) \iff \forall s \in C[C].P'(s)$ for some predicate $P'$ on event sequences.
Contract verification and contract analysis

- Contract verification: Given $C$ and $P$, decide whether $P(C)$ holds. If possible, provide a proof of $\vdash P(C)$, respectively $\vdash \neg P(C)$.
- Contract analysis: Given $C$ and $P$, compute a "good" witness $W$ such that $P(W)(c)$ holds.
Contract analysis: First events

Definition

\[ \text{First}(c) = \{ e \in E \mid \exists s \in E^*. es \in c \} \]

Analysis problem: Given contract specification \( C \) compute a (useful representation of) set \( F \) such that \( \text{First}(\llbracket C \rrbracket) \subseteq F \).

Exercise: How would you do this for CSL contract specifications?
Contract analysis: Fair consideration

Definition

Let transfer effect function \( \text{eff} \) be given. Given valuation function \( \text{Price} : X \rightarrow \mathbb{R} \), the value of a sequence of events \( s \in E^* \) is

\[
\text{Value}(s) = \text{Price}^*(\text{eff}(s)) \in \sum_{A} \mathbb{R}.
\]

A contract \( c \) is \( \epsilon\)-fair under \( \text{Price} \) if

\[
\forall s \in c, a \in A. \ |\text{Value}(s)(a)| \leq \epsilon.
\]

A contract specification \( C \) is \( \epsilon\)-fair under \( \text{Price} \) if \( C[C] \) is so.
Contract analysis: Fair consideration

Why important?

An $\epsilon$-fair contract, where $Price$ reflects market prices, guarantees that no happy path leads to a state where an agent has paid or received significantly more than any other in terms of the market value of exchanged goods, services and money.

Combined with transactional (escrowed) contract execution, this guarantees that neither happy paths nor unhappy paths lead to disproportionate benefits/losses for any agent.

Cannot be a built-in property of contracts since it depends on $Price$, which reflects context-dependent assumptions (market prices) about unit values of resource types.
Linear resources and linear logic

- Linear logic is called a resource logic: Assumptions are “used up” by applications of modus ponens.
- Transfers guarantee that no resources are lost or duplicated; they are treated “linearly”.

Intuitively, there is a connection between these “linearities”. What is it?

- Can sequents be interpreted as transfers and linear logic inference rules as (particular) linear maps on transfers?
- Linear logic has no scalar multiplication. Can linear logic be extended to admit meaningful “counting” operations; e.g. add $k \cdot P$ instead of $!P$. For $k \in \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$, any field $K$?
Linear logic: Structural rules

**Exchange**

\[ \vdash \Gamma, B, A, \Delta \]
\[ \vdash \Gamma, A, B, \Delta \]

**Init**

\[ \vdash A, A \perp \]

**Cut**

\[ \vdash \Gamma, A \quad A \perp, \Delta \]
\[ \vdash \Gamma, \Delta \]
Linear logic: Multiplicative rules

\[
\begin{array}{c}
\text{TENSOR} \\
\Gamma, A \vdash \Delta, B \\
\hline \\
\Gamma, A \otimes B \\
\text{TENSORUNIT} \\
\Gamma \vdash \Gamma, \bot \\
\text{PAR} \\
\Gamma, A, B \vdash \Delta, B \\
\hline \\
\Gamma, A \otimes B \\
\text{PARUNIT} \\\n\Gamma \vdash \Gamma, \bot
\end{array}
\]
Linear logic: Additive rules

\[ \Gamma, A \vdash \quad \Gamma, B \vdash \quad \Gamma, A \& B \]

**Plus1**
\[ \Gamma, A \vdash \quad \Gamma, A \oplus B \]

**Plus2**
\[ \Gamma, B \vdash \quad \Gamma, A \oplus B \]

**Top**
\[ \Gamma \vdash \top \]
Wrap-up
Smart contracts

- Ethereum-style smart contracts:
  - Current standard understanding of term “smart contract”.
  - Contract specification, contract management and resource management combined and expressed in single-threaded program expressed in general-purpose programming language (EVM).
  - Implementation as decentralized replicated state machine, where each replica stores full state.
  - Distributed consensus on total event order of all events across all contracts is required and computed.
Smart digital contracts: Concepts

- Contract: A set of event sequences ("happy paths").
- Contract specification: Syntactic object that denotes a contract (reified contract).
- CSL: Compositional contract specification language with multiple induction principles and supporting equational reasoning.
- Contract manager: System/service ("generic smart contract") that manages a set of contracts passed to it for management.
- Resource: Finite map from arbitrary resource types to number of units of each type.
- Resource manager: System that manages ownership of resources by agents, admits only resource transfers.
- Resource transfer: Ownership changes that guarantee that no resources are duplicated or lost.
Smart digital contracts: Separation of concerns

- Separation of contract and contract manager:
  - Separation of concerns: reified contracts (“digital contracts”) and their flexible, intelligent management (“smart”).
  - Change contracts and contract managers independent of each other.
  - Analyze contracts (written in DSL with mathematical semantics supporting compositional, equational reasoning and having multiple, useful induction principles) independent of their managers (written in any arbitrarily expressive and complex general-purpose languages).
  - Transparently port running contracts among contract managers.

- Separation of resource managers and contract managers: Facilitates
  - privacy and scalability of contract management (each contract managed independently; synchronize only through resource managers; consensus on total event order neither required nor computed);
  - transactional use on multiple resource managers, both decentralized (blockchain/distributed ledger) and centralized (server-based, cloud-hosted) systems;
  - scalable, distributed resource managers by additive (de)composition.
Thank you!

Fritz Henglein
henglein@diku.dk
henglein@deondigital.com