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 $\mathcal{C}[\![C_1]\!] = \mathcal{C}[\![C_2]\!] \implies (P(C_1) \Leftrightarrow P(C_2))$.
It is *universal* if $P(C) \Leftrightarrow \forall s \in \mathcal{C}[\![C]\!].P'(s)$ 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

$\mathrm{First}(c) = \{e \in E \mid \exists s \in E^*.\, es \in c\}$

Analysis problem: Given contract specification C compute a (useful representation of) set F such that $\mathrm{First}(\mathcal{C}[\![C]\!]) \subseteq F$.

Exercise: How would you do this for CSL contract specifications?

Contract analysis: Fair consideration

Definition

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

$$\mathrm{Value}(s) = \mathrm{Price}^*(\mathrm{eff}(s)) \in \sum_A \mathbb{R}.$$

A contract c is *ϵ-fair under* Price if

$$\forall s \in c, a \in A. \, |\mathrm{Value}(s)(a)| \leq \epsilon.$$

A contract specification C is *ϵ-fair under* Price if $\mathcal{C}[\![C]\!]$ is so.

Contract analysis: Fair consideration

- Why important?
- An ϵ-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

$$\frac{\vdash \Gamma, B, A, \Delta}{\vdash \Gamma, A, B, \Delta}$$

INIT

$$\vdash A, A^{\perp}$$

CUT

$$\frac{\vdash \Gamma, A \qquad A^{\perp}, \Delta}{\vdash \Gamma, \Delta}$$

Linear logic: Multiplicative rules

TENSOR

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}$$

TENSORUNIT

$$\vdash 1$$

PAR

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}$$

PARUNIT

$$\frac{\vdash \Gamma}{\vdash \Gamma, \bot}$$

Linear logic: Additive rules

WITH

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \mathbin{\&} B}$$

PLUS1

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}$$

PLUS2

$$\frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}$$

TOP

$$\vdash \Gamma, \top$$

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.

Finis

Thank you!

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