Smart digital contracts: Algebraic foundations for resource accounting

Fritz Henglein

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

OPLSS 2019, 2019-06-26
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

- Business events: Transmission of information and other events whose resource effect is idempotent (e.g. queries)
- Resource events: Producing (transforming) and transferring resources, which have a resource effect (who owns or possesses what)

Resources: Physical (goods, services) or digital (money, rights) resources that cannot/must not be freely copied and discarded

Contract: A classifier of event sequences into “happy” paths (correct contract executions) and “breaches” (incorrect contract executions).
Today

- Algebraic model of resources, with user-definable resource types ("multi-currency")
- Resource ownership via coproducts
- Resource transfers via kernels
- Operations and properties: vector space operations and basic linear algebra
Vector spaces

Definition

- Field: \((K, +, -, 0, \cdot, /, 1)\), commutative ring with multiplication and division
- Vector space over \(K\): \((V, +, -, 0, \cdot)\), usual properties
- Dimension of vector space: Cardinality of smallest subset of \(V\) that spans all of \(V\)

Example

The reals \(\mathbb{R}\) are a field and simultaneously a vector space of dimension 1 over itself.
Vector space constructions

Let $V_x$ be vector spaces.

\[ \prod_{x \in X} V_x \text{ (product): Functions } f \text{ from } x : X \text{ to } V_x \]

\[ \coprod_{x \in X} V_x \text{ (coproduct): Functions } f \text{ from } x : X \text{ to } V_x \text{ with finite support } \]
\[ \text{Supp}(f) = \{x \mid f(x) \neq 0\}; \text{ that is, finite maps with default return value 0.} \]

\[ V \to_1 W \text{ (linear map space): Functions (linear maps) } f \text{ from } V \text{ to } W \]
\[ \text{such that } f(v_1 + v_2) = f(v_1) + f(v_2) \text{ and } f(k \cdot v) = k \cdot f(v). \]

\[ U \subseteq V \text{ (subspace): Subset } U \text{ of } V \text{ that is closed under } 0, +, -, \cdot \]

If $V_x = V$ for all $x \in X$, write

\[ \prod_{X} V = \prod_{x \in X} V \]
\[ \coprod_{X} V = \coprod_{x \in X} V \]
Vector space constructions: Examples

Let \( X \) be a set.

\( \mathcal{V}_1 \oplus \mathcal{V}_2 \) (direct sum): \( \bigsqcup_{x \in \{1,2\}} \mathcal{V}_i (= \mathcal{V}_1 \times \mathcal{V}_2) \)

Free\(_K(X)\) (free vector space): \( \bigsqcup_X K \)

\( \sum : (\bigsqcup_X \mathcal{V}) \rightarrow_1 \mathcal{V} \) (sum, addition):
\[
\sum(\{x_1 : v_1, \ldots, x_n : v_n\}) = v_1 + \ldots + v_n
\]

\( p^* : \text{Free}_K(X) \rightarrow_1 K \) (valuation under price \( p : X \rightarrow K \)): Unique extension of \( p \) to \( \text{Free}_K(X) \).

\( \ker f \subseteq \mathcal{V} \) (kernel of \( f : \mathcal{V} \rightarrow \mathcal{W} \)): \( \{x \in \mathcal{V} \mid f(x) = 0\} \).

\( \text{im } f \subseteq \mathcal{W} \) (image of \( f : \mathcal{V} \rightarrow \mathcal{W} \)): \( \{f(x) \mid x \in \mathcal{V}\} \).
Vector space constructions: Examples of examples

- $(5, 8) \in \mathbb{R} \oplus \mathbb{R} = \mathbb{R}^2$
- $5 \cdot X_1 + 8 \cdot X_2 = \{X_1 : 5, X_2 : 8\} \in \mathbb{R} \bigoplus \{x_1, x_2\}$
- $\sum\{X_1 : 5, X_2 : 8\} = 5 + 8 = 13$
- $p^* (\{X_1 : 5, X_2 : 8\}) = 4 \cdot 5 + 3 \cdot 8 = 44$
  for $p(X_1) = 4$, $p(X_2) = 3$.
- $\ker p^* = \{\{X_1 : x_1, X_2 : x_2\} \mid 4 \cdot x_1 + 3 \cdot x_2 = 0\}$.
- $\text{im } p^* = \mathbb{R}$. 


Agents and resources

Agents $A$: A set. $A = \{\text{Alice, Bob, Charlie, ...}\}$.

Resource types $X$: A set. $X = \{\text{USD, iPhone, ...}\}$.

Resources $R$: A vector space. $R = \bigsqcup_X \mathbb{R}$

Ownership states $O$: A vector space. $O = \bigsqcup_A R$

Transfers $T$: Subspace of $O$. $T = \sum_X R = \ker(\sum : \bigsqcup_A R \to 1 R)$

**Example**

- A *simple* resource: $50 \cdot \text{USD}$
- A *compound* resource: $50 \cdot \text{USD} + 2 \cdot \text{iPhone}$
- A *missing* resource is also a resource: $-50 \cdot \text{USD}$
- An ownership state: $\{\text{Alice} : 50 \cdot \text{USD}, \text{Bob} : 1 \cdot \text{iPhone} + 10 \cdot \text{USD}\}$
- A *simple* (2-party) transfer: $\{\text{Alice} : -30 \cdot \text{USD}, \text{Bob} : 30 \cdot \text{USD}\}$
- A *compound* (multi-party) transfer: $\{\text{Alice} : -30 \cdot \text{USD}, \text{Bob} : 20 \cdot \text{USD}, \text{Charlie} : 10 \cdot \text{USD}\}$
Resource manager

- Credit limit policy: Predicate (Boolean function), classifying ownership states into valid and invalid ones
  - Usually: $P_{A_0,c}(o) = o(a) \geq c(a)$ for all $a \in A_0$ where $A_0 \subseteq A$.

- Resource manager: Object (service) with
  - Internal state $o$: An ownership state satisfying credit limit policy $P$.
  - Method ApplyTransfer:
    Receive transfer $t$.
    If $P(o + t)$, update internal state to $o + t$ and return “success”; otherwise, return “failure”.

Example

<table>
<thead>
<tr>
<th>Credit limit policy:</th>
<th>No credit (no negative amounts of any resource type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial ownership:</td>
<td>$o_1 = {\text{Alice} : 50 \cdot \text{USD}, \text{Bob} : 1 \cdot \text{iPhone} + 10 \cdot \text{USD}}$</td>
</tr>
<tr>
<td>First transfer:</td>
<td>$t_1 = {\text{Alice} : -30 \cdot \text{USD}, \text{Bob} : 30 \cdot \text{USD}}$</td>
</tr>
<tr>
<td>Second transfer:</td>
<td>$t_2 = {\text{Alice} : 1 \cdot \text{iPhone}, \text{Bob} : -1 \cdot \text{iPhone}}$</td>
</tr>
<tr>
<td>Combined transfer:</td>
<td>${\text{Alice} : 1 \cdot \text{iPhone} - 30 \cdot \text{USD}, \text{Bob} : -(1 \cdot \text{iPhone} - 30 \cdot \text{USD})}$</td>
</tr>
<tr>
<td>Final ownership:</td>
<td>$o_2 = {\text{Alice} : 1 \cdot \text{iPhone} + 20 \cdot \text{USD}, \text{Bob} : 40 \cdot \text{USD}}$</td>
</tr>
</tbody>
</table>
Ownership state as balance plus transfer

**Theorem**

Let $f : V \rightarrow W$. Then:

$$V \cong \text{im } f \oplus \text{ker } f$$

$$\dim V = \dim(\text{im } f) + \dim(\text{ker } f).$$

**Corollary**

$$O = \bigoplus_{A} R \cong R \oplus \sum_{A} R = R \oplus T$$

Intuitively: Ownership state $\cong$ a *resource balance* owned by one particular agent $b \in A$ and some transfer; for example:

$$o = \{\text{Bank} : 60 \cdot \text{USD}, \text{Alice} : 30 \cdot \text{USD}, \text{Bob} : 40 \cdot \text{USD}\}$$

$$= \{\text{Bank} : 130 \cdot \text{USD}\} +$$

$$\{\text{Bank} : -70 \cdot \text{USD}, \text{Alice} : 30 \cdot \text{USD}, \text{Bob} : 40 \cdot \text{USD}\}$$
Resource manager properties

- A multiset $M = \{t_1, \ldots, t_n\}$ of transfers can be applied by a resource manager in any order: any two orders that succeed result in the same ownership state. Some orders may fail, however, due to the resource manager’s credit limit policy.

- If there is some successful order of applying $M$ satisfying $P$, then applying the single “netted” transfer $t = \sum M = \sum_{i=1}^{n} t_i$ is valid, too. The converse is not true.

- The internal ownership state can be stored as a pair, a balance and a transfer.

- The balance component in a resource manager is invariant. Only the transfer component is updated by ApplyTransfer.
Zero-balance resource managers

- Balance of a resource manager can be kept in another resource manager.
- *Zero-balance resource manager*: internal state of resource manager consists of a transfer only; resource balance component is implicitly 0.
Zero-balance resource managers: Example

Two resource managers:

\[
\begin{align*}
\sigma_1 &= \{\text{Bank}_1: 60 \cdot \text{USD}, \text{Alice}: 30 \cdot \text{USD}, \text{Bob}: 40 \cdot \text{USD}\} \\
&= \{\text{Bank}_1: 130 \cdot \text{USD}\} + \\
&\quad \{\text{Bank}_1: -70 \cdot \text{USD}, \text{Alice}: 30 \cdot \text{USD}, \text{Bob}: 40 \cdot \text{USD}\} \\
\sigma_2 &= \{\text{Bank}_2: 10 \cdot \text{USD}, \text{Alice}: 100 \cdot \text{USD}, \text{Bob}: 200 \cdot \text{USD}\} \\
&= \{\text{Bank}_2: 310 \cdot \text{USD}\} + \\
&\quad \{\text{Bank}_1: -300 \cdot \text{USD}, \text{Alice}: 100 \cdot \text{USD}, \text{Bob}: 200 \cdot \text{USD}\}
\end{align*}
\]

Replace by three resource managers maintaining transfers only:

\[
\begin{align*}
t_1 &= \{\text{Bank}_1: -70 \cdot \text{USD}, \text{Alice}: 30 \cdot \text{USD}, \text{Bob}: 40 \cdot \text{USD}\} \\
t_2 &= \{\text{Bank}_1: -300 \cdot \text{USD}, \text{Alice}: 100 \cdot \text{USD}, \text{Bob}: 200 \cdot \text{USD}\} \\
t_0 &= \{\text{Bank}_0: -440 \cdot \text{USD}, \text{Bank}_1: 130 \cdot \text{USD}, \text{Bank}_2: 310 \cdot \text{USD}\}
\end{align*}
\]

where $\text{Bank}_0$ is another agent, corresponding to the central bank in the banking system or the equity account in a company's chart of accounts. Note: $\{\text{Bank}_0: -\sum (\sigma_1 + \sigma_2)\} + \sigma_1 + \sigma_2 = t_0 + t_1 + t_2$ is a transfer.
Double-entry bookkeeping

Fundamental principle of double-entry bookkeeping:

- All (scalar) account ($\cong$ agent) balances sum to 0.
- Every transaction consists of multiple ("double") account entries that sum to 0.

“Equity” plays role of resource balance when decomposing ownership state into resource balance and transfer satisfying

\[ \text{Assets} - \text{Liabilities} - \text{Equity} = 0 \]
Resource accounting

Resource accounting: Double-entry bookkeeping, generalized to admit

- arbitrary resources, not just scalars, with
- expressive algebra (vector space) of transfers that are not composed from possibly incorrect adding/subtracting to/from account balances, but from a base of simple transfers; and
- arbitrary report functions on internal state,
  ▶ often linear maps on internal ownership states or on sequences of transfers $T^*$, and then
  ▶ easily incrementalized to maintain report function results online (dynamically) as new transfers arrive.

A resource manager (implemented whichever way) provides digital resource management for arbitrary (including user-defined) resource types.

- Updating by transfers only guarantees resource preservation: No managed resource is duplicated or lost.
- Credit limit enforcement by checking of credit limit policy.
Distributed resource managers by additive decomposition

- Idea: Implement *distributed resource manager* $r$ by a P2P network of resource managers $r_1, \ldots, r_n$ such that $r.o = r_1.o + \ldots + r_n.o$.

- The $r_i$ may be distributed themselves. Advantages:
  - Some transfers can be performed *locally*: If $r_i$ can validate and effect a transfer $t$, then no communication with other resource managers is necessary.\(^1\)
  - In general, decompose transfer $t$ into $t = t_1 + \ldots + t_n$ and transactionally execute all $t_i$ to $r_i$. No communication with $r_i$ is required if $t_i = 0$.

---

\(^1\)Assume credit limit policy of $r$ is conjunction of credit limit policies $r_1, \ldots, r_n$. 
Distributed resource managers: Example

Let \( r \) consist of resource managers \( r_1, r_2 \) with current ownership states

\[
\begin{align*}
\sigma_1 &= \{ \text{Bank}_1 : 60 \cdot \text{USD}, \text{Alice}: 30 \cdot \text{USD}, \text{Bob}: 40 \cdot \text{USD} \} \\
&= \{ \text{Bank}_1 : 130 \cdot \text{USD} \} + \\
&\quad \{ \text{Bank}_1 : -70 \cdot \text{USD}, \text{Alice}: 30 \cdot \text{USD}, \text{Bob}: 40 \cdot \text{USD} \} \\
\sigma_2 &= \{ \text{Bank}_2 : 10 \cdot \text{USD}, \text{Alice}: 100 \cdot \text{USD}, \text{Bob}: 200 \cdot \text{USD} \} \\
&= \{ \text{Bank}_2 : 310 \cdot \text{USD} \} + \\
&\quad \{ \text{Bank}_1 : -300 \cdot \text{USD}, \text{Alice}: 100 \cdot \text{USD}, \text{Bob}: 200 \cdot \text{USD} \}
\end{align*}
\]

and zero-credit policy (only nonnegative balances allowed).

- Transfer \( \{ \text{Alice}: -80 \cdot \text{USD}, \text{Bob}: 80 \cdot \text{USD} \} \) can be performed by \( r_2 \) without communication with \( r_1 \).
- Transfer \( \{ \text{Alice}: -120 \cdot \text{USD}, \text{Bob}: 120 \cdot \text{USD} \} \) cannot be performed by either \( r_1 \) or \( r_2 \), but it can be decomposed into \( t_1 + t_2 \) where \( t_1 = \{ \text{Alice}: -20 \cdot \text{USD}, \text{Bob}: 20 \cdot \text{USD} \} \) and \( t_2 = \{ \text{Alice}: -100 \cdot \text{USD}, \text{Bob}: 100 \cdot \text{USD} \} \) and then performed by transactionally executing \( t_1 \) on \( r_1 \) and \( t_2 \) on \( r_2 \).
Distributed resource managers: Transactionality

Nodes in a distributed resource manager need to support atomic execution of distributed transactions, e.g. for 2-phase commit:

- **Precommit** transfer $t$: Like ApplyTransfer, but with guarantee that, if validated, subsequent execution of $-t$ will succeed. For simple transfers: deducts resource from sender, but does not make it available yet to receiver.

- **Commit** transfer $t$: Apply previously precommitted $t$ (remove requirement that $-t$ must be applicable later on). For simple transfer: releases resource to receiver.

- **Abort** transfer $t$: Apply $-t$ to previously precommitted $t$. For simple transfer: return resource to sender.
Distributed resource managers: Discussion

- Many freely combinable “dimensions” of decomposition possible:
  - By resource type (e.g. land registry managing houses; national banking system (with individual banks as “peers”) managing USD accounts; the Bitcoin network for managing Bitcoin accounts (UTxOs), etc.
  - By agents (e.g. residents divided into countries of residence)
  - By statically or dynamically splitting off resource managers from existing resource managers for privacy and/or load balancing purposes (e.g. state channels, sharding).

- Resource managers should have API for participating in distributed transactions.

- Algebraic resource model as semantic basis for large design space for distributed resource managers.
Summary

- Algebra of transfers: infinite-dimensional vector space.
  - The power of negative: Additive inverses important.
- Separation of resource preservation (unrestricted algebra) and credit limit policies (restrictions).
- Additive decomposition of transfers: partitioning of resource managers for distributed implementation.