Inhaltsverzeichnis

Einleitung

Im Internet werden zunehmend Inhalte kollaborativ erzeugt. Dabei entsteht ein Inhalt durch Beiträge einzelner **Akteure**. Zentral ist die Frage nach dem **Besitz** des Inhaltes sowie die **Bewertung** der einzelnen Beiträge durch die Besitzer und damit ihren Anteil am **Gesamtergebnis**.

In dieser Arbeit wird dieses Problem für Inhalte einer regulären Sprache untersucht. Dafür wird ein dezentral validiertes Transitionssystem vorgestellt welches die Manipulation eines öffentlichen Ihnaltes durch Akteure steuert. Dafür wird eine Grammatik-Erweiterung vorgestellt, dessen Wörter Informationen über die Besitzallokation der Akteure, alternative Inhalte sowie deren Bewertung durch die Akteure beinhalten. Der Konsens der Akteure über den Inhalt wird als Interpretation des Wortes berechnet und ist durch die verfügbaren Informationen determiniert.

Dezentrale Verifizierung

Das auf dem Bitcoin-Protokoll[Nak08] aufbauende Ethereum [Woo14], beschreibt ein P2P Protokoll zur dezentralen Verifizierung von Turing-Vollständigen Programmausführungen. Es stellt einen gebildeten Konsens von Programmen mit aktuellen Zuständen bereit. Neue Inhalte wie neue Programme oder neue Interaktionen von Akteuren mit Programmen werden nach einer Veröffentlichung und Validierung der Ausführung durch das Netzwerk mit dem neu berechnetem Zustand in den Konsens aufgenommen.

Agierende Instanzen sind einerseits externe Akteure, die durch ein asymmetrisches Kryptosystem mit anderen Akteuren im Netzwerk interagieren. Andererseits sind Akteure Turing-Vollständige Programme, die vom Netzwerk bereitgestellt werden. Solche Akteure werden auch (smart-)**contracts** genannt. Akteure sowie Contracts werden durch ein 20-Byte Adresse identifiziert.

Die Validierungsknoten des Ethereum-Netzwerks finanzieren sich einerseits durch die Neuschöpfung einer internen Währung, sowie durch einen Betrag, der für die Validierung einer Programmausführung von einem Akteur bezahlt wird. Für jeden Berechnungsschritt der Ethereum-VM wird dafür ein kleiner Betrag erhoben. Ist nicht genügend Wert verfügbar, terminiert die Transaktion und wird vom Netzwerk abgelehnt. So wird auch das Halteproblem umgangen.

> [...] halting problem: there is no way to tell, in the general case, whether or not a given program will ever halt. [...] our solution works by requiring a transaction to set a maximum number of computational steps that it is allowed to take, and if execution takes longer computation is reverted but fees are still paid.
>
> (Ethereum Whitepaper p. 28 [But14])

Ein Contract besteht aus einem assoziativem Speicher der den Programmcode sowie Daten beinhaltet.

Ein Beispiel für ein Programm ist eine dezentrale Währung, wie sie derzeit vom Bitcoin Protokoll repräsentiert wird (Quelle [But14]):

```
from = msg.sender
to = msg.data[0]
value = msg.data[1]

if self.storage[from] >= value:
self.storage[from] = self.storage[from] - value
self.storage[to] = self.storage[to] + value
```

Eine neue interessante Anwendung ist ein **geteilter** manipulierbar Inhalt, im folgenden Objekt genannt.

Digital Geteiltes Objekt

Definition

Ein **geteiltes digitales Objekt** ist ein Tupel $O_G = (A, K, w, rate, consens)$ bestehend aus:

1. Eine endliche Menge von Akteuren A
2. Eine eindeutige **Besitzverteilung** von Akteuren zum Objekt
$$w : A \to \mathbb{N}$$
3. eine endliche Menge von validen **Kandidaten** mit mindestes einem Element
$$K_G \subseteq L(G) \wedge |K_G| \geq 1$$
4. eine Bewertung der Kandidaten durch die Akteure.
$$rate : A \times K \to [0, 1]$$
5. einer Konsensfunktion
$$consens : A \times K \times w \times rate \to L(G)$$

Sei $\mathbf{SOBJ_G}$ die Menge aller geteilten Objekte in der Grammatik G sowie $\mathbf{P_G} := \{K_G | K_G \subseteq L(G)\}$ die Menge aller möglichen Kandidatenmengen.

Erläuterung

Besitzverteilung

Die Besitzverteilung wird ähnlich dem Aktienmarkt Modelliert. Dabei hat ein geteiltes Objekt eine bestimmte Anzahl an Teilen (geschrieben $|O| \in \mathbb{N}$), die unter den Akteuren aufgeteilt sind.

$$|O| := \sum_{a \in A} w(a)$$

Der Besitz eines Objekts O wird durch das Recht definiert, auf dieses zugreifen zu können und es Kontrollieren zu können[Wal04]. Besitzen mehrere Akteure ein Objekt, so gibt der Anteil am Objekt an, zu welcher Gewichtung jeder einzelne Akteur über das Objekt mitbestimmen kann.

Eine Konsensfunktion entscheidet über die Ausführung der Kontrolle.

Transaktionen

Transaktionen sind Manipulationen des Objektes. Sie werden **immer** von einem Akteur a ausgelöst und besitzen die Form (a, O'). O' ist dabei das manipulierte geteilte Objekt. **Transaktionsbedingungen** entscheiden über die Validität der Transaktion und demnach über ihre Anwendung.

Wir betrachten vorerst nur die Manipulation der Kandidatenmenge K sowie der Kandidatenbewertung $rate$.

Konsens

$$consens : A \times K \times w \times rate \rightarrow L(G)$$

Die Konsensfunktion liefert für jedes Objekt ein Wort aus der $L(G)$ Sprache. Dabei sucht die Funktion nach dem Kandidaten aus der Kandidatenmenge mit der maximalsten Bewertung. Die Bewertung eines Kandidaten ergibt sich dabei durch die Summe der gewichteten Bewertungen der Akteure.

$$consens(A, K_G, w, rate) := \max_{k \in K_G} (\sum_{a \in A} w(a) * rate(a, k))$$

Qualitätskriterien

Für die Implementation ist auf folgende Qualitätskriterien zu achten:

RESMIN:
Es sind möglichst wenig Ressourcen (Speicher und Rechenleistung) vom Ethereum-Netzwerk notwendig, um Transaktionen zu validieren.

INTMIN:
Eine Akteur soll möglichst wenig Interaktionen benötigen, um auf eine gewünschte, von ihm erreichbare Manipulation zu kommen.

Anwendung auf reguläre Grammatiken

Optimierung der Kandidatenmenge

Die reguläre Grammatik wird so zu einer kontextfreien Grammatik erweitert, dass alle Wörter aus der Kandidatenmenge in einem Wort der erweiterten Grammatik codiert werden können. Redundant auftauchende Präfixe der Kandidaten werden zusammengefasst. Dafür wird an jedem Ableitungsschritt eine Mehrdeutigkeit

zugelassen die zu einer **Optionsmenge** zusammengefasst wird. Die Kandidatenmenge ergibt sich durch die Kombinationen der Präfixe mit den Optionen einer Optionsmenge.

Dazu muss es eine Grammatik-Erweiterung $S : REG \to CFG$ gefunden werden. Um zu zeigen, dass es sich bei S um die gesuchte Erweiterung handelt, muss die Existenz der Funktionen $\phi : P(K_G) \to L(S(G))$ und $\phi^{-1} : L(S(G)) \to P(K_G)$ gezeigt werden, für die folgende Bedingungen erfüllt sein müssen:

$$\phi \circ \phi^{-1} = id_{P(K_G)} \tag{1}$$

$$|\phi(K_G)| \lesssim \sum_{k \in K_G} |k| \tag{2}$$

Die Bedingung (1) kann auch abgeschwächt werden zu $K_G \subseteq \phi \circ \phi^{-1}(K_G)$. Dieses würde bedeuten, dass Kandidaten nicht verloren gehen, jedoch hinzukommen können, solange diese valide Kandidaten bilden.

In Bedingung (2) werden Edge-Cases ausgeschlossen, die nur bei kleinen diversen Lösungen auftauchen können. (z.B.: $\{a\ b\}$)

Grammatik-Erweiterung S1

Sei $G = (N, T, S, P)$ eine reguläre Grammatik.

$$\begin{aligned} S_1(G) &:= (N', T', S, P') \\ N' &:= N \cup \{O_R \mid (R \to r) \in P \wedge O_R \notin N\} \\ T' &:= T \cup \{[,] \mid [,] \notin T\} \\ P' &:= P \cup P_{Options} \\ P_{Options} &:= \{R \to [O_R], O_R \to rO_R, O_R \to \varepsilon \mid R \to r \in P \wedge r \in (N \cup T)^*\} \end{aligned}$$

Definition ϕ

TODO: Definition ϕ

Definition ϕ^{-1}

TODO: Definition ϕ^{-1}

zeige $\phi \circ \phi^{-1} = id_{P(K_G)}$

TODO: zeige $\phi \circ \phi^{-1} = id_{P(K_G)}$

zeige $|\phi(K_G)| \lesssim \sum_{k \in K_G}$

TODO: zeige $|\phi(K_G)| \lesssim \sum_{k \in K_G}$

Optimierung der Kandidatenbewertung

Komprimierung

Die Idee hinter der Komprimierung ist, die Bewertung der Kandidaten an Optionen zu binden. Wenn ein Akteur eine Option in einer Optionsmenge bewertet, so gibt er seine Stimme der Option im Kontext zu dem bisherigen Präfix des Wortes. Somit werden alle Kandidaten, die sich mit dem Präfix und der Option erzeugen lassen von der Stimme beeinflusst. Diese wird als ein Attribut der Ableitung vererbt[Knu68] und jeweils von einer Bewertung einer tieferen Ebene ersetzt. Eine wichtige Bedingung ist, dass jede mögliche Bewertungsverteilung in der Optimierung codierbar ist. Dieses ist leicht zu zeigen, da die Bewertung aller Blätter des Ableitungsbaumes ebenfalls die Bewertung der Kandidaten repräsentiert.

Delegationen

Für die Minimierung der Interaktion wird transitives Abstimmen zugelassen. Akteure können nicht nur für die Option selbst abstimmen, sondern auch ihre Stimme für Optionsmengen an andere Akteure so delegieren, dass diese im Kontext der delegierten Option für den Akteur mitbestimmen.

Durch die lineare Struktur der Worte und somit auch der Delegationsmengen ergibt sich nicht nur eine Menge der Delegationen für eine Optionsmenge, sondern auch eine totale Ordnung der Delegationen, womit bei konkurierenden Delegationen immer die Delegation höherer Ordnung genommen wird.

TODO: konkurierende Delegationen

Besitzverteilung

Schließlich wird die Besitzverteilung ebenfalls als Teil der erweiterten Sprache angesehen. Dies erlaubt nun den Konsens des Objektes als Interpretation eines Wortes der $L(S(G))$ Sprache zu definieren. Transformationen werden als valide Manipulationen eines Wortes definiert. Transformationsbedingungen geben dabei Anforderungen an die Ableitungsstruktur des bisherigen Wortes sowie an das Manipulierte Wort. Dieses determiniert die Validität einer Transaktion.

Grammatik-Erweiterung S

number stellt eine ganzzahlige Nummer dar. **float** stellt eine Fließkommazahl dar, so dass $0 \leq float \leq 1$. **hash** identifiziert einen Akteur, hier ist $hash$ ein 20 Bytes HEX-String.

$$
\begin{aligned}
S(G) :=& (N', T', S', P') && (3)\\
N' :=& N \cup \{O_R | (R \to r) \in P \wedge O_R \notin N\} \cup \{D, A\} && (4)\\
T' :=& T \cup \{[,], \oplus, number, float, hash \mid [,], \oplus \notin T\} && (5)\\
P' :=& P \cup P_{Options} \cup P_{Start} \cup P_{Delegations} \cup P_{Voting} \cup P_{Acteurs} && (6)\\
P_{Acteurs} :=& \{A \to [hash\ number]A, A \to \varepsilon\} && (7)\\
P_{Options} :=& \{R \to [O_R][D], O_R \to r \oplus\ [V]O_R, O_R \to \varepsilon\\
& \mid R \to r \in P \wedge r \in (N \cup T)^*\} && (8)\\
P_{Start} :=& \{S' \to [A][O_S][D]\} && (9)\\
P_{Delegations} :=& \{D \to [hash\ hash]D, D \to [hash\ hash]\} && (10)\\
P_{Voting} :=& \{V \to [hash\ float]V, V \to \varepsilon\} && (11)
\end{aligned}
$$

zeige S(G) ist eindeutig

TODO: zeige S(G) ist eindeutig

Konsens

TODO: definiere $consens : L(S(G)) \to L(G)$

Transitionssystem

Die Manipulation wird durch ein **initialisiertes Transitionssysthem** $T = (M, I, \tau)$ beschrieben nach [Gla].

M ist die Zustandsmenge. $I \subseteq M$ ist die Anfangszustandsmenge und τ ist die Transformation von M sowie $\tau : M \to M$. Wir machen weiter die Einschränkung $|I| = 1$.

Wortteil

Sei $G = (S, N, T, P)$ eine Grammatik und $w \in L(G)$, dann definieren wir ein Wortteil:

$$w_{i..j} := \begin{cases} (t_k)_{i \leq k \ \leq j} \text{ mit } t_k \in T & \mid i \leq j \\ \varepsilon & \mid i > j \end{cases}$$

Ablauf

Eine Folge $(s_n)_{n \in \mathbb{N}}$ mit $s_n \in M$ ist ein Ablauf von T, wenn gilt:

$$s_0 \in I$$
$$s_{i+1} = \tau(s_i)$$

Zustände

Sei $G = (N, T, S, P)$ eine rechtsreguläre Grammatik sowie $S(G) = (N', T', S', P')$ die dazugehörige erweiterte Grammatik.

Ein Zustand ist eine Struktur der Form $(U, w) \in M$

U ist die Trägermenge der Struktur und besteht aus den Wörtern der Sprache $L(S(G))$:

$$U := L(S(G))$$

$w \in U$ ist das derzeitige Wort

Aktualisierung

Sei A die Menge der Akteure, $L(S(G))$ eine Sprache zur gegebenen Grammatik G. Dann ist das Tupel $\delta = (a, w')$ mit $a \in A,\ w' \in L(S(G))$ eine Aktualisierung. Sei $\Delta \subseteq A \times L(S(G))$ die Menge aller aktueller Aktualisierungen.

Akteure können jederzeit die Aktualisierungsmenge erweitern:

$$\Delta' := \Delta \cup \{(a, w')\}$$

Transformation

Sei $\Delta \neq \emptyset$

$$\tau(w) = \begin{cases} w' & \exists (a, w') \in \Delta \wedge \beta(a, w, w') = true \\ w & andernfalls \end{cases}$$

Nach jeder Transformation wird die Aktualisierungsmenge ebenfalls aktualisiert, in dem die für die Transformation verwendete Aktualisierung entfernt wird. Wurde keine Aktualisierung verwendet, so sind alle Aktualisierungen invalide:

$$\Delta' = \begin{cases} \Delta \setminus \{(a, w')\} & (a, w') \text{ wurde in der Transformation verwendet} \\ \{\} & andernfalls \end{cases}$$

Validierung der Aktualisierung

Um die Validität einer Transformation zu verifizieren, wird das derzeitige Wort w zusammen mit dem auslösenden Akteur a sowie seinem vorgeschlagenem Wort w' betrachtet. Die Transformation ist valide, falls eines der folgenden Fälle zutrifft. Es gilt also: $\beta(a, w, w') = true$ genau dann wenn:

Optionsmengen

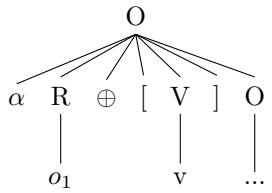
Eine Optionsmenge kann an jeder Position im Wort erzeugt oder erweitert werden, solange die erzeugte Kandidatenmenge durch $\phi^{-1}(w')$ valide bleibt, sowie die Bewertungen der Kandidaten $\phi^{-1}(w)$ nicht verändert werden. Die neu hinzukommenden Optionen dürfen keine Delegationen oder Stimmen enthalten, sowie noch nicht in der bisherigen Optionsmenge enthalten sein.

Erzeugen einer Optionsmenge

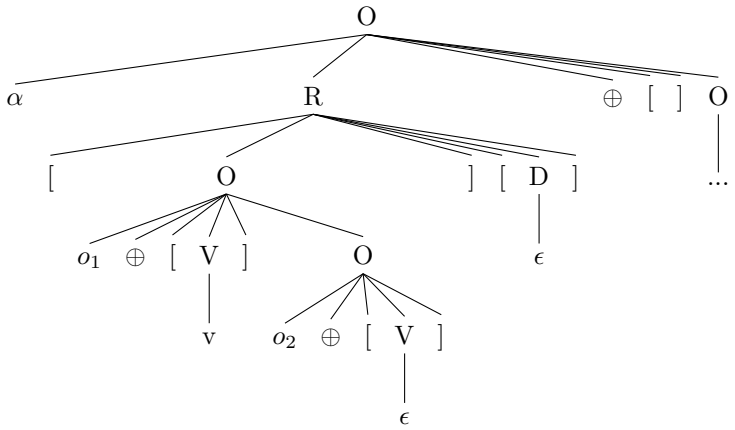
Sei $S(G) = (T', N', S', P')$ eine erweiterte Grammatik. Sei weiter $O, S' \in N'$ sowie $v, o_1, o_2 \in T'^*$ und $0 \leq i \leq j \leq n$ mit $i, j, k, n \in \mathbb{N}$

$$\begin{aligned} w &= w_{0..n} && (12) \\ w_{i..j} &= \alpha o_1 \oplus [v] && (13) \\ S' &\to^* w_{0..(i-1)}\ O\ w_{(j+1)..n} && (14) \\ O &\to^* w_{i..j} && (15) \\ R &\to^* o_1 && (16) \\ R &\to^* o_2 && (17) \\ w' &= w_{0..i-1}\ \alpha[o_1 \oplus [v] o_2 \oplus []][]\ w_{j+1..n} && (18) \end{aligned}$$

Für w hat O folgende Struktur:



Für w' besitzt O folgende Struktur:



Dabei muss o_2 frei von Delegationen und Stimmen sein, sowie o_2 muss relativ zu o_1 neu sein.

Erweitern einer Optionsmenge

Sei $o, d, r \in T'^*$ sowie $R, O_R \in N'$

$$
\begin{aligned}
w &= w_{0..n} && (19)\\
w_{i..j} &= \alpha[o][d] && (20)\\
R &\to^* w_{i..j} && (21)\\
O_R &\to^*_P o && (22)\\
O_R &\to^* r && (23)\\
w' &= w_{0..(i-1)}\ \alpha[or \oplus []][d]\ w_{(j+1)..n} && (24)
\end{aligned}
$$

r muss dabei Delegations- und Stimmfrei sein sowie neu:

$$\phi^{-1}(w) \cap \phi^{-1}(w_{0..i-1}\ \alpha[r \oplus []][d]\ w_{j+1..n}) = \emptyset$$

Hinzufügen einer Stimme

Sei $w_{i..j} = r \oplus [v]$

Dann ist $w' = w_{0..i-1}\ r \oplus [v[hn]]\ w_{j+1..n}$ mit $0 \leq n \leq 1$ eine valide Substitution.

Löschen einer Stimme

Sei $a \in A, v, v' \in T'^*$

$$
\begin{aligned}
w &= w_{0..n} && (25)\\
w_{i..j} &= r \oplus [v[an]v'] && (26)\\
w' &= w_{0..(i-1)}r \oplus [vv']w_{(j+1)..n} && (27)
\end{aligned}
$$

Hinzufügen einer Delegation

Da die Reihenfolge der Delegationen wichtig ist, für die Auflösung konkurierender Delegationen, daher ist das Hinzufügen einer Delegation an jeder Stelle in einer Delegationsmenge möglich.

Sei $o, d_i \in T'^*$ sowie $a' \in A$ und $a' \neq a$

$$
\begin{aligned}
w &= w_{0..n} && (28)\\
w_{i..j} &= [o][d_1..d_n] && (29)\\
& \quad k, n \in \mathbb{N} \wedge k \leq n && (30)\\
d_k &= [hash1_k\ hash2_k] && (31)\\
w' &= w_{0..(i-1)}\ [o][d_1..d_{k-1}[a\ a']d_k..d_n]\ w_{(j+1)..n} && (32)
\end{aligned}
$$

Löschen einer Delegation

Sei $o, d_i \in T'^*$ sowie $a' \in A$ und $a' \neq a$

$$
\begin{aligned}
w &= w_{0..n} && (33)\\
d_k &= [a\ a'] && (34)\\
w_{i..j} &= [o][d_1..d_k..d_n] && (35)\\
w' &= w_{0..(i-1)}\ [o][d_1..d_{k-1}d_{k+1}..d_n]\ w_{(j+1)..n} && (36)
\end{aligned}
$$

Delegationsprogrammierung

Akteure können ihre Stimme nicht nur an externe Akteure delegieren, sondern auch an Contracts, welche in einer autonomen Weise abstimmen und so die Interessen der Akteure automatisch verfolgen.

Beispiele

Sei $G = (T, N, P, S)$ eine rechtsreguläre Grammatik mit:

$$T := \{a, b, c\}, N := \{S\}, P := \{S \to aS \mid bS \mid cS \mid \varepsilon\}$$

Ein Delegationsprogramm kann mit dem Interesse entwickelt werden, nur für Kandidaten der kontextsensitiven Sprache $L = \{a^n b^n c^n \mid n \in N\}$ zu stimmen so, dass falls die Mehrheit der Akteure ihm seine Stimme delegieren, der Konsens-Kandidat ebenfalls ein Element der Sprache L sein wird. Auf diese weise lassen sich von den Akteuren komplexe Programme erstellen um die Kandidatenmenge agil und autonom zu bewerten.

Umfang der Arbeit

Konzeptuell

- Die Punkte unter Aufgaben (TODO's) werden ausformuliert.
- Dieses Papier wird überarbeitet und strukturiert. Es wird mit anschaulichen Beispielen, Erklärungen, Beweisen, Quellen sowie Verweisen grundlegender Konzepte ergänzt. Solche Konzepte sind:
  - reguläre/ kontextfreie Grammatik, Normalformen, Ableitungen, Eindeutigkeit
  - Algebraisches-Transitionssystem
  - Σ-Strukturen/-Signaturen
  - Turing-Vollständigkeit
  - asymmetrische Verschlüsselung
- Die Punkte unter Aussicht werden diskutiert, jedoch weder implementiert, noch im Umfang der Anwendung auf die regulären Grammatiken konzipiert.

Implementation

- Eine Zentralisierte Version der $L(S(G))$-Sprache sowie des Transitionssystem wird implementiert.
  - JISON[1], eine javascript Portierung des Bison[2]/Flex[3] LALR(1) Parser Generators wird so erweitert, dass bei Eingabe einer beliebigen regulären Grammatik G im bison Format[4] dieser einen Parser erzeugt welcher Wörter aus der $L(S(G))$ Sprache als Eingabe bekommt und ein öffentliches Transitionssystem erzeugt.
  - Ein Initialisierungs-Script, welches aus einem Wort der $L(G)$ Sprache und einer Besitzverteilung gegeben als $\{(a,n)|a \in 20byteHexString \wedge n \in \mathbb{N}\}$ ein valides initiales Wort in der $L(S(G))$ Sprache erzeugt.
  - Akteure können durch eine Webseite in die Transaktionshistorie einsehen, sowie das Wort valide manipulieren.
  - Eine Client-seitige asymmetrische Verschlüsselung validiert die Transaktionen der Akteure.
  - Die Konsens-Funktion wird ebenfalls Implementiert und erzeugt zu jedem Stand einen öffentlich verfügbaren Konsens-Kandidaten.
- Einige anschauliche Beispiele regulärer Sprachen mit beispielhaften Delegationsprogrammen werden implementiert und demonstriert.

Todos

Aufgaben

[1] http://jison.org/
[2] http://www.gnu.org/software/bison/
[3] http://flex.sourceforge.net/
[4] http://dinosaur.compilertools.net/bison/bison_6.html

Aussicht

Anwendung auf kontextfreie Grammatiken

Dezentralisierung

Anonymisierung

Manipulation der Besitzverteilung

Manipulation der Grammatik

Quellen

References

[But14] V Butterin. A next-generation smart contract and decentralized application platform. (January):1–36, 2014.

[Gla] Andreas Glausch. Abstract-State Machines Eine Sammlung didaktischer Beispiele. pages 1–19.

[Knu68] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, June 1968.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, pages 1–9, 2008.

[Wal04] Jeremy Waldron. Property and Ownership. September 2004.

[Woo14] Gavin Wood. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. 2014.