Inhaltsverzeichnis

Abstract

Im Internet werden zunehmend Inhalte kollaborativ erzeugt. Dabei entsteht ein Inhalt durch Beiträge einzelner Akteure, die räumlich und zeitlich getrennt sein können. Zentral ist dabei die Frage wie der Konsens über einen Inhalt auf eine dezentrale Weise gebildet werden kann. In dieser Arbeit untersuchen wir den Prozess der verteilten Zusammenarbeit für Inhalte die als Wörter einer regulären Sprache beschrieben werden können. Dafür wird ein Modell mit einer Implementation einer Blockchain-Technologie vorgestellt: Ein öffentliches Transitionssysthem mit dezentral validierter Ausführung. Ebenfalls wird eine Grammatik-Erweiterung mit einer Konsens-Funktion für eine beliebige reguläre Grammatik beschrieben. Die Wörter der erweiterten Grammatik beinhalten Informationen über die Besitzallokation der Akteure, alternative Beiträge und deren Bewertung. Die Konsens-Funktion überführt unter Berücksichtigung der Bewertungen ein Wort der erweiterten Grammatik in ein Wort der ursprünglichen Grammatik.

Einleitung

Nehmen wir z.B. an, es gäbe eine Webseite, die sich im Besitz von Akteuren befindet. Alle Akteure wollen gerecht, also proportional zu ihrem Besitz, über die Inhalte der Webseite entscheiden können, sowie ihre Entscheidungsgewalt in bestimmten Bereichen an andere vertrauenswürdige Akteure delegieren können. Dieses gilt für die medialen Inhalte, die Programmierung, die Architektur, die Wertflüsse wie ein geteiltes Budget oder eine Einkommensverteilung, sowie nicht automatisierbare Prozesse, wie das Validieren neuer Beiträge. Das eigentliche Ergebnis wird anhand von einer Mehrheit der Besitzer bestimmt.

Wikipedia folgt einem streng hierarchischem Modell, in welchem Vertrauenspersonen die Inhalte der Benutzer filtern. Die Verantwortung liegt bei der Organisation. Effizienter sind jedoch selbstregulierende Systeme, bei denen die Benutzer die Inhalte der Anderen bewerten. Beispiele wären die auf Voting und Reputation basierenden Plattformen Reddit und StackOverflow. Ein weiteres Beispiel für die Bewertung von Inhalten ist das Konzept Liquid Democracy[Lin10], ein Vorschlag der Piratenpartei für eine moderne politische Konsensbildung. Ein solches Prinzip könnte man auf das Verwalten aller digital geteilter Inhalte verallgemeinern.

Jedoch eignen sich diese Konzepte nur bedingt um damit geteiltes Eigentum wie z.B. eine Webseite zu modellieren, da Abstimmungen auf eine informale Weise vorgenommen werden. Es fehlt einer formalen Syntax, welche die Implikationen einer potentiellen Entscheidung vor der Wahl durch Simulation klarer zeigt, sowie nach der Wahl automatisch ausführt. Außerdem bedarf es bei den Ansätzen eines zentralisierten Servers, welcher als Angriffspunkt die Glaubwürdigkeit des Prozesses gefährdet, da die Akteure auf die Korrektheit des Servers vertrauen müssen.

Das 2009 eingeführte Konzept des Bitcoins und der Blockchain[Nak08] bietet eine Alternative zur zentralen Server-Architektur. Dieses beschreibt ein Protokoll in einem Netzwerk, welches Inhalte aus einem gebildeten Konsens bereitstellt. Es besitzt eine einfache, nicht turing vollständige Skriptsprache, sowie durch ein asymmetrisches Kryptosystem, Rechte und Rollen. Neue Inhalte werden nach einer Validierung ebenfalls in den Konsens aufgenommen. Die einfachste Interpretation von Bitcoin ist die eines Werteträgers, wobei die Beschränkung der Skriptsprache wenig Spielraum für weitere Interpretationen lässt. Allgemeiner ist das auf dem Bitcoin-Protokoll aufbauende Ethereum[Woo14] , welches eine turing vollständige Skriptsprache besitzt. Es entsteht eine Vielzahl von neuen Anwendungsmöglichkeiten: verbindliche, autonome Verträge zwischen mehreren Parteien, dezentrale autonome Organisationen (DAO) oder profitorientierte Kooperationen (DAC), die allesamt Werte- und Informationsflüsse ermöglichen. Ein Beispiel einer solchen DAO ist das namecoin[1] Konzept, welches als Alternative zur ICANN[2] Organisation die TLD ".bit" verwaltet und in naher Zukunft

[1] http://namecoin.info/
[2] Internet Corporation for Assigned Names and Numbers

die ICANN ablösen könnte.[CN14] Die Regeln unter denen DACs und DAOs funktionieren, wie das Bewilligen einer neuen TLD, werden auf der Ethereum VM programmiert. Die Einigung der Akteure auf ein Programmstand geschieht noch durch eine zentrale Vertrauensinstanz, z.b. bestimmten Schlüsselpersonen.

Theoretisches

Reguläre Grammatiken

Eine reguläre Grammatik definiert nach Noam Chomsky ist ein vier Tupel $G = (N, T, S, P)$ bestehend aus

1. N - einer Menge nichtterminaler Symbole
2. T - einer Menge terminaler Symbole, disjunkt zur nichtterminalen Menge $T \cap N = \emptyset$
3. $S \in N$ - einem Startsymbol
4. $P \subseteq N \times \{\epsilon\} \cup T \cup TN$ - einer Menge von Produktionen

Sei weiter $\Sigma := N \cup T$ das Alphabet der Grammatik.

Eine Produktion $(R, r) \in P$ kann auch als $R \to r$ geschrieben werden. Sie sagt aus das das das Nichtterminal R in einem Schritt überführt werden kann zum Teilwort r:

$$\alpha R \to \alpha r$$

Sind hierfür n Schritte notwendig schreibt man $R \to^n r$. Um auszudrücken, dass r generell aus R ableitbar ist, kann man sich der reflexiv-transitiven Hülle der Produktionsregeln bedienen: $R \to^* r$.

Die Sprache, die von der regulären Grammatik erzeugt wird ist die Menge aller Wörter, die vom Startsymbol ableitbar sind:

$$L(G) := \{w | S \to^* w \wedge w \in T^*\}$$

Existiert eine reguläre Grammatik, die eine Sprache erzeugt, so heißt die Sprache ebenfalls regulär.

Die Klasse der regulären Sprachen heißt REG.

Ein Beispiel für eine reguläre Grammatik mit der dazugehörigen Sprache ist:

$$
\begin{aligned}
G &= (N, T, S, P) \\
N &= \{S, A, B, C\} \\
T &= \{a, b, c\} \\
P &= \{ \\
&\quad S \to aA, S \to bB, S \to cC, S \to \varepsilon, \\
&\quad A \to aA, A \to bB, A \to cC, A \to \varepsilon, \\
&\quad B \to bB, B \to cC, B \to \varepsilon, \\
&\quad C \to cC, C \to \varepsilon, \\
&\quad \}
\end{aligned}
$$

$$L(G) = \{a^x b^y c^z | x, y, z \in \mathbb{N}\}$$

Wortteil

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

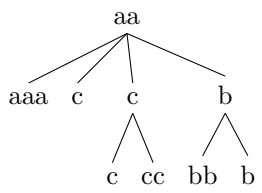
$$w_{i..j} := \begin{cases} (t_k)_{i \leq k \ \leq j} \text{ mit } t_k \in T & | \ i \leq j \\ \varepsilon & | \ i > j \end{cases}$$

Kontextfreie Grammatiken

Kontextfreie Sprachen unterscheiden sich nur in den Produktionsregeln von den regulären. Anders als bei regulären, wird für die rechte Seite einer Produktionsregel keine Einschränkung gemacht: $P \subseteq N \times (N \cup T)^*$

Trie

Ein Trie oder Prefixbaum ist eine Baum-Datenstruktur, die es erlaubt eine Menge von Worten über einem Alphabet effizient zu Speichern. Dabei werden gemeinsame Präfixe von wörtern in Knoten zusammengefasst. Eine Trie verwandte Datenstruktur, die Patricia-Trie reduziert den Speicherverbrauch indem sie alle Knoten mit nur einem Nachfolger zu einem Knoten zusammenfasst. Alle im Trie und Patricia-Trie gespeicherten Wörter können durch ein Tiefenscan wiederhergestellt werden. [Mor68] Ein Beispiel eines Patricia-Trie für die Menge $\{aaaaa, aac, aacc, aaccc, aabbb, aabb\}$ ist:



Transitionsystem

Ein initialisiertes Transitionssystem (T) [Gla] gibt einen formalismus vor um ein zustandbasiertes System zu beschreiben. Die Beschreibung zerteilt sich in die des Zustandraumes, beschrieben durch eine Menge M, die der Dynamik, beschrieben durch eine Übergangsfunktion $\tau : M \to M$ und einer Anfangszustandsmenge $I \subset M$.

$$T = (M, I, \tau)$$

Ablauf

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

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

Kryptografie

TODO: ecdsa

TODO: hash

asymetrische Verschlüsselung

Bei einem asymmetrischem verschlüsselungs verfahren generiert der Benutzer ein schlüsselpaar bestehend aus einem privatem und einem öffentlichem schlüssel. Der öffentliche Schlüssel wird aus dem Privatem erzeugt und wir veröffentlicht. Wir werden im folgenden für den öffentlichen Schlüssel ebenfalls Addresse nennen, die einen Akteur identifiziert. Agiert der Akteur im Netzwert, beweist dieser durch die Signierung der Aktionen, dass dieser tatsächlich im Besitz des Privatem schlüssel ist und beweist dadurch seine Identität.

Signierung

Eine Aktion(oder auch Nachricht (msg) genannt) eines Akteurs kann mit dem privatem schlüssel signiert werden.

$$signMsg : priv \times msg \to sign$$

Mithilfe einer weiteren Funktion kann nun durch die öffentliche Addresse des Akteurs, der nachricht und deren Signatur verifiziert werden, ob die Nachricht auch tatsächlich vom angegebenem Akteur ausgelöst wurde.

$$verifyMsg : msg \times sign \times addr \to \{True, False\}$$

Dadurch besteht nun eine Aktion eines Akteurs aus der nachricht, seiner Addresse sowie der signatur der nachricht.

$$(msg, sign, addr)$$

Decentralized Autonomous Organization (DAO)

Eine DAO ist ein relativ neues Konzept, welches durch das populärwerden Technologien wie Bitcoin und Ethereum erstmals aufgetaucht ist. Die Fachwelt ist sich noch über die genaue Definition und die abgrenzung zu verwanten Konzepten größtenteils uneinig. Zwar exestieren Versuche "Decentralized Applications" (DAs,Dapps) wie BitTorrent oder den "Decentralized Autonomous Corporations" (DACs) von dem der DAOs Abzugrenzen, jedoch gibt es hier keine Garantie auf Persistenz der Begriffe sowie ihrer Definitionen, desshalb werden im folgenden die Begriffe DA, Dapp, DAC, DAO synonym verwendet.

Intuitiv kann eine DAO als eine Gesellschaft verstanden werden, die ohne menschliche Einwirkung existiert und agiert. Sie kann ebenfalls als eine erweiterung des Open-Source Konzeptes betrachtet werden. Dabei ist nicht nur der **Programmcode öffentlich**, sondern auch ihre internen Berechnungen sowie wesentliche Teile ihrer Interkationen. Diese sind durch einen gemeinsamen **Konsens** von der Erzeugung bis zum aktuellen Zustand determiniert und nachvollziehbar. Dieser Konsens wird auf eine dezentrale Weise gebildet, um zu verhindern, dass eine zentrale Instanz den Konsens manipuliert sowie um Robustheit und Beständigkeit zu sichern.

In ihren Aktionen kann die DAO interne, deterministische Berechnungen anstellen, mit anderen DAOs komunizieren und so auf Services zugreifen oder nicht selbst ausführbare Aufgaben, wie beispielsweise das Erweitern und Verbessern des Programmcodes, an Menschen deligieren. Ihr besitz kann durch Anteile determiniert werden, die unter anderen DAOs oder Menschen aufgeteilt sind. Diese berechtigen beteiligte Akteure zu verschiedenen Aktionen innerhalb ihrer Regeln wie z.B. einen Zugriff auf Dividenden oder anderen Ressourcen der DAO, partizipation bei internen Abstimmungen oder anderen spezifischen Aktionen.

Ethereum, Bitcoin und Namecoin sind allesamt Beispiele für unterschiedliche DAOs. Ethereum ist dabei gleichzeitig eine Plattform für andere DAOs.

Im folgenden möchten wir drei Teilberreiche von DAOs näher beleuchten. Als erstes die Funktionsweise des Blockchain Algorithmus, der eigendlichen Neuerung, die zum Entstehen von DAOs führte. Als nächstes dem Teil, der die Interaktionen der DAO steuert. Als letztes der Selbstmanipulation der DAO, also dem Teil, welcher die Manipulation der Regeln der DAO steuert.

Dezentraler Konsens - Blockchain

Bitcoin hat mit der einführung der Blockchain eine elegante und Universelle Lösung für das Problem der Byzantinischen Generäle vorgeschlagen. Das Problem steht repräsentativ in den verteilten Systemen für die Schwierigkeit, einen Konsens zwischen Akteuren herzustellen, wenn einige, für Andere unbestimmte Akteure eigennützig sowie manipulativ agieren. Bitcoin hat hier eine statistische Lösung presentiert: Es führt die dezentralität des Konsens auf die natürliche Knapheit einer Ressource zurück. Für Bitcoin ist es die Anzahl der Berechnungen, die ein Akteur in einer bestimmten Zeit machen kann. Ein General versucht dabei ein Mathematisches Rätsel zu lösen, welches statistisch gesehen 10 Minuten dauern müsste, wenn alle beteiligten Generäle an dem Problem arbeiten würden. Hat ein General eine Lösung gefunden, teilt er dieses an alle Ihm bekannten Generäle mit. Eine solche Lösung wird auch als "Block" bezeichnet. Jeder General arbeitet darauf hin mit dieser Lösung weiter und versucht sie zu erweitern, was wieder statistisch 10 Minuten der Kraft aller Generäle kosten müsste. Jeder General arbeitet mit der von ihm zuerst gesehenen längsten Kette von Blöcken. Mit der Anzahl der Erweiterungnen konvergiert nun die beteiligung loyaler Generäle am Konsens zur majorität, falls diese tatsächlich in Besitz einer Mehrheit der Begrenzten Rechenressourcen besitzen. In einem Block befindet sich neben der vorhergehenden Lösung ebenfalls der aktuelle Konsenses solange er auf den vorergehenden aufbaut und valide ist.

Das Lösen eines Rätsels für das Recht einen neuen Block bestimmen zu können, wird als "proof of work" oder kurz POW bezeichnet, da ein General eine Arbeitsleistung beweisen müssen, um den nächsten Block vorschlagen zu dürfen. Bitcoin und viele andere DAOs benutzen partielle hash invertierung als tatsächlichen POW Algorithmus. Dieser baut auf der Eigenschaft einer Hash-Funktion, dass es sehr einfach ist einen Hash einer Nachricht zu berechnen, jedoch sehr schwehr aus einem Hash eine Nachricht zu rekonstruieren, die diesen hash erzeugt. Der eigendliche Block besteht dabei aus einer referenz des vorherigen Blocks (ref) zusammen mit dem neuen Konsens ($kons$) und einer $nonce$, die frei wählbar ist. Zudem existiert durch die vorherigen Blöcke bestimmte Schwierigkeit (ε), so dass die Rechenzeit für den neuen Block bei 10 Minuten bleibt. Akteure, die im Wettbewerb um einen neuen Block stehen (auch "Miner" genannt) müssen nun eine nonce finden so, dass einen Hashwert des Blockes unter der schwierigkeit liegt:

$$sha256(\ ref \circ kons \circ nonce\ ) < \varepsilon$$

POW ist jedoch umstritten, da das verbrauchen von Rechenressourcen durch strohmkosten, investitionen in neue Hardware und infrastruktur einerseits nicht ökonomisch und zum anderen nicht umweltfreundlich ist. Um die Rechenpower des Bitcoinnetzwerkes in Relation zu setzen hatten die Top 500 Weltbesten Supercomputer der Welt im November 2014 gemeinsam eine Rechenleistung von 309 Pflop/s.[3] Das Bitcoinnetzwerk besitzt derzeit (18.02.2015) eine Rechenleistung

[3] http://www.top500.org/lists/2014/11/ 18.02.2015

von 4172436 Pflop/s.[4].

Neben POW gibt es jedoch auch andere Mechanismen für die Berechtigung einen Block erstellen zu dürfen. Hier gilt "Proof of Stake" (POS) und seine erweiterung "Deligated Proof of Stake"(DPOS) als interessant. Bei POS beweist ein Akteur, dass ihm ein bestimmter Anteil an der Blockchain zugehörigen DAO gehört. Bei DPOS kann ein Anteilseigener zusätzlich sein Anteil an einen Miner deligieren und muss nicht selbst an der erstellung eines Blocks teilnehmen. Jedoch sind hier theoretische Attaken möglich (siehe dazu [Eth] [Vas15]), weshalb diese POW noch nicht abgelöst haben. Bitshares [5] - ein Bitcoin fork mit einem DPOS oder Peercoin[6] ein fork mit reinem POS sind realwirtschaftliche Beispiele für DAOs mit einem Alternativen Konsensmechanismus die zumindest zeigen, dass diese Mechanismen es schaffen ebenfalls in der Praxis zu bestehen.

Das erstellen von Blocks durch Miner ist bei den Blockchain mechanismen der hier vorgestellten DAOs ökonomisch motiviert. Jede DAO schüttet eine Belohnung an den Miner, der einen neuen Block findet. Die Belohnung ist ein interner Token(auch Coin genannt) der aufgrund seiner Knappheit einen Marktwert besitzt. Er kann jedoch auch als Besitzanteil der DAO interpretiert werden oder dient anderen zwecken. Bei Ethereum b.Z. benötigen Akteure diesen Token um in der DAO agieren zu könenn. Im Bitcoin Blockchain Mechanismus werden derzeit (18.02.2015) 25 Bitcoins pro Block ausgeschüttet, was zum derzeitigen Marktpreis (240$/Btc) einem Wert von 6000$/Block entspricht. Zum Zeitpunt der ersten Blocks exestieren noch keine Coin-Bestände, diese werden von dem ersten Block an an konkurierende Miner für die Blockerstellung ausgeschüttet. Die größe der Belohnung wird mit der Zeit kleiner, bis irgendwann eine Menge von 21 Millionen coins ereicht ist. Eine andere Motivation der Miner ist eine Transaktionsgebühr, die ein Akteur seiner Transaktion anhängt. Diese Gebühr bekommt ein Miner, wenn er diese in seinem Block berücksichtigt.

Interaktion

Die Art der Interaktion mit einer DAO hängt von Ihrer Programmierung ab. Handelt es sich um eine Blockchain basierte DAO so ist das bereits beschriebene Erstellen von Blöcken eine Iteraktion sowie das Überweisen von eigenen Coins an einen anderen Akteur. Bitcoin bietet zudem eine einfache Skriptsprache mit der die Akteure kleine Skripte erstellen können wie "Multisignature" bei dem ein Tokenbestand erst freigegeben wird, wenn die mehrheit der angegebenen Adressen ihn freigeben. Diese Skriptsprache ist jedoch nicht Turing-Vollständig. Die Ethereum DAO hingegen besitzt eine turing-vollständige Skriptsprache und wird aufgrund dessen als eine DAO Plattform referenziert. Das ermöglicht das einfache Erstellen von Nicht Blockchain basierter DAOs, da der Berreich der dezentralisierung von der Ethereum Plattform abgenommen wird. Aktionen in

[4] http://bitcoinwatch.com/ 18.02.2015
[5] https://bitshares.org/blog/delegated-proof-of-stake/ 18.02.2015
[6] http://peercoin.net/ 18.02.2015

Ethereum währen uneranderem das hinzufügen eines neuen Programms oder die Übermittlung der Interaktionen an interne DAOs. Akteure können in Ethereum sich einerseits außerhalb der Ethereum DAO befinden, andererseits sind alle DAOs inerhalb von Ethereum ebenfalls Akteure.

Die Interaktion eines Akteurs mit einer Blockchain basierten DAO besitzt folgende Struktur:

Der Akteur:

1. läd sich den für ihn relevanten Teil des Konens herunter
2. Manipuliert diesen, jedoch muss die Manipulation den Regeln der DAO folgen, da sie sonst von den Minern abgelehnt wird
3. Signiert die Manipulation
4. sendet seine Transaktion an ihn bekannte Miner, damit diese zu einen Block hinzugefügt wird.

Der Miner:

1. bekommt von unterschiedlichen Quellen Transaktionen
2. die signatur jeder Transaktion wird validiert: ist der angegebene Akteur auch ihr Erzeuger?
3. die Manipulation des derzeitigen Zustandes wird validiert: ist die Manipulation des aktuellen zustandes nach den aktuellen Regeln valide?
4. falls die Signatur und Manipulation valide ist, wird der derzeitige zustand Aktualisiert und die Transaktion wird der Menge valider, bekannter Transaktionen hinzugefügt, die vom Miner in den nächsten Block hinzugefügt werden sollen.
5. Der Miner versucht nun nach dem "Proof of *" Mechanismus seinen Konsens zum Konsens der DAO zu machen.

Wie berreits erwähnt benötigt ein Akteur für das agieren in der Ethereum DAO ein Bestand an der Interner währung. Wie viel, hängt von der komplexität der ausgelösten Transaktion ab. Für jeden Berechnungsschritt wird 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])

Eine Ethereum interne DAO besteht aus einem assoziativem Speicher der den Programmcode sowie Daten beinhaltet. Diese besitzt eine Adresse und hat einen Bestand an Ether.

Ein Beispiel für eine interne DAO ist eine dezentrale Währung, wie sie derzeit vom Bitcoin 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
```

Ein weiteres Beispiel ist ein dezentrales DNS Systhem wie ihn die Namecoin DAO implementiert:

```
if !contract.storage[msg.data[0]]:
    contract.storage[msg.data[0]] = msg.data[1]
    return(1)
else:
    return(0)
```

Diese beiden Beispiele illustrieren wie vergleichsweise einfach es ist eine DAO auf einer bestehenden Plattform, wie der von Ethereum, zu erstellen, zu den bisherigen Blockchain-Basierten Ansätzen.

(selbst) Modifikationsschicht einer DAO

Durch eine Plattform wie Ethereum wird nun das Erstellen von DAOs um ein vielfaches leichter gemacht, jedoch gibt es noch keine einheitliche Lösungsansätze um die DAOs oder generell dezentrale Inhalte ebenfalls auf eine dezentrale Weise zu erstellen oder diese zu modifizieren. Es Fehlt einer Entkopplung der DAO von den Regeln, wie diese von den Inhabern gerecht, also proportional zu ihrem Besitz, Manipuliert werden können.

Derzeit werden DAOs, entweder von einer einzelnen Schlüsselperson, einer Organisation (wie bei Bitcoin die "Bitcoin-Foundation"[7]) oder einem Unternehmen (wie bei Ethereum "Ethereum Switzerland GmbH"[8]) entwickelt und manipuliert. Trotz der dezentralen natur der DAOs ist die Kontrolle über diese noch Zentralisiert. Einerseits lässt sich der Besitz von Bitcoin und Ethereum nicht determinieren. Falls man die interen Tokens aus der Blockchain Schicht ebenfalls

[7] https://bitcoinfoundation.org/ 25.02.2015
[8] https://www.ethereum.org/ 25.02.2015

als Besitzanteil an den DAOs interpretatiert, so können die Besitzer noch nicht über die Manipulationen der Regeln der DAO mitentscheiden.

Das Problem ist in der Community jedoch bekannt. Im Juni 2014 hat Oliver Janssens, einer der Early Adopter von Bitcoin ein Preisausschreiben von 100 000 USD in BTC an denjenigen ausgeschrieben, der ein Konzept vorschlägt, wie die Bitcoin Foundation auf dezentale weise abgelöst werden kann.

> The Bitcoin foundation [...] is internally recreating the same archaic political system that fails to work for society. Bitcoin is the currency of the internet generation. It puts the power back into the hands of the people. You cannot expect its main representative organisation to be exactly the opposite: A non-transparent, political and secretive elite. (Reddit - Oliver Jansens http://redd.it/25sf4f 19.02.2015)

Der Gewinner war die Bitcoin-Basierte dezentrale Crowdfunding Plattform Lighthouse[9]. Diese erfüllt nicht wirklich die Ansprüche an eine solche selbst-Modifikationsschicht. Der Mechanismus einer solchen Schicht muss einerseits eine große Anzahl an Stimmberechtigten Kandidaten (derzeit wurden rund 200k Bitcoin Adressen verwendet) berücksichtigen sowie eine ähnlich große Menge an Vorschlägen. Er sollte schnell und agil auf aufkommende Probleme reagieren können, jedoch niedrige Einstiegsbarrieren für Benutzer besitzen.

Im folgenden möchte ich meinen Vorschlag einer solchen Schicht vorstellen. Hierfür werden wir als erstes die theoretischen Komponenten betrachten, dann ihre Anwendung auf die regulären Grammatiken und schließlich ihre Implementation.

Transitionen

Theoretisch kann eine solcher Mechanismus durch ein **initialisiertes Transitionssysthem** $T = (M, I, \tau)$ nach [Gla] modelliert werden. Hierzu muss eine passende Zustandsmenge definiert werden, einen Initialzustand sowie die Übergangsfunktion, die valide Manipulationen (Transitionen) der DAO beschreibt.

Durch die Dezentralisierungsschicht wird sichergestellt, dass Transitionen **immer** von einem Akteur a ausgelöst werden und das dieser Akteur ebenfalls ihr Urheber ist. Damit besitzen eine Transition die Form (a, O'). O' beinhaltet dabei die Manipulationen des Akteurs.

Definition

In der selbst-Modifikationsschicht können wir davon ausgehen, dass für die dezentralität der DAO bereits gesorgt ist. Es handelt sich entweder selbst

[9] https://www.vinumeris.com/lighthouse

um eine Blockchain-Basierte DAO oder eine, die eine Plattform wie Ethereum benutzt. Insofern gilt die dezentralität im folgenden als gegeben.

Eine DAO in einer Grammatik G ist ein Tupel $O_G = (A, K, share, vote)$ bestehend aus:

1. eine endliche Menge von Akteuren A
2. eine eindeutige **Besitzverteilung** von Akteuren zur DAO
$$share : A \to \mathbb{N}$$
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.
$$vote : A \times K \to [0, 1]$$

Sei $\mathbf{DAO_G} = \{O_G \mid O_G$ ist DAO in der einer Grammatik $G\}$ die Menge aller DAOs in der Grammatik G.

Zudem exestiert eine Konsensfunktion die eine DAO in eie valides Wort überführt.

$$consens : DAO_G \to L(G)$$

Erläuterung

Grammatik

TODO: Grammatik

Besitzverteilung

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

Die Funktion $share : A \to \mathbb{N}$ gibt den Anteil von O an, der im Besitz von einem Akteur ist.

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

Der Besitz einer DAO O wird durch das Recht definiert, dieses Kontrollieren zu können[Wal04]. Ist der Besitz unter mehreren Akteuren aufgeteilt, so gibt

der Anteil am Objekt an, zu welcher Gewichtung jeder einzelne Akteur über die DAO mitbestimmen kann.

Eine Konsensfunktion entscheidet schließlich über die Ausführung der Kontrolle.

Für die bestimmung der Besitzverteilung können einerseits die Tokens benutzt werden, die in der Dezentralisierungsschicht verwendet werden. Andererseits können auch neue eingeführt werden. Für den Blockchain basierten Ansatz der dezentralisierung muss bei getrennten Besitztokens jedoch sicher gestellt werden, dass die Tokens der Blockchain-Schicht einen Wert für die Miner und damit eine motivierende Funktion am mining Prozess teilzunehmen besitzen.

Konsens

$$consens : DAO_G \to L(G)$$

Die Konsensfunktion überführt jede DAO in ein Wort 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.

$$value(k) := \sum_{a \in A} share(a) * vote(a, k)$$

$$consens(O_G) := k \text{ mit } value(k) = \max_{k' \in K_G} (value(k'))$$

Hier entsteht das Problem, dass es mehrere Kandidaten mit der selben, maximalen Bewertung exestieren. In diesem Fall muss ein mechanismus existieren, der ein k determiniert. Dieser wird in der Implementation nähr betrachtet. Für diese kommt eine strickte Totalordnung für die Kandidatenmenge hinzu, die die Reihenfolge bestimmt, in der diese der Kandidatenmenge hinzugefügt wurden. der Mechanismus entscheidet sich hier immer für den ersten Kandidaten der Ordnung.

TODO: muss diese strickte Totalordnung auch ein element von O sein?

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.

Transitionsysthem

Als Zustandsmenge für das Transitionssystem können wir nun die Menge aller möglichen DAOs nehmen. Allerdings muss man die Menge für die Implementation codieren sowie die Validitätsbedingungen auf der codierten Menge definieren. Eine passende codierung die wir näher betrachen wollen ist eine DAO zu einem Wort einer Sprache zu machen. Diese Sprache nennen wir **Metasprache** der ursprünglichen Sprache ($L(G)$). Eine Bedingung muss jedoch sein, dass das Wort ohne Verlust von Informationen wieder zurück zur selben DAO decodiert werden kann. Damit können wir Transitionsbedingungen als Manipulationen des codierten Wortes definieren.

Da die Wörter der codierten DAOs ebenfalls valide und invalide Teile der Wörter der ursprünglichen Sprache ($L(G)$) erkennen müssen und somit die Metasprache konstruierende Grammatik, ebenfalls die ursprüngliche Grammatik beinhalten muss, wird eine Funktion S angegeben, die die ursprüngliche Grammatik G so erweitert, dass diese zur gesuchten Metasprache wird $L(S(G))$.

Formaler gesagt muss eine Funktion $S : REG \to CFG$ gefunden werden, zusammen mit den Funktionen $\phi : DAO_G \to L(S(G))$ und $\phi^{-1} : L(S(G)) \to DAO_G$ für die die Eigenschaft $\phi \circ \phi^{-1} = id_{DAO_G}$ gilt.

Da die Funktion S, ϕ sowie ϕ^{-1} viele Komponenten haben, werden sie inkrementell eingeführt: Zuerst wird S_1, ϕ_1, ϕ_1^{-1} definiert, diese codieren und decodieren die Kandidatenmenge einer DAO in ein Wort. Anschließend wird die Besitzverteilung sowie die Kandidatenbewertung ebenfalls eingearbeitet, sodass wird die gesuchten Funktionen erhalten.

codierung 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.

Dazu muss es eine Grammatik-Erweiterung $S_1 : REG \to CFG$ gefunden werden. Um zu zeigen, dass es sich bei S_1 um die gesuchte Erweiterung handelt, muss die Existenz der Funktionen $\phi_1 : P(L(G)) \to L(S(G))$ und $\phi_1^{-1} : L(S(G)) \to P(L(G))$ gezeigt werden, für die die Bedingungen $\phi_1 \circ \phi_1^{-1} = id_{P(L(G))}$ erfüllt ist.

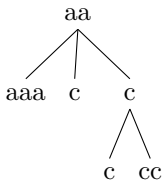
Dieses kann mit hilfe eine Trie-Datenstruktur realisiert werden, die redundant auftauchende Präfixe der Kandidaten zusammengefasst. Die preorder Serialisierung der Tries ist eben ein Wort der Metasprache, die durch die Grammatikerweiterung S_1 konstruiert wird:

Wir wissen, dass es eine Funktion $c : K_G \mapsto T(K_G)$ sowie $c^{-1} : T(K_G) \mapsto K_G$ exestiert, für die $c \circ c^{-1} = id_{P(L(G))}$ gild [Mor68]. Zu finden ist demnach die ein Isomorphismus $s : T(K_G) \mapsto w \in L(S_1(G))$ sowie $s^{-1} : w \in L(S_1(G)) \mapsto T(K_G)$

die ein Tire serialisieren und deserialisieren. Für die Serialisierung wird einfach die preorder notation verwendet. Damit haben wir die Funktionen $\phi_1 := c \circ s$ sowie $\phi_1^{-1} := c^{-1} \circ s^{-1}$ gefunden.

Dieses möchten wir anhand eines Beispiels für die Sprache $L(G) = \{a^x b^y c^z \mid x, y, z \in \mathbb{N}\}$ und die Kandidatenmenge $\{aaaaa, aac, aacc, aaccc\}$ illustrieren.

Sei $T(K_G)$ die Trie Datenstruktur, die die Kandidatenmenge K_G codiert:



Mit der dazugehörigen preorder Serialisierung mit [,] als Klammerzeichen sowie & als Trennzeichen:

$$aa[aaa\&c\&c[c\&cc]]$$

Für die grammatikerweiternde Funktion S_1 wird an jedem Ableitungsschritt in der Grammatik G eine Mehrdeutigkeit zugelassen. Dafür werden die Produktionsregeln folgendermaßen erweitert:

$$P_{Options} := \{R \to [O_R], O_R \to r\&O_R, O_R \to r \mid R \to r \in P \wedge r \in \Sigma^*\}$$

Diese Mehrdeutigkeit ($[O_R]$) nennen wir **Optionsmenge** sowie die darin enthaltetnen Elemente (r) **Option**.

Eine solche Grammatikerweiterung erzeugt eben die Serialisierung einer Trie:

$$aa[aaa\&c\&c[c\&cc]] \in L(S_1(G))$$

Grammatik-Erweiterung S1

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

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

TODO: ist hier ein formaler Beweis notwendig?

Besitzverteilung

Die Akteure lassen sich durch ein 20-Byte String genau identifizieren. Sei $HASH$ die Menge aller 20-Byte Strings. Die Besitzverteilung ist eine Auflistung aller beteiligter Akteure sowie deren Anteile ander DAO. Sei $hash \in HASH$ sowie $number \in \mathbb{N}$.

$$P_{Acteurs} := \{A \to [hash\ number]A, A \to \varepsilon\}$$

Kandidatenbewertung

Um jede DAO ohne Verlust von Inforamtionen codieren zu können, muss jede mögliche Bewertungsverteilung von Akteuren und Kandidaten im Metawort enthalten sein können. Dieses kann erreicht werden, wenn die Bewertung der Kandidaten an Optionen gebunden sind. Da es für jeden Kandidaten aus der Kandidatenmenge genau eine Option gibt, die nur diesem Kandidaten zugeordnet ist (ein Blatt eines Tries), lässt sich jede Bewertungsverteilung des Kandidaten codieren, wenn sie dieser Option zugeordnet sind. Zusätzlich möchten wir Bewertungen einer Option zulassen, die mehreren Kandidaten zugeordnet sind. Dies sind solche, die wiederum eine Optionsmenge beinhalten. Wenn ein Akteur eine Option in einer Optionsmenge bewertet, so gibt er seine Stimme der Option im Kontext zu dem bisherigen Präfix des Wortes. Die Bewertung dieser Option wird als ein Attribut in der Ableitung Synthetisiert[Knu68] und jeweils von einer Bewertung einer tieferen Ebene ersetzt.

Sei $FLOAT = [0, 1]$ die Menge der maschinell darstellbaren Fließkommazahlen zwischen 0 und 1: $float \in FLOAT$

$$P_{Voting} := \{V \to [hash\ float]V, V \to \varepsilon\}$$

Delegationen

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

Ein solcher Akteur kann wiederrum entweder eine Person, ein Zusammenschluss von Personen wie eine Interessensgemeinschaft oder eine Partei in Form einer DAO oder eine DAO ohne Fremdeinwirkung sein, die bestimmte Werte zu optimieren versucht.

Delegationen werden für Optionsmengen verärbt. Durch die lineare Struktur der Worte ergibt sich eine strickte Totalordnung der Delegationen, womit bei konkurierenden Delegationen immer die Delegation höherer Ordnung genommen

wird. Eine Stimme des Akteurs selbst, wird einer Stimme eines Deleganten vorgezogen.

$$P_{Delegations} := \{D \rightarrow [hash\ hash]D, D \rightarrow [hash\ hash]\}$$

TODO: roter faden

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:

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

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.

Grammatik-Erweiterung S

TODO: NUBER, HASH, FLOAT teil der Terminale

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.

$$
\begin{aligned}
S(G) :=& (N', T', S', P') && (1)\\
N' :=& N \cup \{O_R | (R \to r) \in P \wedge O_R \notin N\} \cup \{D, A, S'\} && (2)\\
T' :=& T \cup \{[,], \&, number, float, hash \mid [,], \& \notin \Sigma\} && (3)\\
P_{Acteurs} :=& \{A \to [hash\ number]A, A \to \varepsilon\} && (4)\\
P_{Options} :=& \{R \to [O_R][D], O_R \to r\&\ [V]O_R, O_R \to \varepsilon\\
& \mid R \to r \in P \wedge r \in \Sigma^*\} && (5)\\
P_{Start} :=& \{S' \to [A][O_S][D]\} && (6)\\
P_{Delegations} :=& \{D \to [hash\ hash]D, D \to [hash\ hash]\} && (7)\\
P_{Voting} :=& \{V \to [hash\ float]V, V \to \varepsilon\} && (8)\\
P' :=& P \cup P_{Options} \cup P_{Start} \cup P_{Delegations} \cup P_{Voting} \cup P_{Acteurs} && (9)
\end{aligned}
$$

zeige S(G) ist eindeutig

Leider gibt es kein Algorithmus, der beweist, dass S(G) eindeutig ist. Die eindeutigkeit ist jedoch ein notwendiges kritärium. Zumindest wurde die hier vorgestellte erweiterung für einige reguläre Grammatiken eindeutig von einem LALR(1) Parser akzeptiert wurde, was zumindest für diese eine eindeutigkeit zeigt. Wie man beweist, dass dieses für alle regulären gammatiken der Fall ist, muss noch gefunden werden.

TODO: AUF JEDEN FALL ÜBERARBEITEN - wie eindeutigkeit zeigen?

Konsens

Konsens

Nach dem die Kandidaten, die Akteure sowie ihre Anteile und die Bewertung der Kandidaten durch die Akteure in der Metasprache enthalten sind erlaubt dies den Konsens der DAO als Interpretation des Wortes der $L(S(G))$ Sprache zu definieren.

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

Die Idee hier ist, dass die transitive Hülle der Delegationen als Attribut während der Ableitung vererbt(inherited) wird. Die Stimmenge wird hingegen als Attribut synthetisiert.[Knu68] Delegationen zusammen mit den Stimmen quantifizieren jede Option aus einer Optionsmenge und wählen eine Konsens-Option mit den Maximalsten Stimmen höhster Priorität.

Transitionssystem

Zur Wiederholung besteht ein Transitionssystem $T = (M, I, \tau)$ aus einer Zustandsmenge M, einem Initialzustand I und einer Übergangsfunktion τ. Nach dem wir nun die Zustandsmenge als Metasprache $M = L(S(G))$ sowie die codierung und decodierung einer DAO in ein Wort der Metasprache gefunden haben, widmen wir uns der Übergangsfunktion τ.

Als Initialzustand wird ein beliebiges Wort aus der Metasprache verwendet, welches weder Stimmen, noch Deligationen beinhaltet.

$$I = \{s_0 \in L(S(G))\}$$

Wir werden uns auf die Manipulation der Kandidatenmenge sowie der Kandidatenbewertung beschränken. Weitere Manipulationen werden in Aussicht gestellt. Valide Manipulationen der Kandidatenmenge soll das Erstellen und das Erweitern von Optionsmengen sein. Für die Kandidatenbewertung sollen das Hinzufügen und Entfernen der eigenen Optionsstimmen sowie der eigenen Delegationen für Optionsmengen valide sein.

Falls die Besitzverteilung von der Blockchain-Schickt übernommen wird, greift die dynamik dieser Schickt ebenfalls auf den derzeitigen Zustand. Wir möchten uns jedoch auf solche DAOs beschränken, bei denen die Besitzverteilung ausschließlich in der Selbstmanipulations-Schicht manipuliert wird. Hier möchten wir das Überweisen eines Akteurs von eigenen Anteilen an einen anderen Akteur zulassen.

Aktualisierung

Durch die Blockchain-Schicht ist jede Interaktion mit einer DAO eindeutig und nachweislich einem Akteur zugeordnet. Demnach ist jede Aktualisierung des derzeitigen Zustandes $w := s_i$ ein Tupel $\delta = (a, w')$ mit $hash = a \in A = HASH,\ w' \in L(S(G))$.

Sei $\Delta \subseteq A \times L(S(G))$ die Menge aller vom Miner gesehenen Aktualisierungen, die noch nicht auf validität beprüft wurden und somit nicht Teil seines Konsenses sind.

Akteure können jederzeit die Aktualisierungsmenge erweitern in dem sie eine Aktualisierung an einen Miner schicken:

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

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.

Manipulation der Kandidaten

Eine Optionsmenge kann an jeder Position im Wort erzeugt oder erweitert werden, solange das neu vorgeschlagene Wort, ein Wort der Metasprache ist, sowie die Bewertungen der Kandidaten nicht verändert werden.

$$(A, K', share', vote') = \phi^{-1}(w') \wedge (A, K, share, vote) = \phi^{-1}(w) \Rightarrow vote = vote' \quad (10)$$

$$w' \in L(S(G)) \quad (11)$$

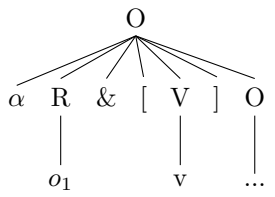
Die neu hinzukommenden Optionen dürfen keine Delegationen oder Stimmen enthalten, sowie noch nicht in der bisherigen Optionsmenge enthalten sein.

Erzeugen einer Optionsmenge

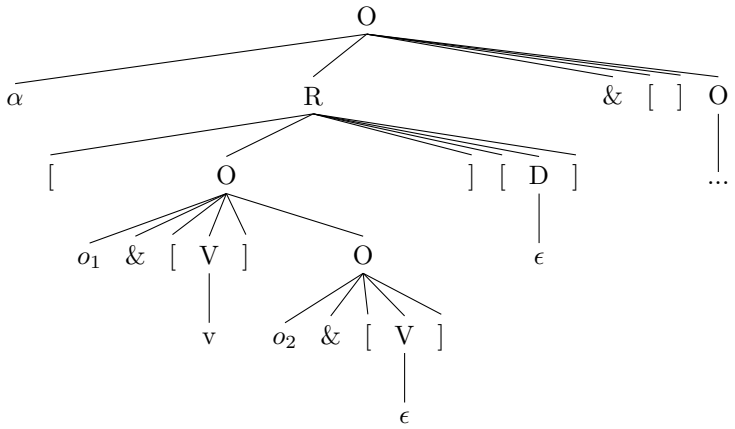
Sei $S(G) = (T', N', S', P')$ eine erweiterte Grammatik. Sei weiter $O, S' \in N'$ sowie $v, o_1, o_2 \in T'^*$, $o_1 \neq 0_2$ und $0 \leq i \leq j \leq n$ mit $i, j, k, n \in \mathbb{N}$

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

Für w hat O folgende Struktur:



Für w' besitzt O folgende Struktur:



Dabei muss o_2 frei von Delegationen und Stimmen sein.

TODO: ist hier ein beweis das die kritärien nicht verletzt werden notwendig?

Erweitern einer Optionsmenge

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

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

r muss dabei Delegations- und Stimmfrei sein sowie neu:

$$(A, K', share', vote') = \phi^{-1}(w_{0..i-1}\ \alpha[r\&[]][d]\ w_{j+1..n}) \tag{25}$$
$$\wedge(A, K, share, vote) = \phi^{-1}(w) \tag{26}$$
$$\Rightarrow K' \cap K = \emptyset \tag{27}$$

Manipulation der Kandidatenbeertung

Sei für alle Manipulationen $a \in HASH$ der auslösende Akteur.

Hinzufügen einer Stimme

Sei $b_1, b_2, v, r \in T'^*$

$$w = w_{0..i-1}\ w_{i..j}\ w_{j+1..n} \tag{28}$$
$$v \neq b_1 a b_2 \tag{29}$$
$$w_{i..j} = r\&[v] \tag{30}$$
$$w' = w_{0..i-1}\ r\&[v[a\ n]]\ w_{j+1..n} \tag{31}$$

Löschen einer Stimme

Sei $v_1, v_2 \in T'^*$ sowie $n \in \mathbb{N}$

$$w = w_{0..i-1}\ w_{i..j}\ w_{j+1..n} \tag{32}$$
$$w_{i..j} = r\&[v_1[a\ n]v_2] \tag{33}$$
$$w' = w_{0..(i-1)}r\&[v_1\ v_2]w_{(j+1)..n} \tag{34}$$

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 $h' \in HASH$ und $h \neq a$

$$w = w_{0..i-1}\ w_{i..j}\ w_{j+1..n} \tag{35}$$
$$u, v \in \{w | w = ([h_1\ h_2])^* \text{ mit } h_1, h_2 \in HASH\} \tag{36}$$
$$w_{i..j} = [o][uv] \tag{37}$$
$$w' = w_{0..(i-1)}\ [o][u[a\ h']v]\ w_{(j+1)..n} \tag{38}$$

Löschen einer Delegation

Sei $o, d_i \in T'^*$ sowie $h' \in HASH$

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (39)\\
u, v &\in \{w | w = ([h_1\ h_2])^* \text{ mit } h_1, h_2 \in HASH\} && (40)\\
w_{i..j} &= [o][u[a\ h']v] && (41)\\
w' &= w_{0..(i-1)}\ [o][uv]\ w_{(j+1)..n} && (42)
\end{aligned}
$$

Überweisung

Ein Akteur kann seine Anteile an einen anderen überweisen. Dabei müssen die Fälle unterschieden werden, ob der Empfänger bereits anteile hällt, bei dem Fall wird die Überweisungssumme seinen Beständen addiert, oder ob es sich um einen neuen Akteur handelt. Dieser wird mit der Überweisungssumme neu hinzugefügt.

Sei $h' \in HASH$ mit $h' \neq a$ sowie $l, m, n \in \mathbb{N}$

TODO: hier ist zum ersten mal nicht die symbolkonkatination gemeint sondern die interpretation der werte(+,-)

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (43)\\
b_1, b_2, b_3, b_4, b_5, b_6 &\in \{w | w = ([h\ n])^* \text{ mit } h \in HASH \text{ und } n \in \mathbb{N}\} && (44)\\
w_{i..j} &= [b_1 b_2 b_3[a\ n]b_4 b_5 b_6] && (45)\\
m' &= (m + l) && (46)\\
n' &= (n - l) && (47)\\
n - l &\geq 0 && (48)\\
\text{Fall 1:} & \quad b_2 = [h'm] && (49)\\
w' &= w_{0..(i-1)}\ [b_1[h'm']b_3[a\ n']b_4 b_5 b_6]\ w_{(j+1)..n} && (50)\\
\text{Fall 2:} & \quad b_5 = [h'm] && (51)\\
w' &= w_{0..(i-1)}\ [b_1 b_2 b_3[a\ n']b_4[h'm']b_6]\ w_{(j+1)..n} && (52)\\
\text{Fall 3:} & && (53)\\
w' &= w_{0..(i-1)}\ [b_1 b_2 b_3[a\ n']b_4 b_5 b_6[h'l]]\ w_{(j+1)..n} && (54)
\end{aligned}
$$

Beispiele

abc - Grammatik

reddit - Grammatik

memGenerator - Grammatik

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:
- 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[10], eine javascript Portierung des Bison[11]/Flex[12] LALR(1) Parser Generators wird so erweitert, dass bei Eingabe einer beliebigen regulären Grammatik G im bison Format[13] 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.

[10] http://jison.org/
[11] http://www.gnu.org/software/bison/
[12] http://flex.sourceforge.net/
[13] http://dinosaur.compilertools.net/bison/bison_6.html

- Einige anschauliche Beispiele regulärer Sprachen mit beispielhaften Delegationsprogrammen werden implementiert und demonstriert.

Todos

Aufgaben

Zusammenfassung und Ausblick

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.

[CN14] The Internet Corporation and Assigned Names. Identifier Technology Innovation Panel - Draft Report. 2014.

[Eth] https://blog.ethereum.org/2014/07/05/stake/ - Proof of Stake 18.02.2015.

[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.

[Lin10] Lindenberg. Konzeption und Erprobung einer Liquid Democracy Plattform anhand von Gruppendiskussionen. 2010.

[Mor68] Donald R. Morrison. PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, 15:514–534, 1968.

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

[Vas15] Khushita Vasant. A Treatise on Altcoins. Technical report, 2015.

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

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