Ein dezentrales Transitionssystem zur Manipulation von geteilten Wörtern einer regulären Sprache

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

Humboldt-Universität zu Berlin
Institut für Informatik

eingereicht von:	Denis Erfurt
geboren am:	02.04.1988
in:	Novosibirsk
Gutachter(innen):	Prof. Dr. Klaus Reinhardt
	Prof. Dr. Jens-Peter Redlich

eingereicht am: ..... verteidigt am: .....

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Bachelorarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 30. Juni 2015 .....

Statement of authorship

I declare that I completed this thesis on my own and that information which has been directly or indirectly taken from other sources has been noted as such. Neither this nor a similar work has been presented to an examination committee.

Berlin, June 30, 2015 .....

Zusammenfassung

Im Internet werden zunehmend Inhalte kollaborativ erzeugt. Dabei entsteht ein Inhalt durch Beiträge einzelner Akteure, die räumlich und zeitlich getrennt agieren 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 vorgestellt. Dieses Model beinhaltet: (1) Ein öffentliches Transitionssystem mit dezentral validierter Ausführung. (2) 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. (3) Die Konsens-Funktion überführt unter Berücksichtigung der Bewertungen ein Wort der erweiterten Grammatik in ein Wort der ursprünglichen Grammatik. (4) Schließlich wird eine zentralisierte Implementation vorgestellt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Angenommen 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 einer Mehrheit der Besitzer bestimmt.

Wikipedia folgt einem streng hierarchischen 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[Fri10], ein Vorschlag der Piratenpartei für eine moderne politische Konsensbildung. Ein solches Prinzip könnte auf das Verwalten aller digital geteilter Inhalte verallgemeinert werden.

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 eine formale 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 dem 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 auch 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, Decentralized-Autonomous-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 die ICANN ablösen könnte.[Cor14] Die Regeln unter denen DACs und DAOs funktionieren, wie das Bewilligen einer neuen TLD, werden auf der Ethereum Plattform programmiert. Die Einigung der Akteure auf ein Programmstand geschieht noch durch eine zentrale Vertrauensinstanz, z.B. bestimmten Schlüsselpersonen.

Leistung und Struktur der Arbeit

In dieser Arbeit wird versucht die Frage zu beantworten, wie eine Menge von Akteuren auf eine dezentrale Weise einen Konsens über ein Wort einer regulären Sprache bilden kann. Die Arbeit besteht im Wesentlichen aus drei Teilen: Sie fasst notwendige theoretische Grundlagen zusammen. Auf diese aufbauend wird ein Modell für eine Selbst-Modifikation-Schicht einer DAO ausgearbeitet und auf DAOs mit regulären Grammatiken angewandt. Schließlich wird das ausgearbeitete Modell implementiert.

Die weitere Arbeit ist wie folgt aufgebaut: In Kapitel 2 werden theoreti-

[1] http://namecoin.info/

[2] Internet Corporation for Assigned Names and Numbers

sche Konzepte beschrieben auf, denen diese Arbeit aufbaut. Des Weiteren wird in Kapitel 3 eine ausführliche Definition einer Dezentralen-Autonomen-Organisation gegeben, wie sie zum Zeitpunkt der Arbeit vorliegt. Darauf aufbauend wird das Problem der Selbst-Modifikation erläutert und in Kapitel 4 eine Definition, sowie eine Anwendung auf die regulären Sprachen erarbeitet. In Kapitel 5 wird eine Implementation der Anwendung vorgestellt. Schließlich wird in Kapitel 6 ein Ausblick gegeben.

Kapitel 2

Theoretische Grundlagen

Dieses Kapitel fasst theoretische Grundlangen zusammen, auf denen die nachfolgenden Kapitel aufbauen. In Abschnit 2.1 werden die regulären Grammatiken vorgestellt, sowie die kontextfreien Grammatiken in Abschnit 2.2. In Abschnit 2.3 wird die Trie Datenstruktur betrachtet. In Abschnit 2.4 wird das initiale Transitionssystem eingeführt. In Abschnit 2.5 wird die asymmetrische Verschlüsselung betrachtet.

2.1 Reguläre Grammatiken

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

1. N - Eine Menge nichtterminaler Symbole.
2. T - Eine Menge terminaler Symbole, mit $T \cap N = \emptyset$.
3. $S \in N$ - Eine Startsymbol.
4. $P \subseteq N \times \{\epsilon\} \cup T \cup TN$ - eine Menge von Produktionen oder auch Produktionsregeln.

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, dass das Nichtterminal R in einem Schritt zum Teilwort r überführt werden kann:

$$\alpha R \to \alpha r \tag{2.1}$$

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

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

$$L(G) := \{w | S \to^* w \wedge w \in T^*\} \tag{2.2}$$

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:

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

$$L(G) = \{a^x b^y c^z | x, y, z \in \mathbb{N}\} \tag{2.3}$$

2.2 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: $P \subseteq N \times (N \cup T)^*$.

Wortteil

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

$$w_{i..j} := \begin{cases} (t_k)_{i \leq k \leq j} \text{ mit } t_k \in T & : \ i \leq j \\ \varepsilon & : \ i > j \end{cases} \tag{2.4}$$

2.3 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. Weiter dürfen keine Kinder eines Knotens mit gleichem Namen existieren. Eine zu 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 $\{aaaaa, aab, aba, abb, aa\}$ zeigt die Abbildung 2.1.

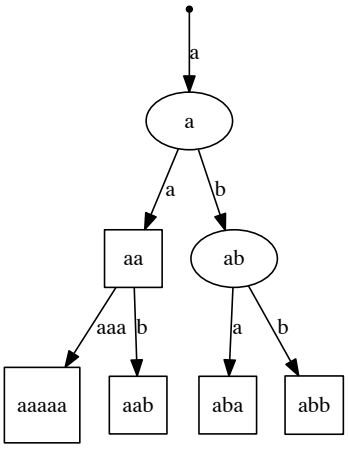


Abbildung 2.1: Beispiel einer Patricia-Trie Datenstruktur

Im Folgenden wird mit einer modifizierten Form der Trie gearbeitet: Für die Wiederherstellung der Wortmenge wird für jeden Knoten eine zusätzliche Information benötigt, ob es sich bei diesem um ein Blatt handelt (In Abbildung 2.1 als Rechteck dargestellt). Darauf kann verzichtet werden indem zwei Kindern eines Knotens erlaubt wird den selben Namen zu tragen, solange einer von beiden ein echtes Blatt, also ein Knoten ohne Nachfolger ist. (Siehe als Beispiel Abbildung 2.2) Dadurch wächst zwar die Größe der Datenstruktur, jedoch bleibt die Eindeutigkeit erhalten, wenn nur noch Pfade von der Wurzel bis zu echten Blättern für die Wiederherstellung der Wortmenge zugelassen werden. Eine solche modifizierte Trie ist zu einer normalen Trie isomorph. Eine Beobachtung der modifizierten Trie ist, dass die Anzahl der echten Blattknoten gleich der Anzahl der Elemente der Wortmenge ist. Dieses erleichtert die weitere Arbeit mit der Datenstruktur.

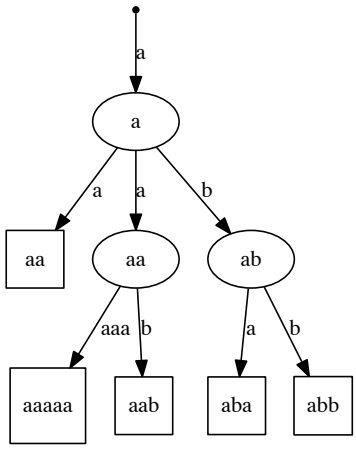


Abbildung 2.2: Modifizierte Patricia-Trie Datenstruktur

2.4 Transitionsystem

Ein initialisiertes Transitionssystem (T) [Gla03] 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 $I \subset M$.

$$T := (M, I, \tau) \tag{2.5}$$

Ablauf

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

$$s_0 \in I \tag{2.6}$$
$$s_{i+1} = \tau(s_i) \tag{2.7}$$

2.5 Asymmetrische Verschlüsselung

Bei einem asymmetrischem Verschlüsselungsverfahren generiert der Benutzer ein Schlüsselpaar bestehend aus einem privaten und einem öffentlichen Schlüssel. Der öffentliche Schlüssel wird aus dem Privaten erzeugt und veröffentlicht. Im Folgenden wird der öffentliche Schlüssel ebenfalls Adresse genannt. Diese identifiziert einen Akteur.

Für die Signierung und Verifizierung von Nachrichten wird ECDSA mit der secp256k1 Kurve verwendet. ECDSA ist ein DSA Verfahren basierend auf der Eliptic Curve Kryptografie. ECDSA bietet viele Vorteile zu alternativen Verfahren wie RSA[1] oder DH[2]. [Pou14]

[1]Rivest, Shamir und Adleman - http://de.wikipedia.org/wiki/RSA-Kryptosystem 25.02.2015

[2]Diffie–Hellman http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange 25.02.2015

Kapitel 3

Decentralized Autonomous Organization (DAO)

In diesem Kapitel wird der Begriff der Decentralized Autonomous Organization ausführlich betrachtet und diskutiert. Dabei folgt nach einer Einführung in Kapitel 3.1 die Betrachtung der dezentralisierenden Technologie der Blockchain. In Kapitel 3.2 wird die Interaktion mit einer DAO betrachtet.

Eine DAO ist ein relativ neues Konzept, welches durch das Populär-werden von Technologien, wie Bitcoin und Ethereum, erstmals ins Interesse der Fachweld gerückt ist. Diese ist sich über die genaue Definition und die Abgrenzung zu verwanten Konzepten größtenteils noch uneinig.

> One of the most popular topics in the digital consensus space [...] is the concept of decentralized autonomous entities. [...] However, one of the hidden problems lurking beneath the space is a rather blatant one: no one even knows what all of these invididual terms mean. (Vitalik Buterin - Ethereum Founder - 06.05.2014[1])

Zwar existieren 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 Terminologie, sowie ihrer Definitionen. Deshalb werden im Folgenden die Begriffe DA, Dapp, DAC, DAO synonym verwendet.

[1] https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/

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 sind nicht nur der Programmcode öffentlich, sondern auch ihre internen Berechnungen sowie wesentliche Teile ihrer Interaktionen. 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 kommunizieren und so auf Services zugreifen oder nicht selbst ausführbare Aufgaben, wie beispielsweise das Erweitern und Verbessern des Programmcodes, an Menschen delegieren. 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 DAOs. Ethereum ist dabei gleichzeitig eine Plattform für andere DAOs.

Im folgenden werden drei Teilbereiche von DAOs näher betrachtet: Als erstes die Funktionsweise des Blockchain Mechanismus, der eigentlichen Neuerung, die zum Entstehen von DAOs führte. Als nächstes den Teil, der die Interaktionen der DAO steuert. Als letztes die Selbstmanipulation der DAO, also den Teil, welcher die Manipulation der Regeln der DAO steuert.

3.1 Dezentraler Konsens - Blockchain

Bitcoin hat mit der Einführung der Blockchain eine elegante und universelle Lösung für das Problem der Byzantinischen Generäle(BG) vorgeschlagen. Das Problem steht repräsentativ im Bereich der verteilten Systeme für die Schwierigkeit, einen Konsens zwischen Akteuren herzustellen, wenn einige, für Andere unbestimmte Akteure, eigennützig sowie manipulativ agieren.

Bitcoin hat hier eine statistische Lösung präsentiert: Es führt die Dezentralität des Konsens auf die natürliche Knappheit 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 daraufhin mit dieser Lösung weiter und versucht sie zu erweitern, was wiederum statistisch gesehen 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 Erweiterungen konvergiert nun die Beteiligung loyaler Generäle am Konsens zur Majorität, falls diese tatsächlich im Besitz einer Mehrheit der begrenzten Rechenressourcen sind. In einem Block befindet sich neben der vorhergehenden Lösung auch der aktuelle Konsens, solange er auf dem Vorhergehenden aufbaut und valide ist.

> The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.
>
> (Satoshi Nakamoto - 2008 [Nak08])

Jedoch gibt es Argumente dafür, dass die Blockchain das Problem der Byzantinischen Generäle nicht vollständig löst. Die Bitcoin - Blockchain hatte jedoch nicht den Anspruch, das BG Problem vollständig zu lösen. Es löst ledignlich eine schwächere Version, nämlich die der Einigung der Generäle.

> This is because in case the adversary finds a solution first, then every honest player will extend the adversary's solution and switch to the adversarial input hence abandoning the original input.
>
> ...
>
> Nakamoto's protocol does not quite solve BA since it does not satisfy Validity with overwhelming probability.
>
> (Garay et. al. 2014 [Gar15])

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 muss, 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 auf. Es ist sehr einfach, einen Hash einer Nachricht zu berechnen, jedoch sehr schwer aus einem gegebenem Hash eine Nachricht zu rekonstruieren, die diesen Hash erzeugt.

Der eigentliche 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 eine, 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, damit ein Hashwert des Blockes unter der Schwierigkeit liegt:

$$sha256(\ ref \circ kons \circ nonce\ ) < \varepsilon \tag{3.1}$$

POW ist jedoch umstritten, da das Verbrauchen von Rechenressourcen durch Stromkosten, Investitionen in neue Hardware und Infrastruktur einerseits nicht ökonomisch und zum anderen nicht umweltfreundlich ist, siehe dazu [BBH+11]. Um die Rechenpower des Bitcoin-Netzwerkes in Relation zu setzen, hatten die Top 500 weltbesten Supercomputer im November 2014 gemeinsam eine Rechenleistung von 309 Pflop/s.[2] Das Bitcoinnetzwerk besitzt derzeit (18.02.2015) eine Rechenleistung von 4172436 Pflop/s.[3].

Neben POW gibt es jedoch auch andere Mechanismen für das Recht einen Block erstellen zu dürfen. Hier gilt “Proof of Stake” (POS) und seine Erweiterung “Deligated Proof of Stake”(DPOS) als wesentlicher Konkurrent. Bei POS beweist ein Akteur, dass ihm ein bestimmter Anteil an der Blockchain zugehörigen DAO gehört. Bei DPOS kann ein Anteilseigner zusätzlich seinen Anteil an einen Miner delegieren und muss nicht selbst an der Erstellung eines Blocks teilnehmen. Jedoch sind hier theoretische Attacken möglich (siehe dazu [Eth] [Vas15]), weshalb diese POW noch nicht abgelöst haben.

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

Bitshares [4] - ein Bitcoin Fork mit einem DPOS oder Peercoin[5], ebenfalls ein Fork mit reinem POS, sind jedoch real wirtschaftliche Beispiele für DAOs mit einem alternativen Konsensmechanismus, die zumindest zeigen, dass diese Mechanismen es schaffen 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 aus, 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 benötigen Akteure diesen Token, um in der DAO agieren zu können. Im Bitcoins 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 Zeitpunkt des ersten Blocks existieren noch keine Coin-Bestände, diese werden von dem ersten Block an konkurierende Miner für die Blockerstellung ausgeschüttet. Die Größe der Belohnung wird mit der Zeit kleiner, bis irgendwann eine Menge von 21 Millionen Coins erreicht 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.

3.2 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 sowie das Überweisen von eigenen Coins an einen anderen Akteur eine Iteraktion. Bitcoin bietet zudem eine einfache Skriptsprache, mit der die Akteure kleine Skripte erstellen können. Beispielsweise kann bei “Multisignature” ein Coinbestand erst freigegeben werden, 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. Dies ermöglicht das einfache Erstellen von nicht

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

Blockchain basierten DAOs, da der Bereich der Dezentralisierung von der Ethereum Plattform abgenommen wird. Aktionen in Ethereum wären unter anderem das Hinzufügen eines neuen Programms oder die Übermittlung der Interaktionen an interne DAOs. In Ethereum können sich Akteure einerseits außerhalb der Ethereum-DAO befinden, andererseits sind alle DAOs innerhalb von Ethereum ebenfalls Akteure.

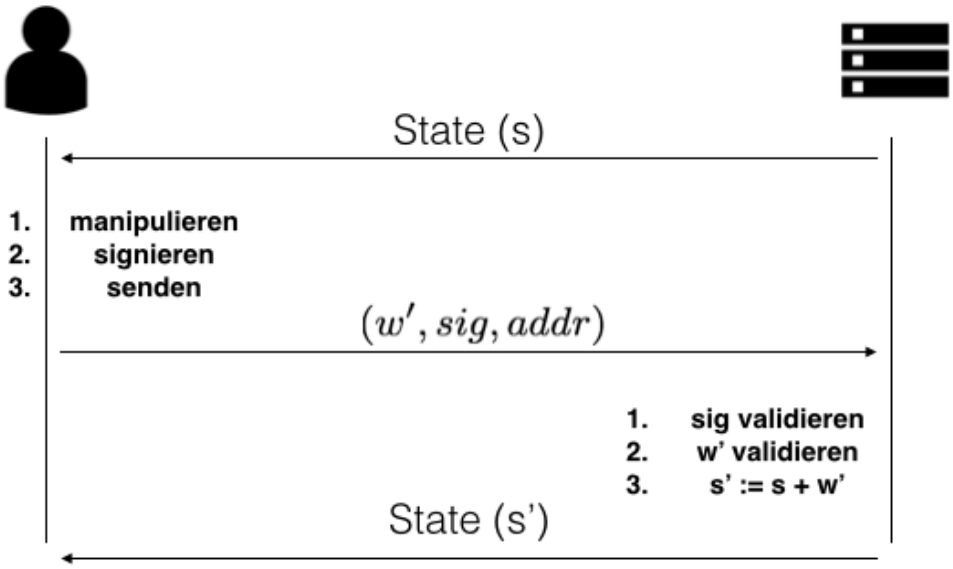


Abbildung 3.1: Interaktion eines Akteurs mit einer DAO.

Die Interaktion eines Akteurs mit einer Blockchain basierten DAO ist in Abbildung 3.1 illustriert und besitzt folgende Struktur:

Der Akteur:

1. Läd sich den für ihn relevanten Teil des Konsens 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. Die Transaktion wird dann 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 eine Berechtigung vom Netzwerk zu erhalten, um seinen privaten Konsens zum allgemeinen Konsens der DAO zu machen.

Ein Akteur benötigt für das Agieren in der Ethereum-DAO einen Bestand an der internen 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. Sie besitzt eine Adresse und hat einen Bestand an interner Währung: Ether.

Ein Beispiel für eine interne DAO ist eine dezentrale Währung, wie sie derzeit vom Bitcoin repräsentiert wird (Quelle [But14]). Sie ist in Abbildung 3.2 illustriert. Ein weiteres Beispiel in Abbildung 3.3 ist ein dezentrales Domain

Name System (DNS), wie ihn die Namecoin DAO implementiert. Es assoziiert Namen ($msg.data[0]$) mit Addressen ($msg.data[1]$), falls mit dem Namen noch keine Assoziation besteht.

```
1  from = msg.sender
2  to = msg.data[0]
3  value = msg.data[1]
4
5  if self.storage[from] >= value:
6  self.storage[from] = self.storage[from] - value
7  self.storage[to] = self.storage[to] + value
```

Abbildung 3.2: Nachbildung einer dezentralen Währung als Ethereum basierte DAO.

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

Abbildung 3.3: Nachbildung eines dezentralen DNS als Ethereum basierte DAO.

Diese beiden Beispiele illustrieren wie vergleichsweise einfach es ist eine DAO auf einer bestehenden Plattform, wie der von Ethereum, zu erstellen, im Gegensatz zu den bisherigen Blockchain-basierten Ansätzen. Ethereum gilt deshalb auch als DAO Plattform, welche die Dezentralität der DAO von ihrer eigentlichen Funktionalität entkoppelt. Eine weitere wichtige Funktionalität einer DAO ist die der Wartbarkeit. Jedoch stellt sich hier die Frage nach dem Besitz und den Berechtigungen der Besitzer die DAO manipulieren zu können. Hierfür muss ebenfalls ein Regelwerk gefunden werden, das die Wartung der DAO, im folgenden auch Selbst-Modifikation genannt, von ihrer eigentlichen Funktionalität entkoppelt.

Kapitel 4

Selbst-Modifikation-Schicht einer DAO

In diesem Kapitel wird die Selbst-Modifikation-Schicht einer DAO betrachtet. Hierfür wird in einer Einführung für die Notwendigkeit einer solchen Schicht argumentiert. In Kapitel 4.1 wird eine Definition einer solchen Schicht vorgeschlagen. In Kapitel 4.2 werden die einzelnen Eigenschaften der Definition diskutiert. Für die Anwendung auf reguläre Grammatiken und die Implementation wird in Kapitel 4.3 eine Kodierung durch eine Grammatikerweiterung vorgestellt. In Kapitel 4.4 wird ein Transitionssystem für die Dynamik der Manipulation und die dazugehörigen validen Transaktionen in Kapitel 4.5 vorgestellt.

Durch eine Plattform wie Ethereum wird nun das Erstellen von DAOs um ein vielfaches leichter gemacht, jedoch gibt es noch keine einheitlichen Lösungsansätze, um die DAOs oder generell dezentrale Inhalte auf eine dezentrale Weise zu erstellen oder zu modifizieren. Es fehlt eine 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 (bei Bitcoin die "Bitcoin-Foundation"[1]) oder einem Unternehmen (bei Ethereum "Ethereum Switzerland GmbH"[2]) entwickelt und manipuliert. Trotz der dezentralen Natur der DAOs ist die Kontrolle über diese noch

[1] https://bitcoinfoundation.org/ 25.02.2015
[2] https://www.ethereum.org/ 25.02.2015

zentralisiert.

Da es sich bei Bitcoin und Ethereum um dezentrale Anwendungen handelt, ist es nicht trivial, wem diese Anwendungen gehören. Der Besitz lässt sich nicht eindeutig festlegen. Falls die interen Tokens aus der Blockchain-Schicht ebenfalls als Besitzanteil an den DAOs interpretiert werden, so können die Besitzer 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 Preisgeld von 100 000 USD in BTC an denjenigen ausgeschrieben, der ein Konzept vorschlägt, wie die Bitcoin Foundation auf dezentrale 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[3]. Diese erfüllt die Ansprüche an eine solche Selbst-Modifikation-Schicht nicht wirklich. Der Mechanismus einer solchen Schicht muss einerseits eine große Anzahl an stimmberechtigten Kandidaten berücksichtigen (derzeit wurden rund 200k Bitcoin Adressen verwendet) sowie eine ähnlich große Menge an Vorschlägen. Er sollte schnell und agil auf aufkommende Probleme reagieren können, jedoch niedrige Einstiegsbarrieren für Benutzer besitzen. Die Besitzverteilung sollte genau festgelegt sein und die Manipulation der DAO gerecht - also proportional zu dem Besitz.

Im folgenden wird ein Vorschlag einer solchen Schicht vorgestellt. Hierfür werden als Erstes die theoretischen Komponenten betrachten, anschließend ihre Anwendung auf die regulären Grammatiken und schließlich ihre Implementation.

Theoretisch kann ein solcher Mechanismus durch ein **initialisiertes Transitionssystem** $T = (M, I, \tau)$ nach [Gla03] modelliert werden. Hierzu muss

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

eine passende Zustandsmenge definiert werden, einen Initialzustand sowie die Übergangsfunktion, die valide Manipulationen (Transitionen) der DAO beschreibt.

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

4.1 Definition

In der Selbst-Modifikation-Schicht wird davon ausgegangen, dass für die Dezentralität der DAO bereits gesorgt ist. Es handelt sich entweder selbst 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. Einer endliche Menge von Akteuren A.
2. Einer eindeutige **Besitzverteilung** von Akteuren zur DAO.
$$share : A \to \mathbb{N} \tag{4.1}$$
3. Einer endlichen Menge von validen **Kandidaten** mit mindestes einem Element.
$$K_G \subseteq L(G) \wedge |K_G| \geq 1 \tag{4.2}$$
4. Eine strickten Totalordnung der Kandidaten.
$$< \subset K \times K \tag{4.3}$$
5. Einer Bewertung der Kandidaten durch die Akteure.
$$vote : A \times K \to [0, 1] \tag{4.4}$$

Sei $\mathbf{DAO_G} = \{O_G \mid O_G \text{ ist DAO in der einer Grammatik } G\}$ die Menge aller DAOs in der Grammatik G. Zudem existiert eine Konsensfunktion, die eine DAO in ein valides Wort überführt.

$$consens : DAO_G \to L(G) \tag{4.5}$$

4.2 Diskussion der Eigenschaften

Im Folgenden werden die Eigenschaften einer DAO näher betrachtet: in Kapitel 4.2.1 die Kandidaten, in Kapitel 4.2.2 die Besitzverteilung der Akteure zur DAO, in Kapitel 4.2.3 wird die Einigung der Akterue auf einen Konsenskandidaten beschrieben, in Kapitel 4.2.4 wird ein anschauliches Beispiel einer DAO beschrieben. Schließlich werden in Kapitel 4.2.5 Qualitätskritärien für die Implementierung beschrieben.

4.2.1 Kandidaten

Der hier verfolgte Ansatz ist nur DAOs in regulären Grammatiken zu betrachten. Diese Eigenschaft schränkt die Menge der möglichen Kandidaten ein. Zu der Kandidatenmenge existiert eine strickte Ordnungsrelation, welche die Reihenfolge angibt, in der die Kandidaten hinzugefügt wurden.

4.2.2 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_G an, der im Besitz von einem Akteur ist.

$$|O_G| := \sum_{a \in A} share(a) \tag{4.6}$$

Der Besitz einer DAO O_G 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 die Tokens benutzt werden, die in der Dezentralisierung-Schicht verwendet werden. Es können auch Neue eingeführt werden. Für den Blockchain-Basierten Ansatz der Dezentralisierung

muss bei getrennten Besitz-Tokens 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.

4.2.3 Konsens

$$consens : DAO_G \rightarrow L(G) \tag{4.7}$$

Die Konsensfunktion überführt jede DAO in ein Wort der Sprache $L(G)$. Dabei sucht die Funktion nach dem Kandidaten aus der Kandidatenmenge mit der maximalen Bewertung. Die Bewertung eines Kandidaten ergibt sich dabei aus der Summe der gewichteten Bewertungen der Akteure. Falls mehrere Kandidaten die maximale Bewertung besitzen, wird der älteste Kandidat genommen.

$$value(k) := \sum_{a \in A} share(a) \cdot vote(a, k) \tag{4.8}$$

$$consens(O_G) := min_{<}(\{k|\ value(k) = \max_{k' \in K_G}(value(k'))\}) \tag{4.9}$$

4.2.4 Beispiel einer DAO

Sei $HASH$ die Menge aller 20-Byte Strings, dann ist die DAO $O^1_{G_{abc}} := (A, K, <, share, vote)$ ein Beispiel für die Grammatik G_{abc} definiert wie in 2.1.

$$A := HASH \tag{4.10}$$
$$h_1 := 1HTN35UxBFTbcN8g2KfXC6TW2ipbytsysh \tag{4.11}$$
$$h_2 := 16iF9qZWG1tKhnjnX5KKCYdYdLv4A1FQ4b \tag{4.12}$$
$$h_3 := 1P3wGbbgDgLxivHw5BGGbLCHugsHBnjnjP \tag{4.13}$$
$$share : hash \mapsto \begin{cases} 10 & : hash = h_1 \\ 7 & : hash = h_2 \\ 6 & : hash = h_3 \\ 0 & : otherwise \end{cases} \tag{4.14}$$
$$K := \{aaaaa, aac, aacc, aaccc\} \tag{4.15}$$
$$< := \{(aaaaa, aac), (aac, aacc), (aacc, aaccc)\}^+ \tag{4.16}$$
$$vote : hash, k \mapsto \begin{cases} 1.0 & : hash = h_1, k = aaaaa \\ 0.9 & : hash = h_2, k = aacc \\ 0.9 & : hash = h_3, k = aacc \\ 0.9 & : hash = h_2, k = aaccc \\ 0.9 & : hash = h_3, k = aaccc \\ 0 & |otherwise \end{cases} \tag{4.17}$$

$$consens(O^1_{G_{abc}}) = aacc \tag{4.18}$$

4.2.5 Qualitätskriterien

Für die Anwendung ist auf folgende Qualitätskriterien zu achten:

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

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

4.3 Grammatikerweiterung

Als Zustandsmenge für das Transitionssystem kann nun die Menge aller möglichen DAOs genommen werden. Allerdings muss die Menge für die Implementation kodiert sowie die Validitätsbedingungen auf der kodierten Menge definiert werden. Eine passende Kodierung die näher betrachtet wird, ist es eine DAO zu einem Wort einer Sprache zu machen. Diese Sprache wird **Metasprache** der ursprünglichen Sprache ($L(G)$) genannt. Eine Bedingung muss jedoch sein, dass das Wort ohne Verlust von Informationen wieder zurück zur selben DAO dekodiert werden kann. Damit können Transitionsbedingungen als Manipulationen des kodierten Wortes definiert werden.

Die Wörter der kodierten DAOs müssen ebenfalls valide und invalide Teile der Wörter der ursprünglichen Sprache ($L(G)$) erkennen. 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 $L(S(G))$ wird.

Formal 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 welche 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 kodieren und dekodieren die Kandidatenmenge einer DAO in ein Wort. Anschließend wird die Besitzverteilung sowie die Kandidatenbewertung eingearbeitet, sodass sich die gesuchten Funktionen ergeben.

4.3.1 Kodierung der Kandidatenmenge

Die reguläre Grammatik wird so zu einer kontextfreien Grammatik erweitert, so dass alle Wörter aus der Kandidatenmenge in einem Wort der erweiterten Grammatik kodiert werden können.

Dazu muss 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 sind.

Dieses kann mit Hilfe einer mod. Patricia Trie-Datenstruktur realisiert werden, die redundant auftauchende Präfixe der Kandidaten zusammengefasst. Die Serialisierung der Tries ist ein Wort der Metasprache, die durch die Grammatikerweiterung S_1 konstruiert wird:

Sei $T(K_G)$ die mod. Patricia Trie Datenstruktur, welche die Menge K_G kodiert. Es ist bekannt, dass eine Funktion $c : K_G \mapsto T(K_G)$ sowie $c^{-1} : T(K_G) \mapsto K_G$ existiert, für die $c \circ c^{-1} = id_{P(L(G))}$ gilt [Mor68]. Zu finden ist demnach 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 Trie serialisieren und deserialisieren. Da eine Trie ein Baum ist, können wir als Serialisierung die geklammerte preorder-Notation verwendet, die uns den gewünschten Isomorphismus liefert. Damit ergeben sich die Funktionen $\phi_1 := c \circ s$ sowie $\phi_1^{-1} := c^{-1} \circ s^{-1}$.

Dieses wird anhand eines Beispiels für die Sprache $L(G) = \{a^x b^y c^z \mid x, y, z \in \mathbb{N}\}$ und die Kandidatenmenge $\{aaaaa, aac, aacc, aaccc\}$ in Abbildung 4.1 durch die Trie $T(K_G)$ illustriert.

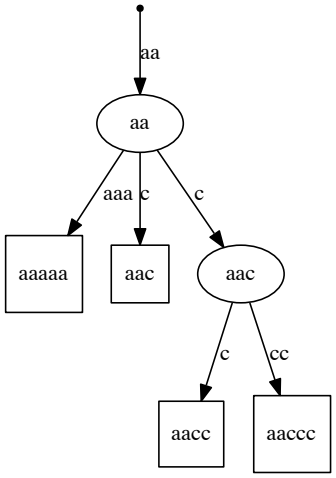


Abbildung 4.1: mod. Patricia Trie für die Kandidatenmenge $\{aaaaa, aac, aacc, aaccc\}$

Mit der dazugehörigen preorder-Serialisierung die [und] als Klammerzeichen sowie & als Trennzeichen verwendet:

$$aa[aaa\&c\&c[c\&cc]] \tag{4.19}$$

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

$$P_{Options} := \{R \to [O_R], O_R \to r\&O_R, O_R \to r \mid R \to r \in P \wedge r \in \Sigma^*\} \tag{4.20}$$

Diese Mehrdeutigkeit ($[O_R]$) wird **Optionsmenge** sowie die darin enthaltenen Elemente (r) **Option** genannt. Eine solche Grammatikerweiterung erzeugt die Serialisierung einer mod. Patricia Trie, welche eine Menge valider Worte kodiert:

$$aa[aaa\&c\&c[c\&cc]] \in L(S_1(G)) \tag{4.21}$$

4.3.1.1 Grammatik-Erweiterung S1

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

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

4.3.2 Kodierung der Besitzverteilung

Die Akteure lassen sich durch einen 20-Byte String genau identifizieren. Sei $HASH$ die Menge aller 20-Byte Strings. Im folgenden werden die Hash-Strings in der Base58[4] Notation angegeben. Die Besitzverteilung ist eine Auflistung aller beteiligter Akteure sowie deren Anteile an der DAO. Sei $hash \in HASH$, $MAXINT = 2^{32} - 1$ sowie $number \in NUMBER := [0, MAXINT]$.

[4] https://en.bitcoin.it/wiki/Base58Check_encoding 26.02.2015

$$P_{Acteurs} := \{A \to [hash\ number]A, A \to \varepsilon\} \tag{4.22}$$

Eine Beispiel-Verteilung ist:

$$\begin{aligned}
h_1 &:= 1HTN35UxBFTbcN8g2KfXC6TW2ipbytsysh && (4.23)\\
h_2 &:= 16iF9qZWG1tKhnjnX5KKCYdYdLv4A1FQ4b && (4.24)\\
h_3 &:= 1P3wGbbgDgLxivHw5BGGbLCHugsHBnjnjP && (4.25)\\
a &:= [h_1\ 10][h_2\ 7][h_3\ 5] && (4.26)\\
A &\to^* a && (4.27)
\end{aligned}$$

In diesem Beispiel gehören Akteur h_1 - 10 Anteile, h_2 - 7 Anteile und h_3 - 5 Anteile. Somit gehören zur DAO 22 Anteile.

4.3.3 Kodierung der Kandidatenbewertung

Um jede *DAO* ohne Verlust von Information kodieren zu können, muss jede mögliche Bewertungsverteilung von Akteuren und Kandidaten im Metawort enthalten sein. Dieses kann erreicht werden, wenn die Bewertungen der Kandidaten (V) an Optionen gebunden sind ($O_R \to r\&[V]O_R$). 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 kodieren, wenn sie dieser Option zugeordnet.

Zusätzlich werden Bewertungen einer Option zugelassen, 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 damit jeweils von einer Bewertung einer tieferen Ebene ersetzt.

Sei $MAXINT = 2^{32} - 1$ sowie $number \in NUMBER := [0, MAXINT]$.

$$P_{Voting} := \{V \to [hash\ number]V, V \to \varepsilon\} \tag{4.28}$$

Ein Beispiel ist:

$$
\begin{aligned}
h_1 &:= 1HTN35UxBFTbcN8g2KfXC6TW2ipbytsysh && (4.29)\\
v &:= [h_1\ MAXINT] && (4.30)\\
V &\rightarrow^* v && (4.31)\\
O_R &\rightarrow^* r\&[v] && (4.32)
\end{aligned}
$$

4.3.4 Kodierung der 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 entweder eine Person, ein Zusammenschluss von Personen, wie eine Interessengemeinschaft in Form einer DAO, oder eine DAO ohne Fremdeinwirkung sein, die bestimmte Werte zu optimieren versucht.

Delegationen werden für Optionsmengen vererbt. Durch die lineare Struktur der Worte ergibt sich eine strickte Totalordnung der Delegationen, womit bei konkurierenden Delegationen immer die Delegation niedrigerer Ordnung genommen wird. Eine Stimme des Akteurs selbst wird einer Stimme eines Deleganten vorgezogen.

$$P_{Delegations} := \{D \rightarrow [hash\ hash]D, D \rightarrow [hash\ hash]\} \quad (4.33)$$

Eine Beispielverteilung ist:

$$
\begin{aligned}
h_2 &:= 16iF9qZWG1tKhnjnX5KKCYdYdLv4A1FQ4b && (4.34)\\
h_3 &:= 1P3wGbbgDgLxivHw5BGGbLCHugsHBnjnjP && (4.35)\\
d &:= [h_2\ h_3] && (4.36)\\
D &\rightarrow^* d && (4.37)
\end{aligned}
$$

4.3.5 Vollständige Grammatik-Erweiterung S

$$
\begin{aligned}
S(G) :=& (N', T', S', P') && (4.38)\\
N' :=& N \cup \{O_R | (R \to r) \in P \wedge O_R \notin N\} \cup \{D, A, S'\} && (4.39)\\
T' :=& T \cup \{[,], \&, \mid [,], \& \notin \Sigma\} \\
& \cup HASH \cup NUMBER && (4.40)\\
P_{Acteurs} :=& \{A \to [hash\ number]A, A \to \varepsilon && (4.41)\\
& | hash \in HASH, number \in NUMBER\} && (4.42)\\
P_{Options} :=& \{R \to [O_R][D], O_R \to r\&[V]O_R, O_R \to \varepsilon \\
& \mid R \to r \in P \wedge r \in \Sigma^*\} && (4.43)\\
P_{Start} :=& \{S' \to [A][O_S][D]\} && (4.44)\\
P_{Delegations} :=& \{D \to [hash\ hash]D, D \to [hash\ hash] \\
& | hash \in HASH\} && (4.45)\\
P_{Voting} :=& \{V \to [hash\ number]V, V \to \varepsilon \\
& | hash \in HASH, number \in NUMBER\} && (4.46)\\
P' :=& P \cup P_{Options} \cup P_{Start} \cup P_{Delegations} \cup P_{Voting} \cup P_{Acteurs} && (4.47)
\end{aligned}
$$

Beispielwort

Das zugehörige Metawort zur DAO $O^1_{G_{abc}}$ definiert wie in 4.2.4 $\phi(O^1_{G_{abc}}) = w \in L(S(G))$ wird mit dem dazugehörige Konsens (4.48) in Abbildung 4.2 dargestellt.

$$consens(w) = aacc \tag{4.48}$$

4.3.6 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.

```
1   [
2     [h_1 10]
3     [h_2 7]
4     [h_3 5]
5   ] [
6     aa[
7       aaa&[h_1 4294967295]
8       c&[]
9       c[
10        c&[]
11        cc&[]
12      ][]&[h_3 3865470565]
13    ][h_2 h_3]&[]
14  ][]
```

Abbildung 4.2: Metawort $w \in L(S(G))$

$$consens : L(S(G)) \to L(G) \tag{4.49}$$

Die Idee dabei ist, dass die transitive Hülle der Delegationen als Attribut wärend der Ableitung vererbt(inherited) wird. Die Stimmmenge wird hingegen als Attribut synthetisiert [Knu68]. Delegationen zusammen mit den Stimmen quantifizieren jede Option aus einer Optionsmenge und wählen eine Konsens-Option mit den maximalen Stimmen niedrigster Ordnung.

4.4 Transitionssystem

Ein Transitionssystem $T = (M, I, \tau)$ besteht aus einer Zustandsmenge M, einem Initialzustand I und einer Übergangsfunktion τ. Nachdem die Zustandsmenge als Metasprache $M = L(S(G))$ sowie die Kodierung und Dekodierung einer DAO in ein Wort der Metasprache gefunden wurde, wird die Übergangsfunktion τ betrachtet.

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

$$I := \{s_0 \in L(S(G))\} \tag{4.50}$$

Es wird sich auf die Manipulation der Kandidatenmenge sowie der Kandidatenbewertung beschränkt. Weitere Manipulationen werden in Aussicht gestellt. Valide Manipulationen der Kandidatenmenge sollen das Erstellen und das Erweitern von Optionsmengen sein. Für die Kandidatenbewertung sollen das Hinzufügen und Entfernen der eigenen Stimmen sowie der eigenen Delegationen für Optionsmengen valide sein.

Falls die Besitzverteilung von der Blockchain-Schicht übernommen wird, wirken sich alle Manipulationen der Besitzverteilung ebenfalls auf das Metawort aus. Es wird sich jedoch auf solche DAOs beschränkt, bei denen die Besitzverteilung ausschließlich in der Selbst-Modifikation-Schicht manipuliert wird. Hierbei wird das Überweisen eigener Anteile eines Akteurs an einen anderen Akteur zulassen.

4.4.1 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 überprüft wurden und somit nicht Teil seines privaten Konsenses sind.

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

$$\Delta' := \Delta \cup \{(a, w')\} \tag{4.51}$$

4.4.2 Transformation

Sei $\Delta \neq \emptyset$, so kann die Transitionsfunktion τ folgendermaßen definiert werden:

$$\tau(w) := \begin{cases} w' & : \exists (a, w') \in \Delta \wedge \beta(a, w, w') = true \\ w & : andernfalls \end{cases} \tag{4.52}$$

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

$$\Delta' := \begin{cases} \Delta \setminus \{(a, w')\} & : (a, w') \text{ wurde in der Transformation verwendet} \\ \{\} & : andernfalls \end{cases} \tag{4.53}$$

4.5 Validierung der Aktualisierung

Um die Validität einer Transformation zu testen, wird das derzeitige Wort w zusammen mit dem auslösenden Akteur a sowie seinem manipuliertem Wort w' betrachtet. Die Transformation ist valide, falls die Kandidatenmenge wie in Abschnitt 4.5.1 vorgestellt, die Kandidatenbewertung wie in Abschnitt 4.5.2 vorgestellt oder die Anteile, wie in Abschnitt 4.5.3 vorgestellt, manipuliert werden.

4.5.1 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.

$$\begin{aligned} (A, K', share', vote') &= \phi^{-1}(w') \\ \wedge (A, K, share, vote) &= \phi^{-1}(w) \\ \Rightarrow vote &= vote' \qquad (4.54) \\ w' &\in L(S(G)) \qquad (4.55) \end{aligned}$$

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

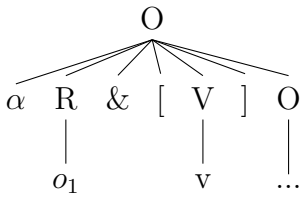
Erzeugen einer Optionsmenge

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 $i, j, k, n \in [0, MAXINT]$,

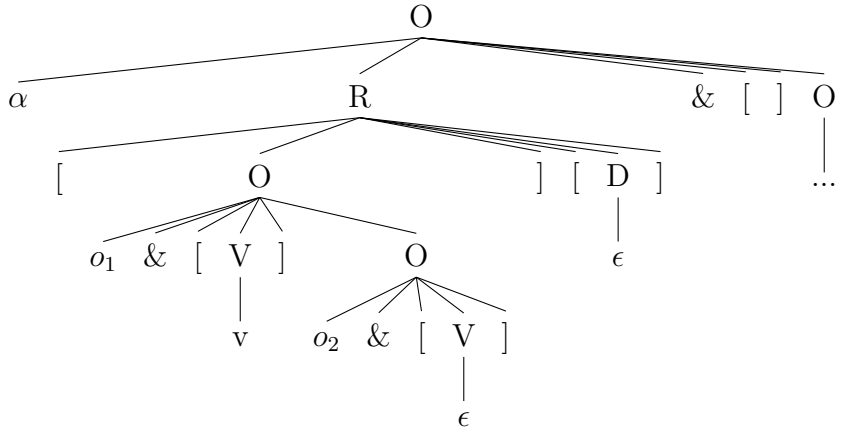
dann ist das Erzeugen einer Optionsmenge valide, wenn folgende Bedingungen erfüllt sind:

$$w = w_{0..i-1}\ w_{i..j}\ w_{j+1..n} \tag{4.56}$$
$$w_{i..j} = \alpha o_1 \& [v] \tag{4.57}$$
$$O \rightarrow^* \alpha R \& [v] \tag{4.58}$$
$$R \rightarrow^* o_1 \tag{4.59}$$
$$S' \rightarrow^* w_{0..(i-1)}\ O\ w_{(j+1)..n} \tag{4.60}$$
$$R \rightarrow^* o_2 \tag{4.61}$$
$$w' = w_{0..i-1}\ \alpha[o_1 \& [v] o_2 \& [][]\ w_{j+1..n} \tag{4.62}$$

Für w hat O folgende Struktur:



Für w' besitzt O folgende Struktur:



Dabei muss o_2 frei von Delegationen und Stimmen sein.

Erweitern einer Optionsmenge

Sei $o, d, r \in T'^*$ sowie $R, O_R \in N'$, dann ist das Erweitern einer Option valide, wenn folgende Bedingungen erfüllt sind:

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (4.63)\\
w_{i..j} &= \alpha[o][d] && (4.64)\\
R &\rightarrow^* w_{i..j} && (4.65)\\
O_R &\rightarrow_P^* o && (4.66)\\
O_R &\rightarrow^* r && (4.67)\\
w' &= w_{0..(i-1)}\ \alpha[or\&[]][d]\ w_{(j+1)..n} && (4.68)
\end{aligned}
$$

r muss dabei Delegations- und Stimmfrei sein sowie neu:

$$
\begin{aligned}
(A, K', share', vote') = \phi^{-1}(w_{0..i-1}\ \alpha[r\&[]][d]\ w_{j+1..n}) && (4.69)\\
\wedge(A, K, share, vote) = \phi^{-1}(w) && (4.70)\\
\Rightarrow K' \cap K = \emptyset && (4.71)
\end{aligned}
$$

4.5.2 Manipulation der Kandidatenbewertung

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

Hinzufügen einer Stimme

Sei $b_1, b_2, v, r \in T'^*$, dann ist das Hinzufügen einer Stimme valide, wenn folgende Bedingungen erfüllt sind:

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (4.72)\\
v &\neq b_1 a b_2 && (4.73)\\
w_{i..j} &= r\&[v] && (4.74)\\
w' &= w_{0..i-1}\ r\&[v[a\ n]]\ w_{j+1..n} && (4.75)
\end{aligned}
$$

Löschen einer Stimme

Sei $v_1, v_2 \in T'^*$ sowie $n \in [0, MAXINT]$, so lässt sich eine Stimme valide löschen, wenn folgende Bedingungen erfüllt sind:

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (4.76)\\
w_{i..j} &= r\&[v_1[a\ n]v_2] && (4.77)\\
w' &= w_{0..(i-1)}r\&[v_1\ v_2]w_{(j+1)..n} && (4.78)
\end{aligned}
$$

Hinzufügen einer Delegation

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

Sei $o, d_i \in T'^*$ sowie $h' \in HASH$ und $h \neq a$, dann ist das Hinzufügen einer Delegation valide, wenn folgende Bedingungen erfüllt sind:

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (4.79)\\
u, v &\in \{w | w = ([h_1\ h_2])^* \text{ mit } h_1, h_2 \in HASH\} && (4.80)\\
w_{i..j} &= [o][uv] && (4.81)\\
w' &= w_{0..(i-1)}\ [o][u[a\ h']v]\ w_{(j+1)..n} && (4.82)
\end{aligned}
$$

Löschen einer Delegation

Sei $o, d_i \in T'^*$ sowie $h' \in HASH$, so ist das Löschen einer Delegation valide, wenn folgende Bedingungen erfüllt sind:

$$
\begin{aligned}
w &= w_{0..i-1}\ w_{i..j}\ w_{j+1..n} && (4.83)\\
u, v &\in \{w | w = ([h_1\ h_2])^* \text{ mit } h_1, h_2 \in HASH\} && (4.84)\\
w_{i..j} &= [o][u[a\ h']v] && (4.85)\\
w' &= w_{0..(i-1)}\ [o][uv]\ w_{(j+1)..n} && (4.86)
\end{aligned}
$$

4.5.3 Überweisung der Anteilen

Ein Akteur kann seine Anteile an einen anderen überweisen. Dabei müssen zwei Fälle unterschieden werden. Hält der Empfänger bereits Anteile, wird die Überweisungssumme seinen Beständen addiert. Handelt es sich um einen neuen Akteur, wird die Überweisungssumme neu hinzugefügt.

Sei $h' \in HASH$ mit $h' \neq a$ sowie $l, m, n \in [0, MAXINT]$, dann ist das Überweisen von Anteilen valide, wenn folgende Bedingungen erfüllt sind:

$$w = w_{0..i-1}\ w_{i..j}\ w_{j+1..n} \tag{4.87}$$

$$b_1, ..., b_6 \in \{w | w = ([h\ n])^* \text{ mit } h \in HASH \text{ und } n \in [0, MAXINT]\} \tag{4.88}$$

$$w_{i..j} = [b_1 b_2 b_3 [a\ n] b_4 b_5 b_6] \tag{4.89}$$

$$m' = (m + l) \tag{4.90}$$

$$n' = (n - l) \tag{4.91}$$

$$n - l \geq 0 \tag{4.92}$$

$$\text{Fall 1:} \quad b_2 = [h'm] \tag{4.93}$$

$$w' = w_{0..(i-1)}\ [b_1 [h'm'] b_3 [a\ n'] b_4 b_5 b_6]\ w_{(j+1)..n} \tag{4.94}$$

$$\text{Fall 2:} \quad b_5 = [h'm] \tag{4.95}$$

$$w' = w_{0..(i-1)}\ [b_1 b_2 b_3 [a\ n'] b_4 [h'm'] b_6]\ w_{(j+1)..n} \tag{4.96}$$

$$\text{Fall 3:} \tag{4.97}$$

$$w' = w_{0..(i-1)}\ [b_1 b_2 b_3 [a\ n'] b_4 b_5 b_6 [h'l]]\ w_{(j+1)..n} \tag{4.98}$$

Dabei ist $(m + l)$[5], $(n - l) \in [0, MAXINT]$.

[5]Bei dieser Operationen muss in der Implementation darauf geachtet werden, dass es zu keinem Überlauf der Werte kommt.

Kapitel 5

Implementation

In diesem Kapitel wird die Implementation der Anwendung einer Selbst-Modifikations-Schicht auf DAOs mit regulärer Grammatik vorgestellt. Dafür wird in Kapitel 5.1 die gewählte Architektur vorgestellt sowie in Kapitel 5.2 der Umgang mit Grammatiken und der Metasprache.

Implementiert wurde im Rahmen dieser Arbeit eine **zentralisierte** Version der Selbst-Modifikation-Schicht. Die Dezentralität wurde ersteinmal außen vor gelassen, da die Funktionsweise der Selbst-Modifikation im Vordergrund steht. Jedoch sind alle Grundlagen dafür gelegt, die hier erarbeitete Lösung auf Ethereum oder eine andere turing-vollständige DAO Plattform zu portieren und damit zu dezentralisieren.

5.1 Architektur

Ziel der Implementation ist eine lauffähige Implementierung des Konzeptes. Um es einem Benutzer zu ermöglichen mit Anwendungsbeispielen zu experimentieren, wie reguläre Grammatiken anzulegen und diese mit der Angabe der Besitzverteilung und eines Initialwortes zu einer DAO zu machen. Implementiert ist die Interaktion mit der DAO Selbst-Modifikation-Schicht wie in 3.2 beschrieben. Ein Akteur kann dabei entweder eine vorhandene DAO manipulieren oder eine Neue erstellen. Technisch wurde hier eine Klassische Client-Server Architektur benutzt wie sie in Abbildung 5.1 dargestellt ist.

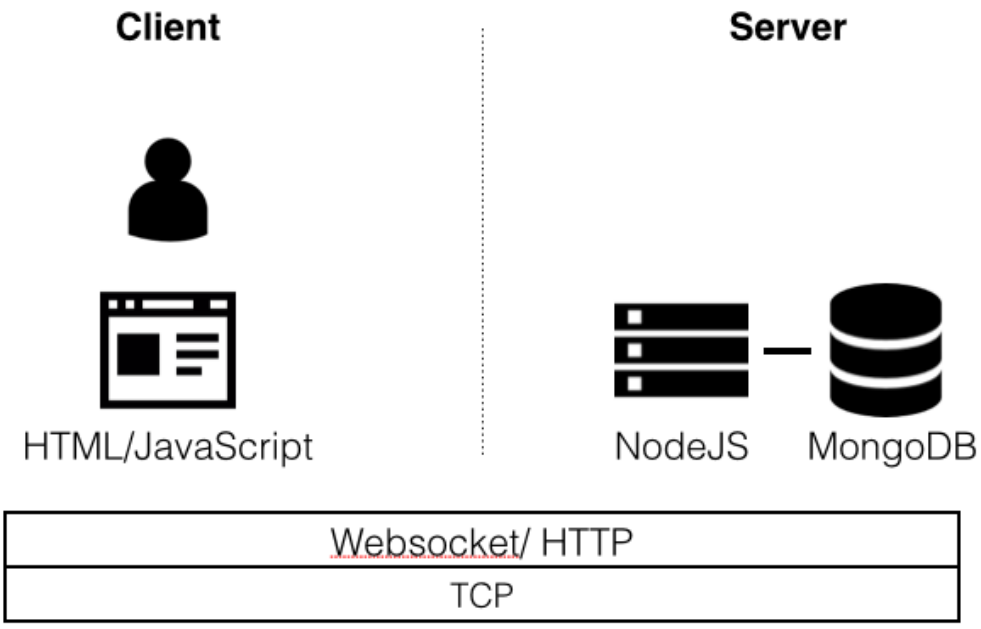


Abbildung 5.1: Benutzte Technologie

Dafür wurde das MeteorJS[1] Web Framework verwendet welches auf NodeJS[2] basiert - einem serverseitigen JavaScript Compiler. Für die Datenspeicherung wird MongoDB[3] verwendet - eine NoSQL Datenbank. Der Client kommuniziert über HTTP und WebSocket mit dem Server.

Das Datenmodell (Abbildung 5.1) ist in zwei Bereiche unterteilt: Einerseits die DAOs und andererseits die Grammatiken.

Ein Akteur hat nun die Möglichkeiten

1. Vorhandene Grammatiken aufzulisten.
2. Eine vorhandene Grammatik zu inspizieren.
3. Eine neue Grammatiken hinzuzufügen.
4. DAOs aufzulisten.

[1]http://meteor.com 25.02.2015
[2]http://nodejs.org/ 25.02.2015
[3]http://www.mongodb.org/ 25.02.2015

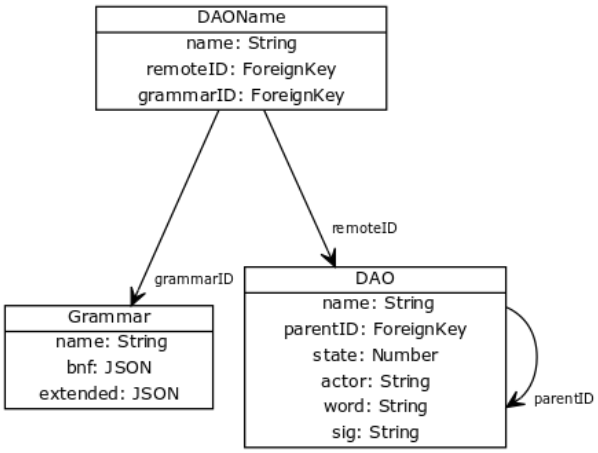


Abbildung 5.2: Datenmodell

5. Den Aufbaue und die Historie einer DAO einzusehen.
6. Eine DAO zu manipulieren.
7. Eine neue DAO zu erstellen.

Für die Signierung wird die Bibliothek BitcoinJS-lib[4] verwendet. Sie stellt die notwendigen ECDSA sowie SHA Funktionen zu Verfügung. Für dem Umgang mit Grammatiken und Worten wird JISON[5], eine JavaScript Portierung des Bison/Flex LALR(1) Parser-Generators, verwendet.

BitcoinJS-lib stellt notwendige kryptografische Funktionen bereit: das Erstellen von einem Private-Key auf Basis eines Strings, das Erstellen eines Public-Key und einer Adresse auf Basis des Private-Keys, das Signieren von Nachrichten mit einem Private-Key sowie das Verifizieren von Nachrichten mit ihrer Signatur und einer Adresse. Dieses deckt alle Funktionen ab, die für die sichere Kommunikation von einem Akteur und der DAO notwendig sind.

5.2 Metasprache

Für das Erzeugen von Parsern wird der Parser-Generator JISON so erweitert, dass dieser bei Eingabe einer regulären Grammatik G einen Parser für die zur

[4]https://github.com/bitcoinjs/bitcoinjs-lib 26.02.2015
[5]http://zaach.github.io/jison 26.02.2015

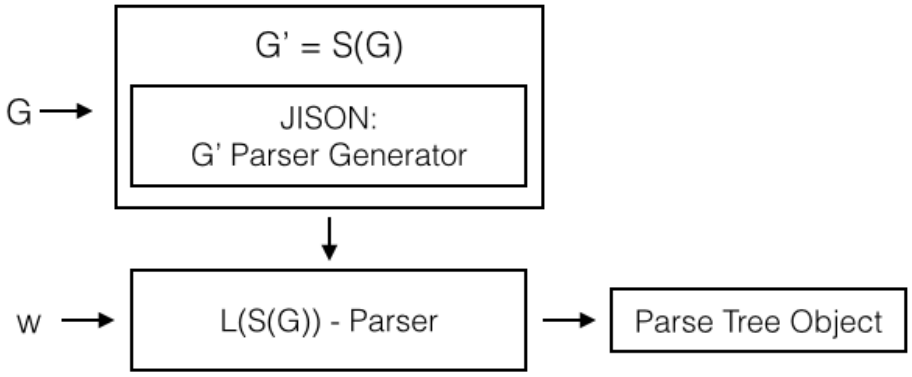


Abbildung 5.3: Grammatikerweiterung und Parser Generator

Grammatik zugehörigen Metasprache erzeugt. Dieser Parser interpretiert ein Wort der $L(S(G))$ Sprache als ein Parse-Baum-Objekt (PTO). Eine Grammatik wird als Bison Grammatik (Abbildung 5.4) in der EBNF Schreibweise angegeben.

Das Parse-Baum-Objekt besitzt folgende Funktionen:

toString wandelt das Parse-Baum-Objekt wieder in ein Wort der Metasprache um, welches dieses Parse-Baum-Objekt konstruiert.

getConsensString wählt den Konsens-Kandidaten aus: wendet die Konsensfunktion an und liefert ein Wort der $L(G)$ Sprache zurück.

validate erhält ein weiteres PTO, die eine Manipulation des derzeitigen PTOs beinhaltet, sowie eine Akteur-Adresse und überprüft, ob die Manipulation des derzeitigen PTOs vom Akteur valide ist.

add erhält ein Wort w der $L(G)$ Sprache und fügt es dem Derzeitigen PTO hinzu. Diese Operation fügt der DAO einen neuen Kandidaten hinzu.

getCandidates liefert K_G der DAO zurück: ein Array aller Kandidaten mit ihren Bewertungen. Dadurch lässt sich die Funktion *vote* leicht rekonstruieren.

```
1  %lex
2  %%
3  \s                              /* IGNORE */
4  (a)                             return 'a'
5  (b)                             return 'b'
6  (c)                             return 'c'
7  .                               return 'INVALID'
8  /lex
9
10 %start A
11 %%
12
13
14 A: 'a' A | 'a'
15   | 'b' B | 'b'
16   | 'c' C | 'c'
17   | ;
18
19 B: 'b' B | 'b'
20   | 'c' C | 'c';
21
22 C: 'c' C | 'c';
23 %%
```

Abbildung 5.4: Bison Grammatik für die Sprache $L(G) = \{a^x b^y c^z | x, y, z \in \mathbb{N}\}$

Kapitel 6

Ausblick

6.1 Anwendung auf kontextfreie Grammatiken

Eine interessante Anwendung des hier vorgestellten Ansatzes einer Selbst-Modifikation-Schicht ist die Anwendung auf die kontextfreien Sprachen. Damit können alle Programmiersprachen zumindest syntaktisch abgebildet werden. Dieses ist jedoch nicht trivial. Das Problem hierbei ist die Kodierung der Kandidatenmenge. Aufgrund der Limitierung der Ableitungsregeln kann ein Ableitungsbaum einer regulären Grammatik keine Knoten haben, die mehrere Kinder mit Nichtterminalen besitzen (Äste). Somit ist der Ableitungsbaum eine lineare Struktur. Dies ermöglicht die Konstruktion eines Präfixbaumes, bei dem Äste als Optionen angesehen werden. Zwar lässt sich die Zusammenfassung der Kandidaten bei kontextfreien Grammatiken mit der selben Methode anhand des Ableitungsbaumes konstruieren, jedoch multiplizieren sich die Optionen in unterschiedlichen Kindern eines Knotens. Somit gilt die in 4.3.1 geforderte Bedingung $\phi_1 \circ \phi_1^{-1} = id_{P(L(G))}$ nicht, da Optionen hinzu kommen. Ebenfalls lässt sich nicht jede Kandidatenbewertung durch das Anhängen an Optionsknoten kodieren.

Im folgenden wird das Beispiel der Sprache $L(G) = \{[ab][abc]\}$ mit der konstruierenden Grammatik betrachtet:

$$
\begin{aligned}
G &= (T, N, P, S) && (6.1)\\
T &= \{a, b\} && (6.2)\\
N &= \{S, B\} && (6.3)\\
P &= \{S \to BB, B \to a, B \to b\} && (6.4)
\end{aligned}
$$

die Kandidatenmenge $K_1 := \{ab, ac, ba\}$ sowie das anhand des Ableitungbaumes valide Metawort:

$$[A][a\&[v_1]b\&[v_2]][][a\&[v_3]b\&[v_4]c\&[v_5]][]][]$$

Die sich aus dem Metawort bildenden Kandidaten sind: $K_2 = \{aa, bb, ab, ba, ac, bc\}$

Zudem gibt es keinen Mechanismus, der jede mögliche Kandidatenbewertung in der Sprache darstellen kann. Es stehen in diesem Beispiel 5 Bewertungsvariablen für 6 Kandidaten zur Verfügung.

6.2 Dezentralisierung

Wie bereits erwähnt, ist die Dezentralisierung der Selbst-Modifikation-Schicht der DAO z.B. durch eine Portierung auf die Ethereum Plattform der nächste notwendige Schritt.

6.3 Anonymisierung

Durch die pseudonyme Natur der Anteilseigener lassen sich Rückschlüsse darüber ziehen, welcher Akteur für welche Option abgestimmt hat. Somit lässt sich der Konsens manipulieren, indem auf strategisch wichtige Akteure Druck ausgeübt wird, um ihre Stimme zu ändern. Auch wird das Abstimmverhalten der Anteilseigener beeinflusst, da diese fürchten könnten ihren pseudonymen Status zu verlieren. Die anonyme Wahl ist ein Kriterium für demokratische Wahlen in vielen Organisationen, sodass eine Anonymisierung

der Abstimmung eine weitere erstrebenswerte Erweiterung ist. Diese Anonymisierung könnte durch homomorphe Verschlüsselung[Gen09] oder durch andere kryptografische Mittel realisiert werden. Siehe dazu [FDL07].

6.4 Manipulation der Besitzverteilung

Derzeit ist die Größe einer DAO fest. Die einzige Möglichkeit an Anteile zu gelangen, ist diese von anderen überwiesen zu bekommen. Eine mögliche Erweiterung wäre die der Erschaffung neuer Anteile. Diese können entweder von der DAO verkauft oder anhand festgelegter Regeln herausgegeben werden. Ein Beispielregelsatz wäre das Binden an Optionsmengen. Derjenige Akteur erhält die neuen Anteile, dessen vorgeschlagene Option für eine bestimmte Dauer den Platz der Konsens-Option einnimmt.

Literaturverzeichnis

[BBH+11] Jörg Becker, Dominic Breuker, Tobias Heide, Justus Holler, Hans Peter Rauer, and Rainer Böhme. *The Bitcoin System*. University of Münster, 2011.

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

[Cor14] Identifier Technology Innovation Panel. Technical report, The Internet Corporation for Assigned Names and Numbers, 2014.

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

[FDL07] Laure Fouard, Mathilde Duclos, and Pascal Lafourcade. *Survey on electronic voting schemes*. 2007.

[Fri10] Friedrich Lindenberg. *Konzeption und Erprobung einer Liquid Democracy Plattform anhand von Gruppendiskussionen*. Bachelorarbeit, Technische Universität Ilmenau, 2010.

[Gar15] Juan A Garay. The Bitcoin Backbone Protocol : Analysis and Applications. In *Proceedings of Eurocrypt 2015*, pages 1–37, 2015.

[Gen09] Craig Gentry. *a Fully Homomorphic Encryption Scheme*. PhD thesis, STANFORD UNIVERSITY, 2009.

[Gla03] Andreas Glausch. *Abstract-State Machines Eine Sammlung didaktischer Beispiele*. Vorlesungsskript, HU Berlin, 2003.

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

[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.

[Ott14] Friedrich Otto. *Formale Sprachen und Automaten*. Vorlesungsskript, University Kassel, 2014.

[Pou14] Nicolas Pouillard. *Master ' s thesis Private , trustless and decentralized message consensus and voting schemes*. PhD thesis, 2014.

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

[Wal04] Jeremy Waldron. Property and Ownership. In *Stanford Encyclopedia of Philosophy*. September 2004.

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