# 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

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:

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

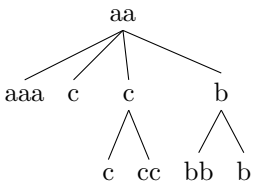
$$L(G) = \{a^x b^y c^z | x, y, z \in \mathbb{N}\}$$

## 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 $I \subset M$.

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

## 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.

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

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.

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

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.

**TODO:** Kapselung - Teilbereiche - Komponenten

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 ($\varepsilon$), 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:

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

POW ist jedoch umstritten, da das verbrauchen von Rechenressourcen nicht ökonomisch und 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 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.

**TODO:** coins einführen

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

Das erstellen von Blocks durch Miner ist zumindest bei Bitcoin ökonomisch motiviert. Die Bitcoin DAO schüttet eine Belohnung an den Miner aus, der einen neuen Block findet. Bei Bitcoin werden derzeit (18.02.2015) 25 Bitcoins 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

### Turing Vollständigkeit

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

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

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

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

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

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

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

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

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

# (selbst) Modifikationsschicht einer DAO

## Definition

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

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

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

## Erläuterung

### Besitzverteilung

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

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

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

Eine Konsensfunktion entscheidet über die Ausführung der Kontrolle.

### Transaktionen

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

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

### Konsens

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

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

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

### Qualitätskriterien

Für die Implementation ist auf folgende Qualitätskriterien zu achten:

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

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

**TODO:** Metasprache einführen

**TODO:** Transitionsysthem einführen: problem der codierung

# Anwendung auf reguläre Grammatiken

## Optimierung der Kandidatenmenge

Die reguläre Grammatik wird so zu einer kontextfreien Grammatik erweitert, dass alle Wörter aus der Kandidatenmenge in einem Wort der erweiterten Grammatik codiert werden können. Dieses kann einfach durch eine Trie-Datenstruktur realisiert werden die redundant auftauchende Präfixe der Kandidaten zusammengefasst. Die Serialisierung der Tries ist eben ein Wort, welcher von der gesuchten Grammatikerweiterung dargestellt werden soll.

Dafür wird an jedem Ableitungsschritt eine Mehrdeutigkeit zugelassen die zu einer **Optionsmenge** zusammengefasst wird. Die Kandidatenmenge ergibt sich durch die Kombinationen der Präfixe mit den Optionen einer Optionsmenge.

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

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

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

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

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

**Grammatik-Erweiterung S1**

Sei $G = (N, T, S, P)$ eine reguläre Grammatik in der Greibach-Normalform.

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

**Definition $\phi$**

Sei $S(\alpha, K)$ die Funktion, die den Größten gemeinsamen Präfix aller Kandidaten liefert, die den Präfix $\alpha$ enthalten:

$$E(\alpha, K) : \exists w \in K : w = \alpha\beta$$

$$A(\alpha, K) : \exists!\beta \forall x, y \in K \exists t, t' \in T \cup \{\varepsilon\} : (x = \alpha\beta t\gamma_1 \wedge y = \alpha\beta t'\gamma_2) \Rightarrow (t = \gamma_1 = \varepsilon \vee t' = \gamma_2 = \varepsilon \vee t \neq t')$$

$$S(\alpha, K) := \begin{cases} \beta & E(\alpha, K) \wedge A(\alpha, K) \\ \varepsilon & otherwise \end{cases}$$

Sei $U$ die Funktion, die einem Präfix und einer Kandidatenmenge die Wörter zuordnet, die nach diesem Präfix kommen.

$$U(\alpha, K) := \{tS(\alpha t, K) |\ \alpha t\beta \in K \wedge (t \in (T \vee t = \beta = \varepsilon))\}$$

Sei $Q$ die Funktion, die bei der übergabe einer Wortmenge K und einem präfix $\alpha$ eine Menge aller Wörter zurückliefert, die den Präfix $\alpha$ enthalten.

$$Q(\alpha, K) := \{x\ | x \in K \wedge x = \alpha\beta \text{ mit } \beta \in T^*\}$$

$$O(\alpha, \beta, M) := \begin{cases} \beta & ||M| = 1 \\ \beta[E(\alpha, U(\alpha, M), M)] & |otherwise \end{cases}$$

$$E(\alpha, B, M) := \begin{cases} \varepsilon & |B = \{\varepsilon\} \\ \beta & |B = \{\beta\} \\ (O(\alpha, \beta, Q(\alpha\beta, M))'\ 'E(\alpha, B \setminus \{\beta\}, M)) \text{ mit } \beta \in B & |otherwise \end{cases}$$

Dann ist $\phi := O(\varepsilon, S(\varepsilon, K), K)$ die gesuchte Funktion.

**Definition** $\phi^{-1}$

$$\phi^{-1}(w) := \begin{cases} U(\alpha, \beta) & |\alpha[\beta] = w \\ \{w\} & |otherwise \end{cases}$$

$$U(\alpha, w) := \begin{cases} \phi^{-1}(\alpha x) \cup U(\alpha, s) & |w = x' \, 's \\ \phi^{-1}(\alpha w) & |otherwise \end{cases}$$

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

Sei $T(K_G)$ die Trie Datenstruktur, die die Kandidatenmenge $K_G$ codiert. Wir wissen, dass es eine Funktion $\phi(K_G) = T(K_G)$ exestiert, sowie $\phi \circ \phi^{-1} = id_{P(L(G))}$. Zu finden ist demnach eine Funktion die ein Tire codiert zum Wort der Metasprache sowie wieder encodiert zur Tire.

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

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

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

## Optimierung der Kandidatenbewertung

### Komprimierung

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

### Delegationen

Für die Minimierung der Interaktion wird transitives Abstimmen zugelassen. Akteure können nicht nur für die Option selbst abstimmen, sondern auch ihre

Stimme für Optionsmengen an andere Akteure so delegieren, dass diese im Kontext der delegierten Option für den Akteur mitbestimmen.

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

**TODO:** konkurierende Delegationen

### Besitzverteilung

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

### Grammatik-Erweiterung S

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

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

### zeige S(G) ist eindeutig

**TODO:** zeige S(G) ist eindeutig

## Konsens

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

## Transitionssystem

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

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

### Wortteil

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

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

### Ablauf

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

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

### Zustände

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

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

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

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

$w \in U$ ist das derzeitige Wort

**Aktualisierung**

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

Akteure können jederzeit die Aktualisierungsmenge erweitern:

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

**Transformation**

Sei $\Delta \neq \emptyset$

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

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

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

**Validierung der Aktualisierung**

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

**Optionsmengen**

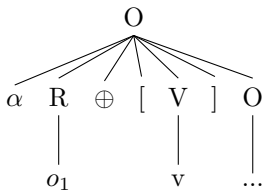
Eine Optionsmenge kann an jeder Position im Wort erzeugt oder erweitert werden, solange die erzeugte Kandidatenmenge durch $\phi^{-1}(w')$ valide bleibt, sowie die Bewertungen der Kandidaten $\phi^{-1}(w)$ nicht verändert werden. Die neu hinzukommenden Optionen dürfen keine Delegationen oder Stimmen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

**Erzeugen einer Optionsmenge**

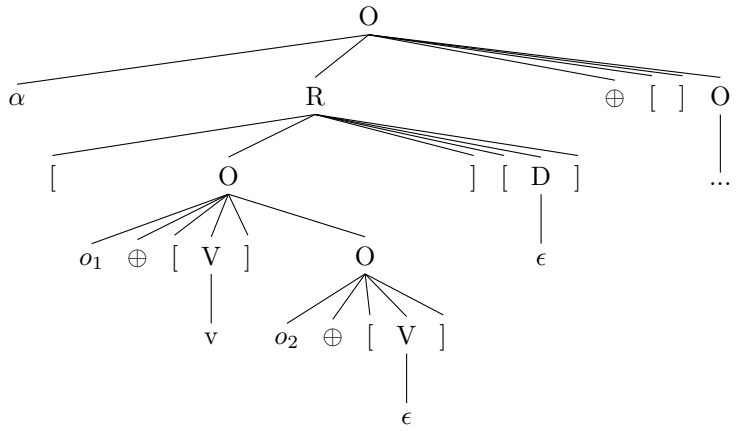
Sei $S(G) = (T', N', S', P')$ eine erweiterte Grammatik. Sei weiter $O, S' \in N'$ sowie $v, o_1, o_2 \in T'^*$ und $0 \leq i \leq j \leq n$ mit $i, j, k, n \in \mathbb{N}$

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

Für $w$ hat $O$ folgende Struktur:




Für $w'$ besitzt O folgende Struktur:




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

### Erweitern einer Optionsmenge

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

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

$r$ muss dabei Delegations- und Stimmfrei sein sowie neu:

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

### Hinzufügen einer Stimme

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

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

### Löschen einer Stimme

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

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

### Hinzufügen einer Delegation

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

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

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

#### Löschen einer Delegation

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

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

## Delegationsprogrammierung

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

#### Beispiele

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

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

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.

# 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:
  - reguläre/ kontextfreie Grammatik, Normalformen, Ableitungen, Eindeutigkeit
  - Algebraisches-Transitionssystem
  - $\Sigma$-Strukturen/-Signaturen
  - Turing-Vollständigkeit
  - asymmetrische Verschlüsselung
- Die Punkte unter Aussicht werden diskutiert, jedoch weder implementiert, noch im Umfang der Anwendung auf die regulären Grammatiken konzipiert.

## Implementation

- Eine Zentralisierte Version der $L(S(G))$-Sprache sowie des Transitionssystem wird implementiert.
  - JISON[7], eine javascript Portierung des Bison[8]/Flex[9] LALR(1) Parser Generators wird so erweitert, dass bei Eingabe einer beliebigen regulären Grammatik $G$ im bison Format[10] dieser einen Parser erzeugt welcher Wörter aus der $L(S(G))$ Sprache als Eingabe bekommt und ein öffentliches Transitionssystem erzeugt.

[7] http://jison.org/
[8] http://www.gnu.org/software/bison/
[9] http://flex.sourceforge.net/
[10] http://dinosaur.compilertools.net/bison/bison_6.html

  - Ein Initialisierungs-Script, welches aus einem Wort der $L(G)$ Sprache und einer Besitzverteilung gegeben als $\{(a,n)|a \in 20byteHexString \wedge n \in \mathbb{N}\}$ ein valides initiales Wort in der $L(S(G))$ Sprache erzeugt.
  - Akteure können durch eine Webseite in die Transaktionshistorie einsehen, sowie das Wort valide manipulieren.
  - Eine Client-seitige asymmetrische Verschlüsselung validiert die Transaktionen der Akteure.
  - Die Konsens-Funktion wird ebenfalls Implementiert und erzeugt zu jedem Stand einen öffentlich verfügbaren Konsens-Kandidaten.
- Einige anschauliche Beispiele regulärer Sprachen mit beispielhaften Delegationsprogrammen werden implementiert und demonstriert.

### Todos

## Aufgaben

# 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.