Contents

Einleitung

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

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

Dezentrale Verifizierung

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

Akteure 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 Berrechnungsschritt der Ethereum-VM 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])

Ein Beispiel für ein Programm währe 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
```

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

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

Digital Geteiltes Objekt

Definition

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

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

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

Erläuterung

Besitzverteilung

Die Besitzverteilung wird änlich 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 mit bestimmen kann.

Eine Konsensfunktion entscheidet über die Ausführung der Kontrolle. Dabei kann eine Zustimmung von $\frac{1}{2}|O|$ eine Ausführung bedingen. Man spricht von einer **majorität**. Eine Zustimmung von $\frac{2}{3}|O|$ heißt **super majorität**.

Transaktionen

Transaktionen sind Manipulationen des derzeitigen Zustandes 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 $rate$.

Eine triviale Transaktionsbedingung ist, dass sich in der Kandidatenmänge nur valide Kandidaten befinden.

Konsens

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

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

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

Bewertung

Qualitätskriterien

Für die Implementation ist auf folgende Qualitätskriterien zu achten:

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

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

Anwendung auf reguläre Grammatiken

Optimierung der Kandidatenmenge

Die reguläre Grammatik wird zu einer contextfreien Grammatik erweitert, so dass alle Wöterer aus der Kandidatenmenge in einem Wort der erweiterten Grammatik codiert werden können. Redundant auftauchende präfixe der Kandidaten werden zusammengefasst. Dafür wird an jedem Ableitungsschrit eine Mehrdeutigkeit zugelassen die zu einer **Optionsmenge** zusammengefasst wird. Die Kandidatenmenge ergibt sich durch die verschiedenen Kombination der mehrdeutigen Ableitung.

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

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

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

Die Bedingung (1) kann auch abgeschwächt werden zu $K_G \subseteq \phi \circ \phi^{-1}(K_G)$. Dieses würde bedeuten, dass Informationen über Kandidaten nicht verlohren 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.: $\{a\ b\}$)

Grammatikerweiterung S1

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

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

Definition ϕ

TODO: Definition ϕ

Definition ϕ^{-1}

TODO: Definition ϕ^{-1}

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

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

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

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

Optimierung der Kandidatenbewertung

Komprimierung

Die Idee hinter der Optimierung der Bewertung ist, dass Aktuere für eine Option stimmen möchten, im kontext zu ihrer historie, gleich welche Optionen sich im verlauf als die Besten herausstellen werden und nicht für individuelle Kandidaten. Wobei die Abstimmung für einen bestimmten Kandidaten ebenfalls mit berücksichtigt werden soll. Daher kann mann Stimmen an Optionen binden, die selbst unbestimmte Optionen enthalten, für die diese Simme ebenfalls geltend gemacht wird.

Eine Bedingung an die Optimierung ist, das jede mögliche Kandidatenbewertung in der Optimierung codierbar ist.

zeige: eine beliebige Kandidatenbewertung ist in S2 codierbar.

TODO: zeige: eine beliebige Kandidatenbewertung ist in S2 codierbar.

Delegationen

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

TODO: konkurierende deligationen

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. Ebenfalls können Transitionen als valide Manipulationen eines Wortes definiert werden. Transitionsbedingungen können so auf der Ableitungsstruktur des Wortes definiert werden.

Akteure werden durch einen 20 byte hex String codiert.

Grammatikerweiterung S

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

TODO: start als optionsmenge

zeige S(G) ist eindeutig

TODO: zeige S(G) ist eindeutig

Konsens

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

Transitionssystem

Die Idee hinter der manipulation ist des es sich bei den Wörter um eine Persistente Datenstruktur handelt. Demnach können Optionen nur hinzu kommen, jedoch nicht gelöscht werden. Jeder Akteur kann Delegationen und Stimmen abgeben sowie die eigenen löschen, hat jedoch keine Macht über die Stimmen und Delegationen der Anderen.

Die Manipulation wird durch ein **initialisiertes Transitionssysthem** T beschrieben.

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

Zustände

Ein Zustand der ist eine Struktur der form $(U, consens, w)$

U ist das Universum der Struktur und wird definiert durch:

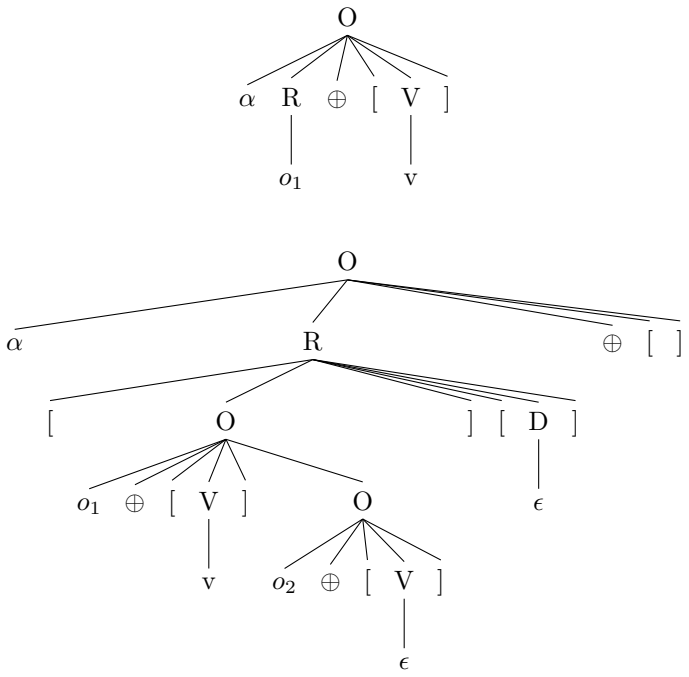
$$U := L(S(G))$$

consens ist die für $L(S(G))$ definierte konsens Funktion: $w \in U$ ist das derzeitige Wort

Transitionenbedingungen

TODO: transitionsbedingungen sind für CFG definiert: ändern

Erzeugen einer Optionsmenge



Sei $\alpha, \beta, o_1, o_2 \in T'^*$ sowie $O \neq R \wedge R \in N$ und $V \rightarrow_P^* v$ Sei $w = \alpha o_1 \beta \oplus [v]$Sei weiter $R \rightarrow_p^* o_1$ sowie $R \rightarrow_p^* o_2$ Dann ist $w := \alpha[o_1 \oplus [v]o_2 \oplus []][]\beta$ eine valide Transition.

Erweitern einer Optionsmenge

Sei $w = \alpha[o]\beta \wedge S' \rightarrow_P^* \alpha R\beta$ mit $O \rightarrow_P^* o, o \in T'^*, R \in N$ Dann ist $w := \alpha[or \oplus []]\beta$ mit $R \rightarrow_P^* r, r \in T'^*$ eine valide Transition.

Hinzufügen einer Stimme

Sei $w = \alpha r \oplus [v]\beta$

Dann ist $w := \alpha r \oplus [v[hn]]\beta$ mit $n = float$ eine valide Transition.

Löschen einer Stimme

Sei $w = \alpha[hn]\beta$

Dann ist $w := \alpha\beta$ mit eine valide Transition.

Hinzufügen einer Delegation

Analog der Stimme

Löschen einer Delegation

Analog der Stimme

Delegationsprogrammierung

Context Delegation

Validator

A/B Tester

Umfang der Arbeit

Konzeptuell

- Die Punkte unter Aufgaben (TODO's) werden ausformuliert.

- Dieses Papier wird mit anschaulichen Beispielen, Erklärungen, Quellen sowie verweisen und Erklärungen grundlegender Konzepte ergänzt. Solche Konzepte währen:
  - reguläre/ contextfreie Grammatik, normalformen
  - algebraisches Transitionssystem
- 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 Transitionsysthems wird implementiert
  - JISON[1], eine javascript Portierung des Bison[2]/Flex[3] LALR(1) Parser Generators wird so erweitert, dass bei Eingabe einer beliebigen Grammatik G dieser einen Parser erzeugt welcher Wörter aus der $L(S(G))$ Sprache als Eingabe bekommt und ein öffentliches Transitionsysthem erzeugt. States
  - Ein Initialscript, welches aus einem Wort der $L(G)$ Sprache und einer Besitzgewichtung gegeben als $\{(a,n)|a \in \ 20byteHexString \wedge n \in \mathbb{N}\}$ ein valiedes initiales Wort in der $L(S(G))$ Sprache erzeugt
  - Akteure können durch eine Öffentliche Schnittstelle(Webseite) in die Historie einsehen, sowie das Wort valide manipulieren.
  - Eine client-seitige asymetrische Verschlüsselung validiert die Akteure.
  - Die Konsensfunktion wird ebenfalls Implementiert und steht öffentlich verfügbar.

Todos

Aufgaben

[1] http://jison.org/
[2] http://www.gnu.org/software/bison/
[3] http://flex.sourceforge.net/

Aussicht

Anwendung auf contextfreie Grammatiken

Dezentralisierung

Anonymisierung

Manipulation der Besitzverteilung

Manipulation der Grammatik

- änderungen der Grammatik berücksichtigen

Quellen

References

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

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

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

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