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 möchte ich eine Meta-Programmiersprache $S(G)$ für eine beliebige eindeutige kontextfreie Grammatik G herrausarbeiten, die die verteilte Steuerung Contextfreier Inhalte ermöglichen soll. Hierfür werden in (1) zwei Funktion angegeben. Eine erweitert eine gegebene kontextfreie Grammatik G zu einer $S(G)$, eine Konsensfunktion erzeugt aus einem Wort der Sprache $L(S(G))$, ein Wort in der ursprünglichen Sprache $L(G)$. Manipulationen eines Wortes einer $L(S(G))$-Sprache sind bedingungen unterworfen, die in (2) angegeben sind.

Dezentrale Verifizierung

Das auf dem Bitcoin-Protokoll aufbauende Ethereum [Woo14], beschreibt ein Protokoll in einem Netzwerk 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 werden durch ein 20-byte Adresse identifiziert. Ihre Interaktion mit dem Netzwerk ist durch ein asymmetrisches Kryptosystem gesichtert.

Das Netzwerk finanziert sich durch die Validierung der Programme. Für jeden Berrechnungsschritt wird ein kleiner Betrag vom Interagierenden Akteur erhoben. Ist nicht genügend Wert verfügbar, terminiert die Transaktion und wird vom Netzwerk abgelehnt. So wird auch das Halteproblem umgangen.

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
```

Digitale Geteiltes Objekt

Digitale Geteilte Objekte müssen in digitaler repräsentation vorliegen, sowie für die Anteilseigner in verschlüsselter oder unverschlüsselter form zugänglich sein.

Beispiele währen:

- medialen Inhalte wie Bilder, Texte oder Videos
- Programme und contextuelle Angaben wie deren Compiler oder die Architektur
- Besitzrepräsentation wie Bitcoin Anteile oder Aktien

Sie können einerseits in direkter Form, oder als Referenz unter angabe einer URI angegeben werden.

Andererseits können digitale geteilte Objekte dynamische Ihnalte repräsentieren, welche eine Syntax besitzen und durch ein Compiler Interpretiert werden können. Da URI's durch reguläre Sprachen (REG) konstruiert werden können und $L(REG) \subsetneq L(CFG)$ ist. Reicht es zur betrachtung aller digitaler Objekte solche zu betrachten, die durch eine CFG Syntax besitzen, oder durch eine Solche referenziert werden können.

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

$$K_G \subseteq \{w | S \to_P^* w \wedge w \in T^*\} \wedge |K_G| \geq 1$$

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.

Repräsentation

Akteure

Da ein Geteiltes Objekt auch Werte in Form einer Kryptowährung wie Bitcoin beinhalten kann, strategische wertvolle Attribute in einem Netzwerk besitzt, wie Reputation oder Namensrechte, geschlossene Informationen beinhaltet auf die nur die Eigentümer zugreifen können oder Gewinnüberschüsse an die Besitzer ausschüttet, besitzt dieser auf einem Freien Markt einen reellen und spekulativen Wert. Somit könenn wir davon ausgehen, dass es im Interesse jedes rationalen Akteurs ist, seinen Besitzanteil zu maximieren. (**AMAX**)

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 werden **immer** von einem Akteur a ausgelöst und besitzen die Form (a, O'). O' ist dabei das manipulierte geteilte Objekt.

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

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.

Implementation

Einleitung

Eine Idee hinter der Implementation ist nah am Parsebaum der Grammatik zu arbeiten um die Kandidatenmenge dadurch zu komprimieren. Die Kandidatenmenge ergibt sich durch die verschiedenen Kombination der mehrdeutigen Optionen, die an jedem Ableitungsschritt entstehen können.

Der derzeitige Ansatz ist es eine Grammatikerweiterung $S : CFG \to CFG$ zusammen mit einem Isomorphismus $\phi : SOBJ_G \to L(S(G))$ und einer Konsensfunktion $c : L(S(G)) \to L(G)$ zu finden, so dass alle Transaktionen des Objektes als Transaktionen eines contextfreien Wortes angesehen werden können. Die Transaktionsbedingungen können als Struktur des Wortes aufgefasst werden.

Sprach-Erweiterung

Bewertung

Diese werden entweder **direkt**, also vom Akteur selber, oder **indirekt** also durch eine Delegation an einen anderen Akteur angegeben.

1.1 erweiterung einer CFG zu einer SCFG

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

Eine Erweiterung auf eine SCFG definieren wir als $S(G) := (N', T', S', P')$, mit:

$N' := N \cup \{O, D, A\}\ mit\ O, D, A \notin N$

$T' := T \cup \{[,], \oplus, number, float, hash\}\ mit\ [,], \oplus \notin T$

number stellt eine ganzzahlige Nummer dar. **float** stellt eine fließkommazahl dar. **hash** ist ein identifikatior für einen akteur, hier ein 20 bytes hex string.

$P' := P \cup P_{Options} \cup P_{Start} \cup P_{Delegations} \cup P_{Voting} \cup P_{Acteurs}$

$P_{Options} := \{R \to [O][D], O \to r \oplus [V]O | R \to r \in P \text{ mit } r \in \Sigma^*\} \cup \{O \to \varepsilon\}$

$P_{Start} := \{S' \to [A]S\}$

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

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

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

zu zeigen

- eindeutigkeit der SCFG

Compile

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

- zirkuläre delegationen
- implekation von Arrow's Theorem auf strategisches voten

2. Manipulation

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 eindeutige Grammatik sowie $G' = S(G) = (N', T', S', P')$ die dazugehörige SCFG.

Qualitätskritärien

1. operationen wie $vote, delegate, addOption$ sind effizient in bezug auf Laufzeit und Speicherverbrauch.
2. Die Verifikation der Transaktion durch das Netzwerk soll möglichst effizient sein.

States

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

U ist das Universum der Struktur und wird definiert durch:

$$U = \{w | S' \rightarrow_p^* w \wedge w \in T'^*\}$$

compile ist die in (1.2) definierte Funktion der Form: $compile : L(G') \rightarrow L(G)$

$w \in U$ ist das derzeitige Wort

Transitionen

Bei der Interaktion eines Programms, ist der Akteur bekannt und wird durch ein Public Key ausgewiesen. Der Akteur kann in einem Interaktionsschritt das derzeitige Wort altuallisieren, wenn es bestimmte Bedingungen erfüllt. Eine Interaktion ist also ein Tupel (h, w') so, dass h der Public Key des Akteurs ist, sowie w' das neue Wort.

Erzeugen einer Optionsmenge

Sei $\alpha, \beta, o_1, o_2 \in T'^*$ sowie $R \in N$ 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

Transaktion

Delegationsprogrammierung

Context Delegation

A/B Tester

zu untersuchen

- wie werden neue anteile vergeben

Umfang der Arbeit

Konzeptuell

- Überarbeitung der Spracherweiterung
  - macht eine Spracherweiterung überhaupt sinn?

Implementation

- Eine Zentralisierte Version der $L(S(G))$-Sprache sowie des Transitionsysthems wird implementiert
  - JISON, ein Bison/Flex ähnlicher javascript LALR(1) parser generator 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.
  - 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.
  - Die Konsensfunktion wird ebenfalls Implementiert.
- Es wird keine Dezentrale version Implementiert.

- Die Delegationen sowie die Abstimmungen werden direkt Implementiert und können jederzeit beteiligten Akteuren zugeordnet werden. Eine anonyme Lösung, mithilfe einer homomorphen verschlüsselung oder eines zero knowledge proofs wird zwar diskutiert, jedoch weder konzipiert noch implementiert

Aussicht

Dezentralisierung

- änderungen der Grammatik berücksichtigen

Quellen

References

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

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

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