Blockchain Engineering

Plutus und Marlowe

Dr. Lars Brünjes

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Ostbayerische Technische Hochschule
Amberg-Weiden

OTH
REGENSBURG

MIND
MODULARES INNOVATIVES
NETZWERK FÜR DURCHLÄSSIGKEIT

AUFSTIEG DURCH
BILDUNG
OFFENE HOCHSCHULEN

19. Oktober 2019

- Plutus und Marlowe sind zwei Smart-Contract-Sprachen, die von IOHK entwickelt wurden.
- Obwohl beide in erster Linie auf Cardano abzielen, könnten sie auf einer Vielzahl von Blockchains verwendet werden.
- Beide sind in Haskell implementiert.
- Plutus ist Turing-vollständig und allgemein, Marlowe ist eine nicht-Turing-vollständige DSL für finanzielle Verträge.

- Wie Bitcoin benutzt Cardano UTxO-basierte Buchführung, und wie Bitcoin-Script erweitert Plutus das normale UTxO-Modell, indem es Skripte mit Inputs und Outputs assoziert, deren Kombination über die Gültigkeit einer Transaktion entscheidet.
- Anders als bei Bitcoin gibt es nicht nur Input- und Output-Skripte, sondern zusätzlich an jedem (Skript-)Output noch ein Daten-Skript, welches *nicht* Teil der Adresse ist. Daten-Skripte können benutzt werden, um beliebig komplexen Status mit Outputs zu assoziieren, was bei Bitcoin nicht möglich ist.
- Zur Validierung werden die Input-, Output- und Daten-Skripte mit der zu validierenden Transaktion kombiniert, d.h. sowohl der Status aus dem Daten-Skript als auch die Transaktion selbst (mit all ihren Inputs und Outputs) stehen zur Verfügung.
- Außerdem ist Plutus Turing-vollständig (und benutzt ein ähnliches System wie Ethereum zur Vermeidung von Endlosschleifen).
- All diese Eigenschaften gemeinsam ermöglichen Smart Contracts in Plutus, die mindestens so mächtig wie Ethereum Smart-Contracts sind.

- Die Plutus zugrunde liegende “Maschinensprache” heißt Plutus Core.
- Im Gegensatz zu Bitcoin Script und der EVM ist Plutus Core nicht Stapel-basiert, sondern im Wesentlichen System Fω, eine Version des Lambda Kalküls mit mächtigem Typ-System, auf der auch Haskell basiert.
- Plutus ist nicht nur in Haskell implementiert, sondern wird auch in Haskell programmiert, d.h. Haskell ist für Plutus Core, was Solidity für die EVM ist.
- Dies ermöglicht ein nahtloses Zusammenspiel von Onchain- und Offchain-Code (wohingegen Solidity nur für Onchain-Code benutzt werden kann und man z.B. Javascript für Offchain-Code benutzen muss.)

- Das Konto-Modell und das UTxO-Modell haben beide ihre Vor- und Nachteile.
- Das Konto-Modell mit seinen sich ständig verändernden Kontoständen benutzt “mutable state”, während Outputs im UTxO-Modell unveränderlich sind.
- Ein ähnliches Verhältnis haben auch imperative und funktionale Programmiersprachen zueinander.
- Daher passt es gut, Smart Contracts für eine Konto-Modell-basierte Blockchain wie Ethereum in einer imperativen Programmiersprache wie Solidity zu programmieren, während Smart Contracts für das UTxO-basierte Cardano in der funktionalen Sprache Plutus erstellt werden.

- Marlowe ist eine DSL (Domain Specific Language) spezialisiert auf finanzielle Verträge.
- Es beruht auf der wegweisenden Veröffentlichung “Composing Contracts: An Adventure in Financial Engineering” von Simon Peyton Jones et al. aus dem Jahre 2000 (Simon Peyton Jones ist einer der Erfinder von Haskell), wurde aber an die speziellen Bedingungen der Blockchain angepasst.
- Marlowe ist *nicht* Turing-vollständig und einfach genug, statische Analysen zu erlauben, die wichtige Eigenschaften eines Marlowe-Contracts automatisch beweisen können.
- Andererseits ist Marlowe mächtig genug, um einen großen Anteil aller gängigen finanziellen Verträge abzubilden.

- Marlowe Contracts haben eine endliche Zahl von Schritten (keine Endlosschleifen), und man kann die Zahl der Schritte durch statische Analyse nach oben beschränken, d.h. vorab wissen, wie lange man höchstens warten muss, bis der Vertrag beendet ist.
- Marlowe garantiert, dass kein Geld im Vertrag “gefangen bleibt”: Bis zum Ende des Vertrages wird alles Geld, was in den Vertrag eingezahlt wurde, an eine oder mehrere der Vertragsparteien zurück gezahlt werden.

- Marlowe unterscheidet zwischen drei Arten von Aktionen (Actions):
  - Deposits: Eine Vertragspartei deponiert Geld in dem Vertrag.
  - Choices: Eine Vertragspartei trifft eine Wahl (entscheidet sich für einen Wert aus einer Liste von Möglichkeiten).
  - Notifications: Ein externes Ereignis benachrichtigt den Vertrag über eine externe Beobachtung.
- Ein Smart Contract kann solche Aktionen nicht erzwingen, weswegen sie in Marlowe immer mit einem Timeout und einer Alternative für den Fall, dass die Aktion nicht stattfindet, versehen sind.

- Marlowe ist eine spezielle Art DSL, eine sogenannte EDSL (Embedded Domain Specific Language), d.h. eine DSL, die in eine “Gastgebersprache” eingebettet ist.
- Die “Gastgebersprache” für Marlowe ist Haskell.
- Eine EDSL hat den Vorteil, dass alle Feautures des “Gastgebers” verfügbar sind, um einen Ausdruck in der DSL zu erstellen.
- Für Marlowe bedeutet dies, dass man Haskell benutzen kann, um Marlowe Contracts zu erzeugen, und dass Marlowe Contracts einfach besondere Werte eines Haskell-Datentyps sind.

- Alternativ können Marlowe Contracts auch in Blockly, einer grafischen Sprache, erstellt werden.
- Dies ist mühsam für komplexere Verträge, aber ideal, um die Sprache zu lernen.

- Die Actus Financial Research Foundation hat einen Standard für Finanzverträge erstellt, der in einer Taxinomie kategorisiert ist und detailliert technisch spezifiziert wird.
- Der Actus Standard beruht auf dem Verständnis, dass Finanzverträge rechtlich bindende Übereinkünfte zwischen zwei oder mehr Parteien über zukünftige Zahlungen sind. Historisch wurden solche Verträge informell mit Worten beschrieben, was zu Mehrdeutigkeiten und Unklarheiten geführt hat.
- Stattdessen beschreibt der Actus Standard Verträge mittels einer Menge vertraglicher Begriffe und deterministischer Funktionen, die diese Begriffe auf zukünftige Zahlungsverpflichtungen abbilden.
- Dadurch ist es möglich, die große Mehrheit aller Finanzinstrumente durch etwa 30 Typen und Muster zu beschreiben.
- IOHK plant, Plutus und Marlowe zu benutzen, um den gesamten Actus Standard auf Cardano zu implementieren.

Schreiben Sie zwei Marlowe Contracts, in denen Alice zuerst 100 Ada deponieren sollte. Falls Sie dies bis Slot 5 nicht tut, geschieht nichts. Wenn Sie es aber tut, dann...

- soll das Geld im ersten Vertrag gleichmäßig an Bob und Charlie ausgezahlt werden und
- im zweiten Vertrag in Slot 10 das Geld an Alice zurückgezahlt werden.

Hinweis

Diese Publikation wurde im Rahmen des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Bund- Länder- Wettbewerbs “Aufstieg durch Bildung: offene Hochschulen” erstellt. Die in dieser Publikation dargelegten Ergebnisse und Interpretationen liegen in der alleinigen Verantwortung der Autor/innen.