Plutus Pioneer Program

2기, 1주차

강지훈 [**GEST**]

2021년 7월 2일

1 필요한 사전 지식

플루투스 개발 문서에 의하면, 플루투스 플랫폼은 다음과 같이 정의되어 있다:

> "플루투스 플랫폼이란 **스크립팅 기능(Scripting Capabilities)**을 가진 **분산 장부(Distributed Ledger)**를 이용하는 **어플리케이션(Application)**을 개발하기 위한 플랫폼(Platform)이다."

플루투스 파이어니어로서, 우리는 그 플랫폼을 사용하고 테스트 해보며, 분산 장부인 '카르다노 블럭체인'위에서 실행되는 다채로운 '어플리케이션'들을 만들 사람들이다. 실로 멋지지 않은가?

지금까지의 내용을 보면 희망으로 가득찰 수도 있고, 막막할 수 도 있다. 볼드체로 쓰여진 단어들에 집중하면서 위 정의를 더 깊이 있게 이해해보자.

어플리케이션 - 이미 우리는 스마트폰에 여러 '앱'들을 설치해보면서 '어플리케이션'이라는 단어에 익숙하다. 어플리케이션이란, 사용자와 디지털 체계의 논리와 구조를 연결해주는 목적과 방향성이 제시된 유용한 소프트웨어이다.

플루투스 어플리케이션 개발자로서 카르다노 블럭체인과 카르다노 사용자들 사이를 이어주는 코드를 짜는 것이 우리의 역할이다. 특히 이 코드를 통해 간단한 송금과 같은 트랜잭션뿐만 가능케 하는 것이 아니라, 더욱 복잡하고 유용한 일들을 위해 작성을 하는 것이 우리의 목표와 바램이다. 바로 인간과 블럭체인 사이, 그 그림을 완성할 퍼즐 한 조각을 만들어 끼워넣는 셈이 되는 것이다!

이는 엄청난 능력처럼 들릴 수도 있다, 하지만 알다시피: "위대한 힘에는 위대한 책임이 따른다(with great power comes great responsibility)". 위와 같은 어플리케이션 개발자들은, 사용자 경험에서부터 블럭체인 동기화 문제들까지 다양한 이슈들과 번외사항들을 염두에 두어야 한다. 특히나, 카르다노 블럭체인 처럼 실질적으로 중요한 자산의 이동이 일어나는 곳에서는, 더욱이나 사용자들의 자산에 대해 책임감을 느끼고 신중히 개발해야 하는 의무를 가져야 한다고 생각한다. 하지만 너무 걱정할 필요는 없다 - 바로 이러한 부분을 돕기 위해 '플루투스 플랫폼'이라는 것이 존재하기 때문이다! 플루투스 플랫폼은 개발자들에게 위와 같은 이슈들이나, 그 어떠한 크고 작은 기능들을 구현하는 것에 있어서 안전하고 표현력 있

는 공간을 제공하려고 한다. 물론 언제나 마지막 단계는 개발자의 몫이다; 언제나 칼 끝에 서서, 앞으로 세상을 바꾸어 나갈 서비스를 만들어 나가는데에는 공부와 실험, 테스트를 게을리 해서는 안된다.

분산 장부 - "좋아, 좋아, 알겠다고, 우리 모두를 위한 선의의 안전한 소프트웨어를 만들겠다고 약속할게, 그럼 도데체 어떤 서비스들을 만들 수 있는 건데? 장부라는게 뭔데?" 역시나 그런 질문이 나올 것이라고 예상하고 있었다! 역사적으로 보면, 장부란 원래 중세 유럽 교회들이 **공개되고 공유되는 공간**에 **언제나 항상 존재**하는, 그래서 누구든지 볼 수 있는 **책**(특히 성경)을 두는 전통에서 시작되었다고 한다. 그 속에 담긴 철학은 현대의 '장부'의 개념에서도 찾아볼 수 있다. 우리는 여러 사람들 사이에서 일어나는 자산의 움직임이나 위치 등에 대한 거래(트랜잭션[Transaction])를 장부라는 것에 기록하고, 속한 모두가 그 정보를 검토하고 확인 할 수 있게 한다.

이러한 '기록'의 목적은 간단하게 **합의(Consensus)**라고 표현해도 과언이 아니다. 그 누가 함부로 "아니야, 나 안 그랬어", "우리가 언제 그러기로 했어?", 또는 "뭐야! 저번에 그렇게 한다고 말해줬잖아!"와 같은 독자적이거나 악의적인 말들이 나오지 않도록 정리해야 되기 때문이다. 장부의 모든 사용자는, 장부에 쓰여 있는 내용들이 그들 사이에서는 부정할 수 없는 진실이라고 약속을 하는 것이다.(그러므로 특히나 거래를 기록할때는, 그 거래에 해당하는 당사자들로 부터 더욱이 '서명'과 같은 인증 체계를 통해 기입된 사실에 믿음을 추가해야 한다.) 이런 식의 "누가 무엇을 언제 누구에게 했는가"가 담긴 정보들의 역사 기록은 자연스럽게 다양한 사용처들을 제시한다.

- 제품/서비스 버젼 관리, 예방접종 기록 확인, 학생부 관리, 송금 거래 추적 등과 같은 간단한 장부 사용에서 부터
- 온/오프라인 소셜 네트워킹, 게임 상태 트래킹, 호스트-클라이언트 매칭, 저작권과 소유권 확인, 공유 컴퓨팅 클러스터 관리, 분산 투표 등과 같은 응용된 사용 등이 예가 될 수 있다.

지금까지 일반적인 '장부'가 무엇인지 한번 같이 알아보았지만, 더욱 자세히 우리의 최애 블럭체인인 카르다노 블럭체인은 어떠한 장부인지 알아보도록 하자.

카르다노 블럭체인은 디지털 **분산** 장부이다. 여기서 '분산' 이라 함은 **'복제되고, 공유되고, 동기화되는'**을 의미한다. 이는 위에서 말한 '합의'를 이루는데에 있어 중앙 관리 체계(위의 교회 예제와 같은)에 비해 더 많은 어려움을 제기 한다. 하지만 만약 그런 합의를 이루어 낼 수만 있다면, **무신뢰(Trustlessness)**가 자연스럽게 찾아올 수 있다. 이는 다양한 (바라옵기는) 긍정적인 사회-경제적인 효과를 불러 일으킨다. 마치 위의 교회 예제에서 모든 사람들이 각자 자기만의 성경을 가지는 것과 비슷하다. 교회가 관리하는 단 하나의 진리의 백서가 관리자인 교회에 의해 수정되지 않았을 것이라는 신뢰를 하지 않아도 되며, 언제든지 자신의 복사본을 가지고 능동적이고 주관적인 생각을 해나갈 수 있게 된다. 누군가의 주장이나 설득에는 빠르게 자신의 장부를 보고 그 진위를 파악 할 수 있으며, 개인정보를 남에게 위탁하여 정보를 얻는 행위를 줄이는 등 다양한 효과들을 볼 수 있다.

카르다노 블럭체인은, 디지털 분산 장부로서 위의 '합의'를 '지분증명'(Proof-of-Stake)이라는 알고리즘으로 해결한다. 제일 간단하게는 '일시적으로 (장부 기입) 권리를 장부가 잘못되거나 잘못된 내용이 기입되었을때 제일 잃을 것이 많은 주체에게 위임하여 선의적 행동을 유도하는' 방식이라고 말 할 수 있지만, 훨씬 더 많은 내용을 내포하고 있다. 또한 지분증명 이외에도 작업증명(Proof-of-Work)이라던가 다양한 합의 알고리즘들이 존재하니, 독자들에게 그러한 부분들에 대해서 더 탐색해보고 검색하여 읽어보기를 권한다! 여기 이 링크에는 카르다노의 합의 프로토콜인 우로보로스 프로토콜(Ouroboros Protocol)을 설명하는 웹사이트가 연결되어 있다.

여기까지 온 우리가 모두 동의할 수 있는 지식을 쌓았기를 바라며(하! 이 또한 합의 아닌가!), 이러한 디지털 분산 장부를 사용하여 만들 수 있는 다양하고 유용한 서비스들에 대한 아이디어가 샘솟고 있길 바란다!

스크립팅 기능 - 자연스럽게 우리는 플루투스 플랫폼의 핵심 부분인 스크립팅 기능으로 넘어오게 된다. 더 깊이 들어가기에 앞서, 우선 분산 장부들이 보통 어떻게 구성되어 있는지 정리해보도록 하자.

계정 기반 - Account based - 이름에서 말하듯 간단하다. 이러한 장부는 포함된 **계정 리스트(List of Accounts)**와 각 계정마다의 **잔액(Balance)**을 기록한다. 여기서의 거래(트랜잭션)에서는 이 장부의 전역적 상태를 받아, 그 거래에 연관된 계정들의 잔액의 증감으로 표현된다. 현재 이더리움 블럭체인의 기본적인 구동 방식이다. 이러한 계정 기반 장부는 시스템의 표현력과 개발-가능성/생산성을 향상 시키는 좋은 구조이나, 거래마다 확인해야 하는 공유되고 있는 전역 계정/잔액 상태는 다양한 부작용을 일으키기 쉽다. 이는 개발자에게 추가적인 고려사항이 되며, 기대하지 못한 부작용이나, 바라지 않는 부작용에 대한 예외 처리등과 같은 어려움을 제공한다.

UTxO 기반 - UTxO(Unspent Transaction Outputs)는 '사용되지 않은 거래 출력값'을 의미한다. 이런 구조의 장부에서는 거래의 단위가 계정과 잔액이 아닌, UTxO들이 된다. UTxO들은 **이름표가 달린 케익**이라고 비유 할 수 있다. 그 이름표는 단순히 그 케익이 누구의 케익인지 정의해주며(그리고 한 사람은 여러 케익을 가질 수도 있다! 이야!), 케익 자체는 어떤 가상화폐 자산 또는 토큰을 의미한다. 거래시에는 언제나 이 케익이 단위이고, 이 케익을 조각 내어서 그 조각만을 가지고 거래를 만들지 않는다. 그 '거래'(트랜잭션)가 케익 전체를 온전하게 받아서 그 거래 내용에 따라 자르거나, 합치거나, 필요한대로 사용하여 새로운 케익을 만드는 역할을 담당한다. 즉 거래에서는 계정에 연관된 잔액이 업데이트가 되는 것이 아닌, 케익의 갯수, 크기와 이름표만이 업데이트 되는 것이다. 온전한 케익만이 거래로 들어가 소모되며, 새로운 소모되지 않은 온전한 출력값 케익들(UTxO)들이 나오게 된다. 거래는 즉 전역적인 정보를 필요로 하지 않으며, 단순히 입력값으로 받게 되는 케익들만을 출력값으로 매핑해주는 것에만 신경을 쓰면 된다. (음? 뭔가 함수 냄새가 나는군! 부작용이 없을 것 같은 느낌이 드는군! 하스켈이 떠오르는군!). 비트코인과 카르다노가 원초적으로는 이런 UTxO 기반 장부로 운영된다.

카르다노에서는 위의 UTxO 기반 장부를 더 확장해서, EUTxO(Extended UTxO) 모델을

기본 장부 모델로 사용하고 있다. 이 '확장'이 바로 카르다노 블럭체인의 '스크립팅 기능'을 제공하게 되는 것이다! **이 스크립트들은 다양하고 복잡한 거래들을 정의하고 생성하는 것에 대한 내용을 담고 있으며, 추가적으로 그렇게 생성된 트랜잭션들이 UTxO들을 소모하기 위한 검증('validation')절차 또한 담고 있다.**

이러한 추가적인 기능은 일반적인 장부의 역할에 **'스마트 컨트랙'**의 기입을 부여하여, 이전에 설명한 "'누가 무엇을 언제 누구에게 했는가'를 정의하는 역사 기록"에 '만약에', '이런 조건에는', '언제까지' 등과 같은 재미있는 논리들을 얹을 수 있게 된다. 아 이것이야 말로 정말 멋진 신세계 아닌가?! 이러한 구조를 가진 장부를 가지고 할 수 있는 것은 그야말로 무궁무진 할 것 아닌가!

바로 그 무궁무진함이 우리가 플루투스 파이어니어로서 꽃 피워나갈 부분이다. EUTxO 모델을 가지고 어떠한 참신한 논리, 필요한 서비스들을 만들어 갈 수 있을까? 그런 걸 하려면 어떻게 무엇을 해야 할까? 고민을 하며 1주차 강의 내용에 대해 정리하도록 하자.

2 1주차 - EUTxO 모델과 잉글리시 옥션(경매)

플루투스 파이어니어 프로그램(2기)의 첫번째 강의는 위에 언급된 '스크립팅 기능'에 대해 더욱 깊게 설명하며 시작한다.

2.1 UTxOs

소모되지 않은 트랜잭션 출력값(UTxO)은 다음과 같이 정의된다.

> "블럭체인에 기입된 **이전 거래들의 출력**으로 나온 **아직 소모되지 않은** 거래 출력값"

각 거래는 **n개의 입력을 받고 m개의 출력을 생산해낸다.**. 여기서 입력과 출력은 모두 UTxO(위의 케익들 ;))이며, 입력으로 소모될때는 온전한 상태로(분할해서 거래에게 주지 않는다) 거래의 입력에게 주어진다. 거래 입장에서는 자신에게 주어진 UTxO를 소모할 수 있는지 확인 하기 위해, UTxO 주인들의 전자 서명이 추가로 주어진다.

만약 누군가에게 당신의 소중한 10 ADA를 보내려고 한다고 하자, 그리고 지금 당신의 명의로 100 ADA의 값을 가진 하나의 UTxO가 존재한다고 하자. 이때 그 거래를 성사 시키고자 그 UTxO를 90 ADA 하나, 10 ADA 하나, 이렇게 2개의 UTxO로 분할해서 10 ADA UTxO만 거래에 넘기지 않는다. 본인 이름 아래 있는 100 ADA 짜리 UTxO를 온전한 상태로 거래에 넘기며, 그 거래가 2개의 출력 UTxO를 만들것이다. 첫번째는 10 ADA의 값을 가지고 수취인의 이름이 달린 UTxO이며, 다른 하나는 90 ADA 값을 가지고 당신의 이름이 달린 거스름돈으로 받는 UTxO이다.

마찬가지로, 만약 당신에게 이전의 여러 거래에서 나온 2개의 UTxO가 있다고 하자. 하나는 10 ADA의 값을 가지고 있고 다른 하나는 20 ADA를 가지고 있다. 이 상태에서 30 ADA를 소모하는 거래를 진행하려고 할 때, 그 두개의 UTxO를 합쳐 하나의 30 ADA 짜리 UTxO로 만들어서 트랜잭션에게 넘기지 않는다. 트랜잭션에게는 10 ADA UTxO와 20 ADA UTxO 둘 다 온전한

상태로 입력값으로 전달되며, 그 합이 30이라는 내용과 메인 로직 자체는 트랜잭션에서 다루는 부분이다.

이는 카르다노의 장부 구조를 깔끔하고 직관적으로 만들어 준다. 각 트랜잭션마다 '값'의 보존이 성립되기 때문이다. 입력 UTxO들의 값들의 합은 언제나 출력으로 나오는 UTxO들의 값들의 합과 일치하게 된다. 거래는 결국 자산의 재분배를 표현하는 것 뿐! 하지만 잠깐! 언제나 그렇듯 예외 사항들이 있다.

그 두가지 예외 사항들은 다음과 같은 사항들을 고려하면서 발현된다:

1. 거래 수수료: 거래가 블럭에 기입되고, 나아가 그 블럭이 장부(카르다노 블럭체인)에 기입될 때, 그 거래의 복잡도에 따라 작은 거래 수수료가 붙는다. 이는 블럭체인에 블럭을 기입하는 행위를 촉진(블럭 기입을 하는 노드를 운영하는 네트워크 노드 운영자들에게 거래 수수료가 '임금'으로 지급된다)시키고 네트워크 자체의 운용이 일어날 수 있게 하기 위해 존재하는 카르다노 네트워크 특성이다. 즉 거래 수수료에 의해 보통은 거래의 입력 UTxO들의 값들의 합이 출력값들 보다 살짝 크다.

2. 네이티브 토큰: 카르다노 블럭체인 위의 토큰을 생성(mint)하거나 소멸(burn)할 때, 입력값과 출력값 사이 합이 일치 하지 않을 수 있다. 토큰을 생성하면, 출력값에 토큰이 추가되어 커지고, 토큰을 소멸하면 입력값의 토큰이 사라짐으로 출력값이 작아진다.

2.2 (E)UTxOs

표현을 빌리자면, EUTxO는 날개 달린 UTxO 모델이다. 여기서 날개란 '스마트 컨트랙'을 의미한다. 그 날개를 설명하기에 앞서 우선 UTxO 모델을 자세하게 설명하고자 한다.

기본적으로 어떤 거래가 UTxO의 소모를 하기 위한 검증에는 전자 서명 확인이 들어간다. 개념적으로는 입력으로 사용되는 UTxO에게 'validator'라는 함수가 추가되어 거래에게 제공된다고 볼 수 있다. 이 함수 속에는 입력 UTxO 주인의 공개키 주소가 들어 있다. 트랜잭션은 위의 UTxO의 소모에 대한 자격을 확인하기 위해 'redeemer'라는 데이터를 가지고 있고, 이 데이터에는 소모하려는 UTxO 주인의 개인키로 만든 해시값이 존재해야 서명확인이 성사된다.

EUTxO 모델에서는, 이러한 전자 서명 확인 절차에 추가적으로 플루투스 코어라는 언어로 컴파일된 **임의의 그 어떠한 로직을 추가**할 수 있게 된다. 'Validator'와 'Redeemer'는 단순히 공개키와 해시값을 가지고 있는 것 뿐만 아니라, 임의의 논리를 펼칠 수 있는 **'스크립트'**라는 것에 대한 주소값을 가지고 있다. 바로 이 '스크립트'가 'redeemer' 속의 정보들과 + (후에 설명될 'datum'), 거래의 인풋과 아웃풋에 대한 정보를 가지고 연결된 UTxO가 소모 자격을 가진 입력값인지에 대한 검증을 진행하게 된다. 자세한 내용은 아래에 더 설명되니, 일단 쭉 일어나가기를 권한다.

2.3 스크립트의 맥락

우리 모두 '곰 세마리' 이야기를 잘 알고 있다. 원작인 'Goldilocks and the Three Bears'라는 스토리에서는 골디락스라는 어린 소녀가 곰 가족의 집에 가서 겪는 이야기를 다룬다. 거기서 곰 가족의 식탁 위에 놓인 죽을 먹어보는데, 어떤 죽은 너무 뜨겁고, 어떤 죽은 너무 차갑고, 세번째로 먹은 죽이 '딱 맞았다'라는 내용이 나온다. 이 비유는 현존하는 3개의 주요 블럭체인의 스크립트 맥락에서 또한 발견할 수 있는데, 과연 우리의 최애 블럭체인인 카르다노는 너무 뜨거운지, 너무 차가운지, 아니면 딱 맞는지 확인해 보도록 하자!

- **비트코인(Bitcoin)** - 비트코인 블럭체인도 비트코인 스크립트로 쓰인 나름의 '스마트(하지 않은) 컨트랙'이 존재한다. 이 스크립트들은 오직 거래 입력값에 대한 redeemer의 정보를 가지고 검증 절차를 거친다. 결국 비트코인의 UTxO들에 있는 스크립트들은 현재 거래와 블럭체인 전역 상태를 보았을때, 제일 소규모 맥락인 현재 거래의 현 입력값 정보만을 가진 상태이다. 깔끔하고 직관적이지만, 다양한 로직들을 표현하기에는 부족하다.

- **이더리움(Ethereum)** - 이더리움에서의 스크립트들은 블럭체인의 공유된 전역 스테이트를 맥락으로 받는다. 이는 이더리움의 스크립트들의 표현력에 한계를 두지 않으며, 블럭체인의 모든 상태에 대한 조건 등을 사용해서 다양한 로직을 짜는 것에 있어서 큰 도움이 된다. 이더리움 위에 현존하는 DApp들만 봐도 이미 알 수 있다. 하지만, 이는 부작용들에 의한 비결정성(non-determinism)이 있으며, 동시성(concurrency)이 문제가 될 수 있다(이는 보안 문제까지 연결될 수 있다).

  거래가 만들어지고 블럭체인에 기입이 되는 시간 동안 다양한 것이 동시다발적으로 일어날 수 있다. 이때 거래를 이루는 스크립트의 맥락이 블럭체인의 전역 상태임으로, 블럭체인 업데이트가 일어나기 전 그 사이 일어나는 수많은 것들을 예측하는 것이 불가능하다. 즉 결국 거래가 성공적으로 성사되는지 안되는지는 그 거래가 블럭체인에 기입되어 업데이트 되려는 시점에 결정되므로 거래가 성사하든 안 하든 가스비(이더리움 거래 수수료)가 항상 요구될 수 밖에 없다.

- **카르다노(Cardano)** - 카르다노 스크립트들의 맥락은 비트코인 스크립트와 이더리움의 스크립트 사이 "딱 맞는" 맥락을 가지고 간다. 비트코인 UTxO의 직관성을 그대로 가져가며, 그렇다고 해서 이더리움의 막대한 표현력을 포기하는 것도 아니다. 이는 다음과 같은 메인 아이디어에 의해서 가능케 되었다.

  1. 스크립트의 맥락이 블럭체인 레벨이 아닌 거래(트랜잭션) 레벨이다. 스크립트는 단순히 자신을 소모하려는 입력값의 정보만이 아닌, 트랜잭션 전체의 맥락을 가지게 된다. 이는 모든 입력값과 출력값에 대한 정보이다. 이러한 닫힌 맥락은 스크립트를 분석하기 쉬게 하며, 예측 가능하게 만들어준다.
  2. 그 맥락에는 UTxO에 새롭게 포함된 **'Datum'**이라는 데이터도 추가된다. 이런 추가 데이터는 트랜잭션에게 블럭체인에 대해 역사적인 상태나, 외부 요소들을 포함 시킬 수 있게 하는 주요 작용이되며, 이더리움 만큼 표현력있게 해준다.

  즉 카르다노에서는, 특정 거래가 블럭체인에 기입되기 전에 트랜잭션의 성사에 대해 미리 그 검증 결과를 예측할 수 있는 것이다. 누군가 거래에 사용될 UTxO를 미리 소모했다면, 블럭에 기입되기 전에 트랜잭션 자체가 실패하여, 거래 수수료를 지불하지 않아도 된다. 이로서 우리는 '딱 맞는' 죽을 찾은 것 같다!

좋다! 카르다노라는 '딱 맞는 죽'을 찾았지만, 그 죽은 어떻게 쑤는 것인가? 트랜잭션에 있어서 누가 Redeemer, Datum, 그리고 Validator(Script)를 제공하게 되는 것인가? 답은 바로 'UTxO를 소모하려는 트랜잭션'이다.

글을 조금 더 자세히 들여다본 독자들은 2.2번 섹션에서는 Script를 제공하는 것은 소모될 UTxO고, 그 UTxO를 소모하려는 트랜잭션의 입력값에 Redeemer가 제공된다고 써 있는 것을 기억할 것이다. 아니, 방금 위에서는 모든 것이 UTxO를 소모하려는 트랜잭션에서 온다고 하지 않았나? 말이 안 맞지 않는가!

개념적으로, 검증 스크립트와 Datum은 UTxO의 일부로서 생각하는게 맞다. 하지만 실질적인 스크립트 그 자체와, Datum은 그 UTxO를 소모하려는 트랜잭션에서 주어진다. UTxO에는 그 스크립트와 그 Datum이 특정 지어질 수 있도록 그 스크립트와 Datum에 대한 해시값들을 가지고 있게 된다. 개념적인 내용과 실질적인 내용이 달라지는 이러한 구조는, EUTxO들의 메모리 사용량을 줄이는 효과를 불러 일으킨다(EUTxO들은 해시값만 가지고 실질적인 것들은 트랜잭션에 담겨 있기 때문에). 이는 다양한 트랜잭션에서 UTxO들을 빠르게 불러 검증 절차를 거치는 접근성에 큰 이점을 가져다 준다.

하지만, 이 말인 즉슨, 특정 UTxO를 소모하는 트랜잭션을 설계하려면 그 UTxO와 결부된 Datum과 Script를 미리 알고 있어야되는것 아닌가? 맞는 말이다, 하지만 이는 항상 가능한 것이 아닐 때가 있고, 어플리케이션 개발에 문제가 될 수도 있다. 그에 선택적으로 해시값이 아닌 전체 Datum을 소모될 EUTxO에 포함 시킬 수도 있다. 또한 아직 1주차에서 설명되지 않은 다양한 오프-체인 방식들을 통해서 위의 문제를 해결할 수 있는데, 앞으로의 강의에서 차차 알아 갈 수 있기를 바란다.

위의 내용들을 정리하는데에 특정 프로그래밍 언어가 꼭 쓰여야 되는 것은 당연히 아니다, 하지만 이 내용이 플루투스 파이어니어 프로그램의 일부인것 처럼, 이 프로그램에서는 플루투스와 그의 모언어 하스켈을 사용한 구현에 집중할 것이다.

EUTxO에 대한 전문적인 견해를 얻기 위해서는 IOHK에서 처음 EUTXO 모델을 제시한 논문을 추천한다. 15페이지 밖에 되지 않고, 사용되는 단어나 표현이 너무 어렵거나 무섭지 않으니 꼭 한번 읽어보길 권한다.

2.4 잉글리시 옥션 예제

먼저 잉글리시 옥션이란, 우리가 흔히 영화에서 많이 보는 형식의 경매이다. 많은 값어치가 나가는 물건이 부를 수 있는 최소 값을 가지고 경매에 오르게 된다. 사람들은 공개적으로 자신의 값을 부르고, 경매 중계인은 계속해서 증가하는 순서대로 그 부른 값을 인정해준다. 현재까지의 최고 값을 부른 사람은 일시적으로 그 물건에 대한 소유권을 가지게 된다. 이 소유권은 더 높은 값을 부르는 사람이 나타나면 바뀌게 되고, 이러한 값 부르기는 주어진 시간 동안 계속 일어난다. 주어진 시간이 다 끝났을 때 최고 값을 부른 사람이 그 물건의 소유권을 최종적으로 얻게 된다. 이러한 시스템을 지금까지 위에서 말한 카르다노 블럭체인과 EUTxO 모델로 어떻게 구현할 수 있을지, 어떻게 동작할지 한번 알아보도록 하자.

일단 이 잉글리시 옥션을 정의하는 파라미터들을 정리해 볼 수 있을 것이다:

- (경매에 오르는) 물건 주인
- (경매에 오르는) 물건 (이 예제에서는 NFT)

- 최소로 부를 수 있는 값
- 현재까지 불려진 최고 값
- 경매 종료 시점

이 파라미터들에 대해서 작용하는 3가지 '거래'들을 정의 할 수 있을 것이다.

- **경매 시작 - Start an Auction** - 위의 파라미터들을 정의하며 경매가 생성된다. 경매가 생성될때는 위의 파라미터들이 '올바른' 값들을 가져야 한다. (경매할 물건이 실존해야 하고, 최소로 부를 수 있는 값은 양수여야 하며, 경매 종료 시점은 지금으로 부터 미래 시점이여야 한다, 등)
- **경매에 값 부르기 - Bid on an Auction** - 시작된 경매에 대해서 사용자는 그 경매에서 다루는 물건에 대한 값을 부를 수 있다. 이 거래는 최소 부를 수 있는 값, 현재까지의 최고 값(부르는 값이 최소값과 현재까지 최고 값 보다 커야 한다), 종료 시점(종료 시점 이전에 불러야 한다)등에 의해 그 거래의 '올바름'이 결정될 것이다.
- **경매 종료 - Close an Auction** - 경매 종료 시점이 지난 후에는 경매를 종료할 수 있을 것이다. 이 종료 거래에서는 경매 물건이 최고 값을 부른 사람에게 가고, 그 값은 물건 주인에게 가는 결산이 일어나는 거래이다.

어플리케이션을 개발할때는 위와 같이 스키마(Schema)를 정리하고 어떤 데이터 타입들이 필요하며, 어떤 거래들이 정의되어야 하는지, 또 어떤 조건들이 붙는지 등을 상시로 업데이트하며 전체적인 그림을 그리는 것이 중요하다.

어쨌거나, 다시 예제로 돌아가 보자.

1. 철수가 자신의 NFT(대체불가토큰 - 카르다노 네트워크에 단 한번 존재하는 네이티브 토큰)를 경매에 올려 팔려고 한다.
2. 그럼 먼저 철수는 NFT가 담긴 UTxO(보통은 토큰 뿐인 UTxO는 가질 수 없다, ADA가 같이 존재해야 한다)를 잉글리시 옥션 스크립트를 담고 있는 주소에 만들어야 한다. 이 옥션 스크립트는 최소로 부를 수 있는 값(100 ADA 라고 하자)과 옥션 종료 시점을 가지고 있다. 또한 현재까지의 최고로 불려진 값은 UTxO의 Datum으로 가져간다고 하자 (지금은 비어있다).
3. 영희는 경매에 올라온 NFT를 가지기 위해 최소로 부를 수 있는 값인 100 ADA를 부른다. 이는 2개의 입력값과 1개의 출력값이 있는 트랜잭션을 만들어서 이루어진다.

   여기서 입력값들은 **옥션 UTxO**와 **영희의 ADA를 담고 있는 UTxO**이다. 이에 트랜잭션은 출력값으로 **수정된 새로운 옥션 스크립트**를 발생시킬 것이고, 이 새로운 스크립트는 값으로 본래 가지고 있던 NFT와 현재 불려진 값이 100 ADA를 가지고 있다. Datum으로는 영희 이름과 불려진 100 ADA라는 정보를 가지고 있다.

   이 거래를 검증하기 위해서는, 거래의 Redeemer가 옥션 UTxO를 입력값으로 받으며 확인된다. Redeemer속에는 영희의 값 부르기에 대한 정보가 담겨있어서 옥션 스크립트 속에 담겨 있는 검증 함수를 거칠 것이다. 예를 들면, 최소로 부를 수 있는 값보다 큰 값을 불렀

는지, Datum이 거래에 의해 제대로 업데이트 되는지, 거래에 올바른 입력값과 출력값이 나오는지 등을 확인하는 함수가 될 것이다. 함수를 성공적으로 통과하면, 그 옥션 UTxO를 소모할 수 있게 된다.

4. 이 상황에서 민수 또한 그 NFT를 가지고 싶어서 더 큰 값인 200 ADA를 부른다고 하자. 똑같이 이 값 부르기 트랜잭션은 옥션 UTxO와 찰리의 ADA가 담긴 UTxO가 소모되며, 업데이트된 새로운 옥션 UTxO가 나오게 될 것이다.

   민수의 값이 영희가 부른 값 보다 높았으니, 새로운 옥션 UTxO는 값에 NFT와 민수의 값을 가지게 되고 Datum 또한 민수의 정보로 업데이트가 된다. 본래의 옥션 UTxO에 들어있던 영희의 값은 추가적인 출력값으로 다시 영희에게 돌아가게 된다.

5. 경매 종료 시점이 지난 후에는 팔린 값을 받고 싶어할 철수나, NFT를 가지고 싶을 민수가 경매 종료 트랜잭션을 구동시킬 것이다.

   여기서는 단지 옥션 UTxO만 트랜잭션에 들어가게 되며, 철수에게 200 ADA값이 들어간 UTxO와 민수에게 NFT가 담긴 UTxO 이렇게 2개의 출력값을 만들게 된다. 언제나 그렇듯이 거래에서도 Redeemer는 옥션 스크립트와 함께 경매 종료 시점이 지났는지, 입/출력값이 올바른지에 대한 검증이 일어날 것이다.

   추가적인 이야기로, 이러한 경매 종료 트랜잭션은 필수적이다. 자동으로 시간이 지나면 알아서 ADA와 NFT를 넘기게 하면 되지 않나라고 생각할 수 있지만, UTxO는 단순히 수동적인 데이터일 뿐이다. 이러한 **데이터에 대해서 블럭체인 상태를 바꾸려면 외부에서 트랜잭션이 구동되어야 한다**. 만약 이런 상황을 자동화하려면, 종료 트랜잭션의 생성을 자동화하는 것을 월렛 단에서 짜야하지, 트랜잭션/검증 스크립트 자체에서 그런 기능을 구현하는 것은 맞지 않다.

6. 만약 철수의 옥션에 아무런 값도 불려지지 않았다면, 경매 종료 시점 이후, 철수가 경매 종료 트랜잭션을 날리면 만들어진 옥션 UTxO를 소모하고 본인에게 다시 NFT가 돌아가게 된다.

2.5 그럼 이걸 다 어디서 짜는 것인가?

와우! 장부를 사용한 어플리케이션 개발에서 부터 EUTxO, 그리고 결국은 잉글리시 옥션까지! 험하면 험했던 많은 길을 지나 여기까지 오게 되었다. 지금까지의 모든 것은 2주차 부터 점차 코드를 직접 보게 되면서 다루게 되는 것에 대해 좋은 기반이 될 것이라고 믿는다! 그전에 플루투스에 코드가 존재하는 2가지 위치에 대해 간략한 정리를 하려고 한다.

온-체인 코드 - On-chain code - 이는 EUTxO 모델에서 표현된 스크립트들을 의미한다. 위에서 'Validator'라고도 표현했던 그 UTxO와 연결된 스크립트들이다. UTxO에 의해서는 그 스크립트의 해시가 주어지고, 트랜잭션 입력값에서 실질적인 플루투스로 컴파일된 코드가 주어진다. 카르다노 노드가 트랜잭션을 받으면, 우선 이 온체인 코드를 가지고 트랜잭션을 검증한 후에 노드의 메모리 풀에 올린다(블럭에 쓰이기 전에). 각 트랜잭션의 입력값의 스크립트 위치마다 해당 스크립트가 실행되며, 모든 것이 성공해야한다. 만약 성공하지 못하는 스크립트가 있다면, 그 거래는 실패한 거래로 간주된다. 실제 코딩은 하스켈로 이루어지며, '플루투스 코어'라는 언어로 컴파일된다.

오프-체인 코드 - Off-chain code - 이 코드는 블럭체인이 아닌 월렛(wallet)에서 도는 외부 코드이다. 오프체인 코드가 바로 특정 UTxO들을 소모하기 위해 만들어지는 트랜잭션들을 짜는 곳이다. 다양한 스마트 컨트랙 어플리케이션을 짜기 위한 메인 공간이 되는 것이다. 트랜잭션에 들어가는 입력값과 출력값을 잘 정의해서 올바른 스마트 컨트랙을 짜야 한다.

보다시피, **온체인과 오프체인 코드는 공생관계를 이룬다.** 오프체인이 능동적으로 트랜잭션들을 설계하고 만들어내면, 온체인 코드는 UTxO들이 그 트랜잭션 맥락을 받고 그 트랜잭션들에 의해 소모되는 부분들을 검증해준다.

이 공생을 더욱 강화시키는 것은 바로 온/오프체인 코드가 모두 하스켈로 쓰인다는 점이다. 이렇게 온/오프체인에서의 로직을 한 언어로 쓸 수 있으면 여러 장점이 있으며, 특히나 추후에 배우게 될 'state machine' 개념에 도움을 많이 준다.

3 마무리

이로서 1주차 플루투스 파이어니어 프로그램 노트를 마무리 지을 수 있다. 플루투스 플레이그라운드 세팅(nix를 사용하면 편하다)이나 잉글리시 옥션 시뮬레이션, 온/오프 체인 코드 분석에 대한 내용은 추가하지 않았지만, 플루투스 파이어니어가 되기 위한 개념적인 도입으로 유용했으면 하는 바램이다. 이 노트들을 통해 토론과 이야기들이 오고가며 서로가 서로의 지식과 관점들을 조화롭게 어루만질 수 있는 공간을 만들고 싶다. 마치 우리의 최애 장부가 노력하듯, 파이어니어들, 더 나아가 카르다노 개발진 사이에서 아름다운 합의들을 많이 만들어 나가 선순환되는 개발문화를 만들어 나가고 싶은 마음이다. 언제나 우리가 세상을 바꾸어 나가고 있다는 점을 잊지 않았으면 한다! Share the love! ♡

ESSE QUAM VIDERI
J Kang.