An attempt to give the most abstract possible definition of a blockchain.

1 BlockChain

1.1 Lists and their validation

Given a set S, let $\textsc{Set}(S)$ be the set of all sets of elements of S, and $\textsc{FList}(S)$ be the set of all finite lists of elements of S. Given $L \in \textsc{FList}(S)$, we use notation $\text{length}(L)$ to refer to the number of elements in L, notation $L[i]$ to refer to the i-th element in L, where $i \in \{1, \ldots, \text{length}(L)\}$, and notation $L[i,j]$ to refer to the sublist $[L[i], \ldots, L[j]]$ of L, where $i, j \in \{1, \ldots, \text{length}(L)\}$ and $i \leq j$. Notice that $\text{length}(L) = 0$ if and only if L is the empty list $[\,]$.

From now on, assume that Σ is a finite alphabet, and that $\mathbf{B} \subseteq \Sigma^*$ is the set of all possible blocks.

Definition 1. *A validation rule is a function* $V : \textsc{FList}(\mathbf{B}) \to \textsc{Set}(\mathbf{B})$

Intuitively, V is a function taking a finite list L of blocks as input, and returning the set of blocks that could be added to L to produce a valid blockchain.

A list $G \in \textsc{FList}(\mathbf{B})$ is said to be a genesis list of V if $\text{length}(G) \geq 1$, $G[1] \in V([\,])$ and $G[i+1] \in V(G[1,i])$, for every $i \in \{1, \ldots, \text{length}(G) - 1\}$. That is, G is a genesis list if G is a non-empty valid blockchain.

Definition 2. *Let V be a validation rule and G be a genesis list. Then a list $L \in \textsc{FList}(\mathbf{B})$ is valid with respect to (G, V) if:*

1. $\text{length}(G) \leq \text{length}(L)$ *and* $G = L[1, \text{length}(G)]$.
2. $L[i+1] \in V(L[1,i])$, *for every* $i \in \{\text{length}(G), \ldots, \text{length}(L) - 1\}$.

The role of G in this definition is to provide the blocks to startup the system. Let $\textsc{Log}(G,V)$ be the set of valid lists with respect to (G,V).

Two lists $L_1, L_2 \in \textsc{FList}(\mathbf{B})$ are said to disagree in the last element if one of the following conditions holds: (1) $\text{length}(L_1) = 0$ and $\text{length}(L_2) > 0$, (2) $\text{length}(L_1) > 0$ and $\text{length}(L_2) = 0$, or (3) $\text{length}(L_1) > 0$, $\text{length}(L_2) > 0$ and $L_1[\text{length}(L_1)] \neq L_2[\text{length}(L_2)]$.

Definition 3. *Let V be a validation rule and G be a genesis list of V. Then $\textsc{Log}(G,V)$ is safe if for every pair $L_1, L_2 \in \textsc{FList}(\mathbf{B})$ that disagree in the last element, it holds that $V(L_1) \cap V(L_2) = \emptyset$.*

1.2 Knowledge

Given a a validation rule V, a genesis list G of V and a subset K of $\textsc{Log}(G,V)$, there is a natural way to visualize K as a graph $\mathcal{G}(K)$. The set of nodes of $\mathcal{G}(K)$ is the set of blocks occurring in the lists in K, and there is an edge from a block b_1 to a block b_2 if there exists a list $L \in K$ such that $b_1 = L[i]$ and $b_2 = L[i+1]$, where $i \in \{1, \ldots, \text{length}(L) - 1\}$.

Lemma 1. *Assume that $\textsc{Log}(G,V)$ is safe. The for every subset K of $\textsc{Log}(G,V)$, it holds that $\mathcal{G}(K)$ is a tree rooted at $G[1]$.*

Juan: Related to the next comment, why do we need $\mathcal{K}(G,V)$? this is just all non-empty finite subset K of $\textsc{Log}(G,V)$

Thus, assuming that $\textsc{Log}(G,V)$ is safe, from now on we refer to every non-empty finite subset K of $\textsc{Log}(G,V)$ as a *knowledge tree* of (G,V). Moreover, we define $\mathcal{K}(G,V)$ as the set of all knowledge trees of (G,V).

1.3 Block chain and protocols

Definition 4. *A relation $\preceq$ on $\textsc{Log}(G,V)$ is said to be a knowledge order over (G,V) if $\preceq$ is a total preorder on $\textsc{Log}(G,V)$, that is, $\preceq$ is reflexive, transitive and total.*

Moreover, P is said to be a blockchain protocol over (G,V) if P is a sequence $\{\preceq_i\}_{i\in\mathbb{N}}$ such that each $\preceq_i$ $(i \in \mathbb{N})$ is a knowledge order over (G,V).

Marcelo: Do we really need to use the notion of knowledge tree in the following definitions? If we say that K is a knowledge tree then we need to use PATHS(K) to refer to the paths in K. On the other hand, if we directly say that $K \in \textsc{Log}(G,V)$, we can just refer to the elements of K (we do not a special notation for paths). Knowledge trees are a nice way to visualize the blocks of set $K \in \textsc{Log}(G,V)$, but I think we should just mention them because of this (and we should not use them in the definitions). What do you think?

Juan: You mean $K \subset \textsc{Log}(G,V)$ right? I agree that here we do not need safeness nor finitiness to make this work, so we could just stick with $\textsc{Log}(G,V)$ instead of all knowledge trees

Definition 5. *Let $P = \{\preceq_i\}_{i\in\mathbb{N}}$ be a blockchain protocol over (G,V), $K \subseteq \textsc{Log}(G,V)$ and $t \in \mathbb{N}$. Then a maximal element of K with respect to $\preceq_t$ is said to be a blockchain of K at time t with respect to the protocol P.*

Marcelo: I stopped here. I am not totally convinced that we need to introduce the following notion of equivalence, this is something that we need to discuss.

Definition 6. *Let $K = (N,E)$ and $K' = (N',E')$ two knowledge trees we say that K and K' are $\equiv$ equivalent if :*

$$\equiv \text{ is an equivalent function}$$
$$\forall L \in PATHS(K), \exists L' \in PATHS(K'), \forall i \in [\![1,|L|]\!], L[i] \equiv L'[i]$$
$$\forall L' \in PATHS(K'), \exists L \in PATHS(K), \forall i \in [\![1,|L'|]\!], L[i] \equiv L'[i]$$

We denote $K^{\equiv}$ the set of knowledge equivalent to K.

Action, states, reward and game

Definition 7. *Considering a set of player P we denote $\mathcal{K}_P$ the set of function $K_P : P \to \mathcal{K}$ mapping a knowledge tree to each player. We denote $\mathcal{K}_P^{\equiv}$ the set of mapping where:*

$$\forall K_P, K'_P \in \mathcal{K}_P^{\equiv}, \forall p \in P, K_P(p) \text{ and } K'_P(p) \text{ are } \equiv \text{ equivalent}$$

Intuitively $\mathcal{K}_P$ represents the true knowledge of each player.

Definition 8. *We call action a for player $w \in P$ function $a_w^{\equiv} : \mathcal{K}_P \to \mathcal{K}_P^{\equiv}$ such that:*

$$\forall K_P \in \mathcal{K}_P, \forall K'_P \in a_w(K_P), \forall u \in P, K_P(u) \subseteq K'_P(u)$$
$$\forall K_P \in \mathcal{K}_P, \forall K'_P \in a_w(K_P), \forall u \in P, K'_P(u) \subseteq K'_P(w) \cup K_P(u)$$

Let $A_w^{\equiv}$ be the set of action for player w.

An action of player w is represented by a modification of w knowledge and a round of communication. We are dealing with equivalence knowledge in order to reduce the number of action possible.

Definition 9. *Let P be a set of player, $\mathcal{A}^{\equiv} = A_{p_1}^{\equiv} \times A_{p_1}^{\equiv} \ldots \times A_{p_{|P|}}^{\equiv}$*

Definition 10. *We call reward for player w a function $r_w : \mathcal{K}_P \times \mathcal{A}^{\equiv} \to \mathbb{R}^+$ such that:*

$$\forall A \in \mathcal{A}^{\equiv}, K_P \text{ equivalent } K'_P \Rightarrow r_w(K_P, A) = r_w(K'_P, A)$$

Intuitively $\mathcal{K}_P$ represent the knowledge of each player assumed by w

Definition 11. *Let P be a set of player, $\mathcal{R} = r_{p_1} \times r_{p_1} \ldots \times r_{p_{|P|}}$*

2 Etienne 's modification space

Definition 12. *Considering a set of player P, a set of function $\mathcal{K}_P$, the set of action $\mathcal{A}$, the set of reward $\mathcal{R}$, and a function $\mathcal{P} : \mathcal{K}_P \times \mathcal{A} \times \mathcal{K}_P^{\equiv} \to [0;1]$ a transition probability ($\mathcal{P}(K_P, A, K_P^{1\equiv})$ is the probability of transitioning from K_P to an element of $K_P^{1\equiv}$ after joint action A). We define a infinite stochastic game Γ such that:*

- *P is the set of player.*
- *$\mathcal{A}$ is the set of available action.*
- *$\mathcal{K}_P$ is the set of states.*
- *$\mathcal{R}$ the set of pay-off function.*
- *$\mathcal{P}$ is the transition probability function.*

Remark. *I don't like what i am doing with equivalence as it complicate stuff a lot. Moreover if i have to prove that my game is well defined as it is not immediate due to that. Finally we have an infinite number of states, equivalence states and finitely repeated game might be a solution.*

Stationary Nash equilibrium, Reasonable knowledge

We know that stochastic games with finite number of states have a discounted nash equilibrium and it is computable : (Nonlinear program with linear constraints). 2 solutions: either we are able to define equivalence classe over states to create a finite number state stochastic game or our game fit in the "On a new class of nonzero-sum discounted stochastic games having stationary Nash equilibrium points", have to understand this paper ...

Definition 13. *We say that K'_P is reasonable regarding a game Γ if exists a strategy*