There are some suggestions with respect to the white paper.

The words “assets”, “data”, and “data assets” are used in the different example use cases (chapter 2) with slightly different meanings. Somtimes assets refer to fungible assets[1] and sometimes to non-fungible assets and sometimes an asset can be duplicated on the different ledger (such as data packages) and sometimes talking about an asset implies that it can only be active/alive in one ledger at any point in time.

Even so these words are used in everyday language interchangeably, I suggest to define them precisely in the white paper and use a consistent terminology throughout all the example use cases. Some of the example use cases might apply to several object types, such as the “2.1 Ethereum to Quorum Asset Transfer” might apply to fungible and non-fungible assets (FA and NFA), whereas the “2.5 Healthcare Data Sharing with Access Control Lists” probably only applies to data (D) objects.

So far I would differentiate between three types of objects that are stored on ledgers and that cactus interacts within the example use cases:
1. FA = fungible asset (alternative naming: value token/coin) – cannot be duplicated on different ledgers
2. NFA = non-fungible asset (alternative naming: asset) – cannot be duplicated on different ledgers
3. D = data – can be duplicated on different ledgers

Difference between an asset (FA or NFA) and data (D):
The same data package can have several representations on different ledgers in active mode while an asset (FA or NFA) can only have one representation active at any time, i.e., an asset exists only on one blockchain while it is locked/burned on all other blockchains. A data package that was once created as a representation of another data package might divert from its original one over time because different ledgers might invoke different transactions on the data packages over time.

There is this table on the website “https://github.com/hyperledger/cactus” explaining the 5 use cases that cactus will be able to handle in the future with respect to value V (means numerical assets (e.g. money)) and data D (means non-numerical assets (e.g. ownership proof)) transfers. The 5 use cases mentioned are as follows:
1. value transfer (V -> V)
2. value-data transfer (V -> D)
3. data-value transfer (D -> V)
4. data transfer (D->D)
5. data merge (D<->D)

1
Difference between fungible and non-fungible asset: A fungible asset is an asset that can be swapped with another one, like a currency. For example, if I have a 1 USD bill, it can be swapped for any other 1 USD bill. It does not matter which 1 USD bill I own. The same is true for any Bitcoin, Ether, ... A non-fungible asset is an asset that cannot be swapped for another. For example, a house is a non-fungible asset. Each house has some unique properties that makes it different. The same applies to e.g. cryptokitties or a product that is tracked on the blockchain in a supply chain. There are two different standards for fungible and non-fungible assets on the Ethereum network which might be useful for further reading (ERC-20 Fungible Token Standard and ERC-721 Non-Fungible Token Standard).

I suggest extending these 5 use cases to a table which allows a new person to easily identify which example use case is best suitable for them.

As a person who is considering using cactus, I might already have a concrete business use case and idea in mind how my two blockchains look like, which objects they are dealing with (NFA, FA or D) and if I want to implement ledger transfer/ ledger coordination or ledger entry point coordination. So far a new user has to go through all the example use cases to find one related to their problem.

I came up with this preliminary table which still needs a lot of improvement and also some help in identifying in which row/column the example use cases from chapter 2 belong to. This table might also help to find new example use cases that have not been included in chapter 2 yet.

		Type I	**Type II**	**Type III**	**Type IV**	**Type V**	**Type VI**
State transition type in blockchain A / B: Invoke (write transaction), Query (read-only transaction)		Invoke/Invoke or Query/Invoke (Query in case data is not locked/deleted in blockchain A)	Invoke/ Invoke or Query/Invoke (Query in case data is not locked/ deleted in blockchain A)	Invoke/ Invoke	Invoke/ Query or Query/Invoke	Query/ Query	Invoke/ Invoke or Invoke/ Query or Query/ Invoke or Query/ Query
Interaction type		Ledger transfer between blockchains [1]		Ledger coordination between blockchains [2]			Ledger entry point coordination [3]
Direction type		one-directional	bi-directional	NA	NA	NA	
Asset/ Data type of Blockchain A	**Asset/Data type of Blockchain B**						
FA	FA	2.1 Ethereum to Quorum Asset Transfer (if asset is FA) 2.4 Stable Coin Pegged to Other Currency					2.7 End User Wallet Authentication/Authorization
NFA	FA			Figure 1 (see below)			

FA	NFA			Figure 1 (see below)			
NFA	NFA	2.1 Ethereum to Quorum Asset Transfer (if asset is NFA)					
FA	D				2.2 Escrowed Sale of Data for Coins (if data is read on one ledger and ownership change of the coins is recorded on the other ledger)		
NFA	D						
D	FA				2.2 Escrowed Sale of Data for Coins (if data is read on one ledger and ownership change of the coins is recorded on the other ledger)		
D	NFA						
D	D	Data transfer (D ->D) 2.8 Blockchain	Data merge (D<->D)		2.5 Healthcare Data Sharing	2.5 Healthcare Data Sharing	

		Migration	2.6 Integrate Existing Food Traceability Solutions		with Access Control Lists (if data is read on one ledger and written to the other ledger)	with Access Control Lists (if data is read from two independent ledgers),	

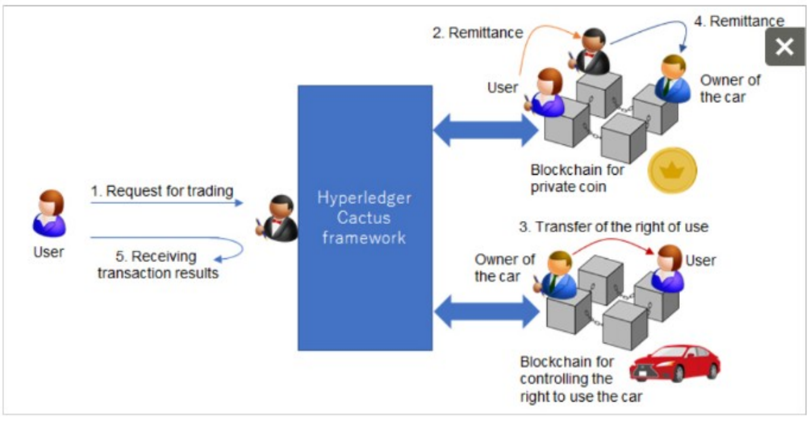
2. Remittance
4. Remittance
User
Owner of the car
Blockchain for private coin
User
1. Request for trading
5. Receiving transaction results
Hyperledger Cactus framework
3. Transfer of the right of use
Owner of the car
User
Blockchain for controlling the right to use the car

Figure 1 (from fujitsu)

There seems to be three different types of modus operandi that cactus is able to support (ledger transfer/ ledger coordination and ledger entry point coordination). Each of these operandi has a different degree of interference with the ledger of the connected blockchains. The ledger transfer has a high degree of interference since the livelihood of a blockchain can be reduced in case too many assets/data are locked/burned in a connected blockchain. The ledger coordination has less degree of interference since all assets/data stay in their respective blockchain environment. The ledger entry point coordination has no degree of interence with the ledger itself.

[1] Ledger transfer between blockchains is defined as follows: An asset gets locked/burned on one blockchain and then a representation of the same asset gets released in the other blockchain. There are never two representations of the same asset alive at any time. Data is an exception since the same data can be transfered to several blockchains.

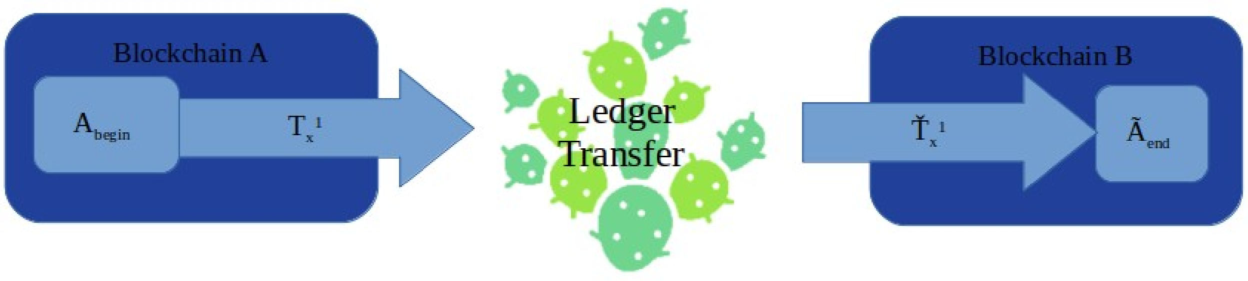

Figure 2: Ledger transfer between blockchains [1] (one-directional)

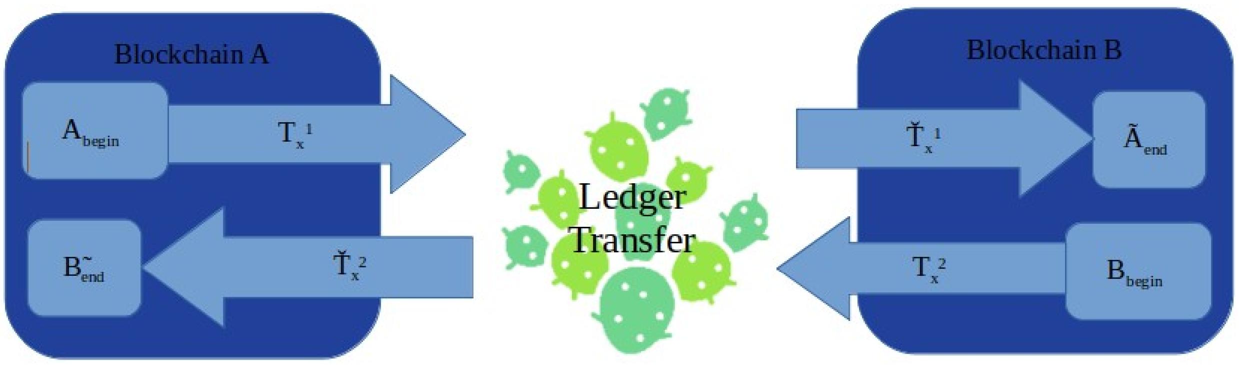

Figure 3: Ledger transfer between blockchains [1] (bi-directional)

[2] Ledger coordination between blockchains is defined as follows: A transaction (read or write) is performed in Blockchain A and another transaction (read or write) is performed in blockchain B. There is no asset/data/coin leaving any blockchain environment. The two blockchain environments are isolated but because cactus coordinates both transactions they are done atomically. That means either both transactions are valid/ committed successfully or none of the transactions are valid/committed successfully.

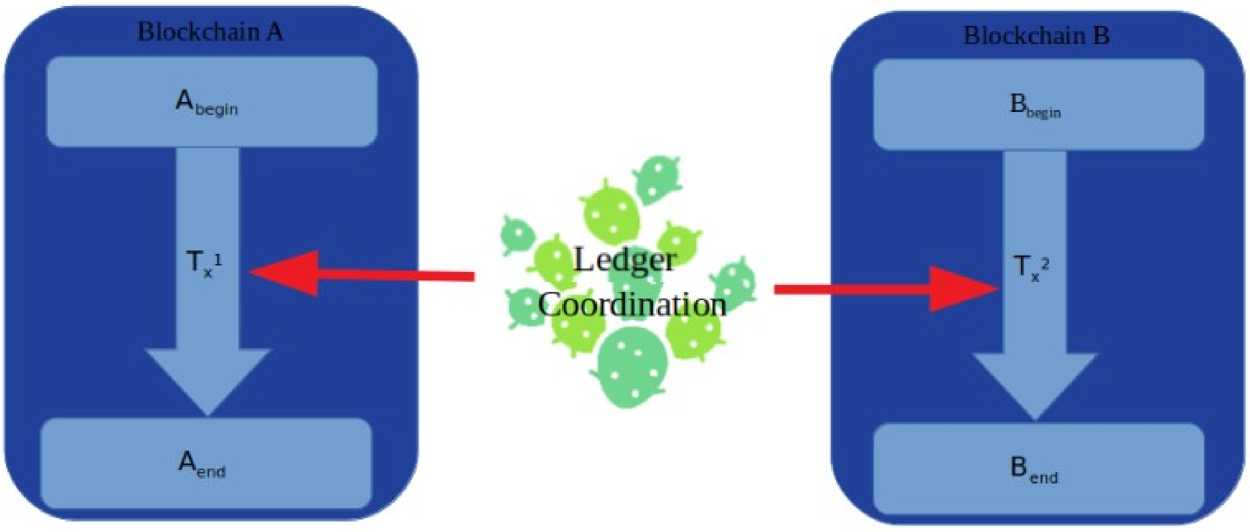

Figure 4: Atomic coordination between blockchains [2]

[3] Ledger entry point coordination between blockchains is defined as follows: Cactus organises end-user wallet authentication/ authorization but does not interfere with wich transaction is committed on any of the connected blockchains. Cactus is just forwarding any of the transactions given by an end-user to the corresponding blockchain.

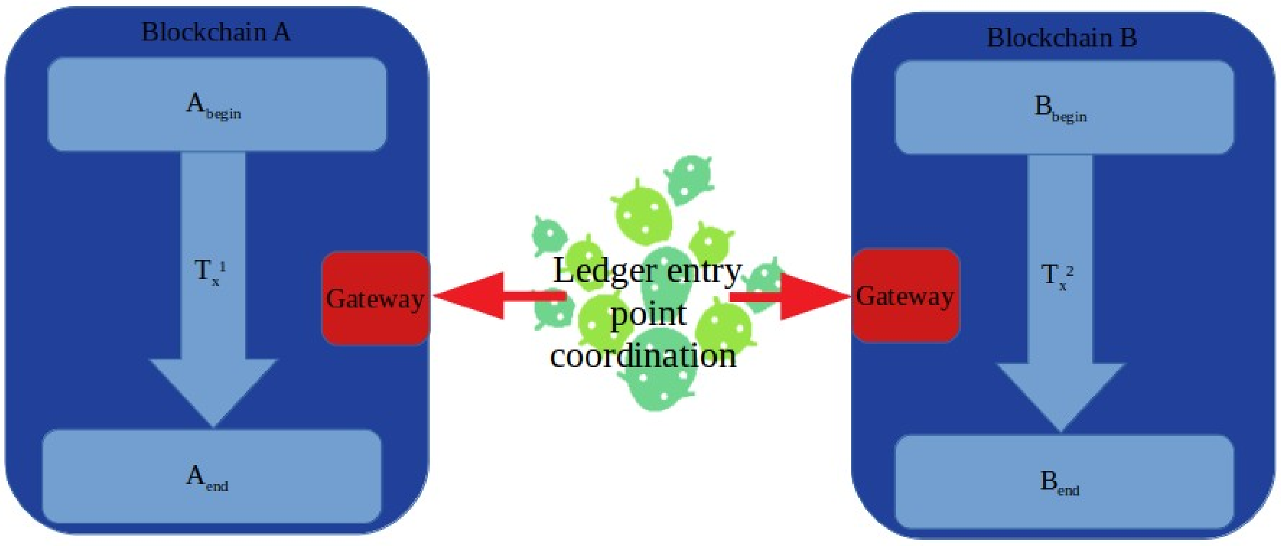

Figure 5: Ledger entry point coordination between blockchains [3]

Legend:

A_{begin} = State of asset A at the beginning of transaction
A_{end} = State of asset A at the end of transaction
$\tilde{A}_{end}$ = Representation of asset A at the end of all transactions

B_{begin} = State of asset B at the beginning of transaction

B_{end} = State of asset B at the end of transaction
$\tilde{B}_{end}$ = Representation of asset B at the end of all transactions

T_x^1 = Invoke transaction 1
T_x^2 = Invoke transaction 2