Transaction functions (when run post-ordering) need a way to query data, such as:

- the current resource of kind k
- the value at content-addressed storage location h

(h

is a hash)

- possibly historical data or other data

I suggest that we add a very simple interface, where transaction functions are passed a pointer to a query function:

type QueryFunction := (Bytes -> Bool) -> Set Bytes

The query function takes a predicate over data blobs and returns the set of blobs which satisfy that predicate (a similar idea has been previously discussed here, and here). This signature is extremely general, but in practice we can start by supporting only a small, finite set of query functions, e.g.:

- \x -> hash x == y

(query content-addressed storage by hash)

- \x -> inCurrentCommitments (hash x) && kind x == y

(pseudocode - query the current resource with kind y

)

The surrounding “resource machine shell” will simply analyze the provided function, see if it matches one of these specific ones, and fail if not. Then we can easily add more sophisticated query options in the future without changing the interface.

Thoughts? @vveiln @Michael @degregat @ray