Recently, while thinking about how to represent different virtual machines, I again came across the excellent work from IPFS/Filecoin on multiformats, which I think might help us with disambiguation and forwards-compatibility. The basic idea of multiformats - as I understand it - is to make values such as hashes and encodings self-describing by adding a simple numerical prefix which maps in some canonical table to a particular hash algorithm, encoding format, etc. I think we could consider using multiformats for hashes and encoding algorithms, as they already do, and I think we could also

consider using it for canonical representations of functions, which would look roughly like (for example):

Code

Virtual machine

0x01

Nockma

0x02

Cairo

0x03

RISC5

Function representations would then be encoded as the concatenation of the virtual machine code, and the actual term (encoded in a way specific to each virtual machine). In order to run a function, then, a node would need to look up the virtual machine in question, and use it to parse and evaluate the term. A node which does not know the virtual machine associated with a particular code can simply fail (and at least the node has meta-knowledge that it doesn't know this particular format). In the future we can add new virtual machines simply with new codes and nodes can gradually upgrade - although consensus-critical code updates will need to wait until all relevant nodes have attested that they can support the new format. Using multiple formats does mean that we lose a clear equality judgement, but we don't really have that anyways… We still have the guarantee that syntactically equal representations are semantically equal, and we can pass around at runtime other (verifiable) judgements of semantic equality.

An interesting alternative to simply using a table would be to write a canonical interpreter for each VM in one particular representation such as Nockma (or even perhaps Geb), and use the hash of this canonical interpreter as the code. This has the nice benefit of being content-addressed, so we could automatically verify equivalences in certain cases. It would be a lot of initial work though - and I think we can do this in the future anyways, simply by adding a code for a further-wrapped content-addressed interpreter mode.

The really neat thing about using multiformats for representation of virtual machines is that we should be able to bootstrap everything else

off one of one single table of VMs in a content-addressed way - because we can define all other functions (encoding, hashes, etc.) in terms of one of these VMs.

cc @degregat @mariari @Moonchild @tg-x @vveiln @terence for further input here