Although it is not required for architectural correctness, or, in principle, protocol adherence, Anoma needs to select a model of computation to standardize on for pragmatic purposes.

Basic assumptions

- We assume that differences in “physical” computational primitives and costs thereof can be modeled as a set of different finite fields, each with bounded constant costs for operations in the field.
- We assume a boundable, known, monotonically increasing cost of nonlocality of reference (ref).

Requirements

This model of computation will be used for, at minimum, transaction execution:

- Post-ordering computation performed by validators to compute final transactions
- Local computation performed by the interaction engine to compose partial transactions

Both of these computations are of type PTX \to PTX

(see here).

These uses impose some requirements; in particular:

- Individual, metered operations (~ instructions, rewrite rules, etc.) must be bounded-constant-cost in space and time (otherwise the cost accounting itself becomes too complex and computationally expensive).
- The model must be sufficiently expressive to handle runtime reflection and evaluation.

Additional possible uses

- Could be used for known solver algorithms for which bleeding-edge efficiency (which will never be possible with metered computation) is not critical.
- With the appropriate VM, could be used as an actual computational substrate for Typhon in the future, if conducive.

Bonus properties

Execution segment transformation

Ideally, the model would have an efficient bidirectional transformation to and fromexecution segments for verification.

Concurrency

Ideally, the model would be able to represent concurrent computations, parts of which could be evaluated in parallel. This should be reflect in the cost model (perhaps by partitioning a program into parallel segments in a DAG, which seems like a similar problem to execution segmentation).

Candidates

(WIP)

- “NockFF” (Nock with finite fields)
- “NockDAG” (Nock with DAGs instead of binary trees)
- “PolyObject” (Prototype-based object systems with internal polynomial logic, probably has a stack)