Draft Status: Very

We (@cwgoes and I) thought a bit more about how to (de)compose Intent Machines and which roles exist within them. Here's a draft on that.

Composition

An Intent Machine

(IM) is composed of a Selector

, a Solver

and a Picker

. IMs can be composed of IMs, in which case the individual roles need to be composed.

Parties implementing these roles can announce that they are forming an Intent Machine

explicitly, or users can request collaboration of different parties to spin up and ad-hoc IM.

IM can state policies they implement, and users can request them but IMs can always choose to accept or not.

Each Role is governed by an Identity

, which can be trivial or composed.

Roles

Selector

The Selector

decides which Intents

go into one solving batch. These Intents

construct and constrain (maybe span

is a better term?) the solving space. Users submit their Intents

to Selectors

.

Selectors can be composed in parallel by just accepting the union of submissions within batch bounds, or sequentially, via consensus.

Note: This seems to have similar structure and issues to a mempool.

Solver

A solver only uses computation to produce matchings between intents. These could be deterministic algorithms for PTIME instances, or heuristics for NP-HARD/NP-COMPLETE ones.

For deterministic algorithms (at least), receipts can be provided via ZK proofs.

Solvers

behave like a single process to the outside, but could be built on top of e.g. privacy preserving MPC schemes running across multiple machines.

Picker

In case of non-unique solutions, the Picker

picks one solution according to some metric.

Policies

An Intent can always request to only be included in batches fulfilling certain IM Policies

, but some policies, called Intent Policies

can be heterogenous within a batch. These can always be extended to the whole batch, but not the other way around.

IM Policies

Example policy features which need to be decided at the IM level:

Solving batches being bounded

Submission of Intents

to solve batches and the solving of them could be bounded by wall-clock time via an oracle, logical time in respect to some consensus provider, or compute cycles (verifiable via receipts/proofs)

Batches could be unbounded

Once started, solve time could be unbounded, e.g. for NP-Hard Intents

and contingent on continued payment for solving.

TODO: Can we have dynamic solving alogrithms, that can incorporate new intents while running?

Retention of Unmatched Intents across batches

E.g. dependent on continued payment.

Privacy

Do the solvers do plaintext solving, or e.g. using a batched MPC scheme.

Intent Policies

This includes most things that are not purely Resource

based, e.g.:

Solving rewards

- A priori, e.g. payment for inclusion and compute attempts
- A posteriori, e.g. outcome dependent

Matching Constraints

- Restrictions on counterparty properties

Incentives and Economics

We assume that most of the MEV-like (SUAVE?) opportunities will lie in the Selector

and Picker

role (in cases where Solving

is verifiably correct).

In adversarial settings, composition between Selectors

and Pickers

needs to be well structured to not implement unwanted equilibria.

MEV Mitigations

- Ferveo, Commit/Reveal Schemes can help bound MEV opportunity.

Questions

- How to incentivize revelation of Intents

to composition components?

- Can we use trust/entanglement metrics to mitigate some of the incentive problems?

Open Problems

- Characterization of IM/IM composition (SUAVE-ish setting)