SUAVE Spec

This is a work-in-progress document. It is NOT the canonical/official SUAVE spec, only a personal interpretation with some AI-MEV flavored research (in section "Cognition and Complexity") on top. The ideas in this doc are inspired by discussions with Barnabé Monnot and based on the original ideas by Phil Daian.

For questions, contact sxysun.

Introduction

SUAVE is a permissionless credible commitment device (PCCD) that programs and privately settles higher-order commitments for other PCCDs.

Manual

- **the goal for this spec** is to answer: "starting only with basic theory, how can we derive the SUAVE design and demonstrate that it is the necessary design," and "if you take this design choice out or modify it, here is how things would go wrong"
- the goal is to be clear about “what are the open problems that we don't know how to solve, but here is the direction and problem statement”
- the goal of this spec is **not** to provide a full context for all of the involved concepts nor a complete API doc
- the goal of this spec is not to give general principle takes, but concrete market structure answers

Background

We assume readers already familiar with the concept of MEV. For an expanded discussion on uncertainty, readers can refer to the appendix.

Permissionless Credible Commitment Devices (PCCDs)

A major value proposition of cryptoeconomic mechanisms is that users can trustlessly collaborate by making credible commitments of their actions. Ultimately, the value of predicated action space and modified incentives comes from the (common) knowledge of the fact thatstrategic uncertainty of the game is reduced, thusimproving equilibrium and easing the reasoning.

For example, a game of Prisoner's Dilemma could be wrapped by players delegating their actions to agents that play cooperate only if the other agent commits to playing cooperate, thus benifiting both agents. In this case, the reduction of *strategic uncertainty* through the use of commitment devices increases welfare.

Similarly, the reduction of *information uncertainty*, which also increases welfare, can be implemented by the reduction of*strategic uncertainty* using commitment devices. For example, retail users commits to reveal that they are retail only if the market maker commits to providing a better slippage if the user's type is retail. This information uncertainty reduction through strategic uncertainty reduction is sometimes called contextual integrity or programmable privacy.

Formalization

Formally, a commitment device $\mathbb{D}$ is a tuple $\langle K, H, F \rangle$ where K is the commitment constructor, H is the history of finalized commitments, and F is the commitment semantics. A PCCD $\mathbb{D}$ wraps a game G by having agents constructing commitments using K, finalizing the constructed commitments into H by first resolving those commitments using the semantics F, and agents play the game G according to the constraints on H.

The program definitions of a commitment device look like (which highly resembles that of a game-theoretic formalization of blockchain):

``` coq

( *define the strategy for the original game G that is to be wrapped*)

( *here we write the strategy in Prisoner's Dilemma game as an example*)

Inductive Strategy :=

| Cooperate

| Defect.

( *define the commitment type, including the commitment constructor*)

( *here we write a higher-order commitment (commitment that depends on other commitments) as an example*)

Inductive Commitment :=

| mkCommit : (Commitment -> Strategy) -> Commitment.

( *define the state of finalized commitments, i.e., the latest blockhash*)

( *state differs from history because state includes settled commitments, i.e., result from executing the semantics*)

Inductive State :=

| genesisBlock : State

| succBlock : State -> list(Strategy) -> State

( *define the semantics for resolving commitments on PCCD to strategies in G*)

Definition semantics (cs: list(Commitment)): list(Strategy) :=

...

(* a PCCD is defined by its commitment constructor, its history of finalized commitments,

and its semantics for resolving (conflicting) commitments into strategeis that change the gameplay of G *)

Definition PCCD := {

commitmentConstructor = mkCommit;

state = genesisBlock;

append = fun cs => succBlock state (semantics cs);

}

```

For example, in the case of modifying a Prisoner's Dilemma game D like the one below (fixing two players):

		Kim	
		Cooperate	Betray
Don	Cooperate	2,2	0,3
	Betray	3,0	1,1

One might use the following commitment device $\mathbb{D}$ to wrap it into $G^{\mathbb{D}}$:

``` coq

Inductive Strategy :=

| Cooperate

| Defect.

Inductive Commitment :=

| mkCommit0 : Strategy -> Commitment0.

| mkCommit1 : (Commitment0 -> Strategy) -> Commitment1.

( *define the history of finalized commitments, i.e., the latest blockhash*)

Inductive History :=

| genesisBlock : History

| succBlock : History -> Option(Strategy) -> History

( *define the semantics for resolving commitments on PCCD to strategies in G*)

Definition semantics (cs: (Commitment, Commitment)): Option(Strategy, Strategy) :=

match cs with

| (mkCommit1 x, mkCommit1 y) => None (* cannot finalize two first-order commitments as they result in non-defined behavior,

    i.e., no strategy can be played in game G, leading to no termination/progress *)

| (mkCommit1 x, mkCommit0 y) => Some (x y, y)

| (mkCommit0 x, mkCommit1 y) => Some (x, y x)

| (mkCommit0 x, mkCommit0 y) => Some (x, y)

end.

(* a PCCD is defined by its commitment constructor, its history of finalized commitments,

and its semantics for resolving (conflicting) commitments into strategeis that change the gameplay of G *)

Definition PCCD := {

commitmentConstructor = mkCommit;

state = genesisBlock;

append = fun cs => succBlock state (semantics cs);

}

```

The game could end up with two players committing to:

``` coq

Definition coopiff (s: Commitment0): Strategy :=

match s with

| mkCommit0 Cooperate => Cooperate

| mkCommit0 Defect => Defect

end.

Definition KimsCommitment := mkCommit1 coopiff.

Definition DonsCommitment := mkCommit0 Cooperate.

```

which resolves to `(Cooperate, Cooperate)` as the finalized strategies that agents must play in the wrapped game $G^{\mathbb{D}}$.

Design Rationale

In the previous example, the PCCD actually doesn't achieve highest welfare for its users because`(Cooperate, Cooperate)` is unrecheable, as it requires one agent to commit to`mkCommit0 Cooperate` and this strategy is not a Nash equilibria in the commitment game. In fact, if there is no randomness allowed in the commitment game (commit to randomly sampled commitments), there is no Nash equilibria. Therefore, the addition of the PCCD to the game does not change the welfare.

However, if we increase the expressivity of the commitment device to also allow transfers conditioning on outcomes (demonstrated below):

``` coq

Inductive Commitment :=

| mkCommit0 : Strategy -> Transfer -> Commitment0.

| mkCommit1 : (Commitment0 -> Strategy) -> Commitment1.
```

```

and the semantics to allow first price auctions for finalizing only one first-order commitments, then the following commitment arise:

```
if Don commits to Cooperate and
   transfer me 1-ε after the game:
     I commit to Cooperate.
else:
     I commit to Betray.
```

which results in both agents frontrunning to finalize their commitment first as it means they can extract the full surplus welfare. Thus attributing value (MEV) to the PCCD.

Essentially, our goal with PCCDs is to increase the social welfare for players in the original game $G$ (users), i.e., make the welfare of the Nash equilibria in the wrapped game $G^\mathbb{D}$ equal the optimal welfare in original game $G$. However, this goal often cannot be achieved due to the inefficient design of PCCDs (e.g., its semantics encourages spamming, its expressivity is too low, etc,.). And those inefficient designs leads to the uncoordinated use of PCCDs, which is the cause of many MEV phenomena.

For example, in an AMM trading game, the highest welfare is approximatable by running a Walrasian Auction, where every user reports their demand (swap amount) conditioning on supply (result from other users' swap demand commitments), and their swap amounts settle each other with the resulting unsettled amount batched together and settled by trading against the AMM LPs. A simplified version of the protocol is the example below:

``` coq

Definition Strategy :=

| Swap: TokenId -> TokenId -> Amount -> Strategy

| Revert: Strategy.

Definition Commitment :=

| mkCommit: (Commitment -> Strategy) -> Commitment.

Fixpoint cowswap (s: Commitment): Strategy :=

match s with

(* is_inverse_direction checks if the commitment functions have the same structure,

e.g., AST, semantic implication, and then only differ in terms of swap direction but having the same amount *)

| mkCommit fx => if is_inverse_direction fx cowswap

    then

        Swap ETH BTC 100

    else

        Revert

end.

Definition UserCommitment := mkCommit cowswap.

```

However, this kind of PCCD commitments is not constructable on most blockchains, because most blockchains only support commitments that are propositional (i.e., they do not depend on other commitments). For example, the Ethereum PCCD looks like:

``` coq

Inductive Opcode :=

| PUSH : Word -> Opcode

| ADD : Opcode

| REVERT : Opcode

| JUMP : Opcode

| JUMPI : Opcode

| STOP : Opcode.

Definition Program := list Opcode.

Definition Strategy :=

| Transaction: Program -> Signature -> Strategy.

Definition Commitment :=

| mkCommit0: Strategy -> Commitment.

```

which only allows agents to commit to a single strategy that is not dependent on other commitments, which results in the highest welfare state being not achievable.

As we demonstrated in previous examples, high welfare state can only be achieved through elimination of strategic and informational uncertainty, which, in some games, require the PCCD to support more expressive commitment constructors. For example, in the Prisoner's Dilemma/Coincidence-of-Wants case, the highest welfare state is not achievable if we only have propositional commitments like the one Ethereum supports.

Theory
```

In game theory, propositional commitments are commonly modeled as Stackelberg games or disarmament games. Numerous studies have explored more expressive commitment devices, such as Reverse Stackelberg games, which correspond to first-order commitments (e.g., `mkCommit: (Commitment0 -> Strategy) -> Commitment1`). Another example is Program Equilibria, which correspond to higher-order commitments (e.g., `mkCommit: (Commitment -> Strategy) -> Commitment`). Mediated Equilibria and Correlated Equilibria also correspond to higher-order commitments that utilize a fixed templating language, for instance, a domain-specific language (DSL) to describe the commitment in mkCommit rather than a Turing Complete language.

Game theory research (specifically, folk theorem of commitment games) indicates that commitment devices can help achieve a broad range of payoffs. In fact, any individually rational and feasible payoff set in normal form games can be achieved through the use of commitment devices with higher-order commitments. Concretely, this means that with a PCCD that has higher-order commitments, there exists a Nash equilibria of the wrapped game $G^{\mathbb{D}}$ that achieves the maximum welfare of the original game G. The generalized result holds only for higher-order commitments (i.e., for any commitment constructor that is not higher-order, one could always construct a game where the optimal welfare of G is not achievable/not an equilibrium in $G^{\mathbb{D}}$).

|Constructor |Equilibria Notions| Pro | Con|

|-----|--------|--------|--------|

|Propositional|Stackelberg games, Disarmament games|current Ethereum|creates much MEV due to the uncoordinated use of commitment devices

|First-order |Reverse Stackelberg games|different payoff set from Propositional|different payoff set from Propositional

|Finite State Machines| FSM-commitment games|large payoff set, works well for many games|confusing to use and hard to design commitments to modify/embetter games

|Templated/DSL|Mediated equilirbria, Correlated equilibria|simple, always achieves highest welfare and eliminates MEV|brittle, has to be application specific, requires central mechanism designer, assumes the mechanism desiger a-priori-ly knows the best mechanism and that the fixed mechanism is always the best mechanism regardless of its environment and conditions (or that there needs to be a pivotal/centralized foundations team doing frequent hard-fork upgrades)

|Higher-order/Turing Complete|Program equilibria|robust, adaptive, works for all applications and x-domain MEV, has the potential to achieve highest welfare, allows truly decentralized mechanism design|to achieve equilibria in the wrapped game in a non-compute intensive way, needs establishment of focal points, which require a large community of mechanism designers that compete with each other to establish those focal points in the way of deploying commitment constructor templates as applications

Necessity

We present the intuition for why "solving MEV necessitates the PCCD to have higher-order commitments."

I. zero MEV

High welfare can only be achieved through coordination, and coordination requires the implementation of common knowledge. Blockchains act as PCCDs, which is a technology to implement common knowledge and therefore coordination. But exisiting designs of PCCDs fail to achieve this goal because the use of those commitment devices are not coordinated. For example, in the Prisoner's Dilemma and Coincidence-of-Wants case, welfare is reduced due to both strategic and informational uncertainty. Numerous other scenarios like stale quote sniping and gas bidding wars are all cases where uncoordinated use of commitment devices reduce welfare.

MEV is the value that originates from the uncoordinated use of commitment devices.

The phenomena of MEV appear to be paradoxical, as PCCDs' purpose is for coordination, and yet they fail to solve the problem of coordinating the use of itself.

But MEV is not a paradox, as it clearly exists (unlike Escher's staircase).

The reason why MEV appears to be paradoxical is because the utterance of "PCCDs solve coordination" presupposes too much ability of the PCCD to implement coordination/eliminate uncertainty (be it strategic or informational).

Concretely, suppose a PCCD has blocktime of t (i.e., it takes t time for it to implement the common knowledge of the most recent history/finalized commitments), then clearly it cannot be used to coordinate any events within t, simply for its inability to reduce the uncertainty via implementation of common knowledge, i.e., *the speed of common knowledge/making commitments credible/credibility is slower than the speed of the game.*

This is why, despite existing PCCDs have pseudo-complete Turing complete virtual machines and therefore able to universally simulate any kind of other PCCDs on top of it, including ones that have higher-order commitments, we still say they only support propositional commitments (because for the game of "using this PCCD," it is propositional, or even just a normal-form game).

This seems to imply that PCCDs cannot coordinate its own use, and because MEV is the uncoordinated use of PCCDs (this uncoordination happens before commitments gets finalized within t). Thus, there exists no PCCD that can "contain" its own MEV, therefore MEV always exists. Formally, *for any PCCD $\mathbb{D}$, there at least exists one game (the game of coordinating the usage of itself) that $\mathbb{D}$ can never coordinate.*

II. Assumptions

In this proof of non-zero MEV, we assumed that regardless of the environment where the PCCD is used and no matter how fast t is, there will always exist some uncoordinated use of the device, resulting in $W(Eq(G^{\mathbb{D}})) < W(\text{OPT}(G))$. This assumption holds in general because to reach coordination, we need:

Assumption 1. all agents to be aware of the game G

Assumption 2. the payoff set supported by the PCCD covers the optimal welfare outcome of G

We proved the non-zero MEV result under assumption of those two conditions (which is strong). Next, we relax those two assumptions for a more general non-zero MEV result:

Challenge 1. *the game G that is being played is too complex and almost infinite. In fact, G is often dynamic and constantly changes over time. Thus, some agents will always play "irrationally" because they are not aware of the full structure of the game G.*

For example, in the AMM trading game, the game G that is being played not only includes liquidity in that single AMM pool, but also liquidity in other pools in this AMM (optimal routing), liquidity in other AMMs (aggregators), liquidity that is off-chain (x-domain MEV), liquidity that is not yet present but could statistically emerge in the future (statistical arbitrage/market making).

Many MEV originates from the information asymmetry between agents on what the game G looks like.

Challenge 2. *existing PCCDs are Turing Complete, so they do cover the optimal welfare for any game that has a slower coordination time than its blocktime. However, the game of its own coordination is played as a normal form game (or some variant of a Stackelberg game, or some non-credible variant of a commitment game with propositional commitments).* Therefore, unless the MEV game (the game of its own coordination) already achieves optimal welfare/have zero Price of Anarchy, then the coordinated outcome payoff isn't covered by the PCCD.

III. Overview

To recap the proof up to now:

Proof Steps

Step 1. PCCDs implement coordination by making commitments credible

Step 2. The speed at which PCCDs make commitments credible is the speed at which they implement common knowledge.

Step 3. The speed that PCCDs implement common knowledge is their block time.

Step 4. Given a fixed PCCD with block time t, the game of agents using this PCCD is a game that is played and finishes at $t' < t$ (in addition to many other games). We refer to this game as the MEV game.

Step 5. Suppose the MEV game has a non-zero price of anarchy, i.e., it doesn't achieve optimal welfare.

Step 6. From 3, 4, and 5, we get that no PCCD can solve its own MEV (and its own MEV is not zero)

Some useful observations:

Observation 1. Although there are many games with coordination times $t' > t$ (the block time) that can be solved via PCCDs (e.g., the public goods funding game in Gitcoin, the payment transfers in Bitcoin), the MEV games are surprisingly significant, as seen in Ethereum today. In fact, one might argue that most value is in MEV games.

Some corollaries of the proof:

Corollary 1. If we can coordinate the game using PCCDs only, then MEV will always exist. This is because, for any PCCD deployed to solve the MEV game of another PCCD, there will always be at least one game that it cannot coordinate, leading to sub-optimal outcomes.

IV. Attempting a solution

One may attempt to coordinate MEV games via the following:

Proposal 1. Use a coordination mechanism other than PCCDs to coordinate the MEV game of PCCDs. It is possible that this "other coordination mechanism" does not have its own MEV game because the way it implements coordination is different.

For example, one could deploy a central mechanism that implements mediated equilibria for the MEV game, and this mechanism will not have the MEV game of itself because mediated equilibria restricts the commitment game to carving out bad equilibrias of the commitment usage.

Some examples of central mechanisms that solve MEV games include the flashbots auction, the Cowswap protocol, RFQ-based orderflow auctions like 0x protocol, etc,. Theoretically, for any fixed known normal form game G, it is always possible to solve for a mechanism that achieves optimal welfare.

However, this approach has the same cons as Templated/DSL-version of PCCDs, in that they are application-specific, brittle and centralizing.

Proposal 2. Make a PCCD that have infinite speed of credibility such that the time it implements common knowledge is 0 and there exists no game that it cannot coordinate.

This corresponds with first-come-first-serve (FCFS) ordering, implemented by most centralized exchanges. Indeed, FCFS solves MEV in the theoretical sense, but it is solving it in a hugely centralizing way, by admitting dictatorial control of colocated actors. It also ignores physics and encourages latency arms race which waste public resources.

Proposal 3. Make the MEV game has zero price of anarchy such that all agents are sufficiently coordinating the use of PCCDs from the beginning.

This approach is clearly not possible for non-trivial games. Moreover, per*Concern 1*, most games that contain economic potential have huge price of anarchy in their mev games.

However, one could intentionally "trivialize" the game by giving up efficiency. For example, enforcing every user of the commitment device to only do a very limited amount of computation/optimization. One such idea is chaotic iteration. The problem is that the efficiency loss from trivialization of the game might overweigh the gain from elimination of MEV.

Another approach to trivialize the game is to lengthen the time at which all the games are played like using VDFs, this approach is the inverse of FCFS as it is slowing down the speed of MEV games such that the speed of credbility/common knowledge can catch up, instead of making the speed of credibility faster. But clearly this approach gives up too much efficiency and will have a spam problem (as it degrades into searcher sybiling/PoW).

Proposal amendments. Here we address the generalized result in*Concern 1* and *Concern 2*, because in reality, without solving those two, MEV can never be solved.

Specifically to mitigate *Concern 1*, we need to optimize game structure and agent awareness. The assumption that the game G is too complex or dynamic, leading to agents being unaware of its full structure, can be challenged by creating mechanisms that simplify the game structure or improve agent awareness. This could involve providing more accessible information about the game or developing mechanisms that reduce the complexity of agent interactions.

In order to mitigate*Concern 2*, we can propose introduce the PCCD that has MEV with higher-order commitment constructors. The goal is to have a more expressive commitment device that is capable of covering a broader range of optimal welfare outcomes for various games (per folk theorem result earlier).

V. Solution

To solve MEV, we need to draw insights from all previous proposals and their amendments. We introduce SUAVE, a PCCD with programmable higher-order commitment constructors, as an approximate solution.

The problem we want to solve is MEV on existing PCCDs.

We know that those MEV on existing PCCDs arise from the uncoordination in their MEV games (game of coordinating to use those PCCDs). And we want to coordinate those MEV games.

Suppose the way we implement this coordination is to use SUAVE.

Suppose we implement SUAVE as a PCCD itself.

This necessitates that SUAVE has a shorter blocktime, ensuring its common knowledge implementation speed surpasses the speed of MEV games on platforms like Ethereum.

From *Concern 2* and folk theorem result of commitment devices, we know that only higher-order commitments or mediated commitments can achieve the optimal welfare solution. As discussed before, the mediated commitments are brittle and centralizing. Thus, SUAVE must provide higher-order commitment constructors for existing PCCDs. And because higher-order commitments are Turing-complete (since you can construct Curry's Paradox from its type), SUAVE must also be Turing-complete for it to be able to program higher-order commitment constructors for Ethereum.

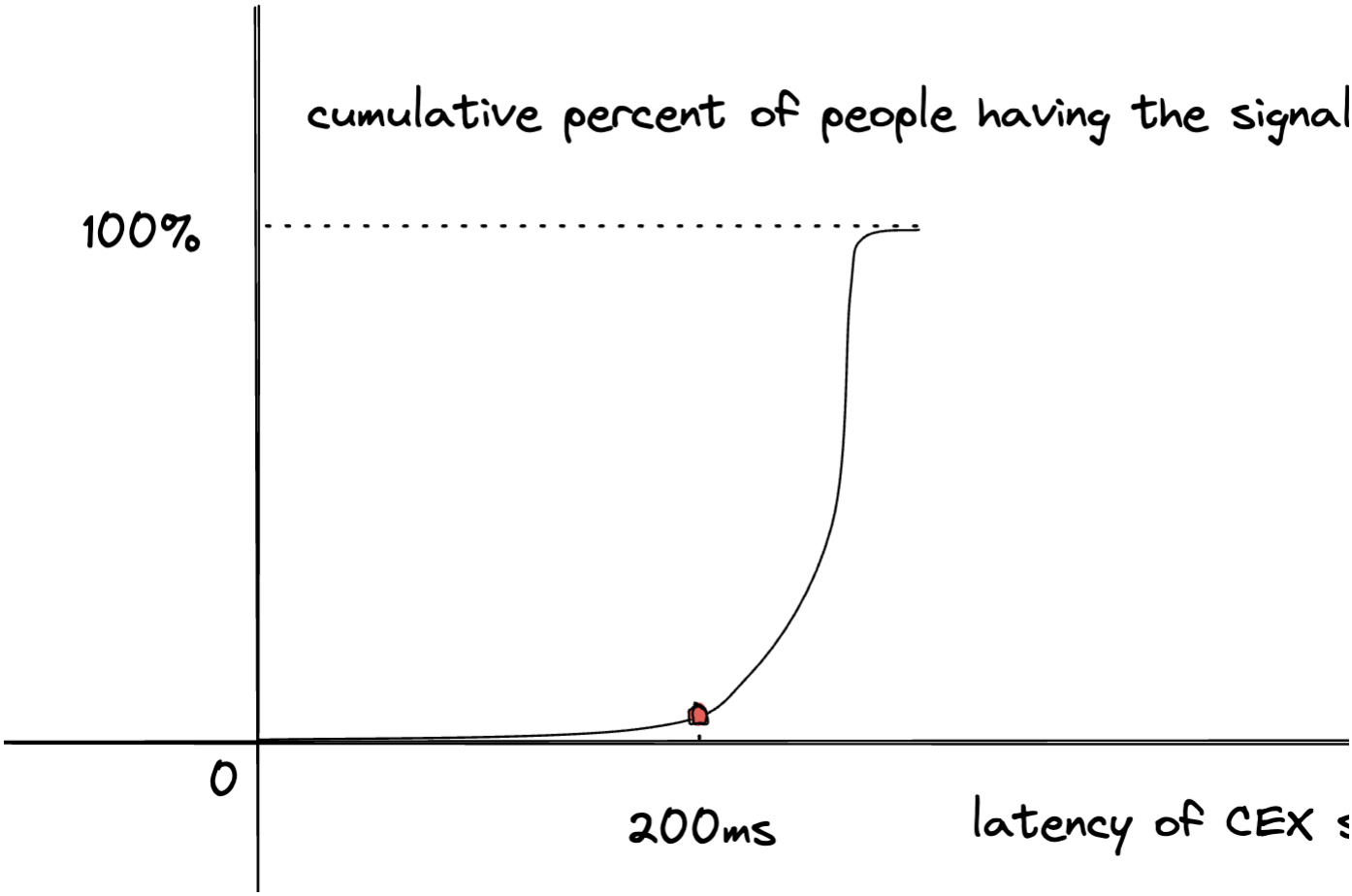
From *Concern 1*, we see that uncertainty of the game and the asymmetry in uncertainty for agents creates MEV. We separate those uncertainties into fundamental uncertainty and strategic/information uncertainty.

Higher-order commitments eliminates strategic/information uncertainty for a given fundamental uncertainty level, and it mitigates the Stackelberg leader (latency race) effect greatly if the level of fundamental uncertainty can be reduced. In fact, with lower expressivity in the commitment constructors, the strategic uncertainties just become fundamental uncertainties due to the inability to predicate/determine on the strategy space or the game structure.

In reality, fundamental uncertainty cannot be eliminated, but their quantity can be effectively controlled. For example, price of an asset is fundamentally uncertain, and it can be modeled as a brownian process that scales as square root of time, so if we reduce the time window of fundamental uncertainty down to some level then any player with a reasonable amount of capital can effectively hedge against the uncertainty.

Furthemore, because one cannot a-priori-ly prepare for fundamental uncertainty, it must be reflected by*cancellation* of prior commitments and re-submission of new commitments. Thus encouraging latency wars, centralization, and profit accrual to misaligned off-chain parties (as the ability to gather information/reduce latency is not commoditized, unlike the ability to compute, so profit flows more to tradfi trading firms that already have regulated centralized capital controls).

For example, building a reliable 200ms Binance price signal is not easy, thus creating a large asymmetry between player's ability of reducing fundamental uncertainty (which fosters huge centralization), as illustrated below:



This can be mitigated by introducing more certainty for the commitments to condition on. For example, offering low-latency centralized exchange price signals as an oracle service:

``` coq

( *allowing higher-order commitments to condition on real-time binance price*)

Inductive Commitment :=

| mkCommit: Commitment -> Oracles -> Commitment.

( *oracles provide additional information for*)

Inductive Oracles :=

| binancePrice: rational -> Oracles

| NYSEPrice: rational -> Oracles.

```

This idea is usually referred to MEV-time Oracles, which are essentially applications (higher-order commitment constructors) that people program on SUAVE.**We define MEV-time of a PCCD as the time before finalization of its commitments (i.e., its blocktime).** The idea is that since SUAVE is permissionless to use, mechanism designers can compete on building better higher-order commitment constructors for MEV games that is played on Ethereum/other domains that help to eliminate uncertainty. And as we devise better mechanisms/commitment constructors, equilibria payoff of $G^{\mathbb{D}}$ approach optimal welfare in G. For example, the classic adverse selection/free options/stale commitment sniping problem is about asymmetry in both strategic uncertainty and fundamental uncertainty, and they will be mitigated in the case of better commitment constructors.

However, there is a limit of fundamental uncertainty that one can achieve, so there will always be gains to information gathering. For example, there will always be someone with a better Binance price signal, and that person has no incentive to program the signal as a MEV-time oracle in commitment constructors. The idea with SUAVE is to make it easier for new mechanism designers to enter the market and drive up competition for existing information holders (i.e., the "someone with a better signal"), thus taking profit away from priviledged parties and drive centralization down.

VI. SUAVE MEV

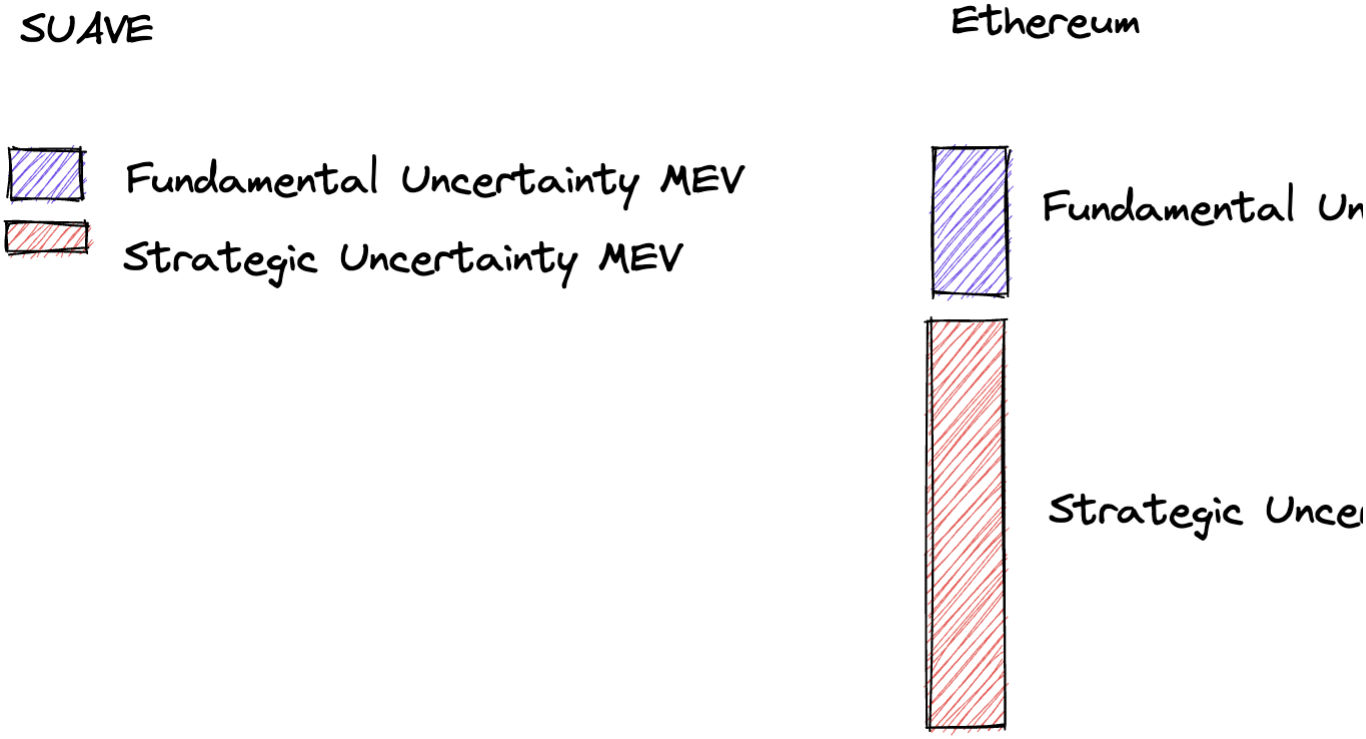
Of course, SUAVE, being a PCCD, has its own MEV game. This implies that SUAVE may need another PCCD that has even faster speed of common knowledge to solve SUAVE MEV.

This is possible. We analyze the sources of SUAVE MEV to see mitigations.

strategic uncertainty: One could prove the existence of SUAVE strategic uncertainty MEV by the following: since SUAVE is used to coordinate the MEV game on other domains, then it must be used as higher-order commitment constructors for other domains, which implies SUAVE must be Turing complete, which means one could always deploy/encode some arbitrary game G on SUAVE, and now the game G is being played at a time shorter than SUAVE's blocktime, which means SUAVE cannot coordinate the commitments there, thus leading to strategic uncertainty MEV.

Overall, the strategic uncertainty MEV of SUAVE originates from people's uncoordinated use of SUAVE's commitment constructors. And those commitments are either about (i) higher order commitments of MEV games on other domains, or (ii) commitments of games on top of SUAVE. For (i) there is no strategic uncertainty MEV as the coordination between higher-order commitments can be encoded in the higher-order commitment themself. And for (ii) we've proved the existence of it, but in reality the quantity should be low (it's abusing type systems for the purpose they were not designed for).

Therefore, we would expect most of non-SUAVE MEV comes from strategic uncertainty, which SUAVE solves almost completely (per the reasoning above, folk theorem of higher order commitments and the competition between mechanism designers to build better commitment constructors/focal points), while the MEV coming from fundamental uncertainty doesn't decrease much from lower latency of PCCDs.



fundamental uncertainty: We elaborate below.

For example, if we treat fundamental uncertainty just as random walk in price of assets, then for every latency improvement of 100%, the volatility you face decreases 30%. Suppose we hedge using volatility options, then the amount of capital needed to hedge this volatility (vega measure) is linear with respect to volatility. So an opponent with 200% your latency only need 43% more capital for hedging to compete (i.e., *they don't need huge investments into latency infra for cancellation and re-submission of higher-order commitments*).

Therefore, latency improvements on PCCDs have diminishing returns in elimination of MEV, as fundamental uncertainty, below a certain threshold, doesn't give much advantage to information/latency asymmetry players, which means, in this model, fundamental uncertainties can be almost perfectly hedged and conflicting commitments compete on price bids rather than latency/information investments. *This drives to a lower bound of MEV that cannot be improved easily. As a result, we don't expect to see many "SUAVE for SUAVE"s.*

However, in reality, fundamental uncertainties are not purely random walks, and previledged actors with information/capital advantage can still extract huge MEV. For example, insider information like knowledge of price impacting trades or whitehat hacks are innately discrete events that can never be hedged. More generally, for any kind of fundamental uncertainty that have discrete impacts, the commitment game will devolve into latency races or centralization to private SUAVE relays. *The quantification of the relative percentage of this kind of MEV is still an open problem.*

combination of fundamental and strategic uncertainty: The combination of fundamental and strategic uncertainty creates demand for privacy.

Specifically, without privacy, SUAVE commitments cannot predicate on the use of its information (which creates the free options/adverse selection/last look problem). For example, when sending a commitment on privacy-less SUAVE, one is facing great strategic uncertainty as other players can use the commitment information in any possible way, including frontrunning it. And there is no way to eliminate this strategic uncertainty using SUAVE because the usage is a game played within SUAVE blocktime, thus SUAVE cannot coordinate it (PCCDs face fundamental uncertainty for any game that is played within its "MEV-time"/block time).

In other words, the speed of certainty/predicatability of SUAVE is too slow to capture its own MEV game. We distinguish between speed of certainty and speed of common knowledge because the latter involves consensus/public broadcast of information while the former does not.

Therefore, the combination of both kinds of uncertainties means that SUAVE could have huge amounts of MEV from if there is no privacy in its commitments. For example, a generalized frontrunner frontrunning a whitehat hack submitted on SUAVE. In this case, privacy is required as agents cannot predicate on the context of use of their information fully (and that the context of use is discrete, i.e., the usage differs vastly before and after receival of the information).

One attempting solution is to eliminate this unpredicated use of commitment information of SUAVE via the use of another PCCD. But this doesn't work because it devolves into latency races as that PCCD cannot predicate on its own unrestricted commitment information usage. And because there will always be some commitments whose information is *discreet*, latency races to provide higher speed of certainty/predicatability will persist.

Therefore, privacy-less PCCDs doesn't solve or even mitigate this kind of MEV games (altho as mentioned before, the quantification of the magnitude of this kind of MEV is an open problem).

SUAVE solves this speed arms race via the use of SGX, which implements infinite speed of certainty, because if information cannot be taken out of it (i.e., there is privacy), then there exists no uncertainty in the predicated use of it. Thus, SUAVE solves the MEV game of unpredicated pathway of commitment information usage (programmable privacy) for Ethereum, and itself is free from the same MEV game. However, SGX itself has flaws like covert channels which usually can only be solved via custom compilers (which definitely is infeasible to do), *it is an open research problem on how to mitigate covert channels in SGX, and if there exists some better low-latency privacy primitives one could use instead of SGX.*

Other important open problems include:

- *suave transactions needs to be private, so there needs to be both a way for people to verify the correctness of the SUAVE settlement of higher-order commitments and not have their higher-order commitments be publicly visible ex-post.* This should be solvable using some kind of cryptography.
- *SUAVE settlement function's output is a bunch of Ethereum transactions, but those transactions needs to be submitted to Ethereum in a bundle as a private relay, which kind of adds centralization to the relay level and implies we need another PBS thing. This private sending is needed because the Ethereum transactions, if not private, will get frontran/sandwiched etc,.* This should be solvable using ePBS.

VII. Preference-gas

Since higher-order commitments are Turing complete, we need a way to ensure its termination. We choose to rely on focal points and fee markets to solve this (for an expanded reasoning on why this choice, refer to this doc). Essentially, each higher-order commitment will have a resource called preference gas, and it gets charged when SUAVE computes the settlement of those commitments (i.e., turning commitments into strategies).

However, the usage of preference-gas depends on collective behavior, i.e., how much resource is used to compute and resolve preferences cannot be easily attributed to a single actor. For example, it is possible that how much preference-gas you gets charged depends on how other people construct their commitments. Of course, we can avoid this dependency (and possibly a griefing attack vector) by running some version of shapley value style preference-gas mechanism (i.e., charge everybody their marginal cost to compute commitment settlements), but it could be computationally intensive. *The economics of preference-gas is an open research problem. Ideally, we want a mechanism that is robust to malicious attacks, computationally feasible, and agents can relatively easily determine a-*

priori what the preference-gas amount of their commitment should be.

Preference-gas is used to upper bound the amount of computation for resolving commitments. For efficient resolution, we rely on focal points (e.g., godel numbering)*One open problem is devising more efficient higher-order commitment settlement methods/algorithms. For example, does the lob's theorem/probabilistic convergence method in game theory help here? How do we match preferences robustly (as the constructors people use might be different) given the large space of possible commitments? How do we more efficiently match preferences in presence of composability (as the constructors people use might be composed of other constructors)?*

Cognition and Complexity

Higher-order commitments require users to predicate on the context of usage of their commitments, which is requiring each user (principal) to delegate their MEV-time actions to a program (agent) that bargains with other delegated programs (agents).

Essentially, a higher-order commitment of Ethereum (i.e., the SUAVE commitment) is an AI that is aligned with the sender's interest that enters into a MEV-time coordination game with other (individually aligned) AIs. And SUAVE's settlement of those higher-order commitments (i.e., the semantics F of the SUAVE PCCD) is a sandbox simulation of all those AIs' bargaining process, where the result of the settlement (i.e., the finalized commitments) is the result of sandboxed AIs' bargaining.

For example, one could implement a mini version ofCicero on SUAVE by first encoding the game of Diplomacy on Ethereum, and encode the higher-order commitment constructors on SUAVE as neural networks. Then, each user just submits their own neural network representing their AI player as a higher-order commitment to SUAVE and the semantics/settlement function of SUAVE is simply to simulate all of those neural networks given the current state in the game of Diplomacy and settle the bargained result, which is a contract between the AIs (e.g., Turkey agrees to declare war on Russia if Italy moves its troops to the Hungarian border).

Clearly, this ties the design of SUAVE with studies of cooperative AI, establishing more correspondences between the two is still an open problem.

In general, the higher the resolution, the less one needs cancellation/latency, because each cancellation/resubmission is an update on the piece of consciousness, and the higher the resolution, the more robust it is to uncertainty (which increases as time goes on). So if one already has high resolution copies, one doesn't need to update frequently as their copy is robust to MEV-time uncertainty.

One example is that statistical arbitraguers need cancellation in mev-boost bundles/blocks because their resolution is purely "swap a to b," which is extremely low resolution. At the same time, end-of-block searchers don't need any cancellation because their searching algorithm logic is on-chain and therefore already have a high-resolution of consciousness.

One could implement this resolution of consciousness by treating higher-order commitments as a lookup table from other commitments and oracles to transactions (much like the Chinese room argument). The benefit of this is that one can compute a SNARK of the lookup table locally using low latency infrastructure, and then send the SNARK to SUAVE. In this way, SUAVE can compute high-resolution consciousness with relatively slow validator setup inside SGX. *How to implement such a scheme for succint computation of consciousness is still an open problem.*

One concrete direction is to study the relation between the amount of compute/data needed to mimic consciousness (i.e., build a good enough bargaining AI, or compute the optimal commitment, which is undecidable) and the duration of MEV-time. Because as the time of uncertainty (blocktime/MEV-time) grows, the amount of compute and information that's needed for the delegated agent to mimic the true user's behavior grows drastically.

This is evidented by the fact that existing AIs can only surpass humans at short-term optimization tasks and fails terribly at longer-term tasks that require strategy and coordination. Ultimately, the blocktime of SUAVE is the time at which the consciousness model is updated. The longer the blocktime, the worse the consciousness model behave, and more likely they become "stale quotes" that gets sniped off by latency searchers.

Centralization

One might ask: for any problem that SUAVE solves, we have primitive solutions (e.g., Cowswap, flashbots auction, 0x orderflow auction, etc,.) already, why do we need a separate PCCD to solve those problems?

The reason is that current solutions (MEV-time applications/mechanisms that coordinate Ethereum commitments) are centralizing, because they operate as offchain systems whose credibility is guaranteed by trust and legitimacy. And trust/legitimacy has huge network effects. For example, users submit orders to Cowswap's centralized private offchain relay because they expect other people to also send their orders to them (because they expect most people expecting most people to know of Cowswap and its reputation), such that they get a higher probability of coincidence of wants (better execution).

As a result, even if someone has a better MEV-time mechanism than those solutions, they cannot compete with because of lack of trust/legitimacy, as those off-chain systems have converged to being the focal point of coordination of Ethereum commitments. For example, credible/reputed relays/wallets/blockbuilders can cartelize/vertically integrate and centralize their position by offering MEV-time mechanisms that are offchain (e.g., mevblocker).

For example, a reputed DEX/blockbuilder can offer an orderflow auction (coordinating user commitments on Ethereum) solution that is based on private market maker deals and RFQ systems, such that even if someone were to have a better MEV-aware DEX design, they cannot move the existing DEX's monopoly position because the off-chain MEV-time mechanism has already ended up as the focal point. This discourages free market competition and poses huge centralization risk as those private MEV-time applications can start extracting monopoly rent and it's extremely easy for them to get regulated by governments and nation states (as we've seen in the case of OFAC censorship), which enforces geographical centralization.

Therefore, the goal with SUAVE is to remove this centralization risk. As the SUAVE PCCD acts as a credibility lender that lends any MEV-time mechanism deployed on it credibility at low interest rates. In this way, we prevent the centralized MEV-time application dystopia because (i) the deployment cost of new mechanisms is lowered and (ii) the switching cost between mechanisms is lowered.

Ultimately, high welfare can only be achieved if we have decentralization in the sense of Hayekian "use of knowledge in society." In the face of fundamental uncertainty in the world, we can't possibly have confidence in "the mechanism we have at hand is the best mechanism for MEV and is robust in face of future changes," therefore we must design the market in a way such that MEV-time mechanism designers can and are incentivized to freely compete. Robustness in MEV solutions can only be achieved via efficiency in MEV mechanism markets.

SUAVE Design

We derived the design of SUAVE starting only from properties of commitment devices, i.e., SUAVE is a permissionless credible commitment device (PCCD) that program and privately settle higher-order commitments for other PCCDs.

System Overview

1. Introduction

SUAVE is a permissionless credible commitment device (PCCD) that program and privately settle higher-order commitments for other PCCDs. It should look like a faster blocktime version of Ethereum but with SGX to ensure private mempool.

1. Accounts and State

SUAVE maintains a global state, which consists of accounts. There are two types of accounts: Externally Owned Accounts (EOAs) and Contract Accounts. EOAs are controlled by private keys, while Contract Accounts are controlled by the code they contain. The state is stored in a Merkle Patricia Trie.

1. Transactions

Transactions are the means to transfer value and data between accounts. They contain fields like nonce, gas price, gas limit, recipient, value, data, and signature. Transactions are signed by the sender and submitted to the network.

```
SUAVE transactions are private. SUAVE transactions are meant to act as higher-order commitments to other domains such as Ethereum.
```

1. Preference-gas and Payment

Preference-gas is a measure of computational resources in SUAVE. Every transaction has an associated preference-gas cost. Users pay for preference-gas with Ether when executing transactions or smart contracts. SUAVE validators receive the fees as a reward for validating and including transactions in SUAVE blocks.

1. Blocks and Blockchain

SUAVE's blockchain is an ordered sequence of blocks. Blocks contain a header and a list of transactions. The block header contains fields like parent hash, beneficiary, state root, transactions root, receipts root, logs bloom, difficulty, number, gas limit, gas used, timestamp, extra data, mix hash, and nonce.

1. Consensus

SUAVE uses some consensus algorithm to agree on the state of the network.

1. Smart Contracts and EVM

SUAVE smart contracts are self-executing contracts with the terms of the agreement directly written into code. SUAVE uses the Ethereum Virtual Machine (EVM), a Turing-complete virtual machine.

```
Smart contracts on SUAVE should be higher-order commitment constructors (MEV-time applications) for domains like Ethereum. The semantics of those higher-order commitments should also
```

1. State and Storage

The SUAVE state is updated after each transaction or contract execution. The state consists of account balances, nonces, and contract storage. State transitions follow specific rules, which determine how accounts and storage are updated.

```
SUAVE state include settled higher-order commitments for other domains, i.e., the resolved propositional commitments (e.g., a transaction on Ethereum).
```

1. Networking

SUAVE relies on a peer-to-peer network for communication and synchronization. Nodes use protocols like devp2p or libp2p to discover and connect to peers, share transactions, and synchronize blockchain data.

1. JSON-RPC API

SUAVE nodes expose a JSON-RPC API for interacting with the network. This API allows querying the blockchain, sending transactions, managing accounts, and subscribing to events.

Technical Specification

We describe one implementation of the SUAVE PCCD.

Recall the definition of a PCCD: a tuple $\langle K, H, F \rangle$ where K is the commitment constructor, H is the history of finalized commitments, and F is the commitment semantics.

For SUAVE we have:

``` python
# The commitment constructors of SUAVE
```

class Oracles:

```
# this should be some kind of ERC-equivalent on SUAVE
# i.e., a json that describes the data structure, along with the data
def __init__(self, name, data):
    self.name = name # name of the oracle field
    self.data = data # content of the oracle
```

class Commitment:

```
def __init__(self, action=None):
    self.action = action # the commitment function
    self.other_commitments = [] # the commitments
# it should be some kind of predicated commitment structures
# i.e., P(other_commitments), to prevent unaccounted usage of information
# e.g., P is the MEV-time application of "does there exist another swap in opposite direction"
def add_other_commitments(self, commitment):
    self.other_commitments.append(commitment)
```

class Transaction:

```
def __init__(self, commitment, pref_gas):
    self.commitment = commitment
    self.pref_gas = pref_gas
    # self.pref_gas_price = pref_gas_price
    # there is MEV in choosing which higher-order commitments to settle
    # i.e., the credibility of commitment DA is not guaranteed
    # still possible to create private bundler relays, sell higher-order commitment orderflows, or have censorship
```

```
# The commitment history of SUAVE
```

class State:

```
def __init__(self):
    self.transactions = []
    self.state = [] # the state is the list of blocks of settled commitments
    self.oracles [] # the oracles should be a bunch of contract calls to state
    # hard to do real-time oracles, as there exists credibility problem and people cannot compete on it without centralization (as it requires both low latency and expectation of liqu
def add_transaction(self, transaction):
    self.transactions.append(transaction)
# The commitment semantics of SUAVE
# This is the building algorithm, can be smart about it
def settle(self):
```

```
    pcom = [] # the settled propositional commitments
    for txn0 in self.transactions:
        for txn1 in self.transactions:
            txn0.commitment.add_other_commitments(txn1.commitment)
        pcom.append(EVM.run(txn0.commitment.action(self.state), txn0.pref_gas)) # execute the commitment using SUAVE state
            # deduct pref_gas inside the function
    self.state.append(pcom)
```

def construct_txn(action, pref_gas):

```
return Transaction(Commitment(action), pref_gas)
```

the cowswap commitment constructor

def cowswap_action(commitments, oracles):

```
# you can charge fee here because you expect everybody to use this commitment constructor
# this should be a smart contract on SUAVE that you call
for c in commitments:
    if is_in_opposite_direction(c):
        # match c's data to see if c is constructed using the same constructor, if yes then resolve
        return swap(a,b)
    else:
        return revert
```

Applications and Examples

- orderflow auctions
- flashbots auction
- Cowswap
- Penumbra
- walrasian auction
- programmable privacy

Conclusion

SUAVE is a permissionless credible commitment device (PCCD) that program and privately settle higher-order commitments for other PCCDs. We showed how this design is derived purely from the theory of commitment devices, and provided a technical specification for its implementation.

Appendix

On Uncertainty

Refer to this paper for a formal description of the difference between fundamental and strategic uncertainty.

Expanding on the taxonomy of different uncertainties, the elimination of strategic and informational uncertainties via commitments require the information and strategy/action space to be able to be predicated on and determined without ambiguity. As a result, they must be known unknowns (that can become known knowns).

Of course, in reality, even if we can predicate on the action space, the cost of computing/determining the validity of that predicate is often too high, so some actions/information might as well be modeled as fundamentally uncertain. Therefore, predicating on them loses purpose. Moreover, our physical world limits us to what information can be measured/determined so there are fundamentally some predicates that we cannot use, such as time-based ordering of transactions in distributed systems.

Those uncertainties, despite fundamental, are all known unknowns (and we know they cannot be known, ever), and we can simply not have our commitments predicate on them. On the other hand, the existence of unknown unknowns (unknown uncertainties) poses a greater threat to commitments: because they are truly unknown, we cannot possibly predicate on them or not predicate on them.

As a result, when unknown unknowns becomes known, they can invalidate many commitments. Furthermore, we cannot possibly account for unknown unknowns when making commitments while keeping the commitments credible, as being credible would mean our change to the commitment is able to be predicated on.

Therefore, we use the term fundamental uncertainty (to contrast the predicatable uncertainties that PCCDs can deal with, namely informational and strategic) to refer the three cases of limitations as to coordination using commitments:

- (i) the unknown unknowns
- (ii) the known unknowables
- (iii) the known knowables that cannot be efficiently determined/computed.

We probably live in a world where all those three kinds of fundamental uncertainties are prevalent. As a result, the efficiency boost we can possibly achieve though the use of PCCDs is limited by the boundary of those uncertainties.

An interesting side note is that we can often exchange different kinds of uncertainties with each other: the fundamental uncertainty in one dimension might be transformed into some more tractable uncertainty that is predicatable in another dimension.

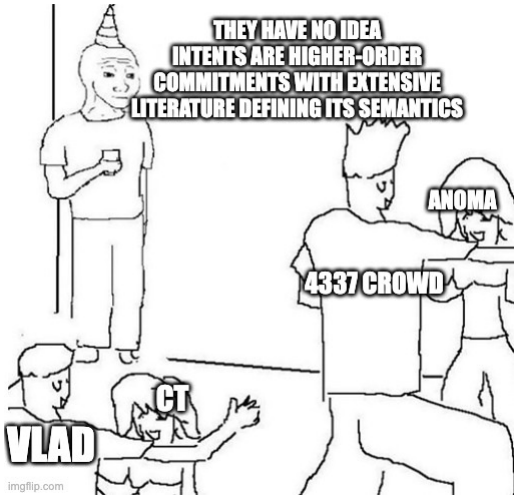
For example, if it is hard to measure the outcome of some game (e.g., a distrust of oracles, or simply because the space is too huge), then maybe we can try to determine the certainty of player's action space by delegating game play to algorithms.

Another effect of the existence of fundamental uncertainties is that they turn one's position of being the first one to commit (the Stackelberg leader) from a blessing into a curse, because the longer one waits to fix the content of the commitment, the less fundamental uncertainties there are. So the one who commits later might actually face less uncertainty (risk) than others who commit earlier.

Essentially, this is the case where commitment "takers" get a free option as they "snipe" the stale quotes from the "maker" of commitments.

Sometimes this elimination of fundamental uncertainty for the one who commits later gives so much advantage such that being the Stackelberg leader becomes an unfavorable position because the gains from elimination of strategic uncertainty and informational uncertainty is less than the gain from elimination of fundamental uncertainty.

Memes


THEY HAVE NO IDEA
INTENTS ARE HIGHER-ORDER
COMMITMENTS WITH EXTENSIVE
LITERATURE DEFINING ITS SEMANTICS
ANOMA
4337 CROWD
CT
VLAD
imgflip.com