NOTE

This article has been published here. Leaving the thread for posterity.

END NOTE

Information flow control, “intentpools”, universality, and the end of supply-side blockchain economics.

tl; dr

- Intents can be understood colloquially as commitments to user preferences over the state space and mathematically as atomic information flow constraints.
- In a practical sense, intents:

Only make sense in a multi-party context. Represent some kind of non-deterministic computation in a complexity theory sense (there may be more than one possibly acceptable state). In order to be concretely efficient, refer to and constrain a small subset of the entire state space which the user is interested in. * Only make sense in a multi-party context. * Represent some kind of non-deterministic computation in a complexity theory sense (there may be more than one possibly acceptable state). * In order to be concretely efficient, refer to and constrain a small subset of the entire state space which the user is interested in. * An intent-centric architecture (such as Anoma) organises all of the internal components and their structure on the basis of this mathematical concept of an intent. * Intents aren’t magic - in a sense, they’re primarily a reconceptualization of how we understand what these kinds of distributed systems look like from the user’s perspective, but they’re a reconceptualization with many concrete implications for protocol design. * Intentpools will require P2P network designs capable of expressing and accounting for heterogeneous roles, instead of assuming that all nodes are exactly equal as most current mempool designs do. * Ultimately, the power in well-designed distributed systems rests with the users. If users select carefully to whom to send their intents, measure what goes on in the network, and reward good behaviour, they will retain control over the incentive equilibria - and if they don’t, no protocol design can change all that much.

Introduction

“Intents” are in vogue, but in the Discourse the concept remains a wee bit ethereal, eluding attempts at concrete characterisation. Intents are everything, nothing, and each point in between. I’ve particularly enjoyed a few genres of take:

From the fully automated luxury space communism wing:

Don’t forget, think of the users:

upload_b53c8f933ab85e4f4af12c5d698f7c42599×626 147 KB

From the penumbral polity of protocol pragmatism:

upload_90aca044d34d424dd8d48c50e24fc823606×673 78.1 KB

From the department of reasonable skepticism:

To which I can only respond

upload_e105b504e3b9b256fb69576217929b28757×330 64 KB

Finally, from the MEV economic realist wing (or possibly Paradigm’s secret underground LAN party), Quintus and Georgios wrote an article on intent-based architectures and their risks. This article does an excellent job of articulating the relation of “intents” to Ethereum’s current architecture and ecosystem topology and describing some risks in transitioning to an intent-centric world. I write partially in (late) response to it, with the hope that the synthesis of an Ethereum-centric perspective and our own can illuminate possible paths forward and areas of collaboration. Let me start off, first of all, with why I think these paths might be interesting.

Why intents?

Developing a new substrate for coordination in the ashes of the old involves a certain conceptual tension. We remain economically tied to the old order, yet must break free from its concepts and measures in order to articulate a clear idea of the new. To me, “intents” is useful as a coordinating concept which can help us sail this space of sociotechnical transition and seek out smooth seas on the other side. I think we want a coordinating concept with three key properties: one which encodes a recognition of the duality between the system and the world, one which possesses a certain universality with respect to coordination systems, and one which guides a technical design philosophy of maximum compositionality. Let me break each of these down in turn.

First, duality. What a protocol sees and what a user sees are not the same. Users are not interacting with blockchain systems in order to provide job guarantees for protocol designers. Users interact with systems because they want something, and they care only about whether or not the system can give them what they want. What they want may include not only directly measured outputs but also higher-level and over-time properties, such as fairness - but still, users care

about outputs and properties of the system, not the implementation details of the interior. When you invoke the Solidity compiler, you want output EVM bytecode which (a) correctly implements the semantics of your program and (b) consumes as little gas as possible. Unless you know some optimisation that the Solidity compiler does not, specifying an exact sequence of EVM bytecode operations would be both time-consuming for you and suboptimal for your program compilation, since it would restrict the freedom of the compiler to find potentially more efficient sequences which still implement your program correctly. As a concept, “intents” takes this intuition of duality already present in compilers and applies it to distributed coordination systems themselves. Just as a high-level language compiler provides mechanisms which allow attention and time-constrained programmers to control complex hardware machines without concerning themselves with all of the low-level details, intents allow resource and knowledge-constrained users to reason about the behavior of a complex distributed system, and ensure that it aligns with their intentions without concerning themselves with all of the low-level details.

Second, universality. Articulating, matching, and settling the intentions of the participants is simply what coordination systems do. We owe the preconditions of our existence to the OG decentralized coordination system - language - and language operates on the basis of discursive commitments - intents par excellence. In using coordination systems, users seek the ability to express what they want, discover others with whom they can cooperate, and agree on a plan which utilizes their joint capabilities to realize a particular future. Properly articulated, a theory of intents can allow us to describe what systems do from the perspective of a user, as opposed to what they look like from the inside. In this sense, an “intent” is not specific to blockchains. In interacting with the American dollar and legacy financial superstructure, for example, we also have intents, just a much more restricted range of choices. A theory of intents can help us move towards clearly articulating the question of “what do we want our economic system to do”, only with which we can choose a configuration of protocols and culture that best suit our needs. If “duality” corresponds to the soundness of an intent system (a user can express an intent with confidence that the execution of the system will be in fact constrained in the way that the intent describes), then “universality” corresponds to the completeness of the intent system (an intent system allows anything a user might want to express for the purposes of coordination, because the most a user can ever do is express their intent).

This universality also helps illuminate what protocols, tokens, and language can’t do:

Indeed.

Third, compositionality. If working on distributed systems has taught me one thing, it is that distributed consensus is damn expensive, and distributed consensus on a distributed consensus protocol doubly so. If we had an infinite time horizon, perhaps we’d manage to finally reach agreement on the one true serialization scheme (it is definitely not any of these), but we don’t. As Moxie put it, the ecosystem is moving, and the existential risk we face is not that we’ll fail to develop a correct set of protocols eventually, but rather that we won’t do it in time. Open source and building in public provides the prerequisite context for efficient ecosystem collaboration, but alone it is insufficient, for we still end up with a plethora of protocols which are ever-so-slightly incompatible with each other because they make ever-so-slightly different design assumptions the effects of which ripple downstream. Similar concepts go under many names - modular blockchains, dependency inversion, denotational design - but whatever you call it, intent-centricity as a philosophy of software design offers a way out of this coordination conundrum, because minimalist intent-centric software components compose automatically, without any coordination or even communication necessary on the part of the software designers.

So, duality, universality, compositionality - quite the laundry list - but wait…

What are intents?

Theory time. Let’s start from the top: in the context of our design domain of discrete information systems, what is an intent?

Quintus and Georgios define an intent (informally) as a signed set of declarative constraints which allow a user to outsource transaction creation to a third party without relinquishing full control to the transacting party. I think this definition describes how intents work with existing blockchain infrastructure, which was designed to work with transactions, very well: a third party can create transactions on behalf of the user, and a signed intent can give them the freedom to choose the execution path while still respecting user-defined programmable constraints. However, I do not think it captures the essence of the concept.

I have a confession to make: I’ve used the word “intent” for a long time, but I haven’t been entirely consistent. Initially, it didn’t strike me as a particularly important word - it just seemed like a way to describe why users would bother to use distributed ledgers in the first place. As part of writing this article, I trawled a bit through the history, and the earliest written mention I found was 2018:

upload_57e10048a8b9e8718df7f8fb6b19461a759×104 15.1 KB

upload_27a53f92323b945b3770b58fa2740060753×141 25 KB

upload_be445b5340fad2c7c8588a38f119f7d2724×340 60.2 KB

Here, “intent” refers to the underlying user preferences, not a precise technical component (intents are “expressed by” orders), and the protocol handles only settement (counterparty discovery is left to “off-chain infrastructure”). Let’s say that the first component is “commitments to preferences”.

After launching Wyvern, I didn't keep working on intent systems design - at the time, there wasn't much interest in the Ethereum ecosystem, and I thought blockchain systems would need to facilitate political polycentrism, so I instead went to work on Cosmos and IBC. I also kept tabs on Zcash (earlier, I'd written a block explorer), since it seemed like blockchain systems would also need privacy. I've always found it a bit tragic that these three ecosystems have remained so separate - in part, Anoma aims to offer a synthesis, and do our part to help knit the communities together.

Anoma comes at this question from an opposite but complementary angle: coordination -

upload_3348e9d8d8ea878307796268727fe33f1156×481 86.3 KB

I find philosophical language personally helpful for reasoning about systems, since it can capture the common shapes of relations shared by many different specific instances, but I've also found that it lacks the precision of mathematical definitions, without which agreeing on what exact protocols to actually run can be quite tricky.

In that spirit, mathematically, I understand intents as atomic information flow constraints. By authoring an intent, a user can describe which future information flows are compatible with their preferences, often including disclosure of information or authorisation of state changes contingent on disclosure of information or state changes from or by other users. Intents can be understood, equivalently, as functions which define what histories are allowed to include them. Implicitly, intents commit a user to accepting future histories including and consistent with their intent, as these histories will be compatible with their preferences insofar as those were expressed in the intent itself.

This definition is still awfully abstract, so I think it's helpful to illustrate by construction. Imagine a world - let's call it Spherical Cow Land. This world has some cows (no, not that sort), and these cows want to coordinate:

upload_e869c0fe05fdddfe51a0f2c7c41854e4512×512 435 KB

In this world, there are:

- Some number of agents (the cows) A_i, indexed by i.
- An external reality which they can measure and take action in. Actions could include moving, kicking, mooing, etc. - typical cow shenanigans. We can understand these measurements and actions as messages sent to and received from the world.
- A magic intent computer with instantaneous encrypted messaging to all agents (cows).

This magic intent computer takes a progression of states over time.

upload_c254540d3bdc9751bdae63c77a3df4a21539×364 16.1 KB

- Each state S contains i partitions, each of which is the partition tracking what is known by cow A_i (these partitions may include pointers to common state known by multiple cows which doesn't need to be repeated).
- The history H is a succession of states: H : List\ S
- An intent I is a relation over the history and new state: I : H \to S \to 0 | 1.

Each cow can send intents to the magic intent computer whenever they like. Periodically, at some internal clock rate, the computer performs an update action with the following steps:

1. Calculate the Pareto-efficient frontier of new states which include as many received intents as possible.
2. Randomly pick a new state from this frontier.
3. Append the new state to the history.
4. Send messages to all cows who have learned new information, according to the state partitions.

For illustration, think of the progression of intent computer steps in terms of who knows what. Atomically, with the processing of each batch of intents, cows learn new information:

upload_02f220ba59d2e1034bb7ffcbb5015a691348×916 73.4 KB

Now, nothing constrains the cows to actually acting based on this information. The intents, state, and intent computer processing rules (just as for any computing system) are purely syntax - they have no necessary semantics, or correspondence to the world. However, this system of intent computation has some very nice game theoretic properties. Informally, any state mutually preferred by any subset of the cows who can themselves enact the actions required to reach that state can be reached by the intent computer if the cows express their preferences as intents. The intent computer can also model physical constraints - for example, uniqueness and linearity of physical resources. See this paper for a further formal exposition.

Now, in tandem with this intent computer, the cows can build a culture - of semantic relations between the state, measurements, and actions in the world, of expressing their preferences as intents, and of adherence to the conditional commitments to action expressed in their intents. This last part is particularly important - as of course, any cow who previously committed to say "moo" can choose not to - but this contradiction in word and deed will be visible to the other cows, who can socially ostracise the violator. The intent computer itself, just like a written constitution, statute, or legal precedent, possesses no force except that which is granted by cultural interpretation. In modern culture, we conceptualise this notion of adherence as the "rule of law".

Realizing an intent computer

In this construction, we relied on a magical intent computer, which can communicate with all agents instantaneously, search through the space of possible solutions instantly, and return the best one (by some definition) immediately. This instantaneous intent computer doesn't really exist - it would violate several physical laws - but when we build blockchain systems, in a sense we are trying to approximate it: the fairer and faster the communication, the deeper and wider the search, and the more mutually satisfying the solution, the better.

In theory, there is really only one problem: time, or the physicality of computation. There is no such idealized intent computer - rather, the approximations must be physically instantiated in a particular location and with particular operators, who cannot in general prove that they have behaved perfectly, and could go offline. Often, in blockchain systems, we make this role distributed and fault-tolerant using consensus protocols. There is no one-size-fits-all solution to consensus, because it depends on the physical details and semantics of the applications - who is using them? where are they? what latency and security do they need? etc.

In practice, there are cryptographic questions - how can we instantiate this information flow control? When are certain primitives most appropriate? What security assumptions are we willing to make about the capabilities available to potential adversaries? Different folks may be willing to make different assumptions here, but there are some that are pretty universal, at least at the moment - if P = NP, for example, we'd all better head back to the drawing board.

Physical computers have an architecture - a specific way of organising subcomponents into a structure designed to serve the purpose of the overall system. There can be many different architectures with the same purpose - they might differ in the components, the design of internal abstractions, and the structure of their composition. Anoma is an intent-centric architecture for this sort of intent computer - which might prompt the question…

What is an intent-centric architecture?

An intent-centric architecture, as Anoma understands it, uses this concept of an intent to organise all of the internal components: each interaction with a subcomponent describes an atomic operation and information flow constraints, but the choice of exactly how to perform the operation is made by the subcomponent itself.

Two concrete examples:

- P2P routing: Anoma's P2P routing system works on the basis of cryptographic identities. In order to send a message, you specify the public key(s) of the recipients who should receive it, information flow constraints around who you do or do not want to learn any metadata during routing, and preferences concerning bandwidth usage, latency, etc. With this description, the P2P intelligence engine figures out how to actually send the message, including physical addresses (IP), intermediate hops, redundant paths, etc, while satisfying the provided constraints.
- Execution: Anoma's transaction execution system works on the basis of declarative specifications of what keys transactions might read and write and a requirement of serializability. With this description, the execution engine figures out how to actually run each transaction, balancing them between many parallel processors, while satisfying the provided constraints.

Fully realized, an intent-centric architecture as defined in this way has two interesting properties. First, an intent-centric architecture is maximally modular: as interfaces with subcomponents specify only exactly the observable behaviour necessary, new implementations of those subcomponents can always be swapped over time (e.g. new, faster ZKP systems). Second, for a particular scope of problem, an intent-centric architecture is unique: there is only one. Further exposition and formalization of these properties is an area of active research.

There are many roads to Rome. Lots of fine folks are working on infrastructure for intents these days - Skip, Essential, Agoric, and PropellerHeads, to name a few - and others are working on specific intent application verticals - CoW Protocol, Penumbra, and Seaport, for example. Anoma takes this fully general road because we can and because we think someone should. I think inductive (specific) and deductive (general) approaches pursued simultaneously in a culture of open research are very complementary, as they can use each other for test data, guidance, and theoretical consistency checks.

One fruit of the inductive approaches so far is an excellent set of implementation constraints, concerns, and pitfalls to avoid. I think these have been best synthesised in Paradigm's post, and I want to address several topics they bring up in particular.

Intent-pools and false dichotomies

Georgios and Quintus focus in particular on the concept of an "intentpool" - how exactly will these intents be gossiped around?

In the absence of the Ethereum mempool, intent system designers are now faced with some design questions. One high-level decision is whether intents will be propagated to a permissioned set or be available in a permissionless manner so that any party can execute the intent.

(diagram)

I think there are two primary dichotomies in play here. The first is the dichotomy between public and private intentpools. I understand public as they describe it to mean that any node can join and see the intents, and private to mean that only specific pre-authorized nodes can see the intents. The second is the dichotomy between permissioned and permissionless intentpools. I understand permissionless as they describe it to mean that any executor (specific party involved in intent settlement) can execute any intent, and permissioned to mean that only specific pre-authorized executors can execute the intents. Used in this sense, a private intentpool is also implicitly permissioned, as executors who do not see the intents cannot execute them. Another dimension of permissioning not explicitly covered in their post is is also important: submission - where we could say that anyone can submit intents to an intentpool with permissionless submission, while only a specifically designated set of parties can submit intents to an intentpool with permissioned submission. Let's put this all in a table of possible configurations:

Intentpool type Who can submit intents? Who can see intents? Who can execute intents?

Permissioned submission, private, permissioned execution Specific parties Specific parties Specific parties

Permissioned submission, private, permissionless execution Specific parties Specific parties Anyone (who can see the intents)

Permissioned submission, public, permissioned execution Specific parties Anyone Specific parties

Permissioned submission, public, permissionless execution Specific parties Anyone Anyone

Permissionless submission, private, permissioned execution Anyone Specific parties Specific parties

Permissionless submission, private, permissionless execution Anyone Specific parties Anyone (who can see the intents)

Permissionless submission, public, permissioned execution Anyone Anyone Specific parties

Permissionless submission, public, permissionless execution Anyone Anyone Anyone

While these distinctions are conceptually workable, they do not actually map to real systems. The Ethereum mempool is not permissionless. Rather, it is permissioned to anyone who holds enough ETH to pay for gas fees for a particular transaction. That set of potential transaction authors is quite dynamic, but it is still a permissioned set, just with a complex update function (the EVM state transition function itself). Truly permissionless systems do not exist in the real world, because they have no way to prevent DoS. The difference between the permissioning of the Ethereum mempool and the permissioning of someone's API is merely the distribution function of the tokens providing authentication.

I think there are two important points to note here. First, these three dimensions are not binary choices, but rather spectra. On the most permissioned end, a given intent pool might have, say, only one designated executor. On the most permissionless end, anyone can execute. Many interesting points lie in between: any executor staking a certain token, any executor with such-and-such an execution history, any executor within a particular executor DAO, just to name a few examples. Permissioning of submission and visibility of information, similarly, are spectra, and many points in between the two extrema may be compelling for different specific application needs.

Second, and more fundamentally, I think there is a subtle, implicit assumption in play here: that whether a system is permissionless or permissioned and private or public is a choice made by the intent system or intentpool designer. I do not think this assumption holds. At an operational level, intents are just messages sent between peers. What messages to accept (submission permissioning) and who to allow to read them (privacy) are choices made by individual nodes, not intent system designers. What executors to allow to execute an intent is a choice made by intent authors, since intent authors define the validity conditions of settlement. These choices by intent authors and P2P node operators may be informed by software defaults, but ultimately they will be dictated by cryptoeconomic equilibria.

Let's talk a bit about how Anoma's intent gossip system actually works. In order to do this, it will be helpful to first introduce the concept of roles. In distributed systems, there are many individual nodes, each of which can receive and send messages, store data, perform computation, and provide attestations (signatures). The way in which they do so often takes particular patterns, and many of these patterns are expected to be adhered to in order for the system to work as designed. For example, one very common role in proof-of-stake systems is that of a validator. Validators are expected to receive most messages for a particular chain (blocks, transactions, and votes), sign and send around their votes according to particular consensus logic, and typically execute the transactions (so they can sign, say, only valid blocks).

Considering just the peer-to-peer networking part, a role is defined by what messages a node wants to receive and send. Nearly all blockchain P2P networks in existence assume that all participating nodes take the same role - that of full node, or validator - and that they want to receive and send all messages. This assumption may make sense for a P2P network intended to relay transactions between full nodes and validators who are all supposed to be processing all of them, but it does not make sense for a P2P network intended to relay intents, for two reasons:

- First, there will be a lot of intents. Many - perhaps most - intents will never be matched and make their way into transactions - they'll just be gossiped around for awhile, then expire. This is intentional - broadcasting intents should be cheap - but it makes P2P designs where all nodes receive all messages economically unworkable.
- Second, most nodes are probably not interested in most intents. Receiving and processing messages costs energy, and nodes will only want to receive and process messages when they have some reason to expect that doing so will

be of benefit to them - perhaps through fees, through later payment from a community, through reciprocity with friends, etc. - whatever the specifics are, nodes will want to be able to express preferences about what intents they want to receive, and share those preferences with the network so that they receive only intents they want to process.

Anoma's approach is to standardise the protocol and make these roles explicit in descriptions of preferences for what kinds of intents each node wants to receive, and what commitments they are willing to make about processing those intents (e.g. signing blocks). Application-specific intent systems such as CowSwap and Penumbra fix a particular topology of certain roles as part of the architecture. For example, in Penumbra, the roles of gossip, ordering, solving, and execution all coincide - the same nodes are performing all of them. In CowSwap, intents are submitted to a single relayer, which compares solutions sent by different solvers, then periodically sends the best to Ethereum in a settlement transaction. Different specific topologies will make sense for different application designs and goals. Anoma aims to make these toplogies of roles explicit and programmable, and let different users, applications, and communities experiment with what configurations make most sense for them. For more details on how Anoma's P2P layer is instantiated, please see the recent paper.

(had to split into two posts due to Discourse length restrictions)