Getting to know about an ~intense

~ intent-centric world & some other thoughts.

Intents have been slowly getting popular in the research discourse in the crypto space, and the concept is being built upon by various protocols like Anoma, Essential, etc.

This article is meant to be a primer to the various views, ending with what an intent resolution architecture could look like where intents are expressed in natural language. If successful, intents show the promise of revolutionizing application architecture at every level. Hence, the concept merits discussion, delving into its nuances and what it might take to reach there.

Understanding an Intent

Intents allow users to specify certain transaction conditions or preferences without providing exact message calls. This allows for more flexibility and potentially less on-chain complexity.

In "Intent-based architectures and their risks," the definition is, "an intent is a set of declarative constraints which allow a user to outsource transaction creation to a third party without relinquishing full control to the transacting party.

"

In a recent podcast, Chris Goes from Anoma defines it in two ways: "Credible commitments to preferences over a state of some system" and "credible commitments to information flow constraints."

An intuitive way of thinking about intents is that intents are basically, Intents are desired outcomes. When you express an intent, you simply define the outcome you want, not the process to get there.

Let's say your desired outcome is trading some Tether (USDT) for Ethereum (ETH). Instead of managing the whole process yourself - picking the exchange, making accounts/signing transactions, handling transfers (or converting dust in your wallet(, etc. - you submit an intent stating:

"I want the outcome of trading my 1 Ethereum for 2000 USDT."

Some other entity, called a Solver, takes your intent and figures out how to fulfill it. The solver handles the messy details of trying to optimize for the best possible outcome for you.

The key is that intents focus on outcomes rather than processes. You define the desired results, while someone else determines the "how." Intents greatly simplify the transaction flow that most users use in crypto by allowing you to specify outcomes and not worry about steps.

Now that we have built the basic idea, the higher-level idea is that users define what they want without specifying the contracts they want to route their transactions from (which we can call computational paths or simply transaction routes). The users might also constrain this by saying they prefer certain pathways or contracts over others.

A few examples from the past and present

Cowswap

Cowswap utilizes batch auctions as its core price-finding mechanism. Rather than executing trades immediately like AMMs, Cowswap aggregates orders off-chain and settles them in batches. This enables establishing uniform clearing prices across all trades in a batch, eliminating issues like front-running common with immediate execution models. Batch auctions also optimize gas costs by settling many trades simultaneously. An open competition between solvers takes place to submit order settlement solutions that maximize trader welfare for each batch. The best solution sets the finalized uniform prices. Overall, batch auctions bring fairness, efficiency, and MEV protection that immediate execution cannot.

A key innovation enabled by Cowswap's batch auction model is the ability to find coincidences of wants (CoWs) among orders. CoWs are direct peer-to-peer settlements between trades with reciprocal wants. This liquidity sharing means no external liquidity providers are necessary to facilitate these trades. CoWs can also involve multiple assets simultaneously in ring trades. By maximizing CoWs, batch auctions access more liquidity than isolated pools. Settlements will utilize CoWs when possible, with any remainder executed against on-chain liquidity sources. Combining batch auctions and CoW liquidity sharing gives traders superior pricing and execution.

The CoWswap model is similar to the intent model, where users elicit their intention to trade in the form of a limit order, which goes into the order book, where solvers use the order book state to match them in the form of ring trades or route through AMMs (that is the user only mentions the price, they do not mention the computational path or where exactly they want to execute).

Uniswap-X

The Uniswap X paper proposes a decentralized trading protocol that uses signed off-chain orders and on-chain settlement via Dutch auctions. Users sign orders specifying parameters like input/output tokens, quantities, and price bounds. These orders are disseminated to "fillers" who compete for the best execution price.

Uniswap X suggests setting initial Dutch auction prices through an off-chain RFQ system. Users can poll a network of fillers for quotes and grant a brief exclusivity period to the best quote, incentivizing honest pricing. The order then proceeds to an open Dutch auction.

There are some similarities between Uniswap X and Cowswap:

- Both use off-chain signed orders aggregated and settled on-chain in batches. This provides gas savings compared to on-chain orders.
- Both aim to find the best execution prices by promoting competition among liquidity providers (Called solvers in the case of cowswap and fillers in the case of uniswapX)
- Cowswap emphasizes the coincidence of wants to power direct peer-to-peer trades, whereas Uniswap X focuses more on integrating off-chain and on-chain liquidity sources.

Both use off-chain signed orders aggregated and settled on-chain in batches. This provides gas savings compared to on-chain orders.

Both aim to find the best execution prices by promoting competition among liquidity providers (Called solvers in the case of cowswap and fillers in the case of uniswapX)

Cowswap emphasizes the coincidence of wants to power direct peer-to-peer trades, whereas Uniswap X focuses more on integrating off-chain and on-chain liquidity sources.

The RFQ (Request for quotation) system in Uniswap X and the signing model with your intent specifying your wishes and letting someone else fill your order are similar to intent architectures.

Towards a formal definition of intents

Instead, users simply express their intent, such as "I want to trade X asset for Y asset." Sophisticated actors called solvers then figure out how to fulfill that intent optimally, handling all the blockchain specifics behind the scenes. Solvers give proofs that the intent was fulfilled and may participate in mechanisms like auctions to fulfill intents in a decentralized way.

Some definitions are discussed in this blog:

1st Model: An intent i

i

i

is defined as a tuple (

B

,

E

,

T

)

(B, E, T)

(

B

,

E

,

T

)

:

- B

B

B

represents the set of supported "begin" states.

- E

E

E

represents the set of supported "end" states.

- T

T

T

is a set of preferred sequences of transactions.

- The state transition function s

:

Q

×

T

s : Q × T

s

:

Q

×

T

→ Q

Q

Q

moves from a beginning state to an end state through a sequence of transactions t

t

t

.

B

B

B

represents the set of supported "begin" states.

E

E

E

represents the set of supported "end" states.

T

T

T

is a set of preferred sequences of transactions.

The state transition function s

:

Q

×

T

s : Q × T

s

:

Q

×

T

→ Q

Q

Q

moves from a beginning state to an end state through a sequence of transactions t

t

t

.

An intent is considered fulfilled if it starts in a state q

0

q_0

q

0

∈ B

B

B

and ends in a state q

n

q_n

q

n

∈ E through a transaction sequence t

∈

T

t ∈ T

t

∈

T

.

Intent Clearing: A set of intents ı

0

,

…

,

ı

m

ı_0 , … , ı_m

ı

0

,

…

,

ı

m

can be cleared if their B

,

E

,

T

B , E , T

B

,

E

,

T

sets have non-empty intersections, allowing the creation of a meta-intent ı

′

ı ′

ı

′

with the intersections.

As we mentioned earlier, Intents are elicited by the user and then solved by a solver; when it is represented in any such format, the landscape of intents becomes an optimization problem for the solver. A very layman's way of thinking of this is the user might elicit an intent like “I want to buy 4 ETH worth of BTC,” and the solver would generally find a place to fill or swap this order. But intents don’t stop here; they also allow for constraints to be applied like “Slippage as low as possible”, and “Don’t transact on DEXes that do not allow US users”, which then become additional constraints the solver has to keep in mind.

While there’s another model discussed, the challenges include:

- The need for simplified expressions of intents.
- Possible welfare implications of specific intents, e.g., zero slippage in DEXs.
- Situations where the execution trace may matter due to risks or legal reasons.

The need for simplified expressions of intents.

Possible welfare implications of specific intents, e.g., zero slippage in DEXs.

Situations where the execution trace may matter due to risks or legal reasons.

The aim is to balance the need for expressivity in capturing user intents and preferences against the practical considerations of computational efficiency and user experience.

There’s also a Lagrangian interpretation of Intent search mentionedhere.

To me, the formulation of intents looked like a Markov decision process. Still, while Markov decision processes have stochastic state transitions, this would be a Deterministic MDP with absolute state transition values, which could then be solved with value iteration, policy iteration, or MCTS (Monte Carlo tree search) (This last part was also used to solve the game of Go in Alphago).

Intents as a boon for UX

Intents could be the next stage of UX evolution on-chain. The current way of on-chain UX is the transaction level, where a user signs every transaction that is a part of the action. So, every step on the chain is expressed through a transaction. Intents, in very simple words, are meta transactions where the activity is expressed at a very abstract level, and it is left to the solver to do their best to meet the user's intent demand. This could include buying some ETH for X amount, and you want the best possible trade, which could either be structured in a single huge swap on Uniswap on Ethereum or sliced across rollups and then bought to Ethereum (calculating fees, too).

Today, a simple swap from USDC to eth would include you approving the limit of the token, approving the token type, and then approving the trade, whereas in an intent-centric world, you could be abstracted away from these details and work with the actions you are interested in taking. An unofficial rule that exists in web design is the idea of no action taking more than three clicks to perform; today, to swap, you have to select both the tokens and perhaps adjust the slippage and the transactions as elicited above, which might not seem like a lot for one swap but over time becomes cumbersome UX.

Unibot, in a very elementary way, offers a glimpse into what intent-specific architecture could look like. It takes away the complex parts of the transaction and offers a simple and accessible UX for traders to trade coins, albeit with some constraints on possible flexibility. Even though there are purported risks around key handling and thus introducing an attack vector, the existence of a consistent user base for the app despite taxes and fees shows unexplored UX opportunities in crypto.

A conversational Intent flow

Where does AI come into the conversation in an intent-centric blockchain world? The idea of intent recognition has been around in Natural language processing for a few decades and is heavily studied in the context of dialogue. Suppose, for example, you go to a travel website and talk to a chatbot; you initially have a purpose, which might be booking the flight or checking on the reservation or status, after which you provide various details. In case of booking a flight, you give the "to" and "from" destinations, the time, the date, and the class of ticket you are interested in; in some instances, you may also have to choose the airport. In this example, your purpose is the intent of the conversation, and the various details you gave are slots that need to be filled to accomplish that intent.

Annotated Dialogue States in a conversation

Another example of intent recognition and slot filling is when you intend to play a song, and there are various slots (details) in the sentence related to the song, like the song name and song artist.

Another Intent Example

In a dialogue world, Intent classification and slot filling is a highly complex problem because your dialogue can span multiple turns, and sometimes there is a global intent and a local intent, and you have to keep track of numerous states. Whenever you've used Siri and Google Assistant in the past decade to schedule an alarm or note something in the calendar or birthday, some form of intent classification and slot filling was active in the background.

How does this relate to blockchains? As we move from a transaction-specific world to an intent-centric one, the details of how we go from intents to transactions have yet to be actively present in popular discourse. The interface between the "intent pool" and the "mempool" is non-existent. Accessing On-chain models and using them for intent recognition and slot filling provides one way of a natural language interface intent pools and solvers (and, in my opinion, a most natural one).

The broad idea of this would involve accessing a set of models on a chain that reduces every intent to a DSL (Domain Specific language); this domain-specific language would include details like the core intent ("buy," "sell," "bridge," "borrow/lend") and have other slots like "address," "size," "slippage preferences" (depending on the type of intent). Global DSL allows anyone to deploy a model to reduce intent to this specific DSL. In the presence of multiple such models, a voted majority is taken over the ensemble of models.

The availability of on-chain models helps us develop this interface securely and provably, where proof of computation could be attested for every intent/proof of solving. In some way, capturing the majority voting of various models in some cases might give us insight into how the intent was chosen and, in some rare cases, even help solvers solve these intents better.

The onchain model used here could be a standard deep learning model, like BERT, trained for this very purpose or use inference of large language models in the ensemble; this detail can depend on the various participants or the solvers. In the case of encrypted intent pools, we would need to use Homomorphic encryption or private inference methods to ensure data privacy while still computing over it. Every epoch or every few epochs, a proof could be posted on the chain where the model is the prover. A verifier, either a human or another model, posts a statement about the model's efficacy. Whether it can still accurately process an intent, this last part ensures that the models' lifecycle is considered. Sometimes, when the verifier is a sophisticated participant, it might find pitfalls of the model that can be quickly resolved and replaced with a newer model.

As we see below, for the action/idea "buy with stablecoins and dust in my wallet," once it goes into the intent pool, it is then passed through a variety of models and resolved to a DSL with various details like the intent, the sub-actions and the slots that need to be filled. This resolution to a DSL could be as detailed or abstract as possible; the intent conversation could last a few turns since the threshold of "dust" might have to be determined. Once the DSL is in place, solvers could choose optimal paths to convert these balances to ETH and then pass on the transactions to the mempool.

A sample intent resolution architecture

Another DSL Architecture - Essential

Account abstraction is a concept that decouples accounts from signers in Ethereum by making all accounts smart contracts. This allows accounts to be customized with different authorization logic tailored to users' needs. However, implementing full account abstraction requires substantial changes to Ethereum's core protocol.

EIP 4337 takes a different approach to deliver the benefits of account abstraction without consensus layer changes. It introduces "user operations" - pseudo-transactions submitted to an alternate mempool and bundled by "bundlers" into transactions calling an EntryPoint smart contract.

This allows features like social recovery, paying fees in any token, and transaction batching. Developers can build custom accounts adapted to different use cases. By avoiding protocol changes, EIP 4337 can bring these benefits to Ethereum more quickly. However, it does introduce new complexity and actors like bundlers and paymasters. The resulting dynamics around multiple mempools, incentives, and transparency will require thoughtful management.

Intents allow users to specify a desired outcome rather than a specific action. Solvers then help users achieve that outcome in the best way. However, current implementations have limitations around centralization, lack of composability, and inadequate competition among solvers.

An EIP proposed by Essential sets to change this. Account abstraction through initiatives like EIP 4337 enables smart contract-based accounts rather than traditional Externally Owned Accounts (EOAs). This unlocks the possibility for users to submit generalized intents rather than pure transactions. Intents represent a user's desired outcome and can be combined by "solvers" to maximize participant satisfaction.

EIP 7521 proposes a framework to support an evolving landscape of intent standards without requiring constant smart contract wallet upgrades. Users sign "User intents," specifying which "IntentStandard" contract should process the intent. These get submitted to an "EntryPoint" contract that handles signature verification as they did in EIP 4337. The User Intents mempool exists alongside the ERC 4337 mempool. Solvers process the intents.

Anoma

Anoma is an intent-centric architecture centered around programmatic intents rather than transactions to build infrastructure layers. Intents are partial state changes signed by users that express preferences, unlike complete state change transactions. This intent-centric design enables decentralized counterparty discovery and solving. The paradigm that Anoma is trying to shift to is a declarative paradigm from an imperative one.

From Adrian Brink's Talk on Intent Centric apps

Users broadcast intents, which propagate across an intent gossip network. Various nodes can specialize in gossiping about certain intents based on their computational resources and the kind of intents they might want to serve. Solvers observe intents and attempt to combine compatible ones into valid transactions that can be settled on-chain. Transactions are submitted to an encrypted mempool using threshold cryptography, so they cannot be front-run. Anoma also has a partial intent model, allowing for intent combinations.

Chris goes talk on Intent x Rollups - Anoma partial intent model

The focus on privacy when it comes to Anoma is about the “choices at the user level,” i.e., giving a user the flexibility to reveal information about their intents and also choose which parts they want to reveal if they want to do it at all.

The architecture consists of multiple components. The Tiger execution engine handles transparent, shielded, and private data using ZKPs and homomorphic encryption. Typhon is the consensus algorithm. The compiler stack includes the Juvix language, AnomaVM, and VampIR.

The architecture has a homogeneous protocol but a heterogeneous security model. It can be deployed as a standalone blockchain or used to decentralize apps on Ethereum, for instance, as a ZK rollup or to decentralize order books. Users with different security needs can leverage the same protocol while choosing their desired security tradeoffs.

Anoma makes building decentralized applications easier compared to transaction-centric models. Intents enable new applications like runtime rollups, multivariate bartering, and private DAOs. Overall, Anoma provides a flexible and modular architecture tailored to the requirements of contemporary decentralized applications. Focusing on intents rather than transactions solves the issues of counterparty discovery and coordination while preserving privacy.

Anoma ultimately philosophically designed its protocol, looking at intents as “information flows” and “constrained/private information flows” from which the architecture and design choices follow. This also points to the fact that the Anoma intent composition model gives rise to generalized intent models, which may be technically difficult to solve under privacy constraints since efficiency tradeoffs would limit the amount of information that can be private.

Conclusion

Intents currently are a very interesting space within crypto as a research and engineering problem.

Open Problems with Intents:

- A formal definition of intent
- What do Intent-centric architectures for apps beyond Dexes look like?
- The privacy-utility tradeoff to solve any optimization problem is that you need as much information as possible. If private intents are to become a thing, a certain amount of information must be revealed for the intent to be solved.
- What is the minimum knowledge you need to solve an intent problem?
- What are you trading off by cutting access to the rest of the knowledge?
- Formulate this privacy-efficiency tradeoff in a generalized manner.
- What is the minimum knowledge you need to solve an intent problem?
- What are you trading off by cutting access to the rest of the knowledge?

- Formulate this privacy-efficiency tradeoff in a generalized manner.

A formal definition of intent

What do Intent-centric architectures for apps beyond Dexes look like?

The privacy-utility tradeoff to solve any optimization problem is that you need as much information as possible. If private intents are to become a thing, a certain amount of information must be revealed for the intent to be solved.

- What is the minimum knowledge you need to solve an intent problem?
- What are you trading off by cutting access to the rest of the knowledge?
- Formulate this privacy-efficiency tradeoff in a generalized manner.

What is the minimum knowledge you need to solve an intent problem?

What are you trading off by cutting access to the rest of the knowledge?

Formulate this privacy-efficiency tradeoff in a generalized manner.

As spoken about at the end of the last section, A general intent might be too vast to solve for and, in the case of seemingly big state space like ethereum, a very difficult problem to solve. This would point to the fact that Intents are best solved with some constraints and should also be limited when trying to be combined. (when there are common intents). In my opinion, Generalized intents are extremely difficult to implement in practice, and intent-centric architectures will be application-specific in nature.

While these are research problems, various engineering problems also arise with the design choices of implementing intent. It might result in an overreliance (permissioned) middlemen, which risks the centralization of infrastructure across various stacks (In the case of UniswapX, 77% of the volume is filled by off-chain inventory). It can also entrench trusted intermediaries, create high barriers to entry, and stifle innovation, something that is already being seen in MEV. Any intent protocol design must balance permissionlessness, privacy, transparency, and decentralization.

References

1. Intent-Based Architectures and their Risks By Quintus and Georgios
2. Christopher Goes - Anoma: an intent-centric
3. Are Intents, SUAVE, Account Abstraction, & Cross-Chain Bridging all the same thing? - Uma Roy
4. Realizing Intents with a Resource Model - Chris goes
5. Exploring Intents - No execution podcast
6. Adrian Brink - Intent-centric (intent-solver pattern) architectures for fully decentralized dApps
7. Adrian Brink - The 3rd generation is intent-centric
8. Anoma ❤ Celestia: intent-centric rollups - Chris Goes
9. A formalish definition of intents
10. Lagrangian mechanics of intent solving I by Fabrizio Genovese
11. Intents with Chris Goes from Anoma, ZK podcast
12. Wtf is Anoma? Wtf are Intents by Delphi Creative
13. Anoma
14. Essential
15. Cowswap Docs
16. Uniswap X whitepaper

Intent-Based Architectures and their Risks By Quintus and Georgios

Christopher Goes - Anoma: an intent-centric

Are Intents, SUAVE, Account Abstraction, & Cross-Chain Bridging all the same thing? - Uma Roy

Realizing Intents with a Resource Model - Chris goes

Exploring Intents - No execution podcast

Adrian Brink - Intent-centric (intent-solver pattern) architectures for fully decentralized dApps

Adrian Brink - The 3rd generation is intent-centric

Anoma ❤ Celestia: intent-centric rollups - Chris Goes

A formalish definition of intents

Lagrangian mechanics of intent solving I by Fabrizio Genovese

Intents with Chris Goes from Anoma, ZK podcast

Wtf is Anoma? Wtf are Intents by Delphi Creative

Anoma

Essential

Cowswap Docs

Uniswap X whitepaper